

# L4Re Operating System Framework

Generated on Sat Aug 24 2024 16:07:24 for L4Re Operating System Framework by Doxygen  
1.9.8

Sat Aug 24 2024 16:07:24



<b>1 Overview</b>	<b>1</b>
<b>2 Introduction</b>	<b>3</b>
2.1 L4Re Microkernel	3
2.1.1 Communication	3
2.1.2 Kernel Objects	4
2.2 L4Re System Structure	4
2.3 L4Re Runtime Environment	6
<b>3 Tutorial</b>	<b>7</b>
3.1 Customizations	8
<b>4 Programming for L4Re</b>	<b>9</b>
4.1 L4 Inter-Process Communication (IPC)	9
4.1.1 IPC mechanism	10
4.1.1.1 IPC Flags	10
4.1.1.2 Partner capability selector	11
4.1.1.3 IPC Label	11
4.1.1.4 IPC Message Tag	11
4.1.1.5 IPC Timeouts	12
4.1.1.6 User-level Thread Control Block	13
4.1.1.7 Transfer of Typed Send Items	15
4.1.2 Examples	15
4.1.2.1 User Thread to Kernel Object	16
4.1.2.2 User Thread to User Thread	16
4.1.2.3 User Thread to User Object	18
4.2 Capabilities and Naming	18
4.3 Spaces and Mappings	19
4.4 Initial Environment and Application Bootstrapping	20
4.4.1 Configuring an application before startup	21
4.4.2 Connecting clients and servers	21
4.5 Memory management - Data Spaces and the Region Map	22
4.5.1 User-level paging	22
4.5.1.1 Data spaces	22
4.5.1.2 Virtual Memory Handling	22
4.5.1.3 Memory Allocation	22
4.6 Program Input and Output	23
4.7 Initial Memory Allocator and Factory	23
4.8 Application and Server Building Blocks	23
4.8.1 Creating Additional Application Threads	23
4.8.2 Providing a Service	24
4.9 Pthread Support	24
4.10 Interface Definition Language	25

4.10.1 Parameter types for RPC	26
4.10.2 RPC Return Types	27
4.10.3 RPC Method Declaration	27
4.11 L4Re Build System	28
4.11.1 Building L4Re	28
4.11.2 Writing BID Make Files	29
4.11.3 prog.mk - Application Role	30
4.11.4 include.mk - Header File Role	32
4.11.5 test.mk - Test Application Role	33
4.12 Kernel Factory	40
4.12.1 Passing parameters for the create stream	41
<b>5 L4Re Servers</b>	<b>43</b>
5.1 Sigma0, the Root-Pager	45
5.1.1 Factory	45
5.2 Moe, the Root-Task	45
5.2.1 Moe objects	45
5.2.1.1 Factory	46
5.2.1.2 Namespace	47
5.2.1.3 Dataspace	48
5.2.1.4 Log Subsystem	48
5.2.1.5 DMA Space	48
5.2.1.6 Scheduler subsystem	48
5.2.1.7 Region Map	48
5.2.2 Command Line Options	48
5.3 Ned, the Init Process	49
5.3.1 Lua Bindings for L4Re	49
5.3.1.1 Capabilities in Lua	50
5.3.1.2 Access to L4Re::Env Capabilities	50
5.3.1.3 Constants	50
5.3.1.4 Application Startup Details	51
5.3.1.5 Reacting on task termination	52
5.3.1.6 Access to the kernel debugger	52
5.3.1.7 Using the interactive ned prompt	53
5.3.2 Command Line Options	53
5.4 Io, the Io Server	53
5.5 AHCI driver	59
5.6 Cons, the Console Multiplexer	61
5.7 Mag, the GUI Multiplexer	62
5.8 NVMe server	64
5.9 Uvmm, the virtual machine monitor	67
5.9.1 RAM configuration	75



5.10 l4vio_net_p2p, a virtual network point-to-point link . . . . .	76
<b>6 Bootstrap, the L4 kernel bootstrapper</b>	<b>79</b>
<b>7 Deprecated List</b>	<b>83</b>
<b>8 Topic Index</b>	<b>85</b>
8.1 Topics . . . . .	85
<b>9 Namespace Index</b>	<b>89</b>
9.1 Namespace List . . . . .	89
<b>10 Hierarchical Index</b>	<b>91</b>
10.1 Class Hierarchy . . . . .	91
<b>11 Data Structure Index</b>	<b>103</b>
11.1 Data Structures . . . . .	103
<b>12 File Index</b>	<b>119</b>
12.1 File List . . . . .	119
<b>13 Topic Documentation</b>	<b>131</b>
13.1 Base API . . . . .	131
13.1.1 Detailed Description . . . . .	134
13.1.2 Basic Macros . . . . .	134
13.1.2.1 Detailed Description . . . . .	136
13.1.2.2 Macro Definition Documentation . . . . .	136
13.1.2.3 Function Documentation . . . . .	137
13.1.3 C++ IPC Interface Definition. . . . .	138
13.1.3.1 Detailed Description . . . . .	138
13.1.3.2 Internal Helpers . . . . .	138
13.1.4 Cache Consistency . . . . .	139
13.1.4.1 Detailed Description . . . . .	139
13.1.4.2 Function Documentation . . . . .	139
13.1.5 Capabilities . . . . .	142
13.1.5.1 Detailed Description . . . . .	143
13.1.5.2 Typedef Documentation . . . . .	143
13.1.5.3 Enumeration Type Documentation . . . . .	143
13.1.5.4 Function Documentation . . . . .	144
13.1.6 Error codes . . . . .	146
13.1.6.1 Detailed Description . . . . .	147
13.1.6.2 Enumeration Type Documentation . . . . .	147
13.1.7 Fiasco extensions . . . . .	148
13.1.7.1 Detailed Description . . . . .	149
13.1.7.2 Function Documentation . . . . .	149

13.1.7.3 Fiasco real time scheduling extensions	151
13.1.7.4 Kernel Debugger	152
13.1.7.5 Kernel Tracing	159
13.1.8 Flex pages	162
13.1.8.1 Detailed Description	164
13.1.8.2 Enumeration Type Documentation	164
13.1.8.3 Function Documentation	169
13.1.9 Integer Types	180
13.1.9.1 Detailed Description	181
13.1.10 Kernel Interface Page	182
13.1.10.1 Detailed Description	183
13.1.10.2 Enumeration Type Documentation	183
13.1.10.3 Function Documentation	184
13.1.10.4 Fiasco-UX Virtual devices	188
13.1.10.5 Memory descriptors (C version)	189
13.1.11 Kernel Objects	194
13.1.11.1 Detailed Description	195
13.1.11.2 DMA space	195
13.1.11.3 Factory	196
13.1.11.4 IPC-Gate API	205
13.1.11.5 IRQs	208
13.1.11.6 Interrupt controller	222
13.1.11.7 Kernel-provided semaphore	238
13.1.11.8 L4 kernel object type information	240
13.1.11.9 Platform Control C API	241
13.1.11.10 Scheduler	247
13.1.11.11 Task	255
13.1.11.12 Thread	265
13.1.11.13 Virtual Console	296
13.1.11.14 Virtual Machines	311
13.1.12 Memory operations.	330
13.1.12.1 Detailed Description	330
13.1.12.2 Enumeration Type Documentation	330
13.1.12.3 Function Documentation	331
13.1.13 Memory related	332
13.1.13.1 Detailed Description	333
13.1.13.2 Macro Definition Documentation	333
13.1.13.3 Enumeration Type Documentation	335
13.1.13.4 Function Documentation	336
13.1.14 Object Invocation	340
13.1.14.1 Detailed Description	341
13.1.14.2 Timeouts during IPC	342

13.1.14.3 Enumeration Type Documentation . . . . .	342
13.1.14.4 Function Documentation . . . . .	343
13.1.14.5 Error Handling . . . . .	358
13.1.14.6 Message Items . . . . .	364
13.1.14.7 Message Tag . . . . .	367
13.1.14.8 Realtime API . . . . .	381
13.1.14.9 Timeouts . . . . .	381
13.1.14.10 Virtual Registers (UTCBS) . . . . .	389
13.2 EDID parsing functionality . . . . .	401
13.2.1 Detailed Description . . . . .	402
13.2.2 Enumeration Type Documentation . . . . .	402
13.2.2.1 Libedid_consts . . . . .	402
13.2.3 Function Documentation . . . . .	402
13.2.3.1 libedid_check_header() . . . . .	402
13.2.3.2 libedid_checksum() . . . . .	403
13.2.3.3 libedid_dump() . . . . .	403
13.2.3.4 libedid_dump_standard_timings() . . . . .	403
13.2.3.5 libedid_num_ext_blocks() . . . . .	404
13.2.3.6 libedid_pnp_id() . . . . .	404
13.2.3.7 libedid_prefered_resolution() . . . . .	404
13.2.3.8 libedid_revision() . . . . .	405
13.2.3.9 libedid_version() . . . . .	405
13.3 IO interface . . . . .	405
13.3.1 Detailed Description . . . . .	406
13.3.2 Typedef Documentation . . . . .	406
13.3.2.1 l4io_resource_t . . . . .	406
13.3.3 Enumeration Type Documentation . . . . .	406
13.3.3.1 l4io_device_types_t . . . . .	406
13.3.3.2 l4io_iomem_flags_t . . . . .	407
13.3.3.3 l4io_resource_types_t . . . . .	407
13.3.4 Function Documentation . . . . .	407
13.3.4.1 l4io_has_resource() . . . . .	407
13.3.4.2 l4io_lookup_device() . . . . .	408
13.3.4.3 l4io_lookup_resource() . . . . .	408
13.3.4.4 l4io_release_iomem() . . . . .	409
13.3.4.5 l4io_release_ioport() . . . . .	409
13.3.4.6 l4io_request_iomem() . . . . .	409
13.3.4.7 l4io_request_iomem_region() . . . . .	410
13.3.4.8 l4io_request_ioport() . . . . .	411
13.3.4.9 l4io_request_resource_iomem() . . . . .	411
13.4 IPC Helpers . . . . .	412
13.4.1 Detailed Description . . . . .	412

13.4.2 Function Documentation	412
13.4.2.1 throw_ipc_exception() [1/2]	412
13.4.2.2 throw_ipc_exception() [2/2]	413
13.5 IRQ handling library	414
13.5.1 Detailed Description	414
13.5.2 Interface for asynchronous ISR handlers.	414
13.5.2.1 Detailed Description	415
13.5.2.2 Function Documentation	415
13.5.2.3 Interface for asynchronous ISR handlers with a given IRQ capability.	416
13.5.3 Interface using direct functionality.	417
13.5.3.1 Detailed Description	418
13.5.3.2 Function Documentation	418
13.5.3.3 Interface using direct functionality.	421
13.6 L4 IPC Opcodes	423
13.6.1 Detailed Description	424
13.6.2 Enumeration Type Documentation	424
13.6.2.1 L4_icu_opcode	424
13.6.2.2 L4_ipc_gate_ops	425
13.6.2.3 L4_platform_ctl_ops	425
13.6.2.4 L4_task_ops	426
13.6.2.5 L4_thread_ops	426
13.6.2.6 L4_vcon_ops	427
13.7 L4 VIRTIO Interface	427
13.7.1 Detailed Description	428
13.7.2 L4 VIRTIO Block Device	428
13.7.2.1 Detailed Description	428
13.7.2.2 Enumeration Type Documentation	428
13.7.3 L4 VIRTIO Input Device	429
13.7.3.1 Detailed Description	430
13.7.4 L4 VIRTIO Network Device	430
13.7.4.1 Detailed Description	431
13.7.5 L4 VIRTIO Transport Layer	431
13.7.5.1 Detailed Description	432
13.7.5.2 Typedef Documentation	433
13.7.5.3 Enumeration Type Documentation	433
13.7.5.4 Function Documentation	435
13.8 L4 Vbus functions	439
13.8.1 Detailed Description	440
13.8.2 Enumeration Type Documentation	440
13.8.2.1 L4vbus_dma_domain_assign_flags	440
13.8.3 Function Documentation	441
13.8.3.1 l4vbus_assign_dma_domain()	441

13.8.3.2 l4vbus_get_adr()	442
13.8.3.3 l4vbus_get_device()	442
13.8.3.4 l4vbus_get_device_by_hid()	443
13.8.3.5 l4vbus_get_hid()	444
13.8.3.6 l4vbus_get_next_device()	444
13.8.3.7 l4vbus_get_resource()	446
13.8.3.8 l4vbus_is_compatible()	447
13.8.3.9 l4vbus_release_ioport()	448
13.8.3.10 l4vbus_request_ioport()	448
13.8.3.11 l4vbus_vicu_get_cap()	449
13.8.4 L4vbus GPIO functions	450
13.8.4.1 Detailed Description	451
13.8.4.2 Enumeration Type Documentation	451
13.8.4.3 Function Documentation	451
13.8.5 L4vbus PCI functions	459
13.8.5.1 Detailed Description	460
13.8.5.2 Function Documentation	460
13.8.6 L4vbus power management functions	465
13.8.6.1 Detailed Description	465
13.8.6.2 Function Documentation	465
13.9 L4Re C Interface	467
13.9.1 Detailed Description	469
13.9.2 Capability allocator	469
13.9.2.1 Detailed Description	470
13.9.2.2 Function Documentation	470
13.9.3 DMA Space Interface	470
13.9.3.1 Detailed Description	471
13.9.3.2 Typedef Documentation	471
13.9.3.3 Function Documentation	471
13.9.4 Dataspace interface	474
13.9.4.1 Detailed Description	475
13.9.4.2 Enumeration Type Documentation	475
13.9.4.3 Function Documentation	475
13.9.5 Debug interface	479
13.9.5.1 Detailed Description	479
13.9.5.2 Function Documentation	479
13.9.6 Event interface	480
13.9.6.1 Detailed Description	480
13.9.6.2 Function Documentation	480
13.9.7 Initial Environment	483
13.9.7.1 Detailed Description	484
13.9.7.2 Function Documentation	484

13.9.8 Kumem allocator utility . . . . .	488
13.9.9 L4Re Util C Interface . . . . .	488
13.9.10 Log interface . . . . .	488
13.9.10.1 Detailed Description . . . . .	489
13.9.10.2 Function Documentation . . . . .	489
13.9.11 Memory allocator . . . . .	491
13.9.11.1 Detailed Description . . . . .	492
13.9.11.2 Enumeration Type Documentation . . . . .	492
13.9.11.3 Function Documentation . . . . .	492
13.9.12 Namespace interface . . . . .	495
13.9.12.1 Detailed Description . . . . .	496
13.9.12.2 Enumeration Type Documentation . . . . .	496
13.9.12.3 Function Documentation . . . . .	496
13.9.13 Parent interface . . . . .	499
13.9.14 Region map interface . . . . .	499
13.9.14.1 Detailed Description . . . . .	500
13.9.14.2 Enumeration Type Documentation . . . . .	500
13.9.14.3 Function Documentation . . . . .	501
13.9.15 Video API . . . . .	513
13.9.15.1 Detailed Description . . . . .	514
13.9.15.2 Typedef Documentation . . . . .	514
13.9.15.3 Enumeration Type Documentation . . . . .	514
13.9.15.4 Function Documentation . . . . .	515
13.10 L4Re C++ Interface . . . . .	520
13.10.1 Detailed Description . . . . .	522
13.10.2 Auxiliary data . . . . .	522
13.10.2.1 Detailed Description . . . . .	523
13.10.3 C++ Exceptions . . . . .	523
13.10.3.1 Detailed Description . . . . .	524
13.10.4 Console API . . . . .	524
13.10.4.1 Detailed Description . . . . .	524
13.10.5 Debugging API . . . . .	524
13.10.5.1 Detailed Description . . . . .	525
13.10.6 Event API . . . . .	525
13.10.6.1 Detailed Description . . . . .	525
13.10.7 L4Re ELF Auxiliary Information . . . . .	526
13.10.7.1 Detailed Description . . . . .	527
13.10.7.2 Macro Definition Documentation . . . . .	527
13.10.7.3 Enumeration Type Documentation . . . . .	527
13.10.8 L4Re Protocol identifiers . . . . .	528
13.10.8.1 Detailed Description . . . . .	529
13.10.9 L4Re Util C++ Interface . . . . .	529

13.10.9.1 Detailed Description . . . . .	529
13.10.9.2 Kumem utilities . . . . .	530
13.10.9.3 L4Re Capability API . . . . .	531
13.10.10 Logging interface . . . . .	533
13.10.10.1 Detailed Description . . . . .	534
13.10.11 Name-space API . . . . .	534
13.10.11.1 Detailed Description . . . . .	534
13.10.12 Parent API . . . . .	535
13.10.12.1 Detailed Description . . . . .	535
13.10.13 Region map API . . . . .	535
13.10.13.1 Detailed Description . . . . .	536
13.10.14 Vbus API . . . . .	536
13.10.14.1 Detailed Description . . . . .	537
13.11 L4SHM-based ring buffer implementation . . . . .	538
13.11.1 Detailed Description . . . . .	538
13.11.2 Internal . . . . .	538
13.11.2.1 Detailed Description . . . . .	539
13.11.2.2 Macro Definition Documentation . . . . .	539
13.11.3 Receiver . . . . .	540
13.11.4 Sender . . . . .	540
13.12 Server-Side IPC framework . . . . .	540
13.12.1 Detailed Description . . . . .	541
13.12.2 Enumeration Type Documentation . . . . .	542
13.12.2.1 Reply_mode . . . . .	542
13.13 Shared Memory Library . . . . .	542
13.13.1 Detailed Description . . . . .	543
13.13.2 Function Documentation . . . . .	544
13.13.2.1 l4shmc_area_overhead() . . . . .	544
13.13.2.2 l4shmc_area_size() . . . . .	544
13.13.2.3 l4shmc_area_size_free() . . . . .	544
13.13.2.4 l4shmc_attach() . . . . .	545
13.13.2.5 l4shmc_chunk_overhead() . . . . .	545
13.13.2.6 l4shmc_connect_chunk_signal() . . . . .	545
13.13.2.7 l4shmc_create() . . . . .	546
13.13.2.8 l4shmc_get_client_nr() . . . . .	546
13.13.2.9 l4shmc_get_initialized_clients() . . . . .	547
13.13.2.10 l4shmc_mark_client_initialized() . . . . .	547
13.13.3 Chunks . . . . .	548
13.13.3.1 Detailed Description . . . . .	548
13.13.3.2 Function Documentation . . . . .	548
13.13.3.3 Consumer . . . . .	551
13.13.3.4 Producer . . . . .	555

13.13.4 Signals	558
13.13.4.1 Detailed Description	559
13.13.4.2 Function Documentation	559
13.13.4.3 Consumer	562
13.13.4.4 Producer	566
13.14 Sigma0 API	566
13.14.1 Detailed Description	567
13.14.2 Enumeration Type Documentation	568
13.14.2.1 l4sigma0_return_flags_t	568
13.14.3 Function Documentation	568
13.14.3.1 l4sigma0_debug_dump()	568
13.14.3.2 l4sigma0_map_anypage()	568
13.14.3.3 l4sigma0_map_errstr()	569
13.14.3.4 l4sigma0_map_iomem()	569
13.14.3.5 l4sigma0_map_kip()	570
13.14.3.6 l4sigma0_map_mem()	570
13.14.4 Internal constants	571
13.14.4.1 Detailed Description	572
13.15 Small C++ Template Library	572
13.15.1 Detailed Description	574
13.15.2 Function Documentation	574
13.15.2.1 clamp()	574
13.15.2.2 max() [1/2]	574
13.15.2.3 max() [2/2]	574
13.15.2.4 min() [1/2]	575
13.15.2.5 min() [2/2]	575
13.15.2.6 operator new()	576
13.16 Utility Functions	576
13.16.1 Detailed Description	578
13.16.2 Function Documentation	578
13.16.2.1 l4_sleep()	578
13.16.2.2 l4_touch_ro()	579
13.16.2.3 l4_touch_rw()	580
13.16.2.4 l4_usleep()	580
13.16.2.5 l4util_micros2l4to()	581
13.16.2.6 l4util_splitlog2_hdl()	581
13.16.2.7 l4util_splitlog2_size()	582
13.16.3 Atomic Instructions	583
13.16.3.1 Detailed Description	584
13.16.3.2 Function Documentation	584
13.16.4 Bit Manipulation	600
13.16.4.1 Detailed Description	601



13.16.4.2 Function Documentation . . . . .	601
13.16.5 Bitmap graphics and fonts . . . . .	607
13.16.5.1 Detailed Description . . . . .	608
13.16.5.2 Functions for rendering bitmap data in frame buffers . . . . .	608
13.16.5.3 Functions for rendering bitmap fonts to frame buffers . . . . .	608
13.16.6 CPU related functions . . . . .	608
13.16.6.1 Detailed Description . . . . .	609
13.16.6.2 Function Documentation . . . . .	609
13.16.7 Comfortable Command Line Parsing . . . . .	611
13.16.7.1 Detailed Description . . . . .	612
13.16.7.2 Function Documentation . . . . .	612
13.16.8 ELF binary format . . . . .	613
13.16.8.1 Detailed Description . . . . .	619
13.16.8.2 Macro Definition Documentation . . . . .	619
13.16.8.3 Enumeration Type Documentation . . . . .	620
13.16.9 Functions to manipulate the local IDT . . . . .	636
13.16.9.1 Detailed Description . . . . .	637
13.16.10 IA32 Port I/O API . . . . .	637
13.16.10.1 Detailed Description . . . . .	637
13.16.10.2 Function Documentation . . . . .	637
13.16.11 Internal functions . . . . .	643
13.16.11.1 Detailed Description . . . . .	644
13.16.12 Kernel Interface Page API . . . . .	644
13.16.12.1 Detailed Description . . . . .	644
13.16.12.2 Macro Definition Documentation . . . . .	644
13.16.12.3 Function Documentation . . . . .	645
13.16.13 Low-Level Thread Functions . . . . .	646
13.16.14 Random number support . . . . .	646
13.16.14.1 Detailed Description . . . . .	647
13.16.14.2 Function Documentation . . . . .	647
13.16.15 Timestamp Counter . . . . .	647
13.16.15.1 Detailed Description . . . . .	648
13.16.15.2 Function Documentation . . . . .	648
13.17 vCPU Support Library . . . . .	654
13.17.1 Detailed Description . . . . .	655
13.17.2 Function Documentation . . . . .	655
13.17.2.1 l4vcpu_irq_disable() . . . . .	655
13.17.2.2 l4vcpu_irq_disable_save() . . . . .	656
13.17.2.3 l4vcpu_irq_enable() . . . . .	657
13.17.2.4 l4vcpu_irq_restore() . . . . .	658
13.17.2.5 l4vcpu_is_irq_entry() . . . . .	659
13.17.2.6 l4vcpu_is_page_fault_entry() . . . . .	660

13.17.2.7 l4vcpu_print_state()	660
13.17.2.8 l4vcpu_wait_for_event()	661
13.17.3 Extended vCPU support	662
13.17.3.1 Detailed Description	662
13.17.3.2 Function Documentation	662
<b>14 Namespace Documentation</b>	<b>663</b>
14.1 cxx Namespace Reference	663
14.1.1 Detailed Description	665
14.1.2 Function Documentation	665
14.1.2.1 access_once()	665
14.1.2.2 write_now()	666
14.2 cxx::Bits Namespace Reference	667
14.2.1 Detailed Description	667
14.3 L4 Namespace Reference	667
14.3.1 Detailed Description	671
14.3.2 Enumeration Type Documentation	671
14.3.2.1 anonymous enum	671
14.3.3 Function Documentation	672
14.3.3.1 cap_cast() [1/2]	672
14.3.3.2 cap_cast() [2/2]	672
14.3.3.3 cap_dynamic_cast()	673
14.3.3.4 cap_reinterpret_cast() [1/2]	674
14.3.3.5 cap_reinterpret_cast() [2/2]	674
14.3.3.6 round_order()	675
14.3.3.7 trunc_order()	676
14.4 L4::lpc Namespace Reference	677
14.4.1 Detailed Description	679
14.4.2 Function Documentation	679
14.4.2.1 buf_cp_in()	679
14.4.2.2 buf_cp_out()	679
14.4.2.3 buf_in()	680
14.4.2.4 make_cap()	680
14.4.2.5 make_cap_full()	681
14.4.2.6 make_cap_rw()	682
14.4.2.7 make_cap_rws()	683
14.4.2.8 msg_ptr()	684
14.4.2.9 read()	684
14.4.2.10 str_cp_in()	684
14.5 L4::lpc::Msg Namespace Reference	685
14.5.1 Detailed Description	686
14.5.2 Enumeration Type Documentation	686

14.5.2.1 anonymous enum . . . . .	686
14.5.3 Function Documentation . . . . .	687
14.5.3.1 align_to() [1/2] . . . . .	687
14.5.3.2 align_to() [2/2] . . . . .	688
14.5.3.3 check_size() [1/2] . . . . .	688
14.5.3.4 check_size() [2/2] . . . . .	689
14.5.3.5 msg_add() . . . . .	689
14.5.3.6 msg_get() . . . . .	690
14.6 L4::lpc_svr Namespace Reference . . . . .	690
14.6.1 Detailed Description . . . . .	691
14.7 L4::Typeid Namespace Reference . . . . .	691
14.7.1 Detailed Description . . . . .	692
14.8 L4::Types Namespace Reference . . . . .	692
14.8.1 Detailed Description . . . . .	693
14.9 L4Re Namespace Reference . . . . .	693
14.9.1 Detailed Description . . . . .	695
14.9.2 Typedef Documentation . . . . .	695
14.9.2.1 Shared_cap . . . . .	695
14.9.2.2 shared_cap . . . . .	695
14.9.2.3 Shared_del_cap . . . . .	696
14.9.2.4 shared_del_cap . . . . .	696
14.9.2.5 Unique_cap . . . . .	697
14.9.2.6 unique_cap . . . . .	697
14.9.2.7 Unique_del_cap . . . . .	698
14.9.2.8 unique_del_cap . . . . .	698
14.9.3 Function Documentation . . . . .	698
14.9.3.1 chkcap() . . . . .	698
14.9.3.2 chkipc() . . . . .	700
14.9.3.3 chksys() [1/3] . . . . .	700
14.9.3.4 chksys() [2/3] . . . . .	701
14.9.3.5 chksys() [3/3] . . . . .	702
14.9.3.6 make_shared_cap() . . . . .	704
14.9.3.7 make_shared_del_cap() . . . . .	705
14.9.3.8 make_unique_cap() . . . . .	706
14.9.3.9 make_unique_del_cap() . . . . .	707
14.9.3.10 throw_error() . . . . .	707
14.10 L4Re::Util Namespace Reference . . . . .	709
14.10.1 Detailed Description . . . . .	711
14.10.2 Typedef Documentation . . . . .	711
14.10.2.1 Shared_cap . . . . .	711
14.10.2.2 shared_cap . . . . .	711
14.10.2.3 Shared_del_cap . . . . .	712

14.10.2.4 shared_del_cap . . . . .	713
14.10.2.5 Unique_cap . . . . .	713
14.10.2.6 unique_cap . . . . .	714
14.10.2.7 Unique_del_cap . . . . .	715
14.10.2.8 unique_del_cap . . . . .	715
14.10.3 Function Documentation . . . . .	716
14.10.3.1 make_shared_cap() . . . . .	716
14.10.3.2 make_shared_del_cap() . . . . .	716
14.10.3.3 make_unique_cap() . . . . .	716
14.10.3.4 make_unique_del_cap() . . . . .	718
14.11 L4Re::Vfs Namespace Reference . . . . .	718
14.11.1 Detailed Description . . . . .	719
14.12 L4vbus Namespace Reference . . . . .	719
14.12.1 Detailed Description . . . . .	719
14.13 L4virtio Namespace Reference . . . . .	720
14.13.1 Detailed Description . . . . .	720
<b>15 Data Structure Documentation</b> . . . . .	<b>721</b>
15.1 Block_device::Device_discard_feature Struct Reference . . . . .	721
15.1.1 Detailed Description . . . . .	721
15.2 Block_device::Device_mgr< DEV, FACTORY, SCHEDULER > Class Template Reference . . . . .	722
15.2.1 Detailed Description . . . . .	722
15.3 Block_device::Device_with_notification_domain< DEV > Struct Template Reference . . . . .	723
15.3.1 Detailed Description . . . . .	723
15.4 Block_device::Dma_region_info Struct Reference . . . . .	724
15.4.1 Detailed Description . . . . .	724
15.5 Block_device::Errand::Errand Class Reference . . . . .	724
15.5.1 Detailed Description . . . . .	727
15.5.2 Member Function Documentation . . . . .	727
15.5.2.1 expired() . . . . .	727
15.6 Block_device::Errand::Poll_errand Class Reference . . . . .	728
15.6.1 Detailed Description . . . . .	730
15.6.2 Member Function Documentation . . . . .	730
15.6.2.1 expired() . . . . .	730
15.7 Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, bool > Class Template Reference . . . . .	731
15.7.1 Detailed Description . . . . .	732
15.8 Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, true > Class Template Reference . . . . .	732
15.8.1 Detailed Description . . . . .	732
15.9 Block_device::Inout_block Struct Reference . . . . .	733
15.9.1 Detailed Description . . . . .	733
15.10 Block_device::Inout_memory< DEV > Class Template Reference . . . . .	734

15.10.1 Detailed Description . . . . .	734
15.11 Block_device::Mem_region_info Struct Reference . . . . .	735
15.11.1 Detailed Description . . . . .	735
15.12 Block_device::Notification_domain Struct Reference . . . . .	735
15.12.1 Detailed Description . . . . .	736
15.13 Block_device::Partition_info Struct Reference . . . . .	736
15.13.1 Detailed Description . . . . .	737
15.14 Block_device::Partition_reader< DEV > Class Template Reference . . . . .	737
15.14.1 Detailed Description . . . . .	737
15.15 Block_device::Partitioned_device< BASE_DEV > Class Template Reference . . . . .	738
15.15.1 Detailed Description . . . . .	739
15.16 Block_device::Pending_request Struct Reference . . . . .	739
15.16.1 Detailed Description . . . . .	740
15.16.2 Member Function Documentation . . . . .	740
15.16.2.1 fail_request() . . . . .	740
15.16.2.2 handle_request() . . . . .	740
15.17 Block_device::Rr_scheduler< DEV > Struct Template Reference . . . . .	740
15.17.1 Detailed Description . . . . .	742
15.18 Block_device::Scheduler_base< DEV > Class Template Reference . . . . .	742
15.18.1 Detailed Description . . . . .	743
15.19 cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC > Class Template Reference . . . . .	744
15.19.1 Detailed Description . . . . .	747
15.19.2 Constructor & Destructor Documentation . . . . .	747
15.19.2.1 Avl_map() . . . . .	747
15.19.3 Member Function Documentation . . . . .	748
15.19.3.1 insert() . . . . .	748
15.19.3.2 operator[]() [1/2] . . . . .	749
15.19.3.3 operator[]() [2/2] . . . . .	749
15.20 cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC > Class Template Reference . . . . .	750
15.20.1 Detailed Description . . . . .	754
15.21 cxx::Avl_tree< Node, Get_key, Compare > Class Template Reference . . . . .	754
15.21.1 Detailed Description . . . . .	758
15.21.2 Member Typedef Documentation . . . . .	759
15.21.2.1 Iterator . . . . .	759
15.21.3 Member Function Documentation . . . . .	759
15.21.3.1 insert() . . . . .	759
15.21.3.2 remove() . . . . .	760
15.22 cxx::Avl_tree_node Class Reference . . . . .	761
15.22.1 Detailed Description . . . . .	763
15.23 cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference . . . . .	763
15.23.1 Detailed Description . . . . .	764
15.23.2 Member Enumeration Documentation . . . . .	765

15.23.2.1 anonymous enum . . . . .	765
15.23.3 Member Function Documentation . . . . .	765
15.23.3.1 alloc() . . . . .	765
15.23.3.2 free() . . . . .	766
15.23.3.3 free_objects() . . . . .	767
15.23.3.4 total_objects() . . . . .	768
15.24 cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i Struct Reference . . . . .	769
15.24.1 Detailed Description . . . . .	769
15.25 cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference . . . . .	769
15.25.1 Detailed Description . . . . .	771
15.25.2 Member Enumeration Documentation . . . . .	771
15.25.2.1 anonymous enum . . . . .	771
15.25.3 Member Function Documentation . . . . .	772
15.25.3.1 alloc() . . . . .	772
15.25.3.2 free() . . . . .	772
15.25.3.3 free_objects() . . . . .	773
15.25.3.4 total_objects() . . . . .	774
15.26 cxx::Bitfield< T, LSB, MSB > Class Template Reference . . . . .	774
15.26.1 Detailed Description . . . . .	776
15.26.2 Member Typedef Documentation . . . . .	776
15.26.2.1 Bits_type . . . . .	776
15.26.2.2 Shift_type . . . . .	777
15.26.3 Member Enumeration Documentation . . . . .	777
15.26.3.1 anonymous enum . . . . .	777
15.26.3.2 Masks . . . . .	777
15.26.4 Member Function Documentation . . . . .	777
15.26.4.1 get() . . . . .	777
15.26.4.2 get_unshifted() . . . . .	778
15.26.4.3 set() . . . . .	779
15.26.4.4 set_dirty() . . . . .	779
15.26.4.5 set_unshifted() . . . . .	780
15.26.4.6 set_unshifted_dirty() . . . . .	781
15.26.4.7 val() . . . . .	782
15.26.4.8 val_dirty() . . . . .	783
15.26.4.9 val_unshifted() . . . . .	786
15.27 cxx::Bitfield< T, LSB, MSB >::Value< TT > Class Template Reference . . . . .	786
15.27.1 Detailed Description . . . . .	787
15.28 cxx::Bitfield< T, LSB, MSB >::Value_base< TT > Class Template Reference . . . . .	788
15.28.1 Detailed Description . . . . .	788
15.29 cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT > Class Template Reference . . . . .	789
15.29.1 Detailed Description . . . . .	790
15.30 cxx::Bitmap< BITS > Class Template Reference . . . . .	790

15.30.1 Detailed Description	793
15.30.2 Member Function Documentation	794
15.30.2.1 scan_zero()	794
15.31 cxx::Bitmap_base Class Reference	795
15.31.1 Detailed Description	797
15.31.2 Member Enumeration Documentation	797
15.31.2.1 anonymous enum	797
15.31.3 Member Function Documentation	798
15.31.3.1 atomic_clear_bit()	798
15.31.3.2 atomic_get_and_clear()	798
15.31.3.3 atomic_get_and_set()	798
15.31.3.4 atomic_set_bit()	799
15.31.3.5 bit() [1/2]	799
15.31.3.6 bit() [2/2]	799
15.31.3.7 bit_index()	800
15.31.3.8 clear_bit()	800
15.31.3.9 operator[]() [1/2]	801
15.31.3.10 operator[]() [2/2]	801
15.31.3.11 scan_zero()	802
15.31.3.12 set_bit()	803
15.31.3.13 word_index()	803
15.32 cxx::Bitmap_base::Bit Class Reference	803
15.32.1 Detailed Description	804
15.33 cxx::Bitmap_base::Char< BITS > Class Template Reference	804
15.33.1 Detailed Description	804
15.34 cxx::Bitmap_base::Word< BITS > Class Template Reference	805
15.34.1 Detailed Description	805
15.35 cxx::Bits::Avl_map_get_key< KEY_TYPE > Struct Template Reference	806
15.35.1 Detailed Description	806
15.36 cxx::Bits::Avl_set_get_key< KEY_TYPE > Struct Template Reference	806
15.36.1 Detailed Description	807
15.37 cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > Class Template Reference	807
15.37.1 Detailed Description	809
15.37.2 Member Enumeration Documentation	810
15.37.2.1 anonymous enum	810
15.37.3 Constructor & Destructor Documentation	810
15.37.3.1 Base_avl_set() [1/2]	810
15.37.3.2 Base_avl_set() [2/2]	811
15.37.4 Member Function Documentation	811
15.37.4.1 begin() [1/2]	811
15.37.4.2 begin() [2/2]	812
15.37.4.3 end() [1/2]	812

15.37.4.4 end() [2/2]	813
15.37.4.5 erase()	814
15.37.4.6 find_node()	814
15.37.4.7 insert()	815
15.37.4.8 lower_bound_node()	816
15.37.4.9 rbegin() [1/2]	817
15.37.4.10 rbegin() [2/2]	818
15.37.4.11 remove()	818
15.37.4.12 rend() [1/2]	819
15.37.4.13 rend() [2/2]	820
15.38 cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node Class Reference	820
15.38.1 Detailed Description	821
15.38.2 Member Function Documentation	822
15.38.2.1 operator*()	822
15.38.2.2 operator->()	822
15.38.2.3 valid()	822
15.39 cxx::Bits::Basic_list< POLICY > Class Template Reference	823
15.39.1 Detailed Description	824
15.39.2 Member Function Documentation	824
15.39.2.1 clear()	824
15.39.2.2 iter()	824
15.40 cxx::Bits::Bst< Node, Get_key, Compare > Class Template Reference	825
15.40.1 Detailed Description	828
15.40.2 Member Function Documentation	828
15.40.2.1 begin() [1/2]	828
15.40.2.2 begin() [2/2]	829
15.40.2.3 dir() [1/2]	830
15.40.2.4 dir() [2/2]	830
15.40.2.5 end() [1/2]	831
15.40.2.6 end() [2/2]	831
15.40.2.7 find()	832
15.40.2.8 find_node()	833
15.40.2.9 lower_bound_node()	833
15.40.2.10 rbegin() [1/2]	834
15.40.2.11 rbegin() [2/2]	835
15.40.2.12 remove_all()	836
15.40.2.13 remove_tree()	837
15.40.2.14 rend() [1/2]	838
15.40.2.15 rend() [2/2]	838
15.41 cxx::Bits::Bst_node Class Reference	839
15.41.1 Detailed Description	841
15.42 cxx::Bits::Direction Struct Reference	841



15.42.1 Detailed Description	842
15.42.2 Member Enumeration Documentation	842
15.42.2.1 Direction_e	842
15.42.3 Member Function Documentation	842
15.42.3.1 operator"()()	842
15.43 cxx::Bits::Smart_ptr_list< ITEM > Class Template Reference	843
15.43.1 Detailed Description	845
15.43.2 Member Function Documentation	846
15.43.2.1 pop_front()	846
15.44 cxx::Bits::Smart_ptr_list_item< T, STORE_T > Class Template Reference	846
15.44.1 Detailed Description	847
15.45 cxx::H_list< T, POLICY > Class Template Reference	848
15.45.1 Detailed Description	851
15.45.2 Member Function Documentation	851
15.45.2.1 erase()	851
15.45.2.2 insert()	852
15.45.2.3 insert_after()	852
15.45.2.4 insert_before()	853
15.45.2.5 iter()	854
15.45.2.6 pop_front()	854
15.45.2.7 remove()	855
15.45.2.8 replace()	856
15.46 cxx::H_list_item_t< ELEM_TYPE > Class Template Reference	856
15.46.1 Detailed Description	858
15.46.2 Constructor & Destructor Documentation	858
15.46.2.1 H_list_item_t()	858
15.46.2.2 ~H_list_item_t()	858
15.47 cxx::H_list_t< T > Struct Template Reference	859
15.47.1 Detailed Description	862
15.48 cxx::List< D, Alloc > Class Template Reference	862
15.48.1 Detailed Description	863
15.48.2 Member Function Documentation	863
15.48.2.1 operator[]() [1/2]	863
15.48.2.2 operator[]() [2/2]	864
15.49 cxx::List< D, Alloc >::Iter Class Reference	864
15.49.1 Detailed Description	864
15.50 cxx::List_alloc Class Reference	865
15.50.1 Detailed Description	865
15.50.2 Constructor & Destructor Documentation	865
15.50.2.1 List_alloc()	865
15.50.3 Member Function Documentation	866
15.50.3.1 alloc()	866

15.50.3.2 alloc_max()	866
15.50.3.3 avail()	867
15.50.3.4 free()	867
15.51 cxx::List_item Class Reference	868
15.51.1 Detailed Description	869
15.51.2 Member Function Documentation	869
15.51.2.1 push_back()	869
15.51.2.2 push_front()	870
15.51.2.3 remove()	871
15.52 cxx::List_item::Iter Class Reference	872
15.52.1 Detailed Description	873
15.53 cxx::List_item::T_iter< T, Poly > Class Template Reference	873
15.53.1 Detailed Description	874
15.54 cxx::Lt_functor< Obj > Struct Template Reference	875
15.54.1 Detailed Description	875
15.55 cxx::New_allocator< _Type > Class Template Reference	875
15.55.1 Detailed Description	876
15.56 cxx::Nothrow Class Reference	876
15.56.1 Detailed Description	877
15.57 cxx::Pair< First, Second > Struct Template Reference	877
15.57.1 Detailed Description	878
15.57.2 Constructor & Destructor Documentation	878
15.57.2.1 Pair() [1/2]	878
15.57.2.2 Pair() [2/2]	878
15.58 cxx::Pair_first_compare< Cmp, Typ > Class Template Reference	880
15.58.1 Detailed Description	880
15.58.2 Constructor & Destructor Documentation	881
15.58.2.1 Pair_first_compare()	881
15.58.3 Member Function Documentation	881
15.58.3.1 operator()()	881
15.59 cxx::Ref_obj_list_item< T > Struct Template Reference	882
15.59.1 Detailed Description	883
15.60 cxx::Ref_ptr< T, CNT > Class Template Reference	883
15.60.1 Detailed Description	884
15.60.2 Constructor & Destructor Documentation	885
15.60.2.1 Ref_ptr() [1/3]	885
15.60.2.2 Ref_ptr() [2/3]	885
15.60.2.3 Ref_ptr() [3/3]	885
15.60.3 Member Function Documentation	886
15.60.3.1 get()	886
15.60.3.2 ptr()	886
15.60.3.3 release()	886

15.61 <code>cxx::S_list&lt; T, POLICY &gt;</code> Class Template Reference	887
15.61.1 Detailed Description	889
15.61.2 Member Function Documentation	889
15.61.2.1 <code>pop_front()</code>	889
15.62 <code>cxx::Slab&lt; Type, Slab_size, Max_free, Alloc &gt;</code> Class Template Reference	890
15.62.1 Detailed Description	892
15.62.2 Member Function Documentation	893
15.62.2.1 <code>alloc()</code>	893
15.62.2.2 <code>free()</code>	893
15.63 <code>cxx::Slab_static&lt; Type, Slab_size, Max_free, Alloc &gt;</code> Class Template Reference	894
15.63.1 Detailed Description	896
15.63.2 Member Function Documentation	897
15.63.2.1 <code>alloc()</code>	897
15.64 <code>cxx::static_vector&lt; T, IDX &gt;</code> Class Template Reference	898
15.64.1 Detailed Description	898
15.65 <code>cxx::String</code> Class Reference	899
15.65.1 Detailed Description	901
15.65.2 Constructor & Destructor Documentation	902
15.65.2.1 <code>String()</code>	902
15.65.3 Member Function Documentation	902
15.65.3.1 <code>find()</code>	902
15.65.3.2 <code>from_dec()</code>	903
15.65.3.3 <code>from_hex()</code>	904
15.65.3.4 <code>starts_with()</code>	904
15.66 <code>cxx::Weak_ref&lt; T &gt;</code> Class Template Reference	905
15.66.1 Detailed Description	907
15.67 <code>cxx::Weak_ref_base</code> Class Reference	908
15.67.1 Detailed Description	911
15.68 <code>cxx::Weak_ref_base::List</code> Struct Reference	911
15.68.1 Detailed Description	915
15.69 <code>Elf32_Auxv</code> Struct Reference	915
15.69.1 Detailed Description	915
15.69.2 Field Documentation	916
15.69.2.1 <code>atype</code>	916
15.70 <code>Elf32_Dyn</code> Struct Reference	916
15.70.1 Detailed Description	916
15.70.2 Field Documentation	917
15.70.2.1 <code>d_tag</code>	917
15.71 <code>Elf32_Ehdr</code> Struct Reference	917
15.71.1 Detailed Description	918
15.71.2 Field Documentation	918
15.71.2.1 <code>e_flags</code>	918

15.71.2.2 e_machine	919
15.71.2.3 e_type	919
15.71.2.4 e_version	919
15.72 Elf32_Phdr Struct Reference	920
15.72.1 Detailed Description	920
15.72.2 Field Documentation	921
15.72.2.1 p_flags	921
15.72.2.2 p_type	921
15.73 Elf32_Rel Struct Reference	921
15.73.1 Detailed Description	922
15.74 Elf32_Rela Struct Reference	922
15.74.1 Detailed Description	922
15.75 Elf32_Shdr Struct Reference	923
15.75.1 Detailed Description	924
15.75.2 Field Documentation	924
15.75.2.1 sh_flags	924
15.75.2.2 sh_type	924
15.76 Elf32_Sym Struct Reference	924
15.76.1 Detailed Description	925
15.77 Elf64_Auxv Struct Reference	925
15.77.1 Detailed Description	926
15.77.2 Field Documentation	926
15.77.2.1 atype	926
15.78 Elf64_Dyn Struct Reference	926
15.78.1 Detailed Description	927
15.78.2 Field Documentation	927
15.78.2.1 d_tag	927
15.79 Elf64_Ehdr Struct Reference	927
15.79.1 Detailed Description	928
15.79.2 Field Documentation	928
15.79.2.1 e_flags	928
15.79.2.2 e_machine	929
15.79.2.3 e_type	929
15.79.2.4 e_version	929
15.80 Elf64_Phdr Struct Reference	930
15.80.1 Detailed Description	930
15.80.2 Field Documentation	931
15.80.2.1 p_flags	931
15.80.2.2 p_type	931
15.81 Elf64_Rel Struct Reference	931
15.81.1 Detailed Description	932
15.82 Elf64_Rela Struct Reference	932

15.82.1 Detailed Description	932
15.83 Elf64_Shdr Struct Reference	933
15.83.1 Detailed Description	934
15.83.2 Field Documentation	934
15.83.2.1 sh_flags	934
15.83.2.2 sh_type	934
15.84 Elf64_Sym Struct Reference	934
15.84.1 Detailed Description	935
15.85 gfxbitmap_offset Struct Reference	935
15.85.1 Detailed Description	936
15.86 L4::Alloc_list Class Reference	936
15.86.1 Detailed Description	936
15.87 L4::Arm_smccc Class Reference	937
15.87.1 Detailed Description	937
15.87.2 Member Function Documentation	937
15.87.2.1 call()	937
15.88 L4::Base_exception Class Reference	938
15.88.1 Detailed Description	940
15.89 L4::Basic_registry Class Reference	941
15.89.1 Detailed Description	942
15.89.2 Member Function Documentation	942
15.89.2.1 dispatch()	942
15.89.2.2 find()	943
15.90 L4::Bounds_error Class Reference	944
15.90.1 Detailed Description	946
15.91 L4::Cap< T > Class Template Reference	946
15.91.1 Detailed Description	949
15.91.2 Constructor & Destructor Documentation	950
15.91.2.1 Cap() [1/4]	950
15.91.2.2 Cap() [2/4]	950
15.91.2.3 Cap() [3/4]	950
15.91.2.4 Cap() [4/4]	951
15.91.3 Member Function Documentation	951
15.91.3.1 check_castable_from()	951
15.91.3.2 check_convertible_from()	951
15.91.3.3 copy()	952
15.91.3.4 move()	952
15.92 L4::Cap_base Class Reference	952
15.92.1 Detailed Description	954
15.92.2 Member Enumeration Documentation	955
15.92.2.1 Cap_type	955
15.92.2.2 No_init_type	955

15.92.3 Constructor & Destructor Documentation	955
15.92.3.1 Cap_base() [1/2]	955
15.92.3.2 Cap_base() [2/2]	955
15.92.4 Member Function Documentation	956
15.92.4.1 cap()	956
15.92.4.2 copy()	957
15.92.4.3 fpage()	958
15.92.4.4 is_valid()	959
15.92.4.5 move()	960
15.92.4.6 snd_base()	961
15.92.4.7 validate() [1/2]	962
15.92.4.8 validate() [2/2]	962
15.93 L4::Com_error Class Reference	963
15.93.1 Detailed Description	966
15.93.2 Constructor & Destructor Documentation	966
15.93.2.1 Com_error()	966
15.94 L4::Debugger Class Reference	967
15.94.1 Detailed Description	970
15.94.2 Member Function Documentation	970
15.94.2.1 add_image_info()	970
15.94.2.2 get_object_name()	971
15.94.2.3 global_id()	971
15.94.2.4 kobj_to_id()	972
15.94.2.5 query_log_name()	973
15.94.2.6 query_log_typeid()	974
15.94.2.7 set_object_name()	975
15.94.2.8 switch_log()	975
15.95 L4::Element_already_exists Class Reference	976
15.95.1 Detailed Description	979
15.96 L4::Element_not_found Class Reference	979
15.96.1 Detailed Description	982
15.97 L4::Epiface Struct Reference	982
15.97.1 Detailed Description	984
15.97.2 Member Function Documentation	984
15.97.2.1 dispatch()	984
15.97.2.2 get_buffer_demand()	985
15.97.2.3 obj_cap()	985
15.97.2.4 server_iface()	986
15.97.2.5 set_server()	986
15.98 L4::Epiface_t< Derived, IFACE, BASE, bool > Struct Template Reference	987
15.98.1 Detailed Description	989
15.98.2 Member Function Documentation	990

15.98.2.1 dispatch()	990
15.99 L4::Epiface_t0< RPC_IFACE, BASE > Struct Template Reference	990
15.99.1 Detailed Description	992
15.99.2 Member Function Documentation	992
15.99.2.1 obj_cap()	992
15.100 L4::Exception Class Reference	993
15.100.1 Detailed Description	993
15.100.2 Member Function Documentation	993
15.100.2.1 exception()	993
15.101 L4::Exception_tracer Class Reference	994
15.101.1 Detailed Description	995
15.102 L4::Factory Class Reference	996
15.102.1 Detailed Description	999
15.102.2 Member Function Documentation	999
15.102.2.1 create() [1/2]	999
15.102.2.2 create() [2/2]	1000
15.102.2.3 create_factory()	1001
15.102.2.4 create_gate()	1002
15.102.2.5 create_task()	1003
15.103 L4::Factory::Lstr Struct Reference	1005
15.103.1 Detailed Description	1005
15.103.2 Constructor & Destructor Documentation	1005
15.103.2.1 Lstr()	1005
15.104 L4::Factory::Nil Struct Reference	1006
15.104.1 Detailed Description	1006
15.105 L4::Factory::S Class Reference	1007
15.105.1 Detailed Description	1008
15.105.2 Constructor & Destructor Documentation	1008
15.105.2.1 S() [1/2]	1008
15.105.2.2 S() [2/2]	1008
15.105.3 Member Function Documentation	1009
15.105.3.1 operator l4_msgtag_t()	1009
15.105.3.2 put() [1/5]	1009
15.105.3.3 put() [2/5]	1009
15.105.3.4 put() [3/5]	1010
15.105.3.5 put() [4/5]	1010
15.105.3.6 put() [5/5]	1010
15.106 L4::lcu Class Reference	1011
15.106.1 Detailed Description	1014
15.106.2 Member Function Documentation	1014
15.106.2.1 bind()	1014
15.106.2.2 info()	1015

15.106.2.3 mask()	1016
15.106.2.4 msi_info()	1017
15.106.2.5 set_mode()	1018
15.106.2.6 unbind()	1018
15.107 L4::lcu::Info Class Reference	1019
15.107.1 Detailed Description	1020
15.108 L4::Invalid_capability Class Reference	1021
15.108.1 Detailed Description	1023
15.108.2 Constructor & Destructor Documentation	1023
15.108.2.1 Invalid_capability()	1023
15.108.3 Member Function Documentation	1023
15.108.3.1 cap()	1023
15.109 L4::io_pager Class Reference	1024
15.109.1 Detailed Description	1025
15.109.2 Member Function Documentation	1025
15.109.2.1 io_page_fault()	1025
15.110 L4::lomu Class Reference	1026
15.110.1 Detailed Description	1027
15.110.2 Member Function Documentation	1028
15.110.2.1 bind()	1028
15.110.2.2 unbind()	1028
15.111 L4::IOModifier Class Reference	1028
15.111.1 Detailed Description	1029
15.112 L4::lpc::Array< ELEM_TYPE, LEN_TYPE > Struct Template Reference	1029
15.112.1 Detailed Description	1031
15.113 L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX > Struct Template Reference	1032
15.113.1 Detailed Description	1033
15.114 L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE > Struct Template Reference	1033
15.114.1 Detailed Description	1034
15.115 L4::lpc::As_value< T > Struct Template Reference	1034
15.115.1 Detailed Description	1035
15.116 L4::lpc::Call Struct Reference	1035
15.116.1 Detailed Description	1036
15.117 L4::lpc::Call_t< RIGHTS > Struct Template Reference	1037
15.117.1 Detailed Description	1037
15.118 L4::lpc::Call_zero_send_timeout Struct Reference	1038
15.118.1 Detailed Description	1039
15.119 L4::lpc::Cap< T > Class Template Reference	1039
15.119.1 Detailed Description	1041
15.119.2 Member Enumeration Documentation	1041
15.119.2.1 anonymous enum	1041
15.119.3 Constructor & Destructor Documentation	1041



15.119.3.1 Cap()	1041
15.119.4 Member Function Documentation	1042
15.119.4.1 from_ci()	1042
15.120 L4::lpc::Gen_fpage Class Reference	1042
15.120.1 Detailed Description	1044
15.120.2 Member Function Documentation	1044
15.120.2.1 cap_received()	1044
15.120.2.2 id_received()	1045
15.120.2.3 is_compound()	1045
15.120.2.4 local_id_received()	1045
15.121 L4::lpc::In_out< T > Struct Template Reference	1045
15.121.1 Detailed Description	1046
15.122 L4::lpc::lostream Class Reference	1046
15.122.1 Detailed Description	1050
15.122.2 Constructor & Destructor Documentation	1050
15.122.2.1 lostream()	1050
15.122.3 Member Function Documentation	1051
15.122.3.1 call()	1051
15.122.3.2 get() [1/3]	1052
15.122.3.3 get() [2/3]	1052
15.122.3.4 get() [3/3]	1052
15.122.3.5 put() [1/2]	1053
15.122.3.6 put() [2/2]	1053
15.122.3.7 reply_and_wait() [1/2]	1054
15.122.3.8 reply_and_wait() [2/2]	1054
15.122.3.9 reset()	1056
15.123 L4::lpc::Istream Class Reference	1056
15.123.1 Detailed Description	1059
15.123.2 Constructor & Destructor Documentation	1059
15.123.2.1 Istream()	1059
15.123.3 Member Function Documentation	1060
15.123.3.1 get() [1/3]	1060
15.123.3.2 get() [2/3]	1060
15.123.3.3 get() [3/3]	1061
15.123.3.4 receive()	1062
15.123.3.5 reset()	1063
15.123.3.6 skip()	1064
15.123.3.7 tag() [1/2]	1064
15.123.3.8 tag() [2/2]	1064
15.123.3.9 wait() [1/2]	1065
15.123.3.10 wait() [2/2]	1066
15.124 L4::lpc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS > Struct Template Reference	1067

15.124.1 Detailed Description . . . . .	1068
15.125 L4::Ipc::Msg::Cls_buffer Struct Reference . . . . .	1068
15.125.1 Detailed Description . . . . .	1069
15.126 L4::Ipc::Msg::Cls_data Struct Reference . . . . .	1069
15.126.1 Detailed Description . . . . .	1070
15.127 L4::Ipc::Msg::Cls_item Struct Reference . . . . .	1070
15.127.1 Detailed Description . . . . .	1071
15.128 L4::Ipc::Msg::Dir_in Struct Reference . . . . .	1071
15.128.1 Detailed Description . . . . .	1072
15.129 L4::Ipc::Msg::Dir_out Struct Reference . . . . .	1072
15.129.1 Detailed Description . . . . .	1073
15.130 L4::Ipc::Msg::Do_in_data Struct Reference . . . . .	1073
15.130.1 Detailed Description . . . . .	1074
15.131 L4::Ipc::Msg::Do_in_items Struct Reference . . . . .	1074
15.131.1 Detailed Description . . . . .	1075
15.132 L4::Ipc::Msg::Do_out_data Struct Reference . . . . .	1075
15.132.1 Detailed Description . . . . .	1076
15.133 L4::Ipc::Msg::Do_out_items Struct Reference . . . . .	1076
15.133.1 Detailed Description . . . . .	1077
15.134 L4::Ipc::Msg::Do_rcv_buffers Struct Reference . . . . .	1077
15.134.1 Detailed Description . . . . .	1078
15.135 L4::Ipc::Msg::Elem< Array< A, LEN > & > Struct Template Reference . . . . .	1079
15.135.1 Detailed Description . . . . .	1079
15.136 L4::Ipc::Msg::Elem< Array< A, LEN > > Struct Template Reference . . . . .	1080
15.136.1 Detailed Description . . . . .	1080
15.137 L4::Ipc::Msg::Elem< Array_ref< A, LEN > & > Struct Template Reference . . . . .	1081
15.137.1 Detailed Description . . . . .	1081
15.138 L4::Ipc::Msg::Is_valid_rpc_type< T > Struct Template Reference . . . . .	1082
15.138.1 Detailed Description . . . . .	1083
15.139 L4::Ipc::Msg::Svr_arg_pack< IPC_TYPE > Struct Template Reference . . . . .	1084
15.139.1 Detailed Description . . . . .	1084
15.140 L4::Ipc::Msg::Svr_val_ops< MTYPE, DIR, CLASS > Struct Template Reference . . . . .	1084
15.140.1 Detailed Description . . . . .	1085
15.141 L4::Ipc::Msg_ptr< T > Class Template Reference . . . . .	1085
15.141.1 Detailed Description . . . . .	1086
15.141.2 Constructor & Destructor Documentation . . . . .	1086
15.141.2.1 Msg_ptr() . . . . .	1086
15.142 L4::Ipc::Opt< T > Struct Template Reference . . . . .	1086
15.142.1 Detailed Description . . . . .	1088
15.143 L4::Ipc::Ostream Class Reference . . . . .	1088
15.143.1 Detailed Description . . . . .	1091
15.143.2 Member Function Documentation . . . . .	1091

15.143.2.1 put() [1/2]	1091
15.143.2.2 put() [2/2]	1092
15.143.2.3 send()	1092
15.143.2.4 tag() [1/2]	1093
15.143.2.5 tag() [2/2]	1093
15.144 L4::lpc::Out< T > Struct Template Reference	1094
15.144.1 Detailed Description	1094
15.145 L4::lpc::Rcv_fpage Class Reference	1095
15.145.1 Detailed Description	1097
15.146 L4::lpc::Ret_array< T > Struct Template Reference	1097
15.146.1 Detailed Description	1097
15.147 L4::lpc::Send_only Struct Reference	1098
15.147.1 Detailed Description	1098
15.148 L4::lpc::Small_buf Class Reference	1098
15.148.1 Detailed Description	1099
15.148.2 Constructor & Destructor Documentation	1099
15.148.2.1 Small_buf() [1/2]	1099
15.148.2.2 Small_buf() [2/2]	1099
15.149 L4::lpc::Snd_fpage Class Reference	1100
15.149.1 Detailed Description	1102
15.150 L4::lpc::Str_cp_in< T > Class Template Reference	1102
15.150.1 Detailed Description	1102
15.150.2 Constructor & Destructor Documentation	1103
15.150.2.1 Str_cp_in()	1103
15.151 L4::lpc::Varg Class Reference	1103
15.151.1 Detailed Description	1104
15.151.2 Member Function Documentation	1105
15.151.2.1 data()	1105
15.151.2.2 get_value()	1105
15.151.2.3 is_nil()	1106
15.151.2.4 is_of()	1106
15.151.2.5 is_of_int()	1107
15.151.2.6 length()	1107
15.151.2.7 tag()	1108
15.151.2.8 type()	1108
15.151.2.9 value()	1108
15.152 L4::lpc::Varg_list< MAX > Class Template Reference	1109
15.152.1 Detailed Description	1111
15.153 L4::lpc::Varg_list_ref Class Reference	1111
15.153.1 Detailed Description	1112
15.153.2 Constructor & Destructor Documentation	1113
15.153.2.1 Varg_list_ref()	1113

15.154 L4::lpc::Varg_list_ref::Iterator Class Reference . . . . .	1113
15.154.1 Detailed Description . . . . .	1114
15.155 L4::lpc_gate Class Reference . . . . .	1114
15.155.1 Detailed Description . . . . .	1118
15.155.2 Member Function Documentation . . . . .	1119
15.155.2.1 get_infos() . . . . .	1119
15.156 L4::lpc_svr::Br_manager_no_buffers Class Reference . . . . .	1119
15.156.1 Detailed Description . . . . .	1122
15.156.2 Member Function Documentation . . . . .	1122
15.156.2.1 alloc_buffer_demand() . . . . .	1122
15.157 L4::lpc_svr::Compound_reply Struct Reference . . . . .	1123
15.157.1 Detailed Description . . . . .	1125
15.158 L4::lpc_svr::Default_loop_hooks Struct Reference . . . . .	1125
15.158.1 Detailed Description . . . . .	1128
15.159 L4::lpc_svr::Default_setup_wait Struct Reference . . . . .	1128
15.159.1 Detailed Description . . . . .	1129
15.160 L4::lpc_svr::Default_timeout Struct Reference . . . . .	1129
15.160.1 Detailed Description . . . . .	1131
15.161 L4::lpc_svr::Direct_dispatch< R > Struct Template Reference . . . . .	1131
15.161.1 Detailed Description . . . . .	1132
15.162 L4::lpc_svr::Direct_dispatch< R * > Struct Template Reference . . . . .	1133
15.162.1 Detailed Description . . . . .	1133
15.163 L4::lpc_svr::Exc_dispatch< R, Exc > Struct Template Reference . . . . .	1134
15.163.1 Detailed Description . . . . .	1135
15.164 L4::lpc_svr::Ignore_errors Struct Reference . . . . .	1136
15.164.1 Detailed Description . . . . .	1137
15.165 L4::lpc_svr::Server_iface Class Reference . . . . .	1137
15.165.1 Detailed Description . . . . .	1139
15.165.2 Member Function Documentation . . . . .	1139
15.165.2.1 add_timeout() . . . . .	1139
15.165.2.2 alloc_buffer_demand() . . . . .	1139
15.165.2.3 get_rcv_cap() . . . . .	1140
15.165.2.4 rcv_cap() [1/2] . . . . .	1140
15.165.2.5 rcv_cap() [2/2] . . . . .	1141
15.165.2.6 realloc_rcv_cap() . . . . .	1142
15.165.2.7 remove_timeout() . . . . .	1142
15.166 L4::lpc_svr::Timeout Class Reference . . . . .	1143
15.166.1 Detailed Description . . . . .	1144
15.166.2 Member Function Documentation . . . . .	1145
15.166.2.1 expired() . . . . .	1145
15.166.2.2 timeout() . . . . .	1145
15.167 L4::lpc_svr::Timeout_queue Class Reference . . . . .	1146

15.167.1 Detailed Description . . . . .	1146
15.167.2 Member Function Documentation . . . . .	1147
15.167.2.1 add() . . . . .	1147
15.167.2.2 handle_expired_timeouts() . . . . .	1148
15.167.2.3 next_timeout() . . . . .	1149
15.167.2.4 remove() . . . . .	1149
15.167.2.5 timeout_expired() . . . . .	1150
15.168 L4::lpc_srv::Timeout_queue_hooks< HOOKS, BR_MAN > Class Template Reference . . . . .	1151
15.168.1 Detailed Description . . . . .	1154
15.168.2 Member Function Documentation . . . . .	1155
15.168.2.1 add_timeout() . . . . .	1155
15.168.2.2 remove_timeout() . . . . .	1156
15.169 L4::Irq Class Reference . . . . .	1157
15.169.1 Detailed Description . . . . .	1161
15.169.2 Member Function Documentation . . . . .	1161
15.169.2.1 detach() . . . . .	1161
15.169.2.2 receive() . . . . .	1162
15.169.2.3 unmask() . . . . .	1163
15.169.2.4 wait() . . . . .	1164
15.170 L4::Irq_eoi Class Reference . . . . .	1165
15.170.1 Detailed Description . . . . .	1166
15.170.2 Member Function Documentation . . . . .	1166
15.170.2.1 unmask() . . . . .	1166
15.171 L4::Irq_handler_object Struct Reference . . . . .	1167
15.171.1 Detailed Description . . . . .	1169
15.172 L4::Irq_mux Struct Reference . . . . .	1170
15.172.1 Detailed Description . . . . .	1173
15.172.2 Member Function Documentation . . . . .	1173
15.172.2.1 chain() . . . . .	1173
15.173 L4::Irqp_t< Derived, BASE, bool > Struct Template Reference . . . . .	1174
15.173.1 Detailed Description . . . . .	1177
15.173.2 Member Function Documentation . . . . .	1178
15.173.2.1 dispatch() . . . . .	1178
15.173.2.2 obj_cap() . . . . .	1178
15.174 L4::Kip::Mem_desc Class Reference . . . . .	1179
15.174.1 Detailed Description . . . . .	1180
15.174.2 Member Enumeration Documentation . . . . .	1180
15.174.2.1 Arch_sub_type_common . . . . .	1180
15.174.2.2 Info_sub_type . . . . .	1181
15.174.2.3 Mem_type . . . . .	1181
15.174.3 Constructor & Destructor Documentation . . . . .	1181
15.174.3.1 Mem_desc() . . . . .	1181

15.174.4 Member Function Documentation . . . . .	1182
15.174.4.1 all() [1/2] . . . . .	1182
15.174.4.2 all() [2/2] . . . . .	1183
15.174.4.3 count() [1/2] . . . . .	1183
15.174.4.4 count() [2/2] . . . . .	1184
15.174.4.5 end() . . . . .	1185
15.174.4.6 first() . . . . .	1185
15.174.4.7 is_virtual() . . . . .	1186
15.174.4.8 set() . . . . .	1186
15.174.4.9 size() . . . . .	1187
15.174.4.10 start() . . . . .	1188
15.174.4.11 sub_type() . . . . .	1188
15.174.4.12 type() . . . . .	1188
15.175 L4::Kobject Class Reference . . . . .	1189
15.175.1 Detailed Description . . . . .	1189
15.175.2 Member Function Documentation . . . . .	1190
15.175.2.1 cap() . . . . .	1190
15.175.2.2 dec_refcnt() . . . . .	1191
15.176 L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND > Class Template Reference . . . . .	1192
15.176.1 Detailed Description . . . . .	1193
15.176.2 Member Typedef Documentation . . . . .	1194
15.176.2.1 __iface . . . . .	1194
15.176.2.2 __iface_list . . . . .	1194
15.176.2.3 Class . . . . .	1194
15.176.3 Member Function Documentation . . . . .	1194
15.176.3.1 __check_protocols__() . . . . .	1194
15.176.3.2 c() . . . . .	1195
15.177 L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND > Struct Template Reference . . . . .	1195
15.177.1 Detailed Description . . . . .	1197
15.177.2 Member Typedef Documentation . . . . .	1198
15.177.2.1 __iface . . . . .	1198
15.177.2.2 __iface_list . . . . .	1198
15.177.2.3 Class . . . . .	1198
15.177.3 Member Function Documentation . . . . .	1198
15.177.3.1 __check_protocols__() . . . . .	1198
15.177.3.2 c() . . . . .	1199
15.178 L4::Kobject_demand< T > Struct Template Reference . . . . .	1199
15.178.1 Detailed Description . . . . .	1199
15.179 L4::Kobject_t< Derived, Base, PROTO, S_DEMAND > Class Template Reference . . . . .	1200
15.179.1 Detailed Description . . . . .	1201
15.180 L4::Kobject_typeid< T > Struct Template Reference . . . . .	1201
15.180.1 Detailed Description . . . . .	1202

15.180.2 Member Typedef Documentation . . . . .	1202
15.180.2.1 Demand . . . . .	1202
15.180.3 Member Function Documentation . . . . .	1203
15.180.3.1 demand() . . . . .	1203
15.180.3.2 id() . . . . .	1203
15.180.3.3 proto_dispatch() . . . . .	1203
15.181 L4::Kobject_typeid< void > Struct Reference . . . . .	1204
15.181.1 Detailed Description . . . . .	1205
15.182 L4::Kobject_x< Derived, ARGS > Struct Template Reference . . . . .	1205
15.182.1 Detailed Description . . . . .	1206
15.183 L4::Meta Class Reference . . . . .	1206
15.183.1 Detailed Description . . . . .	1209
15.183.2 Member Function Documentation . . . . .	1209
15.183.2.1 interface() . . . . .	1209
15.183.2.2 num_interfaces() . . . . .	1210
15.183.2.3 supports() . . . . .	1210
15.184 L4::Out_of_memory Class Reference . . . . .	1211
15.184.1 Detailed Description . . . . .	1213
15.185 L4::Pager Class Reference . . . . .	1214
15.185.1 Detailed Description . . . . .	1216
15.185.2 Member Function Documentation . . . . .	1216
15.185.2.1 page_fault() . . . . .	1216
15.186 L4::Platform_control Class Reference . . . . .	1218
15.186.1 Detailed Description . . . . .	1220
15.186.2 Member Function Documentation . . . . .	1220
15.186.2.1 cpu_allow_shutdown() . . . . .	1220
15.186.2.2 cpu_disable() . . . . .	1221
15.186.2.3 cpu_enable() . . . . .	1221
15.186.2.4 system_shutdown() . . . . .	1221
15.186.2.5 system_suspend() . . . . .	1222
15.187 L4::Poll_timeout_counter Class Reference . . . . .	1222
15.187.1 Detailed Description . . . . .	1223
15.187.2 Constructor & Destructor Documentation . . . . .	1223
15.187.2.1 Poll_timeout_counter() . . . . .	1223
15.187.3 Member Function Documentation . . . . .	1224
15.187.3.1 set() . . . . .	1224
15.187.3.2 timed_out() . . . . .	1224
15.188 L4::Poll_timeout_kipclock Class Reference . . . . .	1225
15.188.1 Detailed Description . . . . .	1225
15.188.2 Constructor & Destructor Documentation . . . . .	1226
15.188.2.1 Poll_timeout_kipclock() . . . . .	1226
15.188.3 Member Function Documentation . . . . .	1226

15.188.3.1 set()	1226
15.188.3.2 test()	1227
15.188.3.3 timed_out()	1228
15.189 L4::Proto_t< P > Struct Template Reference	1228
15.189.1 Detailed Description	1228
15.190 L4::Rcv_endpoint Class Reference	1229
15.190.1 Detailed Description	1232
15.190.2 Member Function Documentation	1232
15.190.2.1 bind_thread()	1232
15.191 L4::Registry_iface Class Reference	1233
15.191.1 Detailed Description	1235
15.191.2 Member Function Documentation	1235
15.191.2.1 register_irq_obj()	1235
15.191.2.2 register_obj() [1/3]	1235
15.191.2.3 register_obj() [2/3]	1236
15.191.2.4 register_obj() [3/3]	1236
15.191.2.5 unregister_obj()	1237
15.192 L4::Runtime_error Class Reference	1237
15.192.1 Detailed Description	1240
15.192.2 Constructor & Destructor Documentation	1240
15.192.2.1 Runtime_error()	1240
15.192.3 Member Function Documentation	1241
15.192.3.1 err_no()	1241
15.192.3.2 extra_str()	1241
15.193 L4::Scheduler Class Reference	1241
15.193.1 Detailed Description	1245
15.193.2 Member Function Documentation	1245
15.193.2.1 idle_time()	1245
15.193.2.2 info()	1246
15.193.2.3 is_online()	1247
15.193.2.4 run_thread()	1247
15.194 L4::Semaphore Struct Reference	1248
15.194.1 Detailed Description	1252
15.194.2 Member Function Documentation	1252
15.194.2.1 down()	1252
15.194.2.2 up()	1253
15.195 L4::Server< LOOP_HOOKS > Class Template Reference	1254
15.195.1 Detailed Description	1256
15.195.2 Constructor & Destructor Documentation	1257
15.195.2.1 Server()	1257
15.195.3 Member Function Documentation	1257
15.195.3.1 internal_loop()	1257



15.195.3.2 loop()	1258
15.196 L4::Server_object Class Reference	1259
15.196.1 Detailed Description	1261
15.196.2 Member Function Documentation	1261
15.196.2.1 dispatch() [1/2]	1261
15.196.2.2 dispatch() [2/2]	1262
15.197 L4::Server_object_t< IFACE, BASE > Struct Template Reference	1263
15.197.1 Detailed Description	1266
15.197.2 Member Function Documentation	1267
15.197.2.1 dispatch_meta_request()	1267
15.197.2.2 get_buffer_demand()	1267
15.197.2.3 proto_dispatch()	1267
15.198 L4::Server_object_x< Derived, IFACE, BASE > Struct Template Reference	1269
15.198.1 Detailed Description	1271
15.199 L4::Smart_cap< T, SMART > Class Template Reference	1272
15.199.1 Detailed Description	1276
15.199.2 Constructor & Destructor Documentation	1276
15.199.2.1 Smart_cap()	1276
15.200 L4::String Class Reference	1276
15.200.1 Detailed Description	1277
15.201 L4::Task Class Reference	1277
15.201.1 Detailed Description	1281
15.201.2 Member Function Documentation	1281
15.201.2.1 add_ku_mem()	1281
15.201.2.2 cap_equal()	1282
15.201.2.3 cap_valid()	1283
15.201.2.4 delete_obj()	1284
15.201.2.5 map()	1285
15.201.2.6 release_cap()	1286
15.201.2.7 unmap()	1287
15.201.2.8 unmap_batch()	1288
15.202 L4::Thread Class Reference	1289
15.202.1 Detailed Description	1292
15.202.2 Member Function Documentation	1292
15.202.2.1 control()	1292
15.202.2.2 ex_regs() [1/2]	1293
15.202.2.3 ex_regs() [2/2]	1294
15.202.2.4 modify_senders()	1295
15.202.2.5 register_del_irq()	1296
15.202.2.6 stats_time()	1297
15.202.2.7 switch_to()	1298
15.202.2.8 vcpu_control()	1298

15.202.2.9 <a href="#">vcpu_control_ext()</a>	1299
15.202.2.10 <a href="#">vcpu_resume_commit()</a>	1300
15.202.2.11 <a href="#">vcpu_resume_start()</a>	1301
15.203 <a href="#">L4::Thread::Attr Class Reference</a>	1302
15.203.1 Detailed Description	1303
15.203.2 Constructor & Destructor Documentation	1303
15.203.2.1 <a href="#">Attr()</a>	1303
15.203.3 Member Function Documentation	1304
15.203.3.1 <a href="#">alien()</a>	1304
15.203.3.2 <a href="#">bind()</a>	1304
15.203.3.3 <a href="#">exc_handler()</a> [1/2]	1305
15.203.3.4 <a href="#">exc_handler()</a> [2/2]	1305
15.203.3.5 <a href="#">pager()</a> [1/2]	1306
15.203.3.6 <a href="#">pager()</a> [2/2]	1306
15.203.3.7 <a href="#">ux_host_syscall()</a>	1307
15.204 <a href="#">L4::Thread::Modify_senders Class Reference</a>	1307
15.204.1 Detailed Description	1308
15.204.2 Member Function Documentation	1308
15.204.2.1 <a href="#">add()</a>	1308
15.205 <a href="#">L4::Triggerable Struct Reference</a>	1309
15.205.1 Detailed Description	1312
15.205.2 Member Function Documentation	1312
15.205.2.1 <a href="#">trigger()</a>	1312
15.206 <a href="#">L4::Type_info Struct Reference</a>	1313
15.206.1 Detailed Description	1314
15.207 <a href="#">L4::Type_info::Demand Class Reference</a>	1314
15.207.1 Detailed Description	1316
15.207.2 Constructor & Destructor Documentation	1317
15.207.2.1 <a href="#">Demand()</a>	1317
15.207.3 Member Function Documentation	1317
15.207.3.1 <a href="#">no_demand()</a>	1317
15.208 <a href="#">L4::Type_info::Demand_t&lt; CAPS, FLAGS, MEM, PORTS &gt; Struct Template Reference</a>	1318
15.208.1 Detailed Description	1320
15.208.2 Member Enumeration Documentation	1320
15.208.2.1 <a href="#">anonymous enum</a>	1320
15.209 <a href="#">L4::Type_info::Demand_union_t&lt; D1, D2 &gt; Struct Template Reference</a>	1320
15.209.1 Detailed Description	1323
15.210 <a href="#">L4::Typeid::Detail::_Rpc&lt; OPCODE, O, X &gt; Struct Template Reference</a>	1323
15.210.1 Detailed Description	1324
15.211 <a href="#">L4::Typeid::Detail::_Rpc&lt; OPCODE, O, Default_op&lt; R &gt; &gt;::Rpc&lt; Y &gt; Struct Template Reference</a>	1325
15.211.1 Detailed Description	1325
15.212 <a href="#">L4::Typeid::Detail::_Rpc&lt; OPCODE, O, R, X... &gt; Struct Template Reference</a>	1325

15.212.1 Detailed Description . . . . .	1326
15.213 L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y > Struct Template Reference . . . . .	1327
15.213.1 Detailed Description . . . . .	1327
15.214 L4::Typeid::Detail::_Rpc_end Struct Reference . . . . .	1327
15.214.1 Detailed Description . . . . .	1328
15.215 L4::Typeid::P_dispatch< LIST > Struct Template Reference . . . . .	1328
15.215.1 Detailed Description . . . . .	1329
15.216 L4::Typeid::Raw_ipc< CLASS > Struct Template Reference . . . . .	1329
15.216.1 Detailed Description . . . . .	1329
15.217 L4::Typeid::_Rpc_nocode< OPERATION > Struct Template Reference . . . . .	1330
15.217.1 Detailed Description . . . . .	1331
15.218 L4::Typeid::_Rpc< RPCS > Struct Template Reference . . . . .	1332
15.218.1 Detailed Description . . . . .	1333
15.219 L4::Typeid::_Rpc_code< OPCODE_TYPE > Struct Template Reference . . . . .	1334
15.219.1 Detailed Description . . . . .	1334
15.220 L4::Typeid::_Rpc_code< OPCODE_TYPE >::F< RPCS > Struct Template Reference . . . . .	1335
15.220.1 Detailed Description . . . . .	1336
15.221 L4::Typeid::_Rpc_sys< ARG > Struct Template Reference . . . . .	1337
15.221.1 Detailed Description . . . . .	1338
15.222 L4::Types::Bool< V > Struct Template Reference . . . . .	1339
15.222.1 Detailed Description . . . . .	1339
15.223 L4::Types::False Struct Reference . . . . .	1340
15.223.1 Detailed Description . . . . .	1341
15.224 L4::Types::Flags< BITS_ENUM, UNDERLYING > Class Template Reference . . . . .	1342
15.224.1 Detailed Description . . . . .	1343
15.224.2 Member Enumeration Documentation . . . . .	1344
15.224.2.1 None_type . . . . .	1344
15.224.3 Constructor & Destructor Documentation . . . . .	1344
15.224.3.1 Flags() [1/2] . . . . .	1344
15.224.3.2 Flags() [2/2] . . . . .	1345
15.224.4 Member Function Documentation . . . . .	1345
15.224.4.1 clear() . . . . .	1345
15.224.4.2 from_raw() . . . . .	1345
15.225 L4::Types::Flags_ops_t< DT > Struct Template Reference . . . . .	1346
15.225.1 Detailed Description . . . . .	1347
15.226 L4::Types::Flags_t< DT, T > Struct Template Reference . . . . .	1348
15.226.1 Detailed Description . . . . .	1350
15.227 L4::Types::Int_for_size< SIZE, bool > Struct Template Reference . . . . .	1350
15.227.1 Detailed Description . . . . .	1351
15.228 L4::Types::Int_for_type< T > Struct Template Reference . . . . .	1351
15.228.1 Detailed Description . . . . .	1351
15.229 L4::Types::Same< A, B > Struct Template Reference . . . . .	1352

15.229.1 Detailed Description . . . . .	1353
15.230 L4::Types::True Struct Reference . . . . .	1354
15.230.1 Detailed Description . . . . .	1355
15.231 L4::Uart Class Reference . . . . .	1356
15.231.1 Detailed Description . . . . .	1358
15.231.2 Member Function Documentation . . . . .	1358
15.231.2.1 change_mode() . . . . .	1358
15.231.2.2 char_avail() . . . . .	1358
15.231.2.3 enable_rx_irq() . . . . .	1358
15.231.2.4 generic_write() . . . . .	1359
15.231.2.5 get_char() . . . . .	1359
15.231.2.6 mode() . . . . .	1360
15.231.2.7 rate() . . . . .	1360
15.231.2.8 shutdown() . . . . .	1360
15.231.2.9 startup() . . . . .	1360
15.231.2.10 write() . . . . .	1361
15.232 L4::Uart_apb Class Reference . . . . .	1361
15.232.1 Detailed Description . . . . .	1364
15.232.2 Member Function Documentation . . . . .	1364
15.232.2.1 change_mode() . . . . .	1364
15.232.2.2 char_avail() . . . . .	1365
15.232.2.3 enable_rx_irq() . . . . .	1365
15.232.2.4 get_char() . . . . .	1365
15.232.2.5 shutdown() . . . . .	1366
15.232.2.6 startup() . . . . .	1366
15.232.2.7 write() . . . . .	1366
15.233 L4::Unknown_error Class Reference . . . . .	1367
15.233.1 Detailed Description . . . . .	1369
15.234 L4::Vcon Class Reference . . . . .	1369
15.234.1 Detailed Description . . . . .	1373
15.234.2 Member Function Documentation . . . . .	1373
15.234.2.1 get_attr() . . . . .	1373
15.234.2.2 read() . . . . .	1374
15.234.2.3 read_with_flags() . . . . .	1375
15.234.2.4 send() . . . . .	1376
15.234.2.5 set_attr() . . . . .	1376
15.234.2.6 write() . . . . .	1377
15.235 L4::Vm Class Reference . . . . .	1378
15.235.1 Detailed Description . . . . .	1382
15.235.2 Member Function Documentation . . . . .	1382
15.235.2.1 vgicc_map() . . . . .	1382
15.236 l4_buf_regs_t Struct Reference . . . . .	1383

15.236.1 Detailed Description . . . . .	1384
15.237 l4_exc_regs_t Struct Reference . . . . .	1384
15.237.1 Detailed Description . . . . .	1386
15.237.2 Field Documentation . . . . .	1386
15.237.2.1 flags . . . . .	1386
15.237.2.2 ss . . . . .	1387
15.238 l4_ext_vcpu_state_vmx_t Struct Reference . . . . .	1387
15.238.1 Detailed Description . . . . .	1388
15.239 l4_fpage_t Union Reference . . . . .	1388
15.239.1 Detailed Description . . . . .	1389
15.240 l4_icu_info_t Struct Reference . . . . .	1389
15.240.1 Detailed Description . . . . .	1390
15.240.2 Field Documentation . . . . .	1390
15.240.2.1 features . . . . .	1390
15.241 l4_icu_msi_info_t Struct Reference . . . . .	1390
15.241.1 Detailed Description . . . . .	1391
15.242 l4_kernel_info_mem_desc_t Struct Reference . . . . .	1391
15.242.1 Detailed Description . . . . .	1391
15.243 l4_kernel_info_t Struct Reference . . . . .	1392
15.243.1 Detailed Description . . . . .	1394
15.244 l4_msg_regs_t Union Reference . . . . .	1394
15.244.1 Detailed Description . . . . .	1394
15.245 l4_msgtag_t Struct Reference . . . . .	1395
15.245.1 Detailed Description . . . . .	1396
15.245.2 Member Function Documentation . . . . .	1396
15.245.2.1 flags() . . . . .	1396
15.246 l4_sched_cpu_set_t Struct Reference . . . . .	1397
15.246.1 Detailed Description . . . . .	1397
15.246.2 Member Function Documentation . . . . .	1398
15.246.2.1 granularity() . . . . .	1398
15.246.2.2 offset() . . . . .	1398
15.246.2.3 set() . . . . .	1399
15.246.3 Field Documentation . . . . .	1400
15.246.3.1 gran_offset . . . . .	1400
15.247 l4_sched_param_t Struct Reference . . . . .	1401
15.247.1 Detailed Description . . . . .	1401
15.248 l4_snd_fpage_t Struct Reference . . . . .	1402
15.248.1 Detailed Description . . . . .	1402
15.249 l4_thread_regs_t Struct Reference . . . . .	1403
15.249.1 Detailed Description . . . . .	1403
15.250 l4_timeout_s Struct Reference . . . . .	1404
15.250.1 Detailed Description . . . . .	1404

15.251 <a href="#">l4_timeout_t Union Reference</a> . . . . .	1405
15.251.1 Detailed Description . . . . .	1405
15.252 <a href="#">l4_vcon_attr_t Struct Reference</a> . . . . .	1406
15.252.1 Detailed Description . . . . .	1406
15.252.2 Member Function Documentation . . . . .	1407
15.252.2.1 <a href="#">set_raw()</a> . . . . .	1407
15.253 <a href="#">l4_vcpu_arch_state_t Struct Reference</a> . . . . .	1407
15.253.1 Detailed Description . . . . .	1408
15.254 <a href="#">l4_vcpu_ipc_regs_t Struct Reference</a> . . . . .	1408
15.254.1 Detailed Description . . . . .	1409
15.255 <a href="#">l4_vcpu_regs_t Struct Reference</a> . . . . .	1409
15.255.1 Detailed Description . . . . .	1411
15.255.2 Field Documentation . . . . .	1411
15.255.2.1 <a href="#">ax</a> . . . . .	1411
15.255.2.2 <a href="#">bp</a> . . . . .	1411
15.255.2.3 <a href="#">bx</a> . . . . .	1411
15.255.2.4 <a href="#">cx</a> . . . . .	1411
15.255.2.5 <a href="#">di</a> . . . . .	1412
15.255.2.6 <a href="#">dx</a> . . . . .	1412
15.255.2.7 <a href="#">si</a> . . . . .	1412
15.256 <a href="#">l4_vcpu_state_t Struct Reference</a> . . . . .	1413
15.256.1 Detailed Description . . . . .	1415
15.256.2 Field Documentation . . . . .	1415
15.256.2.1 <a href="#">version</a> . . . . .	1415
15.257 <a href="#">l4_vhw_descriptor Struct Reference</a> . . . . .	1416
15.257.1 Detailed Description . . . . .	1417
15.258 <a href="#">l4_vhw_entry Struct Reference</a> . . . . .	1417
15.258.1 Detailed Description . . . . .	1418
15.259 <a href="#">l4_vm_svm_vmcb_control_area Struct Reference</a> . . . . .	1418
15.259.1 Detailed Description . . . . .	1418
15.260 <a href="#">l4_vm_svm_vmcb_state_save_area Struct Reference</a> . . . . .	1419
15.260.1 Detailed Description . . . . .	1419
15.261 <a href="#">l4_vm_svm_vmcb_state_save_area_seg Struct Reference</a> . . . . .	1420
15.261.1 Detailed Description . . . . .	1420
15.262 <a href="#">l4_vm_svm_vmcb_t Struct Reference</a> . . . . .	1420
15.262.1 Detailed Description . . . . .	1421
15.263 <a href="#">l4_vm_tz_state Struct Reference</a> . . . . .	1421
15.263.1 Detailed Description . . . . .	1422
15.264 <a href="#">l4_vmx_offset_table_t Struct Reference</a> . . . . .	1422
15.264.1 Detailed Description . . . . .	1423
15.265 <a href="#">L4drivers::Mmio_register_block&lt; MAX_BITS &gt; Struct Template Reference</a> . . . . .	1424
15.265.1 Detailed Description . . . . .	1425

15.266 L4drivers::Register_block< MAX_BITS, BLOCK > Class Template Reference . . . . .	1425
15.266.1 Detailed Description . . . . .	1426
15.266.2 Member Function Documentation . . . . .	1426
15.266.2.1 operator[]() [1/2] . . . . .	1426
15.266.2.2 operator[]() [2/2] . . . . .	1427
15.266.2.3 r() [1/2] . . . . .	1427
15.266.2.4 r() [2/2] . . . . .	1428
15.267 L4drivers::Register_block_base< MAX_BITS > Struct Template Reference . . . . .	1428
15.267.1 Detailed Description . . . . .	1429
15.268 L4drivers::Register_block_impl< BASE, MAX_BITS > Struct Template Reference . . . . .	1429
15.268.1 Detailed Description . . . . .	1430
15.269 L4drivers::Register_block_tmpl< BLOCK > Class Template Reference . . . . .	1431
15.269.1 Detailed Description . . . . .	1432
15.270 L4drivers::Register_tmpl< BITS, BLOCK > Class Template Reference . . . . .	1432
15.270.1 Detailed Description . . . . .	1433
15.270.2 Member Function Documentation . . . . .	1434
15.270.2.1 clear() . . . . .	1434
15.270.2.2 modify() . . . . .	1434
15.270.2.3 operator=() . . . . .	1435
15.270.2.4 set() . . . . .	1435
15.270.2.5 write() . . . . .	1435
15.271 L4drivers::Ro_register_block< MAX_BITS, BLOCK > Class Template Reference . . . . .	1436
15.271.1 Detailed Description . . . . .	1436
15.271.2 Member Function Documentation . . . . .	1437
15.271.2.1 operator[]() . . . . .	1437
15.271.2.2 r() . . . . .	1437
15.272 L4drivers::Ro_register_tmpl< BITS, BLOCK > Class Template Reference . . . . .	1438
15.272.1 Detailed Description . . . . .	1439
15.272.2 Member Function Documentation . . . . .	1439
15.272.2.1 operator value_type() . . . . .	1439
15.272.2.2 read() . . . . .	1440
15.273 L4Re::Cap_alloc Class Reference . . . . .	1440
15.273.1 Detailed Description . . . . .	1441
15.273.2 Member Function Documentation . . . . .	1441
15.273.2.1 alloc() [1/2] . . . . .	1441
15.273.2.2 alloc() [2/2] . . . . .	1442
15.273.2.3 free() . . . . .	1442
15.273.2.4 get_cap_alloc() . . . . .	1443
15.274 L4Re::Console Class Reference . . . . .	1443
15.274.1 Detailed Description . . . . .	1446
15.275 L4Re::Dataspace Class Reference . . . . .	1446
15.275.1 Detailed Description . . . . .	1450

15.275.2 Member Function Documentation . . . . .	1450
15.275.2.1 allocate() . . . . .	1450
15.275.2.2 clear() . . . . .	1451
15.275.2.3 copy_in() . . . . .	1451
15.275.2.4 flags() . . . . .	1452
15.275.2.5 info() . . . . .	1453
15.275.2.6 map() . . . . .	1454
15.275.2.7 map_info() . . . . .	1454
15.275.2.8 map_region() . . . . .	1455
15.275.2.9 size() . . . . .	1456
15.276 L4Re::Dataspace::F Struct Reference . . . . .	1457
15.276.1 Detailed Description . . . . .	1457
15.276.2 Member Enumeration Documentation . . . . .	1457
15.276.2.1 anonymous enum . . . . .	1457
15.276.2.2 Flags . . . . .	1458
15.277 L4Re::Dataspace::Stats Struct Reference . . . . .	1458
15.277.1 Detailed Description . . . . .	1459
15.278 L4Re::Debug_obj Class Reference . . . . .	1459
15.278.1 Detailed Description . . . . .	1461
15.278.2 Member Function Documentation . . . . .	1461
15.278.2.1 debug() . . . . .	1461
15.279 L4Re::Default_event_payload Struct Reference . . . . .	1462
15.279.1 Detailed Description . . . . .	1462
15.280 L4Re::Dma_space Class Reference . . . . .	1463
15.280.1 Detailed Description . . . . .	1464
15.280.2 Member Typedef Documentation . . . . .	1464
15.280.2.1 Attributes . . . . .	1464
15.280.3 Member Enumeration Documentation . . . . .	1464
15.280.3.1 Attribute . . . . .	1464
15.280.3.2 Direction . . . . .	1465
15.280.3.3 Space_attrb . . . . .	1465
15.280.4 Member Function Documentation . . . . .	1465
15.280.4.1 associate() . . . . .	1465
15.280.4.2 disassociate() . . . . .	1466
15.280.4.3 map() . . . . .	1466
15.280.4.4 unmap() . . . . .	1468
15.281 L4Re::Env Class Reference . . . . .	1469
15.281.1 Detailed Description . . . . .	1471
15.281.2 Member Function Documentation . . . . .	1472
15.281.2.1 env() . . . . .	1472
15.281.2.2 factory() [1/2] . . . . .	1472
15.281.2.3 factory() [2/2] . . . . .	1472



15.281.2.4 first_free_cap() [1/2]	1473
15.281.2.5 first_free_cap() [2/2]	1473
15.281.2.6 first_free_utcb() [1/2]	1473
15.281.2.7 first_free_utcb() [2/2]	1474
15.281.2.8 get()	1474
15.281.2.9 get_cap() [1/2]	1475
15.281.2.10 get_cap() [2/2]	1476
15.281.2.11 initial_caps() [1/2]	1476
15.281.2.12 initial_caps() [2/2]	1476
15.281.2.13 log() [1/2]	1477
15.281.2.14 log() [2/2]	1477
15.281.2.15 main_thread() [1/2]	1477
15.281.2.16 main_thread() [2/2]	1478
15.281.2.17 mem_alloc() [1/2]	1478
15.281.2.18 mem_alloc() [2/2]	1478
15.281.2.19 parent() [1/2]	1479
15.281.2.20 parent() [2/2]	1479
15.281.2.21 rm() [1/2]	1479
15.281.2.22 rm() [2/2]	1479
15.281.2.23 scheduler() [1/2]	1480
15.281.2.24 scheduler() [2/2]	1480
15.281.2.25 task()	1480
15.281.2.26 utcb_area() [1/2]	1481
15.281.2.27 utcb_area() [2/2]	1481
15.282 L4Re::Event Class Reference	1481
15.282.1 Detailed Description	1485
15.282.2 Member Function Documentation	1485
15.282.2.1 get_axis_info()	1485
15.282.2.2 get_buffer()	1486
15.282.2.3 get_num_streams()	1486
15.282.2.4 get_stream_info()	1486
15.282.2.5 get_stream_info_for_id()	1487
15.282.2.6 get_stream_state_for_id()	1487
15.283 L4Re::Event_buffer_t< PAYLOAD > Class Template Reference	1488
15.283.1 Detailed Description	1490
15.283.2 Constructor & Destructor Documentation	1490
15.283.2.1 Event_buffer_t()	1490
15.283.3 Member Function Documentation	1491
15.283.3.1 next()	1491
15.283.3.2 put()	1491
15.284 L4Re::Event_buffer_t< PAYLOAD >::Event Struct Reference	1492
15.284.1 Detailed Description	1493

15.285 L4Re::Inhibitor Class Reference . . . . .	1493
15.285.1 Detailed Description . . . . .	1496
15.285.2 Member Enumeration Documentation . . . . .	1496
15.285.2.1 anonymous enum . . . . .	1496
15.285.3 Member Function Documentation . . . . .	1497
15.285.3.1 acquire() . . . . .	1497
15.285.3.2 next_lock_info() . . . . .	1497
15.285.3.3 release() . . . . .	1498
15.286 L4Re::Log Class Reference . . . . .	1498
15.286.1 Detailed Description . . . . .	1503
15.286.2 Member Function Documentation . . . . .	1503
15.286.2.1 print() . . . . .	1503
15.286.2.2 printf() . . . . .	1503
15.287 L4Re::Mem_alloc Class Reference . . . . .	1503
15.287.1 Detailed Description . . . . .	1507
15.287.2 Member Enumeration Documentation . . . . .	1507
15.287.2.1 Mem_alloc_flags . . . . .	1507
15.287.3 Member Function Documentation . . . . .	1508
15.287.3.1 alloc() . . . . .	1508
15.288 L4Re::Mmio_space Struct Reference . . . . .	1509
15.288.1 Detailed Description . . . . .	1511
15.288.2 Member Enumeration Documentation . . . . .	1511
15.288.2.1 Access_width . . . . .	1511
15.288.3 Member Function Documentation . . . . .	1512
15.288.3.1 mmio_read() . . . . .	1512
15.288.3.2 mmio_write() . . . . .	1512
15.289 L4Re::Namespace Class Reference . . . . .	1513
15.289.1 Detailed Description . . . . .	1516
15.289.2 Member Enumeration Documentation . . . . .	1516
15.289.2.1 Query_result_flags . . . . .	1516
15.289.2.2 Query_timeout . . . . .	1516
15.289.2.3 Register_flags . . . . .	1516
15.289.3 Member Function Documentation . . . . .	1517
15.289.3.1 query() [1/2] . . . . .	1517
15.289.3.2 query() [2/2] . . . . .	1518
15.289.3.3 register_obj() . . . . .	1519
15.289.3.4 unlink() . . . . .	1519
15.290 L4Re::Ned::Cmd_control Class Reference . . . . .	1520
15.290.1 Detailed Description . . . . .	1520
15.290.2 Member Function Documentation . . . . .	1521
15.290.2.1 execute() [1/2] . . . . .	1521
15.290.2.2 execute() [2/2] . . . . .	1522

15.291 L4Re::Parent Class Reference . . . . .	1522
15.291.1 Detailed Description . . . . .	1525
15.291.2 Member Function Documentation . . . . .	1525
15.291.2.1 signal() . . . . .	1525
15.292 L4Re::Random Struct Reference . . . . .	1526
15.292.1 Detailed Description . . . . .	1530
15.292.2 Member Function Documentation . . . . .	1530
15.292.2.1 get_random() . . . . .	1530
15.293 L4Re::Rm Class Reference . . . . .	1531
15.293.1 Detailed Description . . . . .	1536
15.293.2 Member Typedef Documentation . . . . .	1536
15.293.2.1 Area . . . . .	1536
15.293.2.2 Region . . . . .	1536
15.293.3 Member Enumeration Documentation . . . . .	1536
15.293.3.1 Detach_flags . . . . .	1536
15.293.3.2 Detach_result . . . . .	1537
15.293.3.3 Region_flag_shifts . . . . .	1537
15.293.4 Member Function Documentation . . . . .	1537
15.293.4.1 attach() [1/2] . . . . .	1537
15.293.4.2 attach() [2/2] . . . . .	1538
15.293.4.3 detach() [1/3] . . . . .	1540
15.293.4.4 detach() [2/3] . . . . .	1540
15.293.4.5 detach() [3/3] . . . . .	1541
15.293.4.6 find() . . . . .	1541
15.293.4.7 free_area() . . . . .	1542
15.293.4.8 get_areas() . . . . .	1543
15.293.4.9 get_regions() . . . . .	1543
15.293.4.10 reserve_area() [1/2] . . . . .	1544
15.293.4.11 reserve_area() [2/2] . . . . .	1544
15.294 L4Re::Rm::F Struct Reference . . . . .	1545
15.294.1 Detailed Description . . . . .	1546
15.294.2 Member Enumeration Documentation . . . . .	1546
15.294.2.1 Attach_flags . . . . .	1546
15.294.2.2 Region_flags . . . . .	1546
15.295 L4Re::Rm::Range Struct Reference . . . . .	1547
15.295.1 Detailed Description . . . . .	1548
15.296 L4Re::Rm::Unique_region< T > Class Template Reference . . . . .	1548
15.296.1 Detailed Description . . . . .	1550
15.296.2 Constructor & Destructor Documentation . . . . .	1550
15.296.2.1 Unique_region() [1/3] . . . . .	1550
15.296.2.2 Unique_region() [2/3] . . . . .	1550
15.296.2.3 Unique_region() [3/3] . . . . .	1551

15.296.2.4 ~Unique_region()	1551
15.296.3 Member Function Documentation	1551
15.296.3.1 get()	1551
15.296.3.2 is_valid()	1552
15.296.3.3 operator=()	1553
15.296.3.4 release()	1553
15.296.3.5 reset()	1553
15.297 L4Re::Smart_cap_auto< Unmap_flags > Class Template Reference	1554
15.297.1 Detailed Description	1555
15.298 L4Re::Smart_count_cap< Unmap_flags > Class Template Reference	1555
15.298.1 Detailed Description	1556
15.299 L4Re::Util::Br_manager Class Reference	1556
15.299.1 Detailed Description	1559
15.299.2 Member Function Documentation	1559
15.299.2.1 alloc_buffer_demand()	1559
15.299.2.2 get_rcv_cap()	1560
15.299.2.3 realloc_rcv_cap()	1561
15.299.2.4 set_rcv_cap_flags()	1561
15.300 L4Re::Util::Br_manager_hooks Struct Reference	1562
15.300.1 Detailed Description	1564
15.301 L4Re::Util::Br_manager_timeout_hooks Struct Reference	1564
15.301.1 Detailed Description	1567
15.302 L4Re::Util::Cap_alloc_base Class Reference	1568
15.302.1 Detailed Description	1568
15.303 L4Re::Util::Counter< COUNTER > Struct Template Reference	1569
15.303.1 Detailed Description	1569
15.303.2 Member Function Documentation	1569
15.303.2.1 dec()	1569
15.303.2.2 inc()	1570
15.304 L4Re::Util::Counter_atomic< COUNTER > Struct Template Reference	1571
15.304.1 Detailed Description	1571
15.304.2 Member Function Documentation	1572
15.304.2.1 dec()	1572
15.304.2.2 inc()	1572
15.305 L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg > Class Template Reference	1572
15.305.1 Detailed Description	1574
15.305.2 Constructor & Destructor Documentation	1574
15.305.2.1 Counting_cap_alloc()	1574
15.305.3 Member Function Documentation	1574
15.305.3.1 alloc() [1/2]	1574
15.305.3.2 alloc() [2/2]	1575
15.305.3.3 free()	1575

15.305.3.4 release()	1576
15.305.3.5 setup()	1577
15.305.3.6 take()	1577
15.306 L4Re::Util::Dataspace_svr Class Reference	1578
15.306.1 Detailed Description	1579
15.306.2 Member Function Documentation	1579
15.306.2.1 allocate()	1579
15.306.2.2 clear()	1580
15.306.2.3 copy()	1580
15.306.2.4 is_static()	1581
15.306.2.5 map()	1581
15.306.2.6 map_hook()	1582
15.306.2.7 map_info()	1583
15.306.2.8 page_shift()	1583
15.306.2.9 release()	1584
15.306.2.10 take()	1584
15.307 L4Re::Util::Event_buffer_consumer_t< PAYLOAD > Class Template Reference	1584
15.307.1 Detailed Description	1587
15.307.2 Member Function Documentation	1587
15.307.2.1 foreach_available_event()	1587
15.307.2.2 process()	1588
15.308 L4Re::Util::Event_buffer_t< PAYLOAD > Class Template Reference	1589
15.308.1 Detailed Description	1592
15.308.2 Member Function Documentation	1592
15.308.2.1 attach()	1592
15.308.2.2 buf()	1592
15.308.2.3 detach()	1593
15.309 L4Re::Util::Event_svr< SVR > Class Template Reference	1593
15.309.1 Detailed Description	1595
15.310 L4Re::Util::Event_t< PAYLOAD > Class Template Reference	1595
15.310.1 Detailed Description	1596
15.310.2 Member Enumeration Documentation	1596
15.310.2.1 Mode	1596
15.310.3 Member Function Documentation	1596
15.310.3.1 buffer()	1596
15.310.3.2 init()	1597
15.310.3.3 init_poll()	1598
15.310.3.4 irq()	1598
15.311 L4Re::Util::Item_alloc_base Class Reference	1599
15.311.1 Detailed Description	1599
15.312 L4Re::Util::Names::Name Class Reference	1599
15.312.1 Detailed Description	1603

15.313 L4Re::Util::Names::Name_space Class Reference . . . . .	1603
15.313.1 Detailed Description . . . . .	1604
15.313.2 Member Function Documentation . . . . .	1604
15.313.2.1 alloc_dynamic_entry() . . . . .	1604
15.313.2.2 copy_receive_cap() . . . . .	1604
15.313.2.3 free_capability() . . . . .	1605
15.313.2.4 free_dynamic_entry() . . . . .	1605
15.313.2.5 free_epiface() . . . . .	1606
15.313.2.6 get_epiface() . . . . .	1607
15.314 L4Re::Util::Object_registry Class Reference . . . . .	1607
15.314.1 Detailed Description . . . . .	1610
15.314.2 Constructor & Destructor Documentation . . . . .	1610
15.314.2.1 Object_registry() [1/2] . . . . .	1610
15.314.2.2 Object_registry() [2/2] . . . . .	1610
15.314.3 Member Function Documentation . . . . .	1611
15.314.3.1 register_irq_obj() . . . . .	1611
15.314.3.2 register_obj() [1/3] . . . . .	1611
15.314.3.3 register_obj() [2/3] . . . . .	1612
15.314.3.4 register_obj() [3/3] . . . . .	1612
15.314.3.5 unregister_obj() . . . . .	1613
15.315 L4Re::Util::Ref_cap< T > Struct Template Reference . . . . .	1614
15.315.1 Detailed Description . . . . .	1614
15.316 L4Re::Util::Ref_del_cap< T > Struct Template Reference . . . . .	1615
15.316.1 Detailed Description . . . . .	1615
15.317 L4Re::Util::Registry_server< LOOP_HOOKS > Class Template Reference . . . . .	1616
15.317.1 Detailed Description . . . . .	1619
15.317.2 Constructor & Destructor Documentation . . . . .	1619
15.317.2.1 Registry_server() [1/3] . . . . .	1619
15.317.2.2 Registry_server() [2/3] . . . . .	1619
15.317.2.3 Registry_server() [3/3] . . . . .	1620
15.317.3 Member Function Documentation . . . . .	1620
15.317.3.1 loop() . . . . .	1620
15.318 L4Re::Util::Smart_cap_auto< Unmap_flags > Class Template Reference . . . . .	1621
15.318.1 Detailed Description . . . . .	1621
15.319 L4Re::Util::Smart_count_cap< Unmap_flags > Class Template Reference . . . . .	1622
15.319.1 Detailed Description . . . . .	1622
15.320 L4Re::Util::Vcon_svr< SVR > Class Template Reference . . . . .	1623
15.320.1 Detailed Description . . . . .	1623
15.321 L4Re::Util::Video::Goos_svr Class Reference . . . . .	1624
15.321.1 Detailed Description . . . . .	1625
15.321.2 Member Function Documentation . . . . .	1625
15.321.2.1 get_fb() . . . . .	1625

15.321.2.2 init_infos()	1626
15.321.2.3 refresh()	1626
15.321.2.4 screen_info()	1626
15.321.2.5 view_info()	1627
15.322 L4Re::Vfs::Be_file Class Reference	1627
15.322.1 Detailed Description	1629
15.322.2 Member Function Documentation	1630
15.322.2.1 data_space()	1630
15.322.2.2 fstat64()	1630
15.322.2.3 unlock_all_locks()	1630
15.323 L4Re::Vfs::Be_file_system Class Reference	1631
15.323.1 Detailed Description	1632
15.323.2 Constructor & Destructor Documentation	1633
15.323.2.1 Be_file_system()	1633
15.323.2.2 ~Be_file_system()	1633
15.323.3 Member Function Documentation	1633
15.323.3.1 type()	1633
15.324 L4Re::Vfs::Directory Class Reference	1634
15.324.1 Detailed Description	1635
15.324.2 Member Function Documentation	1636
15.324.2.1 faccessat()	1636
15.324.2.2 link()	1636
15.324.2.3 mkdir()	1636
15.324.2.4 rename()	1637
15.324.2.5 rmdir()	1637
15.324.2.6 symlink()	1638
15.324.2.7 unlink()	1638
15.325 L4Re::Vfs::File Class Reference	1639
15.325.1 Detailed Description	1641
15.326 L4Re::Vfs::File_system Class Reference	1642
15.326.1 Detailed Description	1643
15.326.2 Member Function Documentation	1643
15.326.2.1 mount()	1643
15.326.2.2 type()	1643
15.327 L4Re::Vfs::Fs Class Reference	1644
15.327.1 Detailed Description	1645
15.327.2 Member Function Documentation	1646
15.327.2.1 alloc_fd()	1646
15.327.2.2 free_fd()	1646
15.327.2.3 get_file()	1646
15.327.2.4 mount()	1647
15.327.2.5 set_fd()	1647

15.328 L4Re::Vfs::Generic_file Class Reference	1648
15.328.1 Detailed Description	1649
15.328.2 Member Function Documentation	1649
15.328.2.1 fchmod()	1649
15.328.2.2 fstat64()	1650
15.328.2.3 get_status_flags()	1650
15.328.2.4 set_status_flags()	1650
15.328.2.5 unlock_all_locks()	1651
15.329 L4Re::Vfs::Mman Class Reference	1651
15.329.1 Detailed Description	1653
15.330 L4Re::Vfs::Ops Class Reference	1653
15.330.1 Detailed Description	1656
15.331 L4Re::Vfs::Regular_file Class Reference	1656
15.331.1 Detailed Description	1658
15.331.2 Member Function Documentation	1659
15.331.2.1 data_space()	1659
15.331.2.2 fdatsync()	1659
15.331.2.3 fsync()	1659
15.331.2.4 ftruncate64()	1659
15.331.2.5 get_lock()	1660
15.331.2.6 lseek64()	1660
15.331.2.7 readv()	1661
15.331.2.8 set_lock()	1661
15.331.2.9 writev()	1661
15.332 L4Re::Vfs::Special_file Class Reference	1662
15.332.1 Detailed Description	1663
15.332.2 Member Function Documentation	1663
15.332.2.1 ioctl()	1663
15.333 L4Re::Video::Color_component Class Reference	1664
15.333.1 Detailed Description	1665
15.333.2 Constructor & Destructor Documentation	1665
15.333.2.1 Color_component()	1665
15.333.3 Member Function Documentation	1665
15.333.3.1 dump()	1665
15.333.3.2 get()	1665
15.333.3.3 operator==()	1666
15.333.3.4 set()	1666
15.333.3.5 shift()	1666
15.333.3.6 size()	1667
15.334 L4Re::Video::Goos Class Reference	1668
15.334.1 Detailed Description	1671
15.334.2 Member Enumeration Documentation	1671



15.334.2.1	Flags	1671
15.334.3	Member Function Documentation	1671
15.334.3.1	create_buffer()	1671
15.334.3.2	create_view()	1672
15.334.3.3	delete_buffer()	1672
15.334.3.4	delete_view()	1673
15.334.3.5	get_static_buffer()	1673
15.334.3.6	info()	1673
15.334.3.7	view()	1674
15.335	L4Re::Video::Goos::Info Struct Reference	1674
15.335.1	Detailed Description	1676
15.336	L4Re::Video::Pixel_info Class Reference	1676
15.336.1	Detailed Description	1677
15.336.2	Constructor & Destructor Documentation	1678
15.336.2.1	Pixel_info() [1/2]	1678
15.336.2.2	Pixel_info() [2/2]	1678
15.336.3	Member Function Documentation	1678
15.336.3.1	a() [1/2]	1678
15.336.3.2	a() [2/2]	1679
15.336.3.3	b() [1/2]	1679
15.336.3.4	b() [2/2]	1679
15.336.3.5	bits_per_pixel()	1680
15.336.3.6	bytes_per_pixel() [1/2]	1680
15.336.3.7	bytes_per_pixel() [2/2]	1680
15.336.3.8	dump()	1681
15.336.3.9	g() [1/2]	1681
15.336.3.10	g() [2/2]	1681
15.336.3.11	has_alpha()	1682
15.336.3.12	operator==(())	1682
15.336.3.13	padding()	1683
15.336.3.14	r() [1/2]	1683
15.336.3.15	r() [2/2]	1683
15.337	L4Re::Video::View Class Reference	1684
15.337.1	Detailed Description	1685
15.337.2	Member Enumeration Documentation	1685
15.337.2.1	Flags	1685
15.337.2.2	V_flags	1686
15.337.3	Member Function Documentation	1686
15.337.3.1	info()	1686
15.337.3.2	refresh()	1686
15.337.3.3	set_info()	1687
15.337.3.4	set_viewport()	1687

15.337.3.5 stack()	1688
15.338 L4Re::Video::View::Info Struct Reference	1688
15.338.1 Detailed Description	1690
15.339 l4re_aux_t Struct Reference	1691
15.339.1 Detailed Description	1691
15.340 l4re_ds_stats_t Struct Reference	1692
15.340.1 Detailed Description	1692
15.341 l4re_elf_aux_mword_t Struct Reference	1692
15.341.1 Detailed Description	1693
15.342 l4re_elf_aux_t Struct Reference	1693
15.342.1 Detailed Description	1693
15.343 l4re_elf_aux_vma_t Struct Reference	1693
15.343.1 Detailed Description	1694
15.344 l4re_env_cap_entry_t Struct Reference	1694
15.344.1 Detailed Description	1695
15.344.2 Constructor & Destructor Documentation	1695
15.344.2.1 l4re_env_cap_entry_t()	1695
15.344.3 Field Documentation	1695
15.344.3.1 flags	1695
15.345 l4re_env_t Struct Reference	1696
15.345.1 Detailed Description	1697
15.345.2 Field Documentation	1697
15.345.2.1 caps	1697
15.346 l4re_event_t Struct Reference	1698
15.346.1 Detailed Description	1698
15.347 l4re_video_color_component_t Struct Reference	1699
15.347.1 Detailed Description	1699
15.348 l4re_video_goos_info_t Struct Reference	1699
15.348.1 Detailed Description	1701
15.349 l4re_video_pixel_info_t Struct Reference	1701
15.349.1 Detailed Description	1702
15.350 l4re_video_view_info_t Struct Reference	1702
15.350.1 Detailed Description	1703
15.351 l4re_video_view_t Struct Reference	1703
15.351.1 Detailed Description	1704
15.352 l4shmc_ringbuf_head_t Struct Reference	1704
15.352.1 Detailed Description	1705
15.353 l4shmc_ringbuf_t Struct Reference	1705
15.353.1 Detailed Description	1706
15.354 l4util_idt_desc_t Struct Reference	1706
15.354.1 Detailed Description	1707
15.355 l4util_idt_header_t Struct Reference	1707

15.355.1 Detailed Description . . . . .	1708
15.356 I4util_I4mod_info Struct Reference . . . . .	1708
15.356.1 Detailed Description . . . . .	1709
15.356.2 Field Documentation . . . . .	1709
15.356.2.1 vbe_ctrl_info . . . . .	1709
15.357 I4util_I4mod_mod Struct Reference . . . . .	1709
15.357.1 Detailed Description . . . . .	1710
15.358 I4util_mb_addr_range_t Struct Reference . . . . .	1710
15.358.1 Detailed Description . . . . .	1711
15.359 I4util_mb_apm_t Struct Reference . . . . .	1711
15.359.1 Detailed Description . . . . .	1711
15.360 I4util_mb_drive_t Struct Reference . . . . .	1712
15.360.1 Detailed Description . . . . .	1712
15.360.2 Field Documentation . . . . .	1713
15.360.2.1 drive_cylinders . . . . .	1713
15.360.2.2 drive_mode . . . . .	1713
15.360.2.3 drive_number . . . . .	1713
15.361 I4util_mb_info_t Struct Reference . . . . .	1714
15.361.1 Detailed Description . . . . .	1715
15.362 I4util_mb_mod_t Struct Reference . . . . .	1715
15.362.1 Detailed Description . . . . .	1716
15.363 I4util_mb_vbe_ctrl_t Struct Reference . . . . .	1716
15.363.1 Detailed Description . . . . .	1717
15.364 I4util_mb_vbe_mode_t Struct Reference . . . . .	1717
15.364.1 Detailed Description . . . . .	1720
15.365 L4vbus::Device Class Reference . . . . .	1721
15.365.1 Detailed Description . . . . .	1723
15.365.2 Constructor & Destructor Documentation . . . . .	1723
15.365.2.1 Device() . . . . .	1723
15.365.3 Member Function Documentation . . . . .	1724
15.365.3.1 bus_cap() . . . . .	1724
15.365.3.2 dev_handle() . . . . .	1725
15.365.3.3 device() . . . . .	1725
15.365.3.4 device_by_hid() . . . . .	1726
15.365.3.5 get_resource() . . . . .	1727
15.365.3.6 is_compatible() . . . . .	1727
15.365.3.7 next_device() . . . . .	1728
15.365.3.8 operator!=(()) . . . . .	1729
15.365.3.9 operator==(()) . . . . .	1729
15.366 L4vbus::Gpio_module Class Reference . . . . .	1730
15.366.1 Detailed Description . . . . .	1733
15.366.2 Member Function Documentation . . . . .	1733

15.366.2.1 config_pad()	1733
15.366.2.2 get()	1733
15.366.2.3 pin()	1734
15.366.2.4 set()	1735
15.366.2.5 setup()	1736
15.367 L4vbus::Gpio_module::Pin_slice Struct Reference	1737
15.367.1 Detailed Description	1737
15.368 L4vbus::Gpio_pin Class Reference	1737
15.368.1 Detailed Description	1741
15.368.2 Member Function Documentation	1741
15.368.2.1 config_get()	1741
15.368.2.2 config_pad()	1742
15.368.2.3 config_pull()	1742
15.368.2.4 get()	1743
15.368.2.5 pin()	1743
15.368.2.6 set()	1743
15.368.2.7 setup()	1744
15.368.2.8 to_irq()	1745
15.369 L4vbus::Icu Class Reference	1746
15.369.1 Detailed Description	1749
15.369.2 Member Enumeration Documentation	1749
15.369.2.1 Src_types	1749
15.369.3 Member Function Documentation	1749
15.369.3.1 vicu()	1749
15.370 L4vbus::Pci_dev Class Reference	1750
15.370.1 Detailed Description	1753
15.370.2 Member Function Documentation	1754
15.370.2.1 cfg_read()	1754
15.370.2.2 cfg_write()	1754
15.370.2.3 irq_enable()	1755
15.371 L4vbus::Pci_host_bridge Class Reference	1756
15.371.1 Detailed Description	1760
15.371.2 Member Function Documentation	1760
15.371.2.1 cfg_read()	1760
15.371.2.2 cfg_write()	1761
15.371.2.3 irq_enable()	1761
15.372 L4vbus::Pm< DEC > Class Template Reference	1762
15.372.1 Detailed Description	1764
15.372.2 Member Function Documentation	1764
15.372.2.1 pm_resume()	1764
15.372.2.2 pm_suspend()	1764
15.373 L4vbus::Vbus Class Reference	1765

15.373.1 Detailed Description . . . . .	1772
15.373.2 Member Function Documentation . . . . .	1772
15.373.2.1 assign_dma_domain() [1/2] . . . . .	1772
15.373.2.2 assign_dma_domain() [2/2] . . . . .	1773
15.373.2.3 release_ioport() . . . . .	1774
15.373.2.4 request_ioport() . . . . .	1774
15.373.2.5 root() . . . . .	1775
15.374 l4vbus_device_t Struct Reference . . . . .	1776
15.374.1 Detailed Description . . . . .	1776
15.375 l4vbus_resource_t Struct Reference . . . . .	1777
15.375.1 Detailed Description . . . . .	1777
15.376 L4vcpu::State Class Reference . . . . .	1778
15.376.1 Detailed Description . . . . .	1778
15.376.2 Constructor & Destructor Documentation . . . . .	1778
15.376.2.1 State() . . . . .	1778
15.376.3 Member Function Documentation . . . . .	1779
15.376.3.1 add() . . . . .	1779
15.376.3.2 clear() . . . . .	1779
15.376.3.3 set() . . . . .	1779
15.377 L4vcpu::Vcpu Class Reference . . . . .	1780
15.377.1 Detailed Description . . . . .	1783
15.377.2 Member Function Documentation . . . . .	1783
15.377.2.1 cast() [1/2] . . . . .	1783
15.377.2.2 cast() [2/2] . . . . .	1783
15.377.2.3 entry_ip() . . . . .	1783
15.377.2.4 entry_sp() . . . . .	1784
15.377.2.5 ext_alloc() . . . . .	1784
15.377.2.6 i() [1/2] . . . . .	1785
15.377.2.7 i() [2/2] . . . . .	1785
15.377.2.8 irq_disable_save() . . . . .	1785
15.377.2.9 irq_enable() . . . . .	1786
15.377.2.10 irq_restore() . . . . .	1786
15.377.2.11 is_irq_entry() . . . . .	1787
15.377.2.12 is_page_fault_entry() . . . . .	1787
15.377.2.13 r() [1/2] . . . . .	1788
15.377.2.14 r() [2/2] . . . . .	1788
15.377.2.15 saved_state() [1/2] . . . . .	1788
15.377.2.16 saved_state() [2/2] . . . . .	1788
15.377.2.17 state() [1/2] . . . . .	1789
15.377.2.18 state() [2/2] . . . . .	1789
15.377.2.19 task() . . . . .	1789
15.377.2.20 wait_for_event() . . . . .	1790

15.378 L4virtio::Device Class Reference . . . . .	1791
15.378.1 Detailed Description . . . . .	1795
15.378.2 Member Function Documentation . . . . .	1795
15.378.2.1 config_queue() . . . . .	1795
15.378.2.2 device_config() . . . . .	1796
15.378.2.3 device_notification_irq() . . . . .	1797
15.378.2.4 register_ds() . . . . .	1797
15.378.2.5 set_status() . . . . .	1798
15.379 L4virtio::Driver::Block_device Class Reference . . . . .	1799
15.379.1 Detailed Description . . . . .	1803
15.379.2 Member Function Documentation . . . . .	1803
15.379.2.1 add_block() . . . . .	1803
15.379.2.2 process_request() . . . . .	1804
15.379.2.3 process_used_queue() . . . . .	1805
15.379.2.4 send_request() . . . . .	1805
15.379.2.5 setup_device() . . . . .	1806
15.379.2.6 start_request() . . . . .	1808
15.380 L4virtio::Driver::Block_device::Handle Class Reference . . . . .	1808
15.380.1 Detailed Description . . . . .	1809
15.381 L4virtio::Driver::Device Class Reference . . . . .	1809
15.381.1 Detailed Description . . . . .	1812
15.381.2 Member Function Documentation . . . . .	1812
15.381.2.1 bind_notification_irq() . . . . .	1812
15.381.2.2 config_queue() . . . . .	1813
15.381.2.3 driver_acknowledge() . . . . .	1814
15.381.2.4 driver_connect() . . . . .	1815
15.381.2.5 feature_negotiated() . . . . .	1816
15.381.2.6 max_queue_size() . . . . .	1817
15.381.2.7 register_ds() . . . . .	1818
15.381.2.8 send() . . . . .	1819
15.381.2.9 send_and_wait() . . . . .	1819
15.381.2.10 wait() . . . . .	1820
15.381.2.11 wait_for_next_used() . . . . .	1821
15.382 L4virtio::Driver::Virtio_net_device Class Reference . . . . .	1822
15.382.1 Detailed Description . . . . .	1826
15.382.2 Member Function Documentation . . . . .	1826
15.382.2.1 finish_rx() . . . . .	1826
15.382.2.2 rx_pkt() . . . . .	1826
15.382.2.3 rx_queue_size() . . . . .	1827
15.382.2.4 setup_device() . . . . .	1827
15.382.2.5 tx() . . . . .	1828
15.382.2.6 tx_queue_size() . . . . .	1829

15.382.2.7 wait_rx()	1830
15.383 L4virtio::Driver::Virtio_net_device::Packet Struct Reference	1831
15.383.1 Detailed Description	1831
15.384 L4virtio::Driver::Virtqueue Class Reference	1831
15.384.1 Detailed Description	1835
15.384.2 Member Function Documentation	1836
15.384.2.1 alloc_descriptor()	1836
15.384.2.2 desc()	1836
15.384.2.3 enqueue_descriptor()	1837
15.384.2.4 find_next_used()	1838
15.384.2.5 free_descriptor()	1838
15.384.2.6 init_queue() [1/2]	1839
15.384.2.7 init_queue() [2/2]	1840
15.384.2.8 initialize_rings()	1841
15.385 L4virtio::Ptr< T > Class Template Reference	1842
15.385.1 Detailed Description	1844
15.385.2 Member Enumeration Documentation	1844
15.385.2.1 Invalid_type	1844
15.385.3 Member Function Documentation	1844
15.385.3.1 get()	1844
15.385.3.2 is_valid()	1845
15.386 L4virtio::Svr::Bad_descriptor Struct Reference	1846
15.386.1 Detailed Description	1847
15.386.2 Member Enumeration Documentation	1847
15.386.2.1 Error	1847
15.386.3 Constructor & Destructor Documentation	1847
15.386.3.1 Bad_descriptor()	1847
15.386.4 Member Function Documentation	1847
15.386.4.1 message()	1847
15.387 L4virtio::Svr::Block_dev_base< Ds_data > Class Template Reference	1848
15.387.1 Detailed Description	1852
15.387.2 Constructor & Destructor Documentation	1852
15.387.2.1 Block_dev_base()	1852
15.387.3 Member Function Documentation	1853
15.387.3.1 finalize_request()	1853
15.387.3.2 get_writeback()	1854
15.387.3.3 set_blk_size()	1854
15.387.3.4 set_config_wce()	1854
15.387.3.5 set_discard()	1855
15.387.3.6 set_size_max()	1855
15.387.3.7 set_topology()	1855
15.387.3.8 set_write_zeroes()	1856

15.388 L4virtio::Svr::Block_request< Ds_data > Class Template Reference . . . . .	1856
15.388.1 Detailed Description . . . . .	1857
15.388.2 Member Function Documentation . . . . .	1857
15.388.2.1 data_size() . . . . .	1857
15.388.2.2 next_block() . . . . .	1858
15.389 L4virtio::Svr::Console::Control_message Struct Reference . . . . .	1859
15.389.1 Detailed Description . . . . .	1860
15.389.2 Member Enumeration Documentation . . . . .	1860
15.389.2.1 Events . . . . .	1860
15.390 L4virtio::Svr::Console::Control_request Struct Reference . . . . .	1860
15.390.1 Detailed Description . . . . .	1861
15.391 L4virtio::Svr::Console::Device Class Reference . . . . .	1862
15.391.1 Detailed Description . . . . .	1866
15.391.2 Constructor & Destructor Documentation . . . . .	1866
15.391.2.1 Device() [1/3] . . . . .	1866
15.391.2.2 Device() [2/3] . . . . .	1867
15.391.2.3 Device() [3/3] . . . . .	1867
15.391.3 Member Function Documentation . . . . .	1868
15.391.3.1 do_control_work() . . . . .	1868
15.391.3.2 notify_queue() . . . . .	1869
15.391.3.3 port() . . . . .	1869
15.391.3.4 port_read() . . . . .	1870
15.391.3.5 port_write() . . . . .	1871
15.391.3.6 process_device_ready() . . . . .	1872
15.391.3.7 process_port_open() . . . . .	1873
15.391.3.8 process_port_ready() . . . . .	1874
15.392 L4virtio::Svr::Console::Device_port Struct Reference . . . . .	1874
15.392.1 Detailed Description . . . . .	1877
15.393 L4virtio::Svr::Console::Features Struct Reference . . . . .	1878
15.393.1 Detailed Description . . . . .	1880
15.393.2 Member Typedef Documentation . . . . .	1880
15.393.2.1 console_multiport_bfm_t . . . . .	1880
15.393.2.2 console_size_bfm_t . . . . .	1880
15.393.2.3 emerg_write_bfm_t . . . . .	1880
15.394 L4virtio::Svr::Console::Port Struct Reference . . . . .	1881
15.394.1 Detailed Description . . . . .	1883
15.394.2 Member Enumeration Documentation . . . . .	1883
15.394.2.1 Port_status . . . . .	1883
15.395 L4virtio::Svr::Console::Virtio_con Class Reference . . . . .	1884
15.395.1 Detailed Description . . . . .	1888
15.395.2 Constructor & Destructor Documentation . . . . .	1888
15.395.2.1 Virtio_con() . . . . .	1888



15.395.3 Member Function Documentation . . . . .	1889
15.395.3.1 handle_control_message() . . . . .	1889
15.395.3.2 notify_queue() . . . . .	1890
15.395.3.3 port() . . . . .	1891
15.395.3.4 port_add() . . . . .	1891
15.395.3.5 port_open() . . . . .	1893
15.395.3.6 port_remove() . . . . .	1894
15.395.3.7 process_device_ready() . . . . .	1895
15.395.3.8 process_port_open() . . . . .	1896
15.395.3.9 process_port_ready() . . . . .	1896
15.395.3.10 reset_device() . . . . .	1897
15.395.3.11 send_control_message() . . . . .	1898
15.396 L4virtio::Svr::Data_buffer Struct Reference . . . . .	1899
15.396.1 Detailed Description . . . . .	1900
15.396.2 Constructor & Destructor Documentation . . . . .	1901
15.396.2.1 Data_buffer() . . . . .	1901
15.396.3 Member Function Documentation . . . . .	1901
15.396.3.1 copy_to() . . . . .	1901
15.396.3.2 done() . . . . .	1902
15.396.3.3 set() . . . . .	1902
15.396.3.4 skip() . . . . .	1903
15.397 L4virtio::Svr::Dev_config Class Reference . . . . .	1903
15.397.1 Detailed Description . . . . .	1905
15.397.2 Constructor & Destructor Documentation . . . . .	1905
15.397.2.1 Dev_config() [1/2] . . . . .	1905
15.397.2.2 Dev_config() [2/2] . . . . .	1906
15.397.3 Member Function Documentation . . . . .	1906
15.397.3.1 add_irq_status() . . . . .	1906
15.397.3.2 change_queue_config() . . . . .	1907
15.397.3.3 ds() . . . . .	1908
15.397.3.4 get_cmd() . . . . .	1908
15.397.3.5 guest_features() . . . . .	1908
15.397.3.6 hdr() . . . . .	1909
15.397.3.7 negotiated_features() . . . . .	1910
15.397.3.8 qconfig() . . . . .	1910
15.397.3.9 reset_cmd() . . . . .	1911
15.397.3.10 reset_queue() . . . . .	1912
15.397.3.11 set_device_needs_reset() . . . . .	1913
15.397.3.12 set_status() . . . . .	1914
15.397.3.13 status() . . . . .	1915
15.398 L4virtio::Svr::Dev_features Struct Reference . . . . .	1916
15.398.1 Detailed Description . . . . .	1917

15.399 L4virtio::Svr::Dev_status Struct Reference . . . . .	1918
15.399.1 Detailed Description . . . . .	1919
15.399.2 Member Function Documentation . . . . .	1919
15.399.2.1 running() . . . . .	1919
15.400 L4virtio::Svr::Device_t< DATA > Class Template Reference . . . . .	1919
15.400.1 Detailed Description . . . . .	1922
15.400.2 Member Function Documentation . . . . .	1922
15.400.2.1 add_trusted_dataspaces() . . . . .	1922
15.400.2.2 device_error() . . . . .	1923
15.400.2.3 device_notify_irq() . . . . .	1923
15.400.2.4 handle_mem_cmd_write() . . . . .	1924
15.400.2.5 init_mem_info() . . . . .	1924
15.400.2.6 register_driver_irq() . . . . .	1924
15.400.2.7 reset_queue_config() . . . . .	1925
15.400.2.8 setup_queue() . . . . .	1926
15.401 L4virtio::Svr::Driver_mem_list_t< DATA > Class Template Reference . . . . .	1927
15.401.1 Detailed Description . . . . .	1930
15.401.2 Member Function Documentation . . . . .	1930
15.401.2.1 add() . . . . .	1930
15.401.2.2 find() . . . . .	1931
15.401.2.3 full() . . . . .	1931
15.401.2.4 init() . . . . .	1932
15.401.2.5 load_desc() [ 1 / 3 ] . . . . .	1932
15.401.2.6 load_desc() [ 2 / 3 ] . . . . .	1933
15.401.2.7 load_desc() [ 3 / 3 ] . . . . .	1934
15.401.2.8 remove() . . . . .	1935
15.402 L4virtio::Svr::Driver_mem_region_t< DATA > Class Template Reference . . . . .	1936
15.402.1 Detailed Description . . . . .	1937
15.402.2 Constructor & Destructor Documentation . . . . .	1938
15.402.2.1 Driver_mem_region_t() . . . . .	1938
15.402.3 Member Function Documentation . . . . .	1939
15.402.3.1 contains() . . . . .	1939
15.402.3.2 drv_base() . . . . .	1940
15.402.3.3 ds() . . . . .	1940
15.402.3.4 ds_offset() . . . . .	1941
15.402.3.5 empty() . . . . .	1941
15.402.3.6 flags() . . . . .	1941
15.402.3.7 is_writable() . . . . .	1941
15.402.3.8 local() . . . . .	1941
15.402.3.9 local_base() . . . . .	1942
15.402.3.10 size() . . . . .	1943
15.403 L4virtio::Svr::Request_processor Class Reference . . . . .	1943

15.403.1 Detailed Description . . . . .	1944
15.403.2 Member Function Documentation . . . . .	1945
15.403.2.1 current_flags() . . . . .	1945
15.403.2.2 has_more() . . . . .	1945
15.403.2.3 next() . . . . .	1946
15.403.2.4 start() [1/2] . . . . .	1948
15.403.2.5 start() [2/2] . . . . .	1949
15.404 L4virtio::Svr::Scmi::Base_attr_t Struct Reference . . . . .	1950
15.404.1 Detailed Description . . . . .	1951
15.405 L4virtio::Svr::Scmi::Base_proto Class Reference . . . . .	1951
15.405.1 Detailed Description . . . . .	1953
15.406 L4virtio::Svr::Scmi::Perf_proto Class Reference . . . . .	1953
15.406.1 Detailed Description . . . . .	1955
15.407 L4virtio::Svr::Scmi::Performance_attr_t Struct Reference . . . . .	1956
15.407.1 Detailed Description . . . . .	1956
15.408 L4virtio::Svr::Scmi::Performance_describe_level_t Struct Reference . . . . .	1956
15.408.1 Detailed Description . . . . .	1957
15.409 L4virtio::Svr::Scmi::Performance_describe_levels_n_t Struct Reference . . . . .	1957
15.409.1 Detailed Description . . . . .	1958
15.410 L4virtio::Svr::Scmi::Performance_domain_attr_t Struct Reference . . . . .	1958
15.410.1 Detailed Description . . . . .	1959
15.411 L4virtio::Svr::Scmi::Proto< OBSERV > Struct Template Reference . . . . .	1959
15.411.1 Detailed Description . . . . .	1960
15.412 L4virtio::Svr::Scmi::Scmi_dev Class Reference . . . . .	1960
15.412.1 Detailed Description . . . . .	1963
15.413 L4virtio::Svr::Scmi::Scmi_hdr_t Struct Reference . . . . .	1964
15.413.1 Detailed Description . . . . .	1964
15.414 L4virtio::Svr::Virtqueue Class Reference . . . . .	1964
15.414.1 Detailed Description . . . . .	1968
15.414.2 Member Function Documentation . . . . .	1969
15.414.2.1 consumed() [1/2] . . . . .	1969
15.414.2.2 consumed() [2/2] . . . . .	1969
15.414.2.3 desc() . . . . .	1970
15.414.2.4 desc_avail() . . . . .	1971
15.414.2.5 disable_notify() . . . . .	1971
15.414.2.6 enable_notify() . . . . .	1972
15.414.2.7 finish() [1/2] . . . . .	1972
15.414.2.8 finish() [2/2] . . . . .	1973
15.414.2.9 next_avail() . . . . .	1974
15.415 L4virtio::Svr::Virtqueue::Head_desc Class Reference . . . . .	1975
15.415.1 Detailed Description . . . . .	1976
15.415.2 Member Function Documentation . . . . .	1976

15.415.2.1 desc()	1976
15.415.2.2 operator Null_ptr_check const *()	1976
15.415.2.3 valid()	1977
15.416 L4virtio::Virtqueue Class Reference	1977
15.416.1 Detailed Description	1981
15.416.2 Member Function Documentation	1981
15.416.2.1 avail_align()	1981
15.416.2.2 avail_size()	1981
15.416.2.3 desc_align()	1982
15.416.2.4 desc_size()	1983
15.416.2.5 disable()	1984
15.416.2.6 dump()	1984
15.416.2.7 get_avail_idx()	1985
15.416.2.8 get_tail_avail_idx()	1985
15.416.2.9 no_notify_guest()	1985
15.416.2.10 no_notify_host() [1/2]	1986
15.416.2.11 no_notify_host() [2/2]	1987
15.416.2.12 num()	1988
15.416.2.13 ready()	1988
15.416.2.14 setup()	1989
15.416.2.15 setup_simple()	1990
15.416.2.16 total_size() [1/2]	1991
15.416.2.17 total_size() [2/2]	1992
15.416.2.18 used_align()	1993
15.416.2.19 used_size()	1993
15.417 L4virtio::Virtqueue::Avail Class Reference	1995
15.417.1 Detailed Description	1996
15.418 L4virtio::Virtqueue::Avail::Flags Struct Reference	1996
15.418.1 Detailed Description	1996
15.418.2 Member Typedef Documentation	1997
15.418.2.1 no_irq_bfm_t	1997
15.419 L4virtio::Virtqueue::Desc Class Reference	1997
15.419.1 Detailed Description	1999
15.420 L4virtio::Virtqueue::Desc::Flags Struct Reference	1999
15.420.1 Detailed Description	2000
15.420.2 Member Typedef Documentation	2000
15.420.2.1 indirect_bfm_t	2000
15.420.2.2 next_bfm_t	2000
15.420.2.3 write_bfm_t	2000
15.421 L4virtio::Virtqueue::Used Class Reference	2001
15.421.1 Detailed Description	2001
15.422 L4virtio::Virtqueue::Used::Flags Struct Reference	2002

15.422.1 Detailed Description . . . . .	2002
15.422.2 Member Typedef Documentation . . . . .	2002
15.422.2.1 no_notify_bfm_t . . . . .	2002
15.423 L4virtio::Virtqueue::Used_elem Struct Reference . . . . .	2003
15.423.1 Detailed Description . . . . .	2003
15.423.2 Constructor & Destructor Documentation . . . . .	2003
15.423.2.1 Used_elem() . . . . .	2003
15.424 l4virtio_block_config_t Struct Reference . . . . .	2004
15.424.1 Detailed Description . . . . .	2005
15.425 l4virtio_block_discard_t Struct Reference . . . . .	2005
15.425.1 Detailed Description . . . . .	2005
15.426 l4virtio_block_header_t Struct Reference . . . . .	2006
15.426.1 Detailed Description . . . . .	2006
15.427 l4virtio_config_hdr_t Struct Reference . . . . .	2007
15.427.1 Detailed Description . . . . .	2007
15.428 l4virtio_config_queue_t Struct Reference . . . . .	2008
15.428.1 Detailed Description . . . . .	2009
15.429 l4virtio_input_absinfo_t Struct Reference . . . . .	2009
15.429.1 Detailed Description . . . . .	2009
15.430 l4virtio_input_config_t Struct Reference . . . . .	2010
15.430.1 Detailed Description . . . . .	2010
15.431 l4virtio_input_devids_t Struct Reference . . . . .	2010
15.431.1 Detailed Description . . . . .	2011
15.432 l4virtio_input_event_t Struct Reference . . . . .	2011
15.432.1 Detailed Description . . . . .	2011
15.433 l4virtio_net_config_t Struct Reference . . . . .	2011
15.433.1 Detailed Description . . . . .	2012
15.434 l4virtio_net_header_t Struct Reference . . . . .	2012
15.434.1 Detailed Description . . . . .	2012
<b>16 File Documentation</b>	<b>2013</b>
16.1 asm_access.h . . . . .	2013
16.2 asm_access.h . . . . .	2013
16.3 asm_access.h . . . . .	2014
16.4 asm_access.h . . . . .	2015
16.5 asm_access.h . . . . .	2015
16.6 asm_access.h . . . . .	2016
16.7 asm_access.h . . . . .	2017
16.8 asm_access.h . . . . .	2017
16.9 asm_access_gen.h . . . . .	2018
16.10 hw_mmio_register_block . . . . .	2018
16.11 hw_register_block . . . . .	2019

16.12 io_regblock.h . . . . .	2025
16.13 io_regblock_port.h . . . . .	2027
16.14 Makefile . . . . .	2028
16.15 Makefile . . . . .	2028
16.16 Makefile . . . . .	2028
16.17 poll_timeout_counter.h . . . . .	2028
16.18 uart_16550.h . . . . .	2029
16.19 uart_16550_dw.h . . . . .	2030
16.20 uart_apb.h . . . . .	2030
16.21 uart_base.h . . . . .	2031
16.22 uart_cadence.h . . . . .	2032
16.23 uart_dcc-v6.h . . . . .	2032
16.24 uart_dm.h . . . . .	2033
16.25 uart_dummy.h . . . . .	2033
16.26 uart_geni.h . . . . .	2034
16.27 uart_imx.h . . . . .	2034
16.28 uart_leon3.h . . . . .	2035
16.29 uart_linflex.h . . . . .	2036
16.30 uart_lpuart.h . . . . .	2036
16.31 uart_mvebu.h . . . . .	2037
16.32 uart_of.h . . . . .	2037
16.33 uart_omap35x.h . . . . .	2038
16.34 uart_pl011.h . . . . .	2038
16.35 uart_s3c2410.h . . . . .	2039
16.36 uart_sa1000.h . . . . .	2040
16.37 uart_sbi.h . . . . .	2041
16.38 uart_sh.h . . . . .	2041
16.39 cmd_control . . . . .	2041
16.40 amd64/l4/util/idt.h File Reference . . . . .	2042
16.40.1 Detailed Description . . . . .	2043
16.41 idt.h . . . . .	2043
16.42 x86/l4/util/idt.h File Reference . . . . .	2044
16.42.1 Detailed Description . . . . .	2045
16.43 idt.h . . . . .	2045
16.44 amd64/l4/util/perform.h File Reference . . . . .	2046
16.44.1 Detailed Description . . . . .	2047
16.45 perform.h . . . . .	2047
16.46 x86/l4/util/perform.h File Reference . . . . .	2052
16.46.1 Detailed Description . . . . .	2053
16.47 perform.h . . . . .	2053
16.48 amd64/l4/util/rdtsc.h File Reference . . . . .	2058
16.48.1 Detailed Description . . . . .	2060

16.49 rdtsc.h . . . . .	2060
16.50 x86/i4/util/rdtsc.h File Reference . . . . .	2062
16.50.1 Detailed Description . . . . .	2064
16.51 rdtsc.h . . . . .	2064
16.52 amd64/i4/util/spin.h File Reference . . . . .	2067
16.52.1 Detailed Description . . . . .	2067
16.53 spin.h . . . . .	2067
16.54 x86/i4/util/spin.h File Reference . . . . .	2068
16.54.1 Detailed Description . . . . .	2068
16.55 spin.h . . . . .	2069
16.56 amd64/i4/sys/segment.h File Reference . . . . .	2069
16.56.1 Detailed Description . . . . .	2071
16.56.2 Enumeration Type Documentation . . . . .	2071
16.56.2.1 L4_sys_segment . . . . .	2071
16.56.2.2 L4_task_ldt_x86_consts . . . . .	2071
16.56.3 Function Documentation . . . . .	2071
16.56.3.1 fiasco_amd64_segment_info() . . . . .	2071
16.56.3.2 fiasco_amd64_set_fs() . . . . .	2072
16.56.3.3 fiasco_amd64_set_segment_base() . . . . .	2073
16.57 segment.h . . . . .	2074
16.58 amd64/i4f/i4/sys/segment.h File Reference . . . . .	2075
16.58.1 Detailed Description . . . . .	2076
16.58.2 Function Documentation . . . . .	2076
16.58.2.1 fiasco_amd64_set_fs() . . . . .	2076
16.58.2.2 fiasco_amd64_set_segment_base() . . . . .	2077
16.59 segment.h . . . . .	2078
16.60 x86/i4/sys/segment.h File Reference . . . . .	2079
16.60.1 Detailed Description . . . . .	2080
16.60.2 Enumeration Type Documentation . . . . .	2080
16.60.2.1 L4_task_ldt_x86_consts . . . . .	2080
16.61 segment.h . . . . .	2080
16.62 x86/i4f/i4/sys/segment.h File Reference . . . . .	2081
16.62.1 Detailed Description . . . . .	2082
16.63 segment.h . . . . .	2082
16.64 amd64/i4/util/port_io.h File Reference . . . . .	2083
16.64.1 Detailed Description . . . . .	2083
16.65 port_io.h . . . . .	2084
16.66 amd64/i4f/i4/util/port_io.h File Reference . . . . .	2084
16.66.1 Detailed Description . . . . .	2084
16.67 port_io.h . . . . .	2084
16.68 x86/i4/util/port_io.h File Reference . . . . .	2085
16.68.1 Detailed Description . . . . .	2086

16.69	<a href="#">port_io.h</a>	2086
16.70	<a href="#">x86/l4/l4/util/port_io.h File Reference</a>	2088
16.70.1	Detailed Description	2089
16.70.2	Function Documentation	2090
16.70.2.1	<a href="#">l4util_ioport_map()</a>	2090
16.71	<a href="#">port_io.h</a>	2091
16.72	<a href="#">__kip-arch.h</a>	2091
16.73	<a href="#">__kip-arch.h</a>	2092
16.74	<a href="#">__kip-arch.h</a>	2092
16.75	<a href="#">amd64/l4/sys/__vcpu-arch.h File Reference</a>	2092
16.75.1	Detailed Description	2094
16.75.2	Enumeration Type Documentation	2094
16.75.2.1	anonymous enum	2094
16.76	<a href="#">__vcpu-arch.h</a>	2094
16.77	<a href="#">arm/l4/sys/__vcpu-arch.h File Reference</a>	2095
16.77.1	Detailed Description	2097
16.77.2	Enumeration Type Documentation	2097
16.77.2.1	anonymous enum	2097
16.77.2.2	<a href="#">L4_vcpu_e_field_ids</a>	2097
16.78	<a href="#">__vcpu-arch.h</a>	2097
16.79	<a href="#">x86/l4/sys/__vcpu-arch.h File Reference</a>	2099
16.79.1	Detailed Description	2100
16.79.2	Enumeration Type Documentation	2100
16.79.2.1	anonymous enum	2100
16.80	<a href="#">__vcpu-arch.h</a>	2100
16.81	<a href="#">ktrace_events.h</a>	2101
16.82	<a href="#">ktrace_events.h</a>	2104
16.83	<a href="#">ktrace_events.h</a>	2107
16.84	<a href="#">amd64/l4/sys/linkage.h File Reference</a>	2111
16.84.1	Detailed Description	2111
16.85	<a href="#">linkage.h</a>	2111
16.86	<a href="#">arm/l4/sys/linkage.h File Reference</a>	2112
16.86.1	Detailed Description	2112
16.87	<a href="#">linkage.h</a>	2112
16.88	<a href="#">x86/l4/sys/linkage.h File Reference</a>	2112
16.88.1	Detailed Description	2113
16.89	<a href="#">linkage.h</a>	2113
16.90	<a href="#">arm/l4/sys/mem_op.h File Reference</a>	2113
16.90.1	Detailed Description	2114
16.91	<a href="#">mem_op.h</a>	2115
16.92	<a href="#">vm.h</a>	2116
16.93	<a href="#">arm/l4/sys/vm.h File Reference</a>	2116



16.93.1 Detailed Description . . . . .	2116
16.94 vm.h . . . . .	2117
16.95 vm.h . . . . .	2118
16.96 amd64/l4/util/bitops_arch.h File Reference . . . . .	2118
16.96.1 Detailed Description . . . . .	2118
16.97 bitops_arch.h . . . . .	2118
16.98 arm/l4/util/bitops_arch.h File Reference . . . . .	2121
16.98.1 Detailed Description . . . . .	2122
16.99 bitops_arch.h . . . . .	2122
16.100 x86/l4/util/bitops_arch.h File Reference . . . . .	2122
16.100.1 Detailed Description . . . . .	2122
16.101 bitops_arch.h . . . . .	2122
16.102 amd64/l4/util/cpu.h File Reference . . . . .	2125
16.102.1 Detailed Description . . . . .	2126
16.103 cpu.h . . . . .	2127
16.104 arm/l4/util/cpu.h File Reference . . . . .	2128
16.104.1 Detailed Description . . . . .	2128
16.105 cpu.h . . . . .	2128
16.106 x86/l4/util/cpu.h File Reference . . . . .	2128
16.106.1 Detailed Description . . . . .	2129
16.107 cpu.h . . . . .	2129
16.108 amd64/l4/util/l4_macros.h File Reference . . . . .	2130
16.108.1 Detailed Description . . . . .	2130
16.109 l4_macros.h . . . . .	2131
16.110 arm/l4/util/l4_macros.h File Reference . . . . .	2131
16.110.1 Detailed Description . . . . .	2131
16.111 l4_macros.h . . . . .	2131
16.112 l4/util/l4_macros.h File Reference . . . . .	2132
16.112.1 Detailed Description . . . . .	2132
16.113 l4_macros.h . . . . .	2132
16.114 x86/l4/util/l4_macros.h File Reference . . . . .	2132
16.114.1 Detailed Description . . . . .	2133
16.115 l4_macros.h . . . . .	2133
16.116 amd64/l4/util/mbi_argv.h File Reference . . . . .	2133
16.116.1 Detailed Description . . . . .	2134
16.117 mbi_argv.h . . . . .	2134
16.118 arm/l4/util/mbi_argv.h File Reference . . . . .	2135
16.118.1 Detailed Description . . . . .	2135
16.119 mbi_argv.h . . . . .	2136
16.120 x86/l4/util/mbi_argv.h File Reference . . . . .	2136
16.120.1 Detailed Description . . . . .	2137
16.121 mbi_argv.h . . . . .	2137

16.122 arm/l4/l4/sys/syscall_defs.h File Reference . . . . .	2138
16.122.1 Detailed Description . . . . .	2138
16.123 syscall_defs.h . . . . .	2138
16.124 contrib/libio-io/l4/io/io.h File Reference . . . . .	2139
16.124.1 Function Documentation . . . . .	2140
16.124.1.1 l4io_get_root_device() . . . . .	2140
16.124.1.2 l4io_iterate_devices() . . . . .	2140
16.124.1.3 l4io_request_all_ioports() . . . . .	2141
16.124.1.4 l4io_request_icu() . . . . .	2141
16.125 io.h . . . . .	2141
16.126 types.h . . . . .	2142
16.127 types.h . . . . .	2143
16.128 l4/sys/types.h File Reference . . . . .	2144
16.128.1 Detailed Description . . . . .	2146
16.128.2 Function Documentation . . . . .	2146
16.128.2.1 l4_capability_next() . . . . .	2146
16.129 types.h . . . . .	2146
16.130 l4/cxx/alloc.h File Reference . . . . .	2149
16.130.1 Detailed Description . . . . .	2149
16.131 alloc.h . . . . .	2149
16.132 arith . . . . .	2150
16.133 arm/l4/sys/atomic.h File Reference . . . . .	2150
16.133.1 Detailed Description . . . . .	2151
16.134 atomic.h . . . . .	2151
16.135 l4/cxx/atomic.h File Reference . . . . .	2152
16.135.1 Detailed Description . . . . .	2152
16.136 atomic.h . . . . .	2152
16.137 l4/util/atomic.h File Reference . . . . .	2153
16.137.1 Detailed Description . . . . .	2155
16.138 atomic.h . . . . .	2156
16.139 l4/cxx/avl_map File Reference . . . . .	2160
16.139.1 Detailed Description . . . . .	2161
16.140 avl_map . . . . .	2161
16.141 l4/cxx/avl_set File Reference . . . . .	2163
16.141.1 Detailed Description . . . . .	2164
16.142 avl_set . . . . .	2165
16.143 l4/cxx/avl_tree File Reference . . . . .	2168
16.143.1 Detailed Description . . . . .	2170
16.144 avl_tree . . . . .	2170
16.145 l4/cxx/basic_ostream File Reference . . . . .	2174
16.145.1 Detailed Description . . . . .	2175
16.146 basic_ostream . . . . .	2175

16.147 <a href="#">l4/cxx/basic_vector.h File Reference</a>	2178
16.147.1 Detailed Description	2179
16.148 <a href="#">basic_vector.h</a>	2179
16.149 <a href="#">bitfield</a>	2179
16.150 <a href="#">bitmap</a>	2182
16.151 <a href="#">l4/cxx/bits/bst.h File Reference</a>	2185
16.151.1 Detailed Description	2186
16.152 <a href="#">bst.h</a>	2187
16.153 <a href="#">l4/cxx/bits/bst_base.h File Reference</a>	2189
16.153.1 Detailed Description	2191
16.154 <a href="#">bst_base.h</a>	2191
16.155 <a href="#">l4/cxx/bits/bst_iter.h File Reference</a>	2192
16.155.1 Detailed Description	2193
16.156 <a href="#">bst_iter.h</a>	2194
16.157 <a href="#">list_basics.h</a>	2195
16.158 <a href="#">l4/cxx/bits/smart_ptr_list.h File Reference</a>	2197
16.158.1 Detailed Description	2199
16.159 <a href="#">smart_ptr_list.h</a>	2200
16.160 <a href="#">type_traits.h</a>	2201
16.161 <a href="#">dlist</a>	2204
16.162 <a href="#">l4/cxx/exceptions File Reference</a>	2207
16.162.1 Detailed Description	2209
16.163 <a href="#">exceptions</a>	2209
16.164 <a href="#">hlist</a>	2211
16.165 <a href="#">l4/cxx/iostream File Reference</a>	2214
16.165.1 Detailed Description	2215
16.166 <a href="#">iostream</a>	2215
16.167 <a href="#">l4/cxx/ipc_helper File Reference</a>	2215
16.167.1 Detailed Description	2216
16.168 <a href="#">ipc_helper</a>	2217
16.169 <a href="#">l4/cxx/ipc_server File Reference</a>	2217
16.169.1 Detailed Description	2218
16.170 <a href="#">ipc_server</a>	2219
16.171 <a href="#">ipc_server</a>	2220
16.172 <a href="#">l4/cxx/ipc_stream File Reference</a>	2223
16.172.1 Detailed Description	2226
16.172.2 Function Documentation	2226
16.172.2.1 <a href="#">operator&lt;&lt;()</a> [1/4]	2226
16.172.2.2 <a href="#">operator&lt;&lt;()</a> [2/4]	2227
16.172.2.3 <a href="#">operator&lt;&lt;()</a> [3/4]	2227
16.172.2.4 <a href="#">operator&lt;&lt;()</a> [4/4]	2228
16.172.2.5 <a href="#">operator&gt;&gt;()</a> [1/6]	2229

16.172.2.6 operator>>() [2/6]	2229
16.172.2.7 operator>>() [3/6]	2230
16.172.2.8 operator>>() [4/6]	2231
16.172.2.9 operator>>() [5/6]	2232
16.172.2.10 operator>>() [6/6]	2232
16.173 ipc_stream	2233
16.174 ipc_timeout_queue	2241
16.175 l4/cxx/l4iostream File Reference	2243
16.175.1 Detailed Description	2244
16.176 l4iostream	2244
16.177 l4/cxx/l4types.h File Reference	2245
16.177.1 Detailed Description	2246
16.178 l4types.h	2246
16.179 list	2246
16.180 list_alloc	2250
16.181 l4/cxx/main_thread File Reference	2256
16.181.1 Detailed Description	2257
16.182 main_thread	2257
16.183 minmax	2258
16.184 observer	2259
16.185 l4/cxx/pair File Reference	2259
16.185.1 Detailed Description	2261
16.186 pair	2261
16.187 ref_ptr	2262
16.188 l4/cxx/ref_ptr_list File Reference	2265
16.188.1 Detailed Description	2267
16.189 ref_ptr_list	2267
16.190 slab_alloc	2267
16.191 slist	2271
16.192 static_container	2274
16.193 static_vector	2274
16.194 std_alloc	2275
16.195 l4/cxx/std_exc_io File Reference	2276
16.195.1 Detailed Description	2276
16.196 std_exc_io	2277
16.197 std_ops	2277
16.198 string	2278
16.199 l4/cxx/string.h File Reference	2281
16.199.1 Detailed Description	2282
16.200 string.h	2282
16.201 l4/cxx/thread File Reference	2283
16.201.1 Detailed Description	2284

16.202 thread	2284
16.203 l4/sys/thread File Reference	2285
16.203.1 Detailed Description	2286
16.204 thread	2286
16.205 type_list	2288
16.206 type_traits	2288
16.207 unique_ptr	2293
16.208 l4/cxx/unique_ptr_list File Reference	2295
16.208.1 Detailed Description	2296
16.209 unique_ptr_list	2297
16.210 utils	2297
16.211 weak_ref	2297
16.212 backend	2299
16.213 default_ops_impl.h	2302
16.214 fd_store.h	2303
16.215 fd_store_impl.h	2304
16.216 ns_fs.h	2304
16.217 ns_fs_impl.h	2305
16.218 ro_file.h	2310
16.219 ro_file_impl.h	2311
16.220 vcon_stream.h	2312
16.221 vcon_stream_impl.h	2313
16.222 vfs_impl.h	2315
16.223 vfs.h	2328
16.224 virtio-block	2337
16.225 virtio-block	2340
16.226 virtio-net	2346
16.227 l4virtio	2349
16.228 l4virtio	2351
16.229 l4virtio	2353
16.230 virtio	2364
16.231 virtio-console	2368
16.232 virtio-console-device	2373
16.233 virtio-scmi-device	2377
16.234 virtio.h	2386
16.235 virtio_block.h	2389
16.236 virtio_input.h	2390
16.237 virtio_net.h	2391
16.238 virtqueue	2392
16.239 block_device_mgr.h	2396
16.240 device.h	2401
16.241 errand.h	2402

16.242 gpt.h . . . . .	2404
16.243 inout_memory.h . . . . .	2405
16.244 part_device.h . . . . .	2406
16.245 partition.h . . . . .	2409
16.246 request.h . . . . .	2412
16.247 virtio_client.h . . . . .	2412
16.248 l4/libedid/edid.h File Reference . . . . .	2420
16.249 edid.h . . . . .	2421
16.250 l4/libgfxbitmap/bitmap.h File Reference . . . . .	2422
16.250.1 Detailed Description . . . . .	2423
16.250.2 Macro Definition Documentation . . . . .	2423
16.250.2.1 pSLIM_BMAP_START_LSB . . . . .	2423
16.250.3 Typedef Documentation . . . . .	2424
16.250.3.1 gfxbitmap_color_pix_t . . . . .	2424
16.250.3.2 gfxbitmap_color_t . . . . .	2424
16.250.4 Function Documentation . . . . .	2424
16.250.4.1 gfxbitmap_bmap() . . . . .	2424
16.250.4.2 gfxbitmap_convert_color() . . . . .	2425
16.250.4.3 gfxbitmap_copy() . . . . .	2425
16.250.4.4 gfxbitmap_fill() . . . . .	2425
16.250.4.5 gfxbitmap_set() . . . . .	2426
16.251 bitmap.h . . . . .	2426
16.252 l4/libgfxbitmap/font.h File Reference . . . . .	2427
16.252.1 Detailed Description . . . . .	2429
16.252.2 Enumeration Type Documentation . . . . .	2429
16.252.2.1 anonymous enum . . . . .	2429
16.252.3 Function Documentation . . . . .	2429
16.252.3.1 gfxbitmap_font_data() . . . . .	2429
16.252.3.2 gfxbitmap_font_get() . . . . .	2430
16.252.3.3 gfxbitmap_font_height() . . . . .	2430
16.252.3.4 gfxbitmap_font_init() . . . . .	2430
16.252.3.5 gfxbitmap_font_text() . . . . .	2431
16.252.3.6 gfxbitmap_font_text_scale() . . . . .	2431
16.252.3.7 gfxbitmap_font_width() . . . . .	2432
16.253 font.h . . . . .	2432
16.254 l4/libgfxbitmap/support File Reference . . . . .	2433
16.254.1 Detailed Description . . . . .	2433
16.255 support . . . . .	2434
16.256 l4/re/c/dataspace.h File Reference . . . . .	2434
16.256.1 Detailed Description . . . . .	2435
16.257 dataspace.h . . . . .	2436
16.258 debug.h . . . . .	2437

16.259 l4/re/c/debug.h File Reference . . . . .	2437
16.259.1 Detailed Description . . . . .	2438
16.260 debug.h . . . . .	2438
16.261 l4/re/c/dma_space.h File Reference . . . . .	2439
16.261.1 Detailed Description . . . . .	2440
16.261.2 Enumeration Type Documentation . . . . .	2440
16.261.2.1 l4re_dma_space_direction . . . . .	2440
16.261.2.2 l4re_dma_space_space_attrbs . . . . .	2440
16.262 dma_space.h . . . . .	2441
16.263 event_buffer.h . . . . .	2442
16.264 l4/re/c/inhibitor.h File Reference . . . . .	2442
16.264.1 Detailed Description . . . . .	2443
16.264.2 Function Documentation . . . . .	2443
16.264.2.1 l4re_inhibitor_acquire() . . . . .	2443
16.264.2.2 l4re_inhibitor_next_lock_info() . . . . .	2444
16.264.2.3 l4re_inhibitor_release() . . . . .	2444
16.265 inhibitor.h . . . . .	2445
16.266 l4/re/c/log.h File Reference . . . . .	2445
16.266.1 Detailed Description . . . . .	2446
16.267 log.h . . . . .	2446
16.268 l4/re/c/mem_alloc.h File Reference . . . . .	2447
16.268.1 Detailed Description . . . . .	2448
16.269 mem_alloc.h . . . . .	2449
16.270 l4/re/c/namespace.h File Reference . . . . .	2449
16.270.1 Detailed Description . . . . .	2450
16.271 namespace.h . . . . .	2451
16.272 l4/re/c/parent.h File Reference . . . . .	2451
16.272.1 Detailed Description . . . . .	2452
16.272.2 Function Documentation . . . . .	2452
16.272.2.1 l4re_parent_signal() . . . . .	2452
16.273 parent.h . . . . .	2453
16.274 l4/re/c/rm.h File Reference . . . . .	2453
16.274.1 Detailed Description . . . . .	2454
16.275 rm.h . . . . .	2455
16.276 l4/re/c/util/cap_alloc.h File Reference . . . . .	2457
16.276.1 Detailed Description . . . . .	2458
16.277 cap_alloc.h . . . . .	2458
16.278 l4/re/c/util/kumem_alloc.h File Reference . . . . .	2458
16.278.1 Detailed Description . . . . .	2459
16.278.2 Function Documentation . . . . .	2459
16.278.2.1 l4re_util_kumem_alloc() . . . . .	2459
16.279 kumem_alloc.h . . . . .	2460

16.280 I4/re/c/util/video/goos_fb.h File Reference . . . . .	2460
16.280.1 Detailed Description . . . . .	2461
16.281 goos_fb.h . . . . .	2461
16.282 I4/re/c/video/colors.h File Reference . . . . .	2462
16.282.1 Detailed Description . . . . .	2463
16.283 colors.h . . . . .	2464
16.284 I4/re/c/video/goos.h File Reference . . . . .	2464
16.284.1 Detailed Description . . . . .	2466
16.285 goos.h . . . . .	2466
16.286 I4/re/c/video/view.h File Reference . . . . .	2467
16.286.1 Detailed Description . . . . .	2469
16.287 view.h . . . . .	2469
16.288 I4/re/cap_alloc File Reference . . . . .	2470
16.288.1 Detailed Description . . . . .	2472
16.289 cap_alloc . . . . .	2472
16.290 I4/re/util/cap_alloc File Reference . . . . .	2474
16.290.1 Detailed Description . . . . .	2475
16.291 cap_alloc . . . . .	2476
16.292 console . . . . .	2477
16.293 I4/re/consts File Reference . . . . .	2477
16.293.1 Detailed Description . . . . .	2478
16.294 consts . . . . .	2479
16.295 consts . . . . .	2479
16.296 amd64/I4/sys/consts.h File Reference . . . . .	2480
16.296.1 Detailed Description . . . . .	2480
16.297 consts.h . . . . .	2480
16.298 arm/I4/sys/consts.h File Reference . . . . .	2481
16.298.1 Detailed Description . . . . .	2481
16.299 consts.h . . . . .	2481
16.300 I4/re/consts.h File Reference . . . . .	2482
16.300.1 Detailed Description . . . . .	2483
16.300.2 Enumeration Type Documentation . . . . .	2483
16.300.2.1 anonymous enum . . . . .	2483
16.301 consts.h . . . . .	2483
16.302 I4/sys/consts.h File Reference . . . . .	2484
16.302.1 Detailed Description . . . . .	2485
16.303 consts.h . . . . .	2486
16.304 x86/I4/sys/consts.h File Reference . . . . .	2488
16.304.1 Detailed Description . . . . .	2488
16.305 consts.h . . . . .	2488
16.306 I4/re/dataspace File Reference . . . . .	2489
16.306.1 Detailed Description . . . . .	2490



16.307 dataspace . . . . .	2490
16.308 l4/re/dataspace-sys.h File Reference . . . . .	2492
16.308.1 Detailed Description . . . . .	2492
16.309 dataspace-sys.h . . . . .	2492
16.310 l4/re/debug File Reference . . . . .	2493
16.310.1 Detailed Description . . . . .	2494
16.311 debug . . . . .	2494
16.312 debug . . . . .	2494
16.313 l4/re/dma_space File Reference . . . . .	2496
16.314 dma_space . . . . .	2498
16.315 l4/re/elf_aux.h File Reference . . . . .	2499
16.315.1 Detailed Description . . . . .	2500
16.316 elf_aux.h . . . . .	2500
16.317 l4/re/env File Reference . . . . .	2501
16.317.1 Detailed Description . . . . .	2502
16.318 env . . . . .	2502
16.319 l4/re/env.h File Reference . . . . .	2504
16.319.1 Detailed Description . . . . .	2505
16.319.2 Typedef Documentation . . . . .	2505
16.319.2.1 l4re_env_t . . . . .	2505
16.320 env.h . . . . .	2506
16.321 l4/re/error_helper File Reference . . . . .	2507
16.321.1 Detailed Description . . . . .	2509
16.322 error_helper . . . . .	2509
16.323 event . . . . .	2510
16.324 l4/re/util/event File Reference . . . . .	2513
16.325 event . . . . .	2514
16.326 event-sys.h . . . . .	2515
16.327 l4/re/c/event.h File Reference . . . . .	2516
16.327.1 Detailed Description . . . . .	2517
16.328 event.h . . . . .	2518
16.329 l4/re/event.h File Reference . . . . .	2518
16.329.1 Detailed Description . . . . .	2519
16.330 event.h . . . . .	2519
16.331 event_enums.h . . . . .	2520
16.332 l4/re/impl/dataspace_impl.h File Reference . . . . .	2527
16.332.1 Detailed Description . . . . .	2528
16.333 dataspace_impl.h . . . . .	2528
16.334 l4/re/impl/mem_alloc_impl.h File Reference . . . . .	2529
16.334.1 Detailed Description . . . . .	2530
16.335 mem_alloc_impl.h . . . . .	2530
16.336 l4/re/impl/namespace_impl.h File Reference . . . . .	2531

16.336.1 Detailed Description . . . . .	2532
16.337 namespace_impl.h . . . . .	2532
16.338 l4/re/impl/rm_impl.h File Reference . . . . .	2533
16.338.1 Detailed Description . . . . .	2534
16.339 rm_impl.h . . . . .	2534
16.340 inhibitor . . . . .	2536
16.341 inhibitor-sys.h . . . . .	2536
16.342 l4/re/l4aux.h File Reference . . . . .	2536
16.342.1 Detailed Description . . . . .	2537
16.343 l4aux.h . . . . .	2538
16.344 l4/re/log File Reference . . . . .	2538
16.344.1 Detailed Description . . . . .	2540
16.345 log . . . . .	2540
16.346 l4/re/log-sys.h File Reference . . . . .	2540
16.346.1 Detailed Description . . . . .	2541
16.347 log-sys.h . . . . .	2541
16.348 l4/re/mem_alloc File Reference . . . . .	2541
16.348.1 Detailed Description . . . . .	2543
16.349 mem_alloc . . . . .	2543
16.350 l4/re/mem_alloc-sys.h File Reference . . . . .	2543
16.350.1 Detailed Description . . . . .	2544
16.351 mem_alloc-sys.h . . . . .	2544
16.352 l4/re/mmio_space File Reference . . . . .	2545
16.352.1 Detailed Description . . . . .	2545
16.353 mmio_space . . . . .	2546
16.354 l4/re/namespace File Reference . . . . .	2546
16.354.1 Detailed Description . . . . .	2548
16.355 namespace . . . . .	2548
16.356 l4/re/namespace-sys.h File Reference . . . . .	2549
16.356.1 Detailed Description . . . . .	2549
16.357 namespace-sys.h . . . . .	2550
16.358 l4/re/parent File Reference . . . . .	2550
16.358.1 Detailed Description . . . . .	2552
16.359 parent . . . . .	2552
16.360 l4/re/parent-sys.h File Reference . . . . .	2552
16.360.1 Detailed Description . . . . .	2553
16.361 parent-sys.h . . . . .	2553
16.362 l4/re/protocols.h File Reference . . . . .	2553
16.362.1 Detailed Description . . . . .	2554
16.362.2 Enumeration Type Documentation . . . . .	2554
16.362.2.1 L4re_protocols . . . . .	2554
16.363 protocols.h . . . . .	2554

16.364 I4/re/random File Reference . . . . .	2555
16.364.1 Detailed Description . . . . .	2556
16.365 random . . . . .	2556
16.366 I4/re/rm File Reference . . . . .	2557
16.366.1 Detailed Description . . . . .	2558
16.367 rm . . . . .	2558
16.368 I4/re/rm-sys.h File Reference . . . . .	2562
16.368.1 Detailed Description . . . . .	2563
16.369 rm-sys.h . . . . .	2563
16.370 I4/re/shared_cap File Reference . . . . .	2563
16.370.1 Detailed Description . . . . .	2565
16.371 shared_cap . . . . .	2565
16.372 I4/re/util/shared_cap File Reference . . . . .	2566
16.372.1 Detailed Description . . . . .	2568
16.373 shared_cap . . . . .	2568
16.374 I4/re/unique_cap File Reference . . . . .	2569
16.374.1 Detailed Description . . . . .	2570
16.375 unique_cap . . . . .	2570
16.376 I4/re/util/unique_cap File Reference . . . . .	2571
16.376.1 Detailed Description . . . . .	2573
16.377 unique_cap . . . . .	2573
16.378 I4/re/util/bitmap_cap_alloc File Reference . . . . .	2573
16.378.1 Detailed Description . . . . .	2574
16.379 bitmap_cap_alloc . . . . .	2575
16.380 br_manager . . . . .	2576
16.381 I4/re/util/cap File Reference . . . . .	2578
16.381.1 Detailed Description . . . . .	2579
16.382 cap . . . . .	2580
16.383 I4/re/util/cap_alloc_impl.h File Reference . . . . .	2580
16.383.1 Detailed Description . . . . .	2581
16.384 cap_alloc_impl.h . . . . .	2581
16.385 I4/re/util/counting_cap_alloc File Reference . . . . .	2582
16.385.1 Detailed Description . . . . .	2583
16.386 counting_cap_alloc . . . . .	2583
16.387 dataspace_svr . . . . .	2587
16.388 env_ns . . . . .	2590
16.389 event_buffer . . . . .	2591
16.390 event_svr . . . . .	2592
16.391 icu_svr . . . . .	2594
16.392 I4/re/util/item_alloc File Reference . . . . .	2596
16.392.1 Detailed Description . . . . .	2597
16.393 item_alloc . . . . .	2597

16.394 l4/re/util/kumem_alloc File Reference	2599
16.394.1 Detailed Description	2599
16.395 kumem_alloc	2600
16.396 meta	2600
16.397 l4/sys/meta File Reference	2600
16.397.1 Detailed Description	2601
16.398 meta	2602
16.399 name_space_svr	2602
16.400 object_registry	2608
16.401 poll_timeout_kipclock	2611
16.402 l4/re/util/region_mapping File Reference	2611
16.402.1 Detailed Description	2612
16.403 region_mapping	2612
16.404 region_mapping_svr_2	2618
16.405 vcon_svr	2621
16.406 goos_fb	2622
16.407 goos_svr	2624
16.408 colors	2626
16.409 goos	2627
16.410 l4/re/video/goos-sys.h File Reference	2630
16.410.1 Detailed Description	2631
16.411 goos-sys.h	2631
16.412 view	2632
16.413 l4/shmc/ringbuf.h File Reference	2632
16.413.1 Function Documentation	2633
16.413.1.1 l4shmc_rb_attach_receiver()	2633
16.413.1.2 l4shmc_rb_attach_sender()	2634
16.413.1.3 l4shmc_rb_deinit_buffer()	2634
16.413.1.4 l4shmc_rb_init_buffer()	2634
16.413.1.5 l4shmc_rb_init_receiver()	2635
16.413.1.6 l4shmc_rb_receiver_copy_out()	2636
16.413.1.7 l4shmc_rb_receiver_notify_done()	2636
16.413.1.8 l4shmc_rb_receiver_read_next_size()	2636
16.413.1.9 l4shmc_rb_receiver_wait_for_data()	2637
16.413.1.10 l4shmc_rb_sender_alloc_packet()	2637
16.413.1.11 l4shmc_rb_sender_commit_packet()	2637
16.413.1.12 l4shmc_rb_sender_next_copy_in()	2638
16.413.1.13 l4shmc_rb_sender_put_data()	2638
16.414 ringbuf.h	2639
16.415 l4/shmc/shmc.h File Reference	2640
16.415.1 Detailed Description	2643
16.416 shmc.h	2643

16.417 l4/sigma0/sigma0.h File Reference . . . . .	2645
16.417.1 Detailed Description . . . . .	2647
16.418 sigma0.h . . . . .	2647
16.419 l4/sys/__kernel_object_impl.h File Reference . . . . .	2649
16.419.1 Detailed Description . . . . .	2649
16.420 __kernel_object_impl.h . . . . .	2649
16.421 __kip-32bit.h . . . . .	2650
16.422 __kip-64bit.h . . . . .	2651
16.423 l4/sys/__ktrace-impl.h File Reference . . . . .	2652
16.423.1 Detailed Description . . . . .	2654
16.424 __ktrace-impl.h . . . . .	2654
16.425 __l4_fpage.h . . . . .	2655
16.426 __platform_control-arm.h . . . . .	2659
16.427 __task-arm.h . . . . .	2659
16.428 __timeout.h . . . . .	2660
16.429 l4/sys/__typeinfo.h File Reference . . . . .	2662
16.429.1 Detailed Description . . . . .	2665
16.430 __typeinfo.h . . . . .	2665
16.431 __vcpu-arm.h . . . . .	2675
16.432 l4/sys/__vm-arm.h File Reference . . . . .	2676
16.432.1 Detailed Description . . . . .	2678
16.433 __vm-arm.h . . . . .	2678
16.434 __vm-svm.h . . . . .	2678
16.435 __vm-vmx.h . . . . .	2680
16.436 l4/sys/arm_smccc File Reference . . . . .	2686
16.436.1 Detailed Description . . . . .	2687
16.437 arm_smccc . . . . .	2688
16.438 l4/sys/arm_smccc.h File Reference . . . . .	2688
16.438.1 Detailed Description . . . . .	2689
16.438.2 Function Documentation . . . . .	2689
16.438.2.1 l4_arm_smccc_call() . . . . .	2689
16.439 arm_smccc.h . . . . .	2690
16.440 l4/sys/assert.h File Reference . . . . .	2691
16.440.1 Detailed Description . . . . .	2693
16.440.2 Macro Definition Documentation . . . . .	2693
16.440.2.1 l4_assert . . . . .	2693
16.441 assert.h . . . . .	2693
16.442 l4/util/assert.h File Reference . . . . .	2694
16.442.1 Detailed Description . . . . .	2695
16.443 assert.h . . . . .	2695
16.444 amd64/l4/sys/cache.h File Reference . . . . .	2697
16.444.1 Detailed Description . . . . .	2697

16.445	cache.h	2697
16.446	arm/l4/sys/cache.h File Reference	2698
16.446.1	Detailed Description	2699
16.447	cache.h	2699
16.448	l4/sys/cache.h File Reference	2701
16.448.1	Detailed Description	2702
16.449	cache.h	2702
16.450	x86/l4/sys/cache.h File Reference	2703
16.450.1	Detailed Description	2703
16.451	cache.h	2704
16.452	l4/sys/capability File Reference	2704
16.452.1	Detailed Description	2706
16.452.2	Macro Definition Documentation	2706
16.452.2.1	L4_DISABLE_COPY	2706
16.453	capability	2706
16.454	l4/sys/compiler.h File Reference	2708
16.454.1	Detailed Description	2709
16.455	compiler.h	2710
16.456	capability.h	2712
16.457	ipc_array	2714
16.458	ipc_basics	2718
16.459	l4/sys/cxx/ipc_client File Reference	2722
16.459.1	Macro Definition Documentation	2723
16.459.1.1	L4_RPC_DEF	2723
16.460	ipc_client	2724
16.461	ipc_epiface	2724
16.462	l4/sys/cxx/ipc_iface File Reference	2728
16.462.1	Detailed Description	2730
16.462.2	Macro Definition Documentation	2730
16.462.2.1	L4_INLINE_RPC	2730
16.462.2.2	L4_INLINE_RPC_NF	2731
16.462.2.3	L4_INLINE_RPC_NF_OP	2731
16.462.2.4	L4_INLINE_RPC_OP	2732
16.462.2.5	L4_RPC	2732
16.462.2.6	L4_RPC_NF	2733
16.462.2.7	L4_RPC_NF_OP	2733
16.462.2.8	L4_RPC_OP	2734
16.463	ipc_iface	2734
16.464	ipc_legacy	2739
16.465	ipc_ret_array	2740
16.466	ipc_server_loop	2741
16.467	ipc_string	2744

16.468 I4/sys/cxx/ipc_types File Reference . . . . .	2745
16.469 ipc_types . . . . .	2747
16.470 ipc_varg . . . . .	2754
16.471 I4/sys/cxx/smart_capability_1x File Reference . . . . .	2760
16.472 smart_capability_1x . . . . .	2761
16.473 I4/sys/cxx/types File Reference . . . . .	2763
16.473.1 Macro Definition Documentation . . . . .	2764
16.473.1.1 L4_TYPES_FLAGS_OPS_DEF . . . . .	2764
16.474 types . . . . .	2764
16.475 I4/sys/debugger File Reference . . . . .	2767
16.475.1 Detailed Description . . . . .	2768
16.476 debugger . . . . .	2768
16.477 I4/sys/debugger.h File Reference . . . . .	2769
16.477.1 Detailed Description . . . . .	2770
16.478 debugger.h . . . . .	2770
16.479 I4/sys/err.h File Reference . . . . .	2773
16.479.1 Detailed Description . . . . .	2774
16.480 err.h . . . . .	2775
16.481 I4/sys/exception File Reference . . . . .	2775
16.481.1 Detailed Description . . . . .	2776
16.482 exception . . . . .	2777
16.483 I4/sys/factory File Reference . . . . .	2777
16.483.1 Detailed Description . . . . .	2779
16.484 factory . . . . .	2779
16.485 I4/sys/factory.h File Reference . . . . .	2781
16.485.1 Detailed Description . . . . .	2782
16.486 factory.h . . . . .	2783
16.487 I4/sys/icu File Reference . . . . .	2787
16.487.1 Detailed Description . . . . .	2788
16.488 icu . . . . .	2788
16.489 I4/sys/icu.h File Reference . . . . .	2789
16.489.1 Detailed Description . . . . .	2791
16.490 icu.h . . . . .	2792
16.491 iommu . . . . .	2795
16.492 ipc.h . . . . .	2795
16.493 arm/I4f/I4/sys/ipc.h File Reference . . . . .	2796
16.493.1 Detailed Description . . . . .	2797
16.494 ipc.h . . . . .	2797
16.495 I4/sys/ipc.h File Reference . . . . .	2798
16.495.1 Detailed Description . . . . .	2799
16.495.2 Function Documentation . . . . .	2800
16.495.2.1 I4_ipc_to_errno() . . . . .	2800

16.496 ipc.h . . . . .	2800
16.497 x86/I4f/I4/sys/ipc.h File Reference . . . . .	2803
16.497.1 Detailed Description . . . . .	2804
16.498 ipc.h . . . . .	2804
16.499 I4/sys/ipc_gate File Reference . . . . .	2805
16.499.1 Detailed Description . . . . .	2806
16.500 ipc_gate . . . . .	2806
16.501 I4/sys/ipc_gate.h File Reference . . . . .	2807
16.501.1 Detailed Description . . . . .	2809
16.502 ipc_gate.h . . . . .	2810
16.503 I4/sys/irq File Reference . . . . .	2811
16.503.1 Detailed Description . . . . .	2812
16.504 irq . . . . .	2812
16.505 amd64/I4/util/irq.h File Reference . . . . .	2814
16.505.1 Detailed Description . . . . .	2815
16.505.2 Function Documentation . . . . .	2815
16.505.2.1 I4util_irq_acknowledge() . . . . .	2815
16.506 irq.h . . . . .	2816
16.507 arm/I4/util/irq.h File Reference . . . . .	2817
16.507.1 Detailed Description . . . . .	2817
16.508 irq.h . . . . .	2818
16.509 I4/irq/irq.h File Reference . . . . .	2818
16.509.1 Detailed Description . . . . .	2820
16.510 irq.h . . . . .	2820
16.511 I4/sys/irq.h File Reference . . . . .	2821
16.511.1 Detailed Description . . . . .	2822
16.512 irq.h . . . . .	2823
16.513 x86/I4/util/irq.h File Reference . . . . .	2825
16.513.1 Detailed Description . . . . .	2825
16.513.2 Function Documentation . . . . .	2826
16.513.2.1 I4util_irq_acknowledge() . . . . .	2826
16.514 irq.h . . . . .	2826
16.515 I4/sys/kdebug.h File Reference . . . . .	2827
16.515.1 Detailed Description . . . . .	2829
16.515.2 Enumeration Type Documentation . . . . .	2829
16.515.2.1 I4_kdebug_ops_t . . . . .	2829
16.515.3 Function Documentation . . . . .	2829
16.515.3.1 __kdebug_3_text() . . . . .	2829
16.515.3.2 __kdebug_op() . . . . .	2830
16.515.3.3 __kdebug_op_1() . . . . .	2831
16.515.3.4 __kdebug_text() . . . . .	2833
16.515.3.5 enter_kdebug() . . . . .	2834



16.515.3.6 outchar()	2835
16.515.3.7 outdec()	2836
16.515.3.8 outhex12()	2836
16.515.3.9 outhex16()	2837
16.515.3.10 outhex20()	2838
16.515.3.11 outhex32()	2838
16.515.3.12 outhex64()	2839
16.515.3.13 outhex8()	2839
16.515.3.14 outnstring()	2840
16.515.3.15 outstring()	2841
16.515.3.16 outumword()	2842
16.516 kdebug.h	2842
16.517 l4/sys/kernel_object.h File Reference	2845
16.517.1 Detailed Description	2846
16.518 kernel_object.h	2846
16.519 l4/sys/kip File Reference	2847
16.519.1 Detailed Description	2848
16.520 kip	2848
16.521 l4/sys/kip.h File Reference	2850
16.521.1 Detailed Description	2851
16.521.2 Macro Definition Documentation	2851
16.521.2.1 l4_kip_for_each_feature	2851
16.521.3 Function Documentation	2851
16.521.3.1 l4_kip_kernel_has_feature()	2851
16.522 kip.h	2852
16.523 l4/util/kip.h File Reference	2854
16.524 kip.h	2855
16.525 kobject	2855
16.526 l4/sys/ktrace.h File Reference	2856
16.526.1 Detailed Description	2857
16.527 ktrace.h	2857
16.528 amd64/l4/sys/l4int.h File Reference	2857
16.528.1 Detailed Description	2858
16.529 l4int.h	2858
16.530 arm/l4/sys/l4int.h File Reference	2858
16.530.1 Detailed Description	2859
16.531 l4int.h	2859
16.532 l4/sys/l4int.h File Reference	2859
16.532.1 Detailed Description	2860
16.533 l4int.h	2860
16.534 x86/l4/sys/l4int.h File Reference	2861
16.534.1 Detailed Description	2861

16.535 l4int.h . . . . .	2861
16.536 l4/sys/memdesc.h File Reference . . . . .	2862
16.536.1 Detailed Description . . . . .	2863
16.537 memdesc.h . . . . .	2863
16.538 l4/sys/pager File Reference . . . . .	2865
16.538.1 Detailed Description . . . . .	2867
16.539 pager . . . . .	2867
16.540 l4/sys/platform_control File Reference . . . . .	2867
16.540.1 Detailed Description . . . . .	2868
16.541 platform_control . . . . .	2869
16.542 platform_control.h . . . . .	2869
16.543 l4/sys/platform_control.h File Reference . . . . .	2869
16.543.1 Detailed Description . . . . .	2871
16.544 platform_control.h . . . . .	2871
16.545 l4/sys/rcv_endpoint File Reference . . . . .	2873
16.545.1 Detailed Description . . . . .	2874
16.546 rcv_endpoint . . . . .	2875
16.547 l4/sys/rcv_endpoint.h File Reference . . . . .	2875
16.547.1 Detailed Description . . . . .	2876
16.547.2 Enumeration Type Documentation . . . . .	2877
16.547.2.1 L4_rcv_ep_ops . . . . .	2877
16.548 rcv_endpoint.h . . . . .	2877
16.549 l4/sys/scheduler File Reference . . . . .	2878
16.549.1 Detailed Description . . . . .	2879
16.550 scheduler . . . . .	2879
16.551 scheduler.h . . . . .	2880
16.552 l4/sys/scheduler.h File Reference . . . . .	2883
16.552.1 Detailed Description . . . . .	2885
16.553 scheduler.h . . . . .	2885
16.554 l4/sys/semaphore File Reference . . . . .	2888
16.554.1 Detailed Description . . . . .	2889
16.555 semaphore . . . . .	2889
16.556 l4/sys/semaphore.h File Reference . . . . .	2890
16.556.1 Detailed Description . . . . .	2891
16.557 semaphore.h . . . . .	2891
16.558 l4/sys/smart_capability File Reference . . . . .	2892
16.558.1 Detailed Description . . . . .	2894
16.559 smart_capability . . . . .	2894
16.560 l4/sys/task File Reference . . . . .	2896
16.560.1 Detailed Description . . . . .	2898
16.561 task . . . . .	2898
16.562 task.h . . . . .	2899

16.563 l4/sys/task.h File Reference . . . . .	2899
16.563.1 Detailed Description . . . . .	2900
16.564 task.h . . . . .	2901
16.565 arm/l4/sys/thread.h File Reference . . . . .	2903
16.565.1 Detailed Description . . . . .	2904
16.566 thread.h . . . . .	2904
16.567 l4/sys/thread.h File Reference . . . . .	2905
16.567.1 Detailed Description . . . . .	2907
16.568 thread.h . . . . .	2907
16.569 l4/util/thread.h File Reference . . . . .	2914
16.569.1 Detailed Description . . . . .	2915
16.569.2 Macro Definition Documentation . . . . .	2915
16.569.2.1 __L4UTIL_THREAD_FUNC . . . . .	2915
16.570 thread.h . . . . .	2916
16.571 l4/sys/typeinfo_svr File Reference . . . . .	2916
16.571.1 Detailed Description . . . . .	2918
16.572 typeinfo_svr . . . . .	2918
16.573 amd64/l4/sys/utcb.h File Reference . . . . .	2919
16.573.1 Detailed Description . . . . .	2920
16.574 utcb.h . . . . .	2920
16.575 arm/l4/sys/utcb.h File Reference . . . . .	2922
16.575.1 Detailed Description . . . . .	2923
16.576 utcb.h . . . . .	2923
16.577 l4/sys/utcb.h File Reference . . . . .	2924
16.577.1 Detailed Description . . . . .	2926
16.578 utcb.h . . . . .	2926
16.579 x86/l4/sys/utcb.h File Reference . . . . .	2928
16.579.1 Detailed Description . . . . .	2930
16.580 utcb.h . . . . .	2930
16.581 l4/sys/vcon File Reference . . . . .	2931
16.581.1 Detailed Description . . . . .	2933
16.582 vcon . . . . .	2933
16.583 l4/sys/vcon.h File Reference . . . . .	2933
16.583.1 Detailed Description . . . . .	2936
16.583.2 Enumeration Type Documentation . . . . .	2936
16.583.2.1 L4_vcon_read_flags . . . . .	2936
16.584 vcon.h . . . . .	2936
16.585 l4/sys/vcpu.h File Reference . . . . .	2939
16.585.1 Detailed Description . . . . .	2941
16.585.2 Function Documentation . . . . .	2941
16.585.2.1 l4_vcpu_check_version() . . . . .	2941
16.586 vcpu.h . . . . .	2941

16.587 I4/vcpu/vcpu.h File Reference . . . . .	2942
16.587.1 Detailed Description . . . . .	2944
16.588 vcpu.h . . . . .	2944
16.589 I4/sys/vcpu_context File Reference . . . . .	2946
16.589.1 Detailed Description . . . . .	2946
16.590 vcpu_context . . . . .	2947
16.591 vcpu_context.h . . . . .	2947
16.592 I4/sys/vhw.h File Reference . . . . .	2947
16.592.1 Detailed Description . . . . .	2948
16.593 vhw.h . . . . .	2948
16.594 vm . . . . .	2949
16.595 I4/sys/vm File Reference . . . . .	2949
16.595.1 Detailed Description . . . . .	2950
16.596 vm . . . . .	2951
16.597 I4/util/backtrace.h File Reference . . . . .	2951
16.597.1 Detailed Description . . . . .	2952
16.597.2 Function Documentation . . . . .	2952
16.597.2.1 I4util_backtrace() . . . . .	2952
16.598 backtrace.h . . . . .	2952
16.599 I4/util/base64.h File Reference . . . . .	2953
16.599.1 Detailed Description . . . . .	2953
16.600 base64.h . . . . .	2954
16.601 I4/util/bitops.h File Reference . . . . .	2954
16.601.1 Detailed Description . . . . .	2956
16.602 bitops.h . . . . .	2957
16.603 I4/util/elf.h File Reference . . . . .	2960
16.603.1 Detailed Description . . . . .	2966
16.604 elf.h . . . . .	2966
16.605 I4/util/getopt.h File Reference . . . . .	2976
16.605.1 Detailed Description . . . . .	2977
16.606 getopt.h . . . . .	2977
16.607 I4/util/keymap.h File Reference . . . . .	2978
16.607.1 Detailed Description . . . . .	2978
16.608 keymap.h . . . . .	2979
16.609 I4/util/kprintf.h File Reference . . . . .	2979
16.609.1 Detailed Description . . . . .	2979
16.610 kprintf.h . . . . .	2980
16.611 I4/util/I4mod.h File Reference . . . . .	2980
16.611.1 Detailed Description . . . . .	2981
16.611.2 Enumeration Type Documentation . . . . .	2981
16.611.2.1 I4util_I4mod_mod_info_flag . . . . .	2981
16.612 I4mod.h . . . . .	2981

16.613 I4/util/list_alloc.h File Reference	2982
16.613.1 Detailed Description	2982
16.613.2 Function Documentation	2983
16.613.2.1 I4la_alloc()	2983
16.613.2.2 I4la_avail()	2983
16.613.2.3 I4la_dump()	2983
16.613.2.4 I4la_free()	2983
16.613.2.5 I4la_init()	2984
16.614 list_alloc.h	2984
16.615 I4/util/lock.h File Reference	2984
16.615.1 Detailed Description	2985
16.616 lock.h	2985
16.617 I4/util/mb_info.h File Reference	2986
16.617.1 Detailed Description	2989
16.617.2 Macro Definition Documentation	2989
16.617.2.1 I4util_mb_for_each_mmap_entry	2989
16.617.2.2 L4UTIL_MB_MEMORY	2989
16.617.2.3 MB_ARD_MEMORY	2989
16.617.2.4 MB_ART_MEMORY	2990
16.618 mb_info.h	2990
16.619 I4/util/parse_cmd.h File Reference	2994
16.619.1 Detailed Description	2995
16.620 parse_cmd.h	2995
16.621 I4/util/rand.h File Reference	2995
16.621.1 Detailed Description	2996
16.622 rand.h	2997
16.623 I4/util/splitlog2.h File Reference	2997
16.623.1 Detailed Description	2998
16.624 splitlog2.h	2998
16.625 util.h	2999
16.626 vbus	2999
16.627 I4/vbus/vbus.h File Reference	3001
16.627.1 Detailed Description	3003
16.627.2 Enumeration Type Documentation	3003
16.627.2.1 anonymous enum	3003
16.627.2.2 I4vbus_icu_src_types	3004
16.628 vbus.h	3004
16.629 vbus_generic	3005
16.630 vbus_gpio	3005
16.631 vbus_gpio-ops.h	3007
16.632 vbus_gpio.h	3007
16.633 vbus_i2c.h	3008

16.634 vbus_inhibitor.h . . . . .	3008
16.635 l4/vbus/vbus_interfaces.h File Reference . . . . .	3008
16.635.1 Detailed Description . . . . .	3010
16.635.2 Typedef Documentation . . . . .	3010
16.635.2.1 l4vbus_iface_type_t . . . . .	3010
16.635.3 Enumeration Type Documentation . . . . .	3010
16.635.3.1 anonymous enum . . . . .	3010
16.635.3.2 l4vbus_iface_type_t . . . . .	3010
16.635.4 Function Documentation . . . . .	3011
16.635.4.1 l4vbus_subinterface_supported() . . . . .	3011
16.636 vbus_interfaces.h . . . . .	3011
16.637 vbus_mcspi.h . . . . .	3012
16.638 vbus_pci . . . . .	3012
16.639 vbus_pci-ops.h . . . . .	3013
16.640 vbus_pci.h . . . . .	3014
16.641 vbus_pm-ops.h . . . . .	3014
16.642 vbus_pm.h . . . . .	3015
16.643 l4/vbus/vbus_types.h File Reference . . . . .	3015
16.643.1 Detailed Description . . . . .	3016
16.643.2 Enumeration Type Documentation . . . . .	3016
16.643.2.1 l4vbus_device_flags_t . . . . .	3016
16.643.2.2 l4vbus_resource_flags_t . . . . .	3017
16.643.2.3 l4vbus_resource_type_t . . . . .	3017
16.644 vbus_types.h . . . . .	3017
16.645 vdevice-ops.h . . . . .	3018
16.646 l4/vcpu/vcpu File Reference . . . . .	3019
16.646.1 Detailed Description . . . . .	3019
16.647 vcpu . . . . .	3020
16.648 ipc-invoke.h . . . . .	3021
16.649 ipc-l42-gcc3-nopic.h . . . . .	3022
<b>17 Examples . . . . .</b>	<b>3025</b>
17.1 hello/server/src/main.c . . . . .	3025
17.2 examples/sys/ipc/ipc_example.c . . . . .	3025
17.3 examples/sys/ipc/ipc.cfg . . . . .	3027
17.4 examples/sys/start-with-exc/main.c . . . . .	3027
17.5 examples/sys/singlestep/main.c . . . . .	3029
17.6 examples/sys/aliens/main.c . . . . .	3031
17.7 examples/sys/utcb-ipc/main.c . . . . .	3035
17.8 examples/sys/ux-vhw/main.c . . . . .	3037
17.9 examples/sys/isr/main.c . . . . .	3038
17.10 examples/clntsrv/server.cc . . . . .	3039

17.11 examples/clntsrv/client.cc . . . . .	3040
17.12 examples/clntsrv/clntsrv.cfg . . . . .	3041
17.13 examples/libs/l4re/c/ma+rm.c . . . . .	3041
17.14 examples/libs/l4re/c++/mem_alloc/ma+rm.cc . . . . .	3042
17.15 examples/libs/l4re/c++/shared_ds/ds_clnt.cc . . . . .	3044
17.16 examples/libs/l4re/c++/shared_ds/ds_srv.cc . . . . .	3045
17.17 examples/libs/l4re/c++/shared_ds/shared_ds.cfg . . . . .	3047
17.18 examples/libs/l4re/streammap/server.cc . . . . .	3048
17.19 examples/libs/l4re/streammap/client.cc . . . . .	3049
17.20 examples/libs/l4re/streammap/streammap.cfg . . . . .	3050
17.21 examples/libs/libirq/loop.c . . . . .	3050
17.22 examples/libs/libirq/async_isr.c . . . . .	3051
17.23 examples/sys/migrate/thread_migrate.cc . . . . .	3052
17.24 examples/sys/migrate/thread_migrate.cfg . . . . .	3053
17.25 tmpfs/lib/src/fs.cc . . . . .	3053
17.26 examples/libs/shmc/prodcons.c . . . . .	3060
<b>Index</b>	<b>3063</b>





# Chapter 1

## Overview

Welcome to the documentation of the L4Re Operating System Framework, or L4Re for short. There are two parts to this documentation: a user manual, which provides a birds eye view of L4Re and its environment, and a reference section which documents the complete programming API.

### User Manual

1. [Introduction](#) shortly explains the concept of microkernels and introduces the basic terminology.
2. [Tutorial](#) helps you getting started with setting up the development environment and writing your own first L4Re application.
3. [Programming for L4Re](#) explains in detail the most important programming concepts.
4. [L4Re Servers](#) provides a quick overview over standard services running on the L4Re operating system.

### Reference

The second part provides the complete reference of all classes and functions of the L4Re Operating System Framework as well as a list of example code.



# Chapter 2

## Introduction

The intention of this section is to provide a short overview about the [L4Re](#) Operating System Framework.

The general structure of a microkernel-based system will be introduced and the principal functionality of the servers in the basic environment outlined.

### 2.1 L4Re Microkernel

The [L4Re](#) Microkernel is the lowest-level component of software running in an L4Re-based system. The microkernel is the only component that runs in privileged processor mode. It does not include complex services such as program loading, device drivers, or file systems; those are implemented in user-level programs on top of it (a basic set of these services and abstractions is provided by the [L4](#) Runtime Environment).

Microkernel services are implemented in kernel objects. Tasks hold references to kernel objects in their respective *"object space"*, which is a kernel-protected table. These references are called *capabilities*. System calls to the microkernel are function invocations on kernel objects through the corresponding capabilities. These can be thought of as function invocations on object references in an object-oriented programming environment. Furthermore, if a task owns a capability, it may grant other tasks the same (or fewer) rights on this object by passing the capability from its own to the other task's object space.

From a design perspective, capabilities are a concept that enables flexibility in the system structure. A thread that invokes an object through a capability does not need to care about where this object is implemented. In fact, it is possible to implement all objects either in the kernel or in a user-level server and replace one implementation with the other transparently for clients.

#### 2.1.1 Communication

The basic communication mechanism in L4-based systems is called *"Inter Process Communication (IPC)"*. It is always synchronous, i.e. both communication partners need to actively rendezvous for IPC. In addition to transmitting arbitrary data between threads, IPC is also used to resolve hardware exceptions, faults and for virtual memory management.

### 2.1.2 Kernel Objects

The following list gives a short overview of the kernel objects provided by the [L4Re](#) Microkernel:

- **Task** A task comprises a memory address space (represented by the task's page table), an object space (holding the kernel protected capabilities), and on x86 an IO-port address space.
- **Thread** A thread is bound to a task and executes code. Multiple threads can coexist in one task and are scheduled by the microkernel's scheduler.
- **Factory** A factory is used by applications to create new kernel objects. Access to a factory is required to create any new kernel object. Factories can control and restrict object creation.
- **IPC Gate** An IPC gate is used to create a secure communication channel between different tasks. It embeds a label (kernel protected payload) that securely identifies the gate through which a message is received. The gate label is not visible to and cannot be altered by the sender.
- **IRQ** IRQ objects provide access to hardware interrupts. Additionally, programs can create new virtual interrupt objects and trigger them. This allows to implement a signaling mechanism. The receiver cannot decide whether the interrupt is a physical or virtual one.
- **Vcon** Provides access to the in-kernel debugging console (input and output). There is only one such object in the kernel and it is only available, if the kernel is built with debugging enabled. This object is typically interposed through a user-level service or without debugging in the kernel can be completely based on user-level services.
- **Scheduler** Implements scheduling policy and assignment of threads to CPUs, including CPU statistics.

## 2.2 L4Re System Structure

The system has a multi-tier architecture consisting of the following layers depicted in the figure below:

- **Microkernel** The microkernel is the component at the lowest level of the software stack. It is the only piece of software that is running in the privileged mode of the processor.
- **Tasks** Tasks are the basic containers (address spaces) in which system services and applications are executed. They run in the processor's deprivileged user mode.

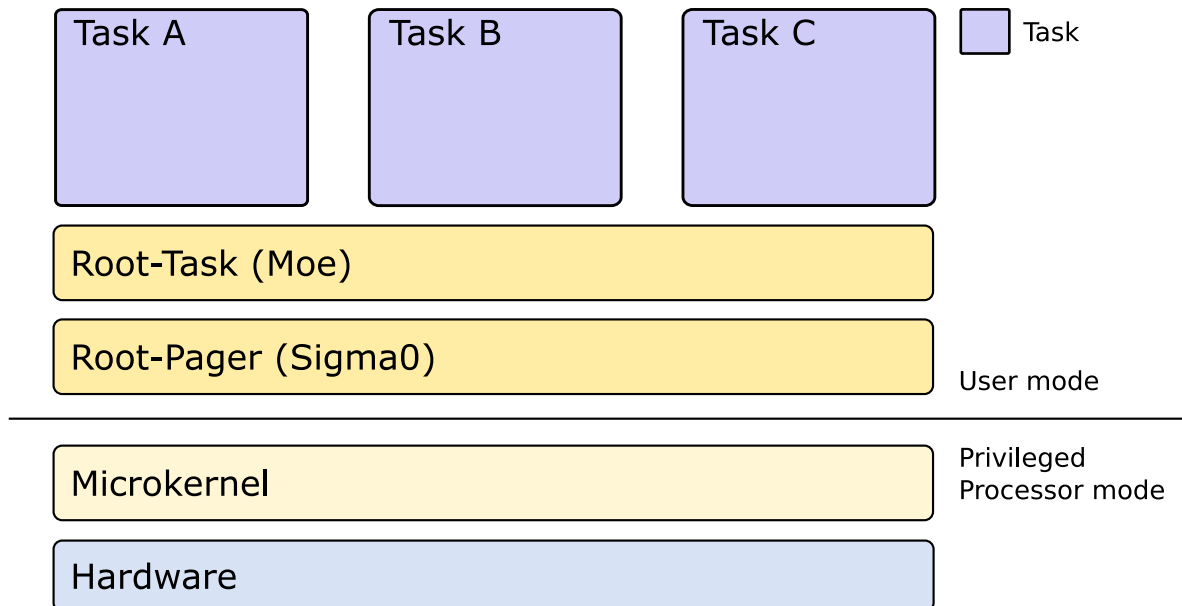


Figure 2.1 Basic Structure of an L4Re based system

In terms of functionality, the system is structured as follows:

- Microkernel** The kernel provides primitives to execute programs in tasks, to enforce isolation among them, and to provide means of secure communication in order to let them cooperate. As the kernel is the most privileged, security-critical software component in the system, it is a general design goal to make it as small as possible in order to reduce its attack surface. It provides only a minimal set of mechanisms that are necessary to support applications.
- Runtime Environment** The small kernel offers a concise set of interfaces, but these are not necessarily suited for building applications directly on top of it. The [L4Re](#) Runtime Environment aims at providing more convenient abstractions for application development. It comprises low-level software components that interface directly with the microkernel. The root pager *sigma0* and the root task *Moe* are the most basic components of the [L4Re](#) Runtime Environment. Other services (e.g., for device enumeration) use interfaces provided by them.
- Applications** Applications run on top of the system and use services provided by the runtime environment – or by other applications. There may be several types of applications in the system and even virtual machine monitors and device drivers are considered applications in the terminology used in this document. They are running alongside other applications on the system.

Lending terminology from the distributed systems area, applications offering services to other applications are usually called *servers*, whereas applications using those services are named *clients*. Being in both roles is also common, for instance, a file system server may be viewed as a server with respect to clients using the file system, while the server itself may also act as a client of a hard disk driver.

## 2.3 L4Re Runtime Environment

The [L4Re](#) Runtime Environment provides a basic set of services and abstractions, which are useful to implement and run user-level applications on top of the [L4Re](#) Microkernel. They form the [L4Re](#) Operating System Framework.

The [L4Re](#) Operating System Framework consists of a set of libraries and servers. [L4Re](#) follows an object-oriented design. Server interfaces are object-oriented, and the implementation is also object-oriented.

A minimal L4Re-based application needs 3 components to be booted beforehand: the [L4Re](#) Microkernel, the root pager (Sigma0), and the root task (Moe). The Sigma0 root pager initially owns all system resources, but is usually used only to resolve page faults for the Moe root task. Moe provides the essential services to normal user applications such as an initial program loader, a region-map service for virtual memory management, and a memory (data space) allocator.

# Chapter 3

## Tutorial

This tutorial assumes that the reader is familiar with the basic L4 concepts that were discussed in the [Introduction](#) section.

Here you can find the first steps to boot a very simple setup. The setup consists of the following components:

- [L4Re](#) Microkernel
- Sigma0 — Root Pager
- Moe — Root Task
- Ned — Init Process
- hello — The classical 'Hello World' Application

The guide assumes that you already compiled the base components and describes how to generate an ISO image, with GRUB as a boot loader, that can for example be booted within QEMU.

First you need a `modules.list` file that contains an entry for the scenario.

```
modaddr 0x002000000

entry hello
  kernel fiasco -serial_esc
  roottask moe rom/hello.cfg
  module l4re
  module ned
  module hello.cfg
  module hello
```

This file describes all the binaries and scripts to put into the ISO image, and also describes the GRUB `menu.lst` contents. What you need to do is to set the `make` variable `MODULE_SEARCH_PATH` to contain the path to your [L4Re](#) Microkernel's build directory and the directory containing your `hello.cfg` script.

The `hello.cfg` script should look like the following. A ready to use version can be found in `l4/conf/examples`.

```
local L4 = require("L4");
L4.default_loader:start({}, "rom/hello");
```

The first line of this script ensures that the [L4](#) package is available for the script. The second line uses the default loader object defined in that package and starts the binary `rom/hello`.

### Note

All modules defined in `modules.list` are available as data spaces ([L4Re::Dataspace](#)) and registered in a name space ([L4Re::Namespace](#)). This name space is in turn available as 'rom' to the init process ([Ned](#)).

Now you can go to your [L4Re](#) build directory and run the following command.

**Note**

The example assumes that you have created the `modules.list` and `hello.cfg` files in the `/tmp` directory. Adapt if you created them somewhere else.

```
make grub2iso E=hello MODULES_LIST=/tmp/modules.list MODULE_SEARCH_PATH=/tmp:<path_to_fiasco_builddir>
```

Now you should be able to boot the image in QEMU by running:

```
qemu-system-x86_64 -m 1024 -cdrom images/hello.iso -serial stdio
```

If you press `<ESC>` in the terminal that shows you the serial output you enter the microkernel's debugger... Have fun.

## 3.1 Customizations

A basic set of bootable entries can be found in `l4/conf/modules.list`. This file is the default for any image creation as shown above. It is recommended that local modification regarding image creation are done in `conf/Makeconf.boot`. Initially you may copy `Makeconf.boot.example` to `Makeconf.boot`. You can overwrite `MODULES_LIST` to set your own modules-list file. Set `MODULE_SEARCH_PATH` to your setup according to the examples given in the file. When configured a `make` call is reduced to:

```
make grub2iso E=hello
```

All other local configuration can be done in a `Makeconf.local` file located in the `l4` directory.



## Chapter 4

# Programming for L4Re

This part of the documentation discusses the concept of microkernel-based programming in more detail.

You should already have a basic understanding of the [L4Re](#) programming environment from the tutorial.

- [L4 Inter-Process Communication \(IPC\)](#)
- [Capabilities and Naming](#)
- [Spaces and Mappings](#)
- [Initial Environment and Application Bootstrapping](#)
- [Memory management - Data Spaces and the Region Map](#)
- [Program Input and Output](#)
- [Initial Memory Allocator and Factory](#)
- [Application and Server Building Blocks](#)
- [Pthread Support](#)
- tasks and threads
- communication channels
- server loops
- [Interface Definition Language](#)
- hardware access
- [L4Re Build System](#)
- [Kernel Factory](#)

### 4.1 L4 Inter-Process Communication (IPC)

Inter-process communication (IPC) is the fundamental communication mechanism in the [L4Re](#) Microkernel.

Basically IPC invokes a subroutine in a different context using input and output parameters. It is used to communicate between userland threads and, as a special case, to communicate between a userland thread and a kernel object. IPC provides the only (non-debugging) way of doing system calls.

### 4.1.1 IPC mechanism

When using this API, an IPC operation can be conducted using the `l4_ipc()` function (or one of its related [helpers](#)). In general it causes a method to be invoked on the called kernel object. An IPC operation consists of a send and receive phase, but either of them can be skipped, that is, an IPC operation can consist of only either a send or a receive phase. IPC is always synchronous and blocking and can be given a timeout. Timeouts can be specified separately for each phase. Invoking the IPC syscall without any phase is deprecated.

On the lowest abstraction level, IPC operations need the following arguments:

- [flags](#) describing the IPC mode,
- the [capability selector](#) of the communication partner,
- a [label](#),
- a [message tag](#), and
- a pair of [timeout](#) values.

During an IPC operation the kernel accesses the UTCB of the current thread to read parameters which are not passed as direct arguments.

As result of an IPC operation the kernel returns a message tag and a label and the kernel also changes UTCB content. For the detailed call signature, refer to `l4_ipc()`. Furthermore, depending on the IPC parameters, the kernel might have transferred the FPU state and capabilities (memory, I/O ports, and/or object capabilities) from the sender to the receiving thread.

The transition between the IPC send phase and the IPC receive phase is atomic, that is, as soon as the send phase has finished, the thread receive phase starts. A relative receive timeout does not start before the send phase has finished (see also below) and a thread which received an IPC call from another thread can assume that the other thread is ready to receive the reply message and the replying thread can therefore reply with a timeout of zero, see [IPC Timeouts](#).

For performance optimization and under certain conditions, the kernel may perform a context switch from the IPC sender to the IPC receiver without consulting the scheduler after the send phase finished. For instance, a client performing an IPC call to a server has to wait for the server anyway. Hence, after the client request was sent to the server, the kernel switches directly to the server thread. This behavior can be disabled by setting the `L4_MSGTAG_SCHEDULE` flag in the sender message tag (see below).

#### 4.1.1.1 IPC Flags

The *flags* defined in `l4_syscall_flags_t` are used by the invoking thread to define the intended IPC operation. The variants of `l4_ipc()` (see [Object Invocation](#)) use the flags

- to request the IPC phases (send-only IPC, receive-only IPC, IPC with send and receive phase), and
- to decide, if the reply capability (see [below](#)) should be used instead of the capability of a dedicated kernel object as target for the send phase (*reply*), and
- to decide, if receiving should wait for an incoming message from any possible sender (*open wait*) instead of a message from a dedicated sender (*closed wait*).

#### 4.1.1.2 Partner capability selector

The *partner capability selector* defines a kernel object as partner of the IPC operation. Some kernel objects forward IPC to a userland thread.

Basically an object capability is represented by `l4_cap_idx_t` where the bits starting from `L4_CAP_SHIFT` are used as index into the local capability table of the current address space.

Specifying `L4_INVALID_CAP` as target for an IPC operation is equivalent to specifying a thread capability of the current thread with full permissions. As a result, the userland thread either communicates with its corresponding kernel thread object (if `L4_PROTO_THREAD` is specified as protocol value, see the description of the message tag below) or the IPC target is the current userland thread. In the latter case, no IPC will be performed and the send phase and the receive phase will block until the corresponding timeout has expired, see below.

A special partner is defined by the *reply capability*. Since it would be impractical (and a security flaw) to always pass an explicit object capability to reply to for each IPC, the kernel generates an implicit one that can be used for just that purpose if the IPC contains an **open wait** phase. The reply capability is valid after a receive phase and points to the kernel object that sent the IPC just received. It can be used only once. The reply capability is selected by setting the `L4_SYSF_REPLY` flag, see above.

#### 4.1.1.3 IPC Label

The IPC label is a machine word which is transferred unchanged from the IPC sender to the IPC receiver when directly sending to a userland thread. However, the primary purpose of the label is to create a relationship between an `L4::Rcv_endpoint` (`L4::lpc_gate` or `L4::lrq`) and the bound thread.

During `L4::Rcv_endpoint::bind_thread()`, a label is specified. If the thread receives an IPC message through the endpoint, that label is delivered to the receiving thread as output parameter of `l4_ipc()` instead of the label specified during the corresponding IPC send operation (see the detailed description of `L4::lpc_gate` for more details on the label with IPC gates). The label is usually used by the receiving thread to invoke the object which is responsible for handling the IPC request of the corresponding endpoint. This mechanism is used by the `L4::Epiface` mechanism for server loops.

#### 4.1.1.4 IPC Message Tag

The *message tag* (`l4_msgtag_t`) describes the payload of the IPC and can also be used to enable certain features. It contains:

- a *protocol value* (cf. `l4_msgtag_t::label()`, also called the tag's *label*),
- the number of items in *UTCB message registers* to transfer (cg. `l4_msgtag_t::words()` and `l4_msgtag_t::items()`), and
- flags (cf. `l4_msgtag_t::flags()` and `L4_msgtag_flags`, may be 0).

The information from the message tag is required by the kernel to transfer the message. The IPC system call returns a message tag as result of the IPC operation. On success, a copy of the message tag specified by the sender is returned if there is a receive phase. Without receive phase, a successful IPC syscall returns the message tag specified as input parameter.

Failures during IPC are signalled using the `L4_MSGTAG_ERROR` flag in the message tag. If this bit is set by the kernel, the content of the returned message tag apart from that bit is undefined and the kernel wrote the actual error code into the `l4_thread_regs_t::error` register of the UTCB (see also *IPC Thread Control Registers*). When an IPC error occurs after the rendezvous of the IPC partners, both partners observe the same error information. If, for

instance, the IPC was aborted using `L4::Thread::ex_regs()`, the sender gets an `L4_IPC_SECANCELED` error while the receiver gets an `L4_IPC_RECANCELED` error. The function `L4Re::chkipc()` can be used to verify that an IPC operation finished successfully: It throws an error if the IPC failed.

The *protocol value* is usually used to distinguish between different protocols of the same interface. Certain protocol IDs are pre-defined when talking to kernel objects, see `L4_msgtag_protocol`. From IPC point of view, the protocol value is just payload that is transferred from sender to receiver and hence doesn't have a dedicated meaning.

By convention, during IPC calls, the protocol value is used for return values, where negative values signify errors. See the [section](#) about L4 RPC return types for further information. The function `L4Re::chksys()` can be used to verify that an RPC call using IPC was successful: It throws an error if the IPC failed or if the returned protocol value is negative.

#### 4.1.1.5 IPC Timeouts

As written above, IPC *timeouts* are specified separately for the send phase (IPC send timeout) and the receive phase (IPC receive timeout). The timeout of one phase is encapsulated by `l4_timeout_s`. Both timeouts are combined into a single `l4_timeout_t` parameter. Timeouts are either relative to the current time of the invoking thread or absolute. In the latter case, the absolute time of the deadline of the respective phase is written into a UTCB buffer register (see `l4_timeout_abs()`). The relative timeout of the receive phase starts immediately after the send phase has finished.

Two specific timeout values are sufficient for most IPC operations:

- `L4_IPC_TIMEOUT_NEVER` is used if the IPC partner might not yet be ready. Usually a client trusting a server will perform an IPC call with an infinite timeout for both phases. The send phase will take some time if the IPC receiver is currently not ready. The receive phase will take some time as the server needs time to serve the request. Also, a server will usually wait with an infinite receive timeout for the next request (see below for a possible exception).
- `L4_IPC_TIMEOUT_0` is used when it is either certain that the IPC receiver is currently ready or if the IPC sender doesn't want to wait if the IPC receiver is not ready. The former case applies to a thread which was called with an IPC call, for example a server got a client request. The reply to the IPC call should use a timeout of zero to ensure that a client doesn't block a server (server could not deliver the result of the request). See also `L4::ipc_srv::Default_timeout`. Another case is an IPC send operation for waking up another thread from an IPC receive operation. If the other thread is not ready to receive, then it might be already woken up and it does not make sense to wait any longer. Also triggering an IRQ is usually done with a send timeout of 0, see `L4::Triggerable::trigger()`.

In certain cases it also makes sense to specify an IPC timeout different from "never" or "zero":

- A server might leave the server loop after some time to perform idle work (see `L4::ipc_srv::Server_iface::add_timeout()`).
- A thread does not want to wait for an infinite time if the partner is not ready. This could be also some safety measure.
- A thread wants to block for a certain amount of time without consuming CPU time. The `l4_ipc_sleep()` function specifies the `L4_INVALID_CAP` as target for an IPC receive operation and specifies the intended relative waiting period as IPC timeout. Waiting for an absolute timeout would be possible with similar code.

#### Note

The kernel IPC path is optimized for the two special cases using `L4_IPC_TIMEOUT_NEVER` and `L4_IPC_TIMEOUT_0`. Specifying a different timeout causes more maintenance effort for the kernel.

Special care is required if a finite timeout for the receive phase of an IPC call is specified: The IPC receive operation could abort before the partner was able to send the reply message. Under certain circumstances the partner may still have the temporary reply capability to the calling thread and may use this capability to reply to the caller at a later, unexpected time specifying an arbitrary IPC label. This case is relevant for servers which call another, possibly untrusted, server while serving a client request.

#### 4.1.1.6 User-level Thread Control Block

The **UTCB** is located on a power-of-2-sized and power-of-2-aligned memory area shared between userland and the kernel (`L4::Task::add_ku_mem()`). The UTCB is bound to a thread during the `L4::Thread::control()` operation with the `L4::Thread::Attr` parameters set up using `L4::Thread::Attr::bind()`. The UTCB is used for IPC-related data exchange and is set up before invoking `l4_ipc()`. To access certain parts of the UTCB, the corresponding functions have to be used (there is no data type `L4_utcb` or similar). The UTCB consists of:

- the **message registers** `MR[0]`, `MR[1]`, ..., `MR[n-1]` with  $n = \text{L4\_UTCB\_GENERIC\_DATA\_SIZE}$  (access using `l4_utcb_mr()`),
- the **buffer descriptor register** `BDR` (access using `l4_utcb_br()`, see `l4_buf_regs_t::bdr`),
- the **buffer registers** `BR[0]`, `BR[1]`, ..., `BR[m-1]` with  $m = \text{L4\_UTCB\_GENERIC\_BUFFERS\_SIZE}$  (access using `l4_utcb_br()`),
- the **thread control registers** (access using `l4_utcb_tcr()`, includes the IPC error code), and
- in case of an exception IPC, the register state of the thread which triggered the exception (access using `l4_utcb_exc()`).

IPC to a kernel object requires the setup of the UTCB of the corresponding userlevel thread. IPC between userlevel threads requires the setup of UTCBs of both partners.

The kernel changes only the following UTCB content:

- The message registers of the UTCB of the receiver of an IPC, and
- the IPC error field of the thread invoking `l4_ipc()` if there was an error during IPC.

##### 4.1.1.6.1 IPC Message registers

The *message registers* contain *untyped items* and *typed items*. The sender's typed items are interpreted by the kernel in conjunction with the receiver's *receive items*. Each typed send item occupies two message registers. The untyped items, on the other hand, are free to be used by the communication partners to exchange data: The content of all message registers used for untyped items (`l4_msgtag_t::words()`) is copied from the UTCB of the IPC sender to the UTCB of the IPC receiver.

A typed send item consists of a *flexpage* (see `l4_fpage()`, `l4_obj_fpage()`, and `l4_iofpage()`) of the to be transferred capabilities (*flexpage word*) and a *message word*. There are two types of send items: *map items* and *void items*. For a void item, the message word is all zero. For a map item, the message word contains:

- the *compound bit* allowing to use the same receive buffer for the subsequent typed send item (scatter-gather behavior, see `L4_ITEM_CONT` of `l4_msg_item_consts_t`),
- the *type bit* defining this typed send item as a *map item*,
- the *grant flag* for delegating the access to the flexpage from the sender to the receiver and atomically removing the rights from the sender (see `l4_msg_item_consts_t`),
- *attributes* with semantics depending on the item type; for memory mappings, they contain cacheability information (see `l4_fpage_cacheability_opt_t`); for objects, they contain additional rights (see `L4_obj_fpage_ctl`),
- the *send base* (also called *hot spot*) defining what is actually mapped when send and receive flexpages have a different size.

A typed send item can be created using `l4_sndfpage_add()`. This function sets the compound bit unconditionally. Alternatively, the functions `l4_map_control()` and `l4_map_obj_control()` can be used to set up the message word of a map item for a memory flexpage respective for objects.

See [below](#) for a description how typed items are transferred.

#### 4.1.1.6.2 IPC Buffer Registers

The *buffer registers* and *buffer descriptor register* are interpreted by the kernel during the receive phase (if any). Buffer registers define *receive items* which are required to receive typed send items (memory, I/O ports or object capabilities). To specify a receive item, either one or two buffer registers are required:

- A *small buffer item* ([L4::lpc::Small\\_buf](#)) occupying one buffer register is sufficient to receive one object capability.
- A *buffer item* ([L4::lpc::Buf\\_item](#)) occupying two buffer registers (*message word* and a *flexpage*) is required to receive memory flexpages, I/O ports, or multiple object capabilities.

#### 4.1.1.6.3 IPC Buffer Descriptor Register

The buffer descriptor register defines indices of buffer registers used to receive dedicated types of send items and also contains a flag:

- 5 bits starting at [L4\\_BDR\\_MEM\\_SHIFT](#) define the index of the first receive item used for memory flexpages.
- 5 bits starting at [L4\\_BDR\\_IO\\_SHIFT](#) define the index of the first receive item used for I/O flexpages.
- 5 bits starting at [L4\\_BDR\\_OBJ\\_SHIFT](#) define the index of the first receive item used for object flexpages.
- The [L4\\_UTCB\\_INHERIT\\_FPU](#) can be set using [l4\\_utcb\\_inherit\\_fpu\(\)](#) and allows to receive the FPU state from the IPC sender. This is only relevant if the sender set [L4\\_MSGTAG\\_TRANSFER\\_FPU](#) in the message tag.

For most use cases, a BDR value of zero is sufficient. In that case, if `BR[0]` contains a void item, no capabilities are received. Otherwise, only one type of capabilities (memory, I/O ports or objects) can be received even if there are several receive items. For more complex setups that require receiving different types of capabilities in one receive operation, other BDR values are necessary.

The BDR of the receiving thread is only used by the kernel if at least one typed item is transferred during the IPC or if [L4\\_MSGTAG\\_TRANSFER\\_FPU](#) is set in the UTCB of the sending thread.

#### 4.1.1.6.4 IPC Thread Control Registers

The [l4\\_thread\\_regs\\_t::error](#) register contains the IPC error code in case the [L4\\_MSGTAG\\_ERROR](#) flag is set in the message tag returned by an IPC syscall. Otherwise this register is not touched by the kernel. See [l4\\_ipc\\_tcr\\_error\\_t](#) for a detailed enumeration of all possible error codes during an IPC operation.

The other members of [l4\\_thread\\_regs\\_t](#) are never touched by the kernel.

#### 4.1.1.7 Transfer of Typed Send Items

The kernel interprets all typed items in the sender UTCB ([l4\\_msgtag\\_t::items\(\)](#)) and performs the following operations while modifying the corresponding typed items in the receiver UTCB:

- If the message item of the sender is void, this item is ignored and the message word of the corresponding typed item in the receiver UTCB is set to zero (making it a void item). The flexpage word of this item is not changed.
- Otherwise, if the type bit of the message item of the sender is not set, the IPC is aborted with [L4\\_IPC\\_SEMSGCUT](#) / [L4\\_IPC\\_REMSGCUT](#).
- Otherwise, if there is a receive item corresponding to the flexpage type of the send item available (see [IPC Buffer Descriptor Register](#)), information described by the flexpage is transferred to the receiver.

In the last case, the message word of the typed item in the receiver UTCB is modified to contain the send base, the type and the size of the transferred flexpage, as well as a copy of the compound bit and the type bit of the send item. If the receiver ordered a local ID in the corresponding receive item, the kernel attempts to apply special rules, see [L4\\_RCV\\_ITEM\\_LOCAL\\_ID](#). Otherwise, regular mappings as described by the flexpage of the send item are created in the receiver space.

A receive item forms a *receive window* of a specific address and size in the receiver space. Each typed send item that is a map item requires a corresponding receive item. By default, there is a one-to-one mapping (one receive item per typed send item) but it is also possible to use one receive item to receive several typed send items: The compound bit (see [l4\\_msg\\_item\\_consts\\_t](#)) of a send item defines if the following typed send item shall use the same receive item as the current one for receiving the flexpage. If the compound bit is set, proper values of the send base shall be used to prevent overlapping of addresses in the receiver space.

The send base is relevant when the size of the receive flexpage differs from the size of the transferred resource. As a typical example, triggering a memory page fault opens a receive window covering the entire memory address space of the faulting thread. The pager will usually reply a memory flexpage smaller than the entire address space of the faulting thread. Hence, the pager has to specify a proper base which is used as offset of the sent object in the receive flexpage in the *receiver space*. If the sender flexpage is bigger than the receive window, a flexpage of the size of the receive window starting at the send base in the *sender space* is transferred to the receiver.

The kernel will stop transmission of typed items before [l4\\_msgtag\\_t::items\(\)](#) is reached either if it finds a void item as receive buffer or if the flexpage type of the send item does not match the flexpage type of the corresponding receive item. Under both conditions, the IPC is aborted with [L4\\_IPC\\_SEMSGCUT](#) / [L4\\_IPC\\_REMSGCUT](#).

#### Note

The kernel ignores the flexpage access rights of the receive items. The actual rights for a transferred resource in the target address space are defined by the access rights to that resource of the IPC sender address space and the flexpage access rights in the typed send item. Additionally, when transferring object capabilities, the transferred rights also depend on the sender's rights on the capability invoked for IPC.

The kernel does not unmap capabilities in the receive window when there is no capability present at the corresponding index at the sender. Further, the receiver cannot reliably detect at which capability indices it received capabilities in its receive windows. Therefore, before receiving from an untrusted source, all receive windows should be cleared. Otherwise the receiver may erroneously associate a capability in one of its receive windows with his last IPC partner although it was actually received in an earlier IPC.

However, the kernel indicates if at least one object capability was received or not, see [L4::lpc::Gen\\_fpage::cap\\_received\(\)](#).

## 4.1.2 Examples

A number of examples show the interplay of the concepts introduced so far.

#### 4.1.2.1 User Thread to Kernel Object

The `L4::Scheduler` kernel object has a method `L4::Scheduler.idle_time()`. It takes a set of CPUs to query, represented by two machine words.

In user space, the function `L4::Cap<L4::Scheduler>->idle_time()` is called, which does the following:

- Fill `MR[0]` with a constant identifying the method being called (`L4_SCHEDULER_IDLE_TIME_OP`).
- Fill `MR[1]` and `MR[2]` with the two words making up the CPU set.
- Set up the message tag with the protocol value (`L4_PROTO_SCHEDULER`), the number of untyped and typed items (3 and 0), and some flags.
- Call `l4_ipc()` with the partner capability ID, the tag, the pointer to the filled UTCB, infinite timeouts, and with flags indicating a send and receive phase. (The label does not matter in this case.)

This function traps into kernel space using standard platform specific mechanisms. The kernel reads the protocol value on the message tag, checks that the partner capability ID refers to a valid object that speaks that protocol, and dispatches the message to the appropriate handler. The handler fills the first 64 bits of the message registers with the computed time value. This will cover `MR[0]` on 64-bit architectures and `MR[0]` and `MR[1]` on 32-bit architectures. It then sets up the return message tag:

- The number of untyped items is 1 or 2.
- The number of typed items is 0.
- The protocol value contains the return value and is set to 0.
- An error would be signalled as a negative protocol value. Under certain conditions (e.g. wrong kernel object capability specified), the error is signalled as IPC error (see `L4_MSGTAG_ERROR` in the description of the `IPC Message Tag`).
- (The return label is as irrelevant in this case as the send label.)

This reply is received by the receive phase of the original `l4_ipc()` call from userland. Finally the `l4_ipc()` function copies the time value out of the message registers and forwards it with a possible error from the message tag flags to its caller.

#### 4.1.2.2 User Thread to User Thread

When the target kernel object is of type `L4::Thread` (or `L4::ipc_gate`, but we will cover this in the example below) and the message tag's protocol value is not `L4_PROTO_THREAD`, the kernel will forward the IPC message to the userland thread represented by the kernel object. There it can be received by a call to `l4_ipc()`. The restriction of the protocol number is necessary because one also wants to invoke `L4::Thread`'s control methods such as `L4::Thread.switch_to()` or `L4::Thread.ex_regs()`. Besides this restriction, the interpretation of all the IPC parameters and the untyped items of the UTCB is up to the communication partners.



#### 4.1.2.2.1 Simple Messages

A simple example is a client calling a server to have a computation performed on a value: Similar to IPC from a userland thread to a kernel object, the client writes the value to `MR[0]`. It sets up the message tag with some agreed upon protocol value (which, as explained above, must be different from `L4_PROTO_THREAD`), number of untyped items to 1, typed items to 0, and no flags set. The client may want to pass a label that identifies itself, as many clients can use the server. In this context, the identifier might also be passed via the message registers, but the label is the proper place for this, as it gets a special treatment by the kernel for IPC gates (covered by the example below). The client then calls `l4_ipc()` with the tag, label and flags indicating it wants a send phase and a receive phase (as it wants a result back). The target is the capability index referring to a capability for the `L4::Thread` kernel object of the server.

To be able to receive an IPC message, the server has set up a UTCB of its own and called `l4_ipc()` with flags indicating it only wants to enter a receive phase and it accepts IPCs from any partner. This is called an **open wait**. (The alternative would be to specify a capability index referring to a `L4::Thread` capability to exclusively receive from.)

Both system calls (the send IPC initiated by the client and the receive IPC initiated by the server) may be seen by the kernel in any order but the IPC will not start before sender and receiver are ready. In that case the kernel will copy the relevant message registers the client specified in the message tag from the client's UTCB to the server's UTCB, in the current example just `MR[0]`. It will then switch the client to the receive phase of its call (which cannot yet be executed) and return the server's call with the message tag and label.

The server inspects the tag for the correct protocol value and number of untyped items passed, inspect the label to decide whether it maybe wants special treatment of this particular client, performs the computation on `MR[0]` and writes the result back to `MR[0]` (or to more words, depending on what exactly it computes). It sets up the tag in the usual way, but probably needs to pass no label, as the client knows who it is talking to.

For the reply, the server makes use the reply capability (see above). Since the client sent the last IPC to the server, the reply capability will point to it. So when the server calls `l4_ipc()` with the computed result in the message registers and using the reply capability as target, the kernel knows to forward this to the client's thread. The kernel copies the message registers from the server's UTCB to the client's UTCB, and the client's `l4_ipc()` system call, which is still stuck in the receive phase, is returned with the tag.

The client looks at the tag and then the message registers for its wanted result and the example is concluded.

#### 4.1.2.2.2 Send Items

IPC between userland threads is also used to transfer typed items: Memory, I/O ports, and objects, all represented as flexpages. Typed items and untyped items can be part of the same IPC. As general rule, the sender specifies what he wants to send, the receiver where and how much it wants to receive, and the kernel checks the required permissions before doing the actual transfer. As written before, this mechanism is synchronous and the receiver cannot be transferred items against its will.

See also

[Flex pages](#)

Suppose a client wants a server to have read only access to a page of its memory. The client sets up a flexpage covering the page and with only the `L4_FPAGE_RO` right set. The server sets up a flexpage of a memory region where it will receive the mapping. This may be larger than one page, suppose for our case four pages, in which case the exact position of the mapping will be resolved by the send base provided by the sender. The client combines the hot spot and some flags into a machine word and writes it to `MR[0]` (see also `l4_map_control()`). At `MR[1]` follows the flexpage it wants to send (see also `l4_fpage()`). The server does almost the same but writes the words to `BR[0]` and `BR[1]`. (The server could also specify a hot spot, but it is currently ignored by the kernel.) The

client specifies 1 typed and 0 untyped items in the message tag. The server writes 0 to `BDR` to specify that the first receive item starts at the first buffer register.

Both client and server initiate their IPC, the client with only a send phase to the server, and the server with an open wait receive phase. The kernel checks the compatibility of the send items and the receive buffers (e.g. that no object capability flexpage is sent to a receive buffer describing a memory mapping flexpage) and updates its internal structures to reflect the change. In our case, the sender's hot spot indicates to which of the four pages that make up the receive buffer the sent page should be mapped. The kernel also translates the typed send item to the server's address space and stores it in the server's UTCB at `MR[0]` and `MR[1]`.

Once the server returns from its syscall, it will have read access to the client's shared page.

#### 4.1.2.3 User Thread to User Object

A common use case for thread to thread communication is when a server implements a number of object interfaces and a client wants to invoke methods on them. For security reasons, the server does not want to hand out its thread capability to everyone it nonetheless wants to serve. It also may not want to allow every client access to everyone of its interfaces. For this purpose, IPC gates implemented by the kernel object `L4::ipc_gate` can be used. An IPC gate can be bound to a thread and forwards IPC to it. In doing so it applies two transformations

1. It sets the label to a predefined value.
2. It changes the rights of transferred items (see `L4_CAP_FPAGE_S`).

For each object of every interface the server implements, it creates an IPC gate and binds it to itself (see `L4::ipc_gate::bind_thread()`). It sets the gate's label to a unique value identifying the object. Then it hands the gate out to clients. Several clients can be handed the same gate and will all end up invoking methods on the same object.

Instead of setting the target as the server's thread kernel object, the client uses the IPC gate's instead. The label the client sends is irrelevant, as the gate will overwrite it with the value the server has set during the bind operation. The server executes an open wait, and the kernel performs the same operation as in the above [example](#) with the transformed IPC finally ending up in the server's thread.

The server checks that the received label refers to one of its objects. It then checks if the protocol value in the message tag matches the interface the object implements. Then it invokes the method specified in `MR[0]` with the rest of the arguments. Finally it returns the results via UTCB and message tag to the reply capability and waits for the next IPC.

## 4.2 Capabilities and Naming

The [L4Re](#) system is a capability based system which uses and offers capabilities to implement fine-grained access control.

Generally, owning a capability means to be allowed to communicate with the object the capability points to. All user-visible kernel objects, such as tasks, threads, and IRQs, can only be accessed through a capability. Please refer to the [Kernel Objects](#) documentation for details. Capabilities are stored in per-task capability tables (the object space) and are referenced by capability selectors or object flex pages. In a simplified view, a capability selector is a natural number indexing into the capability table of the current task.

As a matter of fact, a system designed solely based on capabilities uses so-called 'local names' because each task can only access those objects made available to this task. Other objects are not visible to and accessible by the task.

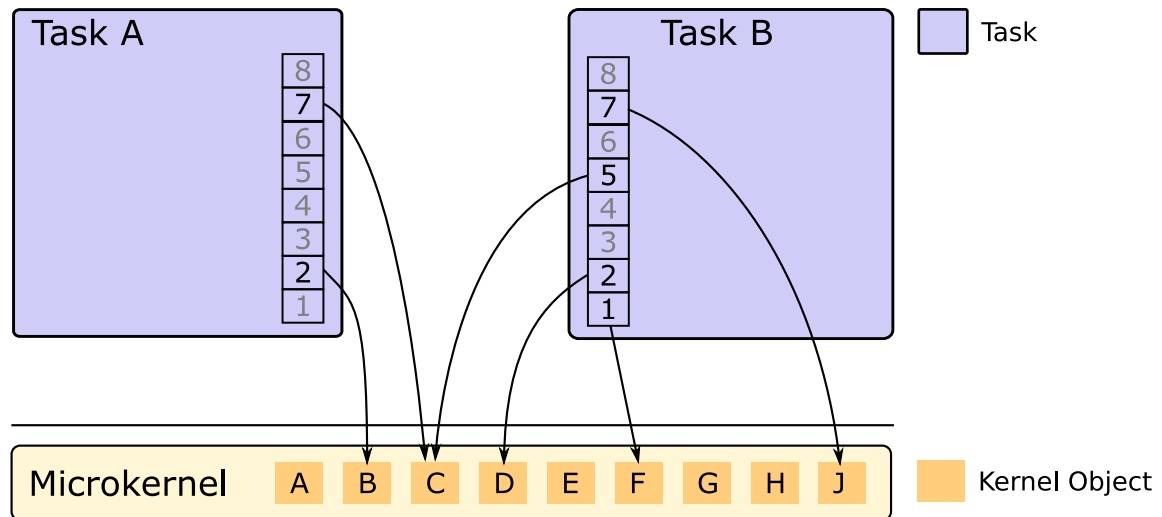


Figure 4.1 Capabilities and Local Naming in L4

So how does an application get access to a service? In general all applications are started with an initial set of available objects. This set of objects is predetermined by the creator of a new application process and granted directly to the new task before starting the first application thread. The application can then use these initial objects to request access to further objects or to transfer capabilities to its own objects to other applications. A central [L4Re](#) object for exchanging capabilities at runtime is the name-space object, implementing a store of named capabilities.

From a security perspective, the set of initial capabilities (access rights to objects) completely define the execution environment of an application. Mandatory security policies can be defined by well known properties of the initial objects and carefully handled access rights to them.

## 4.3 Spaces and Mappings

Each task in the [L4Re](#) system has access to two resource spaces (three on IA32) which are maintained by the kernel.

These are the

1. object space,
2. memory space, and
3. IO-port space (only on IA32).

The entities addressed in each space are capabilities to objects, virtual memory pages, and IO ports. The addresses are unsigned integers and the largest valid address depends on which space is referenced, the hardware, and the configuration of the kernel. Although a program can access memory at byte granularity, from the kernel's point of view the address granularity in the memory space is not bytes but pages, as determined by the hardware. The address of a capability is also called its "capability slot".

Flexpages describe a range in any of the spaces that has a power-of-two length and is also aligned to this length. They additionally hold access rights information and further space specific information.

When a resource is present at some address in a task's corresponding resource space, then we say that resource is mapped to that task. For example, a capability to the task's main thread may be mapped to capability slot 5, or the first page of the code segment a thread executes is mapped to virtual memory page 12345. However, there need not be any resource mapped to an address.

Tasks can exchange resources through a process called "mapping" during IPC and using the [L4::Task::map\(\)](#) method. The sending task specifies a send flexpage and the receiving task a receive flexpage. The resources mapped to the send flexpage will then be mapped to the receive flexpage by the kernel.

Memory mappings and IO port mappings are hierarchical: If a resource of such a type is subject of a map operation, the received mapping is a child mapping of the corresponding mapping in the sending task (parent mapping). The kernel usually respects the relationship between these two mappings (granting is an exception; see below): If rights of a parent mapping are revoked using [L4::Task::unmap\(\)](#), these rights are also removed from its child mappings. Also, if a mapping is completely removed (via [L4::Task::unmap\(\)](#) or by mapping something else at its place), then also all child mappings are removed. In contrast, revoking rights of a child mapping leaves the rights of its parent mapping untouched.

The mapping of a resource can be performed as *grant* operation (see [L4\\_MAP\\_ITEM\\_GRANT](#)): Such an operation includes the removal of all involved mappings from the send flexpage (basically a move operation). While with a map operation without grant the mapping in the send flexpage remains the parent of all child mappings (including the new child mapping in the receive flexpage), a grant operation moves the mappings covered by the send flexpage to the corresponding addresses from the receive flexpage while leaving the parent/child relationship of the moved mappings with other mappings untouched.

During a map operation at most the access rights of the source mapping(s) can be transferred but no additional rights can be added. So only rights that are present in the source mapping and that are specified in the send item/flexpage are transferred. This also holds for grant mappings, however, rights revocation is *not* guaranteed to be applied to descendant mappings in case of grant.

There are cases where a grant operation is not or cannot be performed as requested; see [L4\\_MAP\\_ITEM\\_GRANT](#) for details.

Object capabilities are not hierarchical – they have no children. The result of the map operation on an object capability is a copy of that capability in the object space of the destination task.

## 4.4 Initial Environment and Application Bootstrapping

New applications that are started by a loader conforming to [L4Re](#) get provided an [Initial Environment](#).

This environment comprises a set of capabilities to initial [L4Re](#) objects that are required to bootstrap and run this application. These capabilities include:

- A capability to an initial memory allocator for obtaining memory in the form of data spaces
- A capability to a factory which can be used to create additional kernel objects
- A capability to a Vcon object for debugging output and maybe input

- A set of named capabilities to application specific objects

During the bootstrapping of the application, the loader establishes data spaces for each individual region in the ELF binary. These include data spaces for the code and data sections, and a data space backed with RAM for the stack of the program's first thread.

One loader implementation is the `moe` root task. Moe usually starts an *init* process that is responsible for coordinating the further boot process. The default *init* process is `ned`, which implements a script-based configuration and startup of other processes. Ned uses Lua (<http://www.lua.org>) as its scripting language, see [Ned Script example](#) for more details.

#### 4.4.1 Configuring an application before startup

The default [L4Re](#) init process (Ned) provides a Lua script based configuration of initial capabilities and application startup. Ned itself also has a set of initial objects available that can be used to create the environment for an application. The most important object is a kernel object factory that allows creation of kernel objects such as IPC gates (communication channels), tasks, threads, etc. Ned uses Lua tables (associative arrays) to represent sets of capabilities that shall be granted to application processes.

```
local caps = {
    name = some_capability
}
```

The [L4](#) Lua package in Ned also has support functions to create application tasks, region-map objects, etc. to start an ELF binary in a new task. The package also contains Lua bindings for basic [L4Re](#) objects, for example, to generic factory objects, which are used to create kernel objects and also user-level objects provided by user-level servers.

```
L4.default_loader:start({ caps = { some_service = service } }, "rom/program --arg");
```

#### 4.4.2 Connecting clients and servers

In general, a connection between a client and a server is represented by a communication channel (IPC gate) that is available to both of them. You can see the simplest connection between a client and a server in the following example.

```
local loader = L4.default_loader; -- which is Moe
local svc = loader:new_channel(); -- create an IPC gate
loader:start({ caps = { service = svc:svr() } }, "rom/my_server");
loader:start({ caps = { service = svc:m("rw") } }, "rom/my_client");
```

As you can see in the snippet, the first action is to create a new channel (IPC gate) using `loader:new_channel()`. The capability to the gate is stored in the variable `svc`. Then the binary `my_server` is started in a new task, and full (`:svr()`) access to the IPC gate is granted to the server as initial object. The gate is accessible to the server application as "service" in the set of its initial capabilities. Virtually in parallel a second task, running the client application, is started and also given access to the IPC gate with less rights (`:m("rw")`, note, this is essential). The server can now receive messages via the IPC gate and provide some service and the client can call operations on the IPC gate to communicate with the server.

Services that keep client specific state need to implement per-client server objects. Usually it is the responsibility of some authority (e.g., Ned) to request such an object from the service via a generic factory object that the service provides initially.

```
local loader = L4.default_loader; -- which is Moe
local svc = loader:new_channel():m("rws"); -- create an IPC gate with rws rights
loader:start({ caps = { service = svc:svr() } }, "rom/my-service");
loader:start({ caps = { foo_service = svc:create(object_to_create, "param") } }, "rom/client");
```

This example is quite similar to the first one, however, the difference is that Ned itself calls the `create` method on the factory object provided by the server and passes the returned capability of that request as "foo\_service" to the client process.

#### Note

The `svc:create(...)` call blocks on the server. This means the script execution blocks until the my-service application handles the create request.

## 4.5 Memory management - Data Spaces and the Region Map

### 4.5.1 User-level paging

Memory management in L4-based systems is done by user-level applications, the role is usually called *pager*. Tasks can give other tasks full or restricted access rights to parts of their own memory. The kernel offers means to give access to memory in a secure way, often referred to as *memory mapping*.

The mapping mechanism allows one task to resolve page faults of another: A thread usually has a pager assigned to it. When the thread causes a page fault, the kernel sends an IPC message to the pager with information about the page fault. The pager answers this IPC by either providing a backing page, or with an error. The kernel will map the backing page into the address space of the faulting thread's task.

These mechanisms can be used to construct a memory and paging hierarchy among tasks. The root of the hierarchy is `sigma0`, which initially gets all system resources and hands them out once on a first-come-first-served basis. Memory resources can be mapped between tasks at a page-size granularity. This size is predetermined by the CPU's memory management unit and is commonly set to 4 kB.

#### 4.5.1.1 Data spaces

A data space is the [L4Re](#) abstraction for objects which may be accessed in a memory mapped fashion (i.e., using normal memory read and write instructions). Examples include the sections of a binary which the loader attaches to the application's address space, files in the ROM or on disk provided by a file server, the registers of memory-mapped devices and anonymous memory such as the heap or the stack.

Anonymous memory data spaces in particular (but in general all data spaces except memory mapped IO) can either be constructed entirely from a portion of the RAM or the current working set may be multiplexed on some portion of the RAM. In the first case it is possible to eagerly insert all pages (more precisely page-frame capabilities) into the application's address space such that no further page faults occur when this data space is accessed. In general, however, only the pages for some portion are provided and further pages are inserted by the pager as a result of page faults.

#### 4.5.1.2 Virtual Memory Handling

The virtual memory of each task is constructed from data spaces backing virtual memory regions (VMRs). The management of the VMRs is provided by an object called *region map*. A dedicated region-map object is associated with each task; it allows attaching and detaching data spaces to an address space as well as reserving areas of virtual memory. Since the region-map object possesses all knowledge about the virtual memory layout of a task, it also serves as an application's default pager.

#### 4.5.1.3 Memory Allocation

Operating systems commonly use anonymous memory for implementing dynamic memory allocation (e.g., using `malloc` or `new`). In an L4Re-based system, each task gets assigned a memory allocator providing anonymous memory using data spaces.

See also

[L4Re::Dataspace](#) and [L4Re::Rm](#).

## 4.6 Program Input and Output

The initial environment provides a Vcon capability used as the standard input/output stream.

Output is usually connected to the parent of the program and displayed as debugging output. The standard output is also used as a back end to the C-style printf functions and the C++ streams.

Vcon services are implemented in Moe and the loader as well as by the [L4Re](#) Microkernel and connected either to the serial line or to the screen if available.

See also

[Virtual Console](#)

## 4.7 Initial Memory Allocator and Factory

The purpose of the memory allocator and of the factory is to provide the application with the means to allocate memory (in the form of data spaces) and kernel objects respectively.

An initial memory allocator and an initial factory are accessible via the initial [L4Re](#) environment.

See also

[L4Re::Mem\\_alloc](#)

The factory is a kernel object that provides the ability to create new kernel objects dynamically. A factory imposes a resource limit for kernel memory, and is thus a means to prevent denial of service attacks on kernel resources. A factory can also be used to create new factory objects.

See also

[Factory](#)

## 4.8 Application and Server Building Blocks

So far we have discussed the environment of applications in which a single thread runs and which may invoke services provided through their initial objects.

In the following we describe some building blocks to extend the application in various dimensions and to eventually implement a server which implements user-level objects that may in turn be accessed by other applications and servers.

### 4.8.1 Creating Additional Application Threads

To create application threads, one must allocate a stack on which this thread may execute, create a thread kernel object and setup the information required at startup time (instruction pointer, stack pointer, etc.). In [L4Re](#) this functionality is encapsulated in the pthread library.

## 4.8.2 Providing a Service

In capability systems, services are typically provided by transferring a capability to those applications that are authorised to access the object to which the capability refers to.

Let us discuss an example to illustrate how two parties can communicate with each other: Assume a simple file server, which implements an interface for accessing individual files: `read(pos, buf, length)` and `write(pos, data, length)`.

L4Re provides support for building servers based on the class `L4::Server_object`. `L4::Server_object` provides an abstract interface to be used with the `L4::Server` class. Specific server objects such as, in our case, files inherit from `L4::Server_object`. Let us call this class `File_object`. When invoked upon receiving a message, the `L4::Server` will automatically identify the corresponding server object based on the capability that has been provided to its clients and invoke this object's `dispatch` function with the incoming message as a parameter. Based on this message, the server must then decide which of the protocols it implements was invoked (if any). Usually, it will evaluate a protocol specific opcode that clients are required to transmit as one of the first words in the message. For example, assume our server assigns the following opcodes: `Read = 0` and `Write = 1`. The `dispatch` function calls the corresponding server function (i.e., `File_object::read()` or `File_object::write()`), which will in turn parse additional parameters given to the function. In our case, this would be the position and the amount of data to be read or written. In case the write function was called the server will now update the contents of the file with the data supplied. In case of a read it will store the requested part of the file in the message buffer. A reply to the client finishes the client request.

## 4.9 Pthread Support

L4Re supports the standard pthread library functionality.

Therefore L4Re itself does not contain any documentation for pthreads itself. Please refer to the standard pthread documentation instead.

The L4Re specific parts will be described herein.

- Include pthread-l4.h header file:

```
#include <pthread-l4.h>
```

- Return the local thread capability of a pthread thread:

Use `pthread_l4_cap(pthread_t t)` to get the capability index of the pthread t.

For example:

```
pthread_l4_cap(pthread_self());
```

- Setting the L4 priority of an L4 thread works with a special scheduling policy (other policies do not affect the L4 thread priority):

```
pthread_t t;
pthread_attr_t a;
struct sched_param sp;

pthread_attr_init(&a);
sp.sched_priority = l4_priority;
pthread_attr_setschedpolicy(&a, SCHED_L4);
pthread_attr_setschedparam(&a, &sp);
pthread_attr_setinheritsched(&a, PTHREAD_EXPLICIT_SCHED);

if (pthread_create(&t, &a, pthread_func, NULL))
    // failure...

pthread_attr_destroy(&a);
```



- You can prevent your pthread from running immediately after the call to `pthread_create(..)` by adding `PTHREAD_L4_ATTR_NO_START` to the `create_flags` of the pthread attributes. To finally start the thread you need to call `scheduler()->run_thread()` passing the capability of the pthread and scheduling parameters.

```
pthread_t t;
pthread_attr_t attr;

pthread_attr_init(&attr);
attr.create_flags |= PTHREAD_L4_ATTR_NO_START;

if (pthread_create(&t, &attr, pthread_func, nullptr))
    // failure...

pthread_attr_destroy(&attr);

// do stuff

auto ret = L4Re::Env::env()->scheduler()->run_thread(pthread_l4_cap(t),
                                                    l4_sched_param(2));

if (l4_error(ret))
    // failure...
```

### Constraints on pthread\_t, user-land capability slot, and kernel thread-object

- `pthread_l4_cap()` is guaranteed to return the valid capability slot of the pthread (A) until `pthread_join()` or `pthread_detach()` is invoked on (A)'s `pthread_t`.
- `pthread_l4_cap()` exposes internal state of the pthread management, take the necessary precautions as you would for any shared data in concurrent environments. If you use `pthread_l4_cap()` guarding against concurrency issues is your duty.
- There is no guarantee that a valid capability slot points to a present capability.

#### • Example

It is possible to obtain a valid thread capability slot and for `l4_task_cap_valid()` to return the capability as not present. The following example showcases a possible sequence of events.

```
// Assume: void some_func(void *)
pthread_t pthread = nullptr;
pthread_create(&pthread, nullptr, some_func, nullptr);

// pthread running some_func()
l4_cap_idx_t cap_idx = pthread_l4_cap(pthread);
l4_is_valid_cap(cap_idx); // --> true

long valid = l4_task_cap_valid(L4RE_THIS_TASK_CAP, cap_idx).label();
// valid == 1 --> capability object is present (refers to a kernel object).

// some_func() exits

cap_idx = pthread_l4_cap(pthread);
l4_is_valid_cap(cap_idx); // --> true

valid = l4_task_cap_valid(L4RE_THIS_TASK_CAP, cap_idx).label();
// valid == 0 --> capability object is not present (refers to no kernel object).

pthread_join(pthread, nullptr); // invalidates the cap slot and frees
                                // the pthread's data structures

// using cap_idx here is undefined behavior.
```

## 4.10 Interface Definition Language

An interface definition in [L4Re](#) is normally declared as a class derived from [L4::Kobject\\_t](#).

For example, the simple calculator example declares its interface like that:

```
struct Calc : L4::Kobject_t<Calc, L4::Kobject>
{
    L4_INLINE_RPC(int, sub, (l4_uint32_t a, l4_uint32_t b, l4_uint32_t *res));
```

```
L4_INLINE_RPC(int, neg, (l4_uint32_t a, l4_uint32_t *res));

typedef L4::Typeid::Rpc<sub_t, neg_t> Rpc<sub_t, neg_t>;
```

The signature of each function is first declared using one of the RPC macros (see below) and then all the functions need to be listed in the `Rpcs` type.

Clients invoke these functions with the name given to the RPC macros, `sub` and `neg` above. Servers implement them by defining functions with an `op_` prepended, `op_sub` and `op_neg`. The types of the parameters in the macro definition, on the server side, and on the client side are not the same. The following section describes how they are related to each other.

### 4.10.1 Parameter types for RPC

Generally all value parameters, const reference parameters, and const pointer parameters to an RPC interface are considered as input parameters for the RPC and are transmitted from the client to the server.

#### Note

This means that `char const *` is treated as an input `char` and not as a zero terminated string value, for strings see `L4::lpc::String<>`.

Parameters that are non-const references or non-const pointers are treated as output parameters going from the server to the client.

There are special data types that appear on only one side (client or server) when used, see the following table for details.

```
L4_RPC(long, test, (int arg1, char const *arg2, unsigned *ret1));
```

The example shows the declaration of a method called `test` with `long` as return type, `arg1` is an `int` input, `arg2` a `char` input, and `ret1` an `unsigned` output parameter.

Type	Direction	Client-Type	Server-Type
T	Input	T	T
T const &	Input	T const &	T const &
T const *	Input	T const *	T const &
T &	Output	T &	T &
T *	Output	T *	T &
<a href="#">L4::Ipc::In_out&lt;T &amp;&gt;</a>	In/Out	T &	T &
<a href="#">L4::Ipc::In_out&lt;T *&gt;</a>	In/Out	T *	T &
<a href="#">L4::Ipc::Cap&lt;T&gt;</a>	Input	<a href="#">L4::Ipc::Cap&lt;T&gt;</a>	<a href="#">L4::Ipc::Snd_fpage</a>
<a href="#">L4::Ipc::Out&lt;L4::Cap&lt;T&gt; &gt;</a>	Output	<a href="#">L4::Cap&lt;T&gt;</a>	<a href="#">L4::Ipc::Cap&lt;T&gt; &amp;</a>
<a href="#">L4::Ipc::Rcv_fpage</a>	Input	<a href="#">L4::Ipc::Rcv_fpage</a>	void
<a href="#">L4::Ipc::Small_buf</a>	Input	<a href="#">L4::Ipc::Small_buf</a>	void

Array types can be used to transmit arrays of variable length. They can either be stored in a client-provided buffer ([L4::lpc::Array](#)), copied into a server-provided buffer ([L4::lpc::Array\\_in\\_buf](#)) or directly read and written into the UTCB ([L4::lpc::Array\\_ref](#)). For latter type, the start position of an array type needs to be known in advance. That implies that only one such array can be transmitted per direction and it must be the last part in the message.

Type	Direction	Client-Type	Server-Type
<code>L4::Ipc::Array&lt;const T&gt;</code>	Input	<code>L4::Ipc::Array&lt;const T&gt;</code>	<code>L4::Ipc::Array_ref&lt;const T&gt;</code>
<code>L4::Ipc::Array&lt;const T&gt;</code>	Input	<code>L4::Ipc::Array&lt;const T&gt;</code>	<code>L4::Ipc::Array_in_buf&lt;const T&gt; &amp;</code>
<code>L4::Ipc::Array&lt;T&gt; &amp;</code>	Output	<code>L4::Ipc::Array&lt;T&gt; &amp;</code>	<code>L4::Ipc::Array_ref&lt;T&gt; &amp;</code>
<code>L4::Ipc::Array_ref&lt;T&gt; &amp;</code>	Output	<code>L4::Ipc::Array_ref&lt;T&gt; &amp;</code>	<code>L4::Ipc::Array_ref&lt;T&gt; &amp;</code>

Finally, there are some optional types where the sender can choose if the parameter should be included in the message. These types are for the implementation of some legacy message formats and should generally not be needed for the definition of ordinary interfaces.

Type	Direction	Client-Type	Server-Type
<code>L4::Ipc::Opt&lt;T&gt;</code>	Input	<code>L4::Ipc::Opt&lt;T&gt;</code>	<code>T</code>
<code>L4::Ipc::Opt&lt;const T*&gt;</code>	Input	<code>L4::Ipc::Opt&lt;const T*&gt;</code>	<code>T</code>
<code>L4::Ipc::Opt&lt;T &amp;&gt;</code>	Output	<code>T &amp;</code>	<code>L4::Ipc::Opt&lt;T&gt; &amp;</code>
<code>L4::Ipc::Opt&lt;T *&gt;</code>	Output	<code>T *</code>	<code>L4::Ipc::Opt&lt;T&gt; &amp;</code>
<code>L4::Ipc::Opt&lt;Array↔_ref&lt;T&gt; &amp;&gt;</code>	Output	<code>Array_ref&lt;T&gt; &amp;</code>	<code>L4::Ipc::Opt&lt;Array↔_ref&lt;T&gt;&gt; &amp;</code>

### 4.10.2 RPC Return Types

On the server side, the return type of an RPC handling function is always `long`. The return value is transmitted via the label field in `l4_msgtag_t` and is therefore restricted to its length. Per convention, a negative return value is interpreted as an error condition. If the return value is negative, output parameters are not transmitted back to the client.

#### Attention

The client must never rely on the content of output parameters when the return value is negative.

On the client-side, the return value of the RPC is set as defined in the RPC macro. If `l4_msgtag_t` is given, then the client has access to the full message tag, otherwise the return type should be `long`. Note that the client might not only receive the server return value in response but also an IPC error code.

### 4.10.3 RPC Method Declaration

RPC member functions can be declared using one of the following C++ macros.

For inline RPC stubs, where the RPC stub code itself is `inline`:

- `L4_INLINE_RPC(res, name, (args...), flags)`  
Define an inline RPC call (type and callable).
- `L4_INLINE_RPC_OP(op, res, name, (args...), flags)`  
Define an inline RPC call with specific opcode (type and callable).
- `L4_INLINE_RPC_NF(res, name, (args...), flags)`  
Define an inline RPC call type (the type only, no callable).

- `L4_INLINE_RPC_NF_OP`(opcode, Ret\_type, func\_name, (args...), flags)

Define an inline RPC call type with specific opcode (the type only, no callable).

For external RPC stubs, where the RPC stub code must be defined in a separate compile unit (usually a `.cc` file):

- `L4_RPC`(Ret\_type, func\_name, (args...), flags)  
Define an RPC call (type and callable).
- `L4_RPC_OP`(opcode, Ret\_type, func\_name, (args...), flags)  
Define an RPC call with specific opcode (type and callable).
- `L4_RPC_NF`(Ret\_type, func\_name, (args...), flags)  
Define an RPC call type (the type only, no callable).
- `L4_RPC_NF_OP`(opcode, Ret\_type, func\_name, (args...), flags)  
Define an RPC call type with specific opcode (the type only, no callable).

To generate the implementation of an external RPC stub:

- `L4_RPC_DEF`(class\_name::func\_name)  
Generate the definition of an RPC stub.

The NF versions of the macro generally do not generate a callable member function named `<name>` but do only generate the type `<name>_t`. This data type can be used to call the RPC stub explicitly using `<name>_t::call(L4::Cap<Iface_class> cap, args...)`.

## 4.11 L4Re Build System

L4Re uses a custom make-based build system, often simply referred to as *BID*.

This section explains how to use BID when writing applications and libraries for L4Re.

### 4.11.1 Building L4Re

#### Setting up the Build Directory

L4Re must be built out-of-source. Therefore the first mandatory step is creating and populating a build directory. From the root of the L4Re source tree run

```
make B=<builddir>
```

Other targets that can be executed in the source directory are

**update**

Update the source directory from svn. Only makes sense when you have downloaded L4Re from the official subversion repository.

**help**

Show a short help with the most important targets.

## Invoking Make

Once the build directory is set up, BID make can be invoked in one of two ways:

1. Go to the build directory and invoke make without special options.
2. Go to a source directory with a BID make file and invoke `make O=<builddir> ....`

The default target builds the source (as you would expect), other targets that are available in build mode are

### cleanfast

Quickly cleans the build directory by removing all subdirectories that contain generated files. The configuration will remain untouched.

### clean

Remove generated files. Slower than `make cleanfast` but can be used on selected packages. Use `S=...` to select the target package.

In addition to these targets, there are a number of targets to generate images which are explained elsewhere.

## 4.11.2 Writing BID Make Files

The BID build system exports different roles that define what should be done in the subdirectory. So a BID make file essentially consists of defining the role and a number of role-dependent make variables. The basic layout should look like this:

```
PKGDIR  ?= <path to package's root directory> # e.g., '.' or '..'
L4DIR   ?= <path to L4Re source directory>    # e.g. '$(PKGDIR)/../../'

<various definitions>

include $(L4DIR)/mk/<role>.mk
```

`PKGDIR` in the first line defines the root directory of the current package. `L4DIR` in the next line must be pointed to the root of the [L4Re](#) source tree the package should be built against. After this custom variable definitions for the role follow. In the final line of the file, the make file with the role-specific rules must be sourced.

The following roles are currently defined:

- `project.mk` - Sub-project Role
- `subdir.mk` - Directory Role
- `prog.mk` - [Application Role](#)
- `lib.mk` - Library Role
- `include.mk` - [Header File Role](#)
- `doc.mk` - Documentation Role
- `test.mk` - [Test Application Role](#)
- `idl.mk` - IDL File Role (currently unused)
- `runux.mk` - Tests in FiascoUX Role

## BID-global Variables

This section lists variables that configure how the BID build system behaves. They are applicable for all roles.

Variable	Description
CC	C compiler for target
CXX	C++ compiler for target
HOST_CC	C compiler for host
HOST_CXX	C++ compiler for host

### 4.11.3 prog.mk - Application Role

The prog role is used to build executable programs.

#### General Configuration Variables

The following variables can only be set globally for the Makefile:

MODE

Kind of target to build for. The following values are possible:

- `static` - build a statically linked binary (default)
- `shared` - build a dynamically linked binary
- `l4linux` - build a binary for running on L4Linux on the target platform
- `host` - build for host system
- `targetsys` - build a binary for the target platform with the compiler's default settings

SYSTEMS

List of architectures the target can be built for. The entries must be space-separated entries either naming an architecture (e.g. amd64) or an architecture and ABI (e.g. arm-l4f). When not defined, the target will be built for all possible platforms.

TARGET

Name or names of the binaries to compile. This variable may also be postfixed with a specific architecture.

### Target-specific Configuration Variables

The following variables may either be used with or without a description suffix. Without suffix they will be used for all operations. With a specific description their use is restricted to a subset. These specifications include a target file and the architecture, both optional but in this order, separated by underscores. The specific variables will be used in addition to the more general ones.

`SRC_C / SRC_CC / SRC_F / SRC_S`

.c, .cc, .f90, .S source files.

`REQUIRES_LIBS`

List of libraries the binary depends on. This works only with libraries that export a `pkg_config` configuration file. Automatically adds any required include and link options.

`DEPENDS_PKGS`

List of packages this binary depends on. If one these packages is missing then building of the binary will be skipped.

`CPPFLAGS / CFLAGS / CXXFLAGS / FFLAGS / ASFLAGS`

Options for the C preprocessor, C compiler, C++ compiler, Fortran compiler and assembler. When used with suffix, the referred element is the source file, not the target file.

`LDFLAGS`

Options for the linker ld.

`LIBS`

Additional libraries to link against (with -l).

`PRIVATE_LIBDIR`

Additional directories to search for libraries.

`CRT0 / CRTN`

(expert use only) Files containing custom startup and finish code.

`LDSCRIPT`

(expert use only) Custom link script to use.

#### 4.11.4 include.mk - Header File Role

The header file role is responsible for installing header file at the appropriate location.

The following variables can be used for customizing the process:

`INCSRC_DIR`

Source directory where the headers can be found. Default is the directory where the Makefile resides.

`TARGET`

List of header files to install. If left undefined, then `INCSRC_DIR` will be scanned for files with suffix `.h` or `.i`.

`EXTRA_TARGET`

When `TARGET` is undefined, then add these files to the headers found by scanning the source directory. Has no effect if `TARGET` has been defined.

`CONTRIB_HEADERS`

When set, the headers will be installed in `${BUILDDIR}/include/contrib/${PKGNAME}` rather than `${BUILDDIR}/include/l4/${PKGNAME}`.

`INSTALL_INC_PREFIX`

Base directory where to install the headers. Overwrites `CONTRIB_HEADERS`. The headers will then be found under `${BUILDDIR}/include/${INSTALL_INC_PREFIX}`.

`PC_FILENAME`

When set, a `pkg_config` configuration file is created with the given name.



### 4.11.5 test.mk - Test Application Role

The test role is very similar to the application role, it also builds an executable binary.

The difference is that it also builds for each target a test script that executes the test target either on the host (MODE=host) or a target platform (currently only qemu).

The role accepts all make variables that are accepted by the application role. The only difference is that the `TARGET` variable is not required. If it is missing, the source directory will be scanned for source files that fit the pattern `test_*.c[c]` and create one target for each of them.

#### Note

It is possible to still use `SRC_C[C]` when targets are determined automatically. In that case the specified sources will be used in addition\* to the main `test_*.c[c]` source.

In addition to the variables above, there are a number of variables that control how the test is executed. All these variables may be used as a global variable that applies to all test or, if the target name is added as a suffix, set for a specific target only.

#### TEST\_TARGET

Name of binary containing the test (default: same as `TARGET`).

#### TARGET\_\$ (ARCH)

When `TARGET` is undefined, these targets are added to the list of targets for the specified architecture. For all targets `SRC_C[C]` files must be defined separately.

#### TEST\_KERNEL\_ARGS

Arguments to append to the kernel command line. These are also appended when specifying custom ones via a `.t`-file's `-f` parameter or when using `-d`.

#### TEST\_EXPECTED

File containing expected output. By default the variable is empty, which means the test binary is expected to produce TAP test output, that can be directly processed. When the `TEST_TAP_PLUGINS` variable is given, `TEST_EXPECTED` is ignored.

#### TEST\_EXPECTED\_REPEAT

Number of times the expected output should be repeated, by default 1. When set to 0 then output is expected to repeat forever. This is particularly useful to make sure that stress tests that are meant to run in an endless loop are still alive. Note that such endless tests can only be run by directly executing the test script. They will be skipped when run in a test harness like `prove`.

`TEST_TAP_PLUGINS`

Specify the plugins that are used to process the output of the test run. The syntax is of the values is:

`plugin1:arg1=a,arg2=b plugin2:arg=foo`

Multiple plugins separated by a space are loaded in order. Spaces are not allowed inside a plugin specification. One or more arguments are optionally passed to the plugin separated by commas and delimited by a colon.

If the variable is not specified the plugins for `TAPOutput` and `OutputMatching` (depending on the `TEST_EXPECTED` variable) are automatically loaded.

For the supported plugins and their options please refer to their in-line documentation in `tool/lib/L4/TapWrapper/Plugin/`. The plugin name corresponds to the file stem name in that directory.

`TEST_TIMEOUT`

Non-standard timeout after which the test run is aborted (useful for tests involving sleep).

`NED_CFG`

LUA configuration file for startup to give to Ned

`REQUIRED_MODULES`

Additional modules needed to run the test. By adding `[opts]` to the name of a module you can add module options that are reflected in the generated `modules.list`.

`BOOTSTRAP_ARGS`

Additional parameters to supply to bootstrap.

`QEMU_ARGS`

Additional parameters to supply to QEMU.

`MOE_ARGS`

Additional parameters to supply to moe.

`TEST_ARGS`

Additional arguments for the `TEST_STARTER` (tapper-wrapper per default).

`TEST_ROOT_TASK`

Alternative root task to be used during a test instead of moe.

`TEST_ROOT_TASK_ARGS`

Arguments passed to `TEST_ROOT_TASK` if `TEST_ROOT_TASK` is different from moe.

`KERNEL_CONF`

Features the [L4Re](#) Microkernel must have been compiled with. A space-separated list of config options as used by Kconfig. `run_test` looks for a `globalconfig.out` file in the same directory as the kernel and checks that all options are enabled. If not, the test is skipped. Has only an effect if the `globalconfig.out` file is present.

`L4RE_CONF`

Features the [L4Re](#) userland must have been compiled with. A space-separated list of config options as used by Kconfig. `run_test` will look for these in the `.kconfig` file in the [L4Re](#) build directory.

`L4LINUX_CONF`

Features the L4Linux kernel must have been compiled with. Similar to `KERNEL_CONF` but checks for a `.config` file in the directory of the L4Linux kernel.

`TEST_SETUP`

Command to execute before the test is run. The test will only be executed if the command returns 0. If the exit code is 69, the test is marked as skipped with the reason provided in the final line of stdout.

`TEST_LOGFILE`

Append output of test execution to the given file unless `TEST_WORKDIR` is given.

`TEST_WORKDIR`

Create logs, temp and other files below the given directory. That directory is taken as base dir for more automatically created subdir levels using the current test path, in order to guarantee conflict-free usage when running many different tests with a common workdir. When `TEST_WORKDIR` is provided then `TEST_LOGFILE` is ignored as it is organized below workdir.

`TEST_TAGS`

List of conditions for tags provided during execution of a test. A tag can be set to 1, set to 0 or be unspecified via `TEST_RUN_TAGS` during execution. Therefore there are 4 possible conditions for a tag that can be specified in `TEST_TAGS`: `tag`, `!tag`, `+tag` and `-tag`. The following table shows the conditions they represent.

<code>TEST_RUN_TAGS \ TEST_TAGS</code>	<code>tag</code>	<code>!tag</code>	<code>+tag</code>	<code>-tag</code>
<code>tag</code> or <code>tag=1</code>	y		y	
unspecified		y	y	
<code>tag=0</code>		y		y

*Example usage:*

The tag `long-running` is used by tests which take a long time and should be skipped by default. These tests are marked with the tag `long-running` unprefixed.

The tag `hardware` is set to 1 at runtime when the tests will run on real hardware. Tests that must not run on real hardware are marked with `!hardware`.

The tag `+impl-def` is used by tests that test implementation details. Due to the nature of this flag we

require the "+" prefix to be used, so they are run by default but can be excluded from execution by setting `TEST_RUN_TAGS` to `impl-def=0` at runtime.

If you want to specify multiple tag conditions they need to be separated with a comma.

#### `TEST_PLATFORM_ALLOW` and `TEST_PLATFORM_DENY`

Deny and allow lists of platforms a test is banned from or limited to. If you list platforms in the `TEST_PLATFORM_ALLOW` variable the test will only be run on these listed platforms and will be skipped on any other platform. If you list platforms in the `TEST_PLATFORM_DENY` variable the test will be skipped on the listed platforms and will be run on any other platform. You can only use one of these variables per test, not both. See `mk/platforms/` for the various identifiers.

*Example usage:*

```
# Do not run this test on the Raspberry Pi platform
TEST_PLATFORM_DENY_test_xyz := rpi

# Only run this test on this test on the RCar3 platform.
TEST_PLATFORM_ALLOW_test_abc := rcar3
```

#### `TAPARCHIVE`

Filename for an archive file to store the resulting TAP output.

In addition to compiled tests, it is also possible to create tests where the test binary or script comes from a different source. These tests must be listed in `EXTRA_TARGET` and for each target a custom `TEST_TARGET` must be provided.

## Running Tests

The make role creates a test script which can be found in `<builddir>/test/t/<arch>/<api>`. It is possible to organise the tests further in subdirectories below by specifying a `TEST_GROUP`.

To be able to execute the test, a minimal test environment needs to be set up by exporting the following environment variables:

`KERNEL_<arch>`, `KERNEL`

[L4Re](#) Microkernel binary to use. The test runner is able to check if the kernel has all features necessary for the test and skip tests accordingly. In order for this to work, the `globalconfig.out` config file from the build directory needs to be available in the same directory as the kernel.

`L4LX_KERNEL_<arch>`, `L4LX_KERNEL`

L4Linux binary to use. This is only required to run tests in `mode=l4linux`. If no L4Linux kernel is set then these tests will simply be skipped. The test runner is also able to check if the kernel has all features compiled in that are required to run the test successfully (see make variable `L4LINUX_CONF` above). For this to work, the `.config` configuration file from the build directory needs to be available in the same directory as the kernel.

`LINUX_RAMDISK_<arch>, LINUX_RAMDISK`

Ramdisk to mount as root in L4Linux. This is only required to run tests in `mode=l4linux`. If not supplied, L4Linux tests will be skipped. The ramdisk must be set up to start the test directly after the initial startup is finished. The name of the test binary is supplied via the kernel command line option `l4re_testprog`. The `tool/test` directory contains an example script `launch-l4linux-test`, which can be copied onto the ramdisk and started by the init script.

`TEST_HWCONFIG` and `TEST_FIASCOCONFIG`

Some userland tests rely on external information about the underlying platform and the configuration of the L4Re Microkernel to decide whether or not to test specific features or to determine which and how much resources are available. Some examples for this are whether or not virtualization is supported by the platform, how many cores the platform has, how many cores the kernel supports or how much memory the platform provides. To convey this information to these tests you can set the two environment variables `TEST_HWCONFIG` and `TEST_FIASCOCONFIG`.

Using `TEST_HWCONFIG` requires a plain text document containing key-value pairs separated by a `=` symbol. On top of that comment lines starting with `#` are supported. Simply create a plain text file such as the following and set `TEST_HWCONFIG` to its absolute path.

```
VIRTUALIZATION=y
MP=y
NUM_CORES=4
MEMORY=2048
```

Using `TEST_FIASCOCONFIG` is easier since it only needs to contain the absolute path of the `globalconfig.out` file in the L4Re Microkernel's build directory. The build system will then extract the information when a test is started.

When starting a test the build system will read both files and provide their content as a lua table to the test. A test script can then make decisions based on them. To simplify some decisions the build system merges some information by itself, e.g. virtualization is only available if both the platform and the L4Re Microkernel support this feature. More details can be obtained from the perl module in `tool/lib/L4/TestEnvLua.pm`.

In addition to these variables, the following BID variables can be overwritten at runtime: `PT` (for the platform type) and `TEST_TIMEOUT`. You may also supply `QEMU_ARGS` and `MOE_ARGS` which will be appended to the parameters specified in the BID test make file.

Once the environment is set up, the tests can be run either by simply executing all of them from the build directory with

```
make test
```

or executing them directly, like

```
test/t/amd64_amdfam10/l4f/l4re-core/moe/test_namespace.t
```

or running one or more tests through the test harness `prove`, like

```
prove test/t/amd64_amdfam10/l4f/l4re-core/moe/test_namespace.t
prove -r test/t/amd64_amdfam10/l4f/l4re-core/
prove -rv test/t/amd64_amdfam10/l4f/l4re-core/
```

`TEST_TAGS` allow for a way to include or exclude whole groups of tests during execution, primarily with `prove`. You can specify which tests to run at runtime using one of the following ways:

```
$ test/t/amd64_amdfam10/l4f/l4re-core/test_one.t --run-tags slow,gtest-shuffle=0
$ test/t/amd64_amdfam10/l4f/l4re-core/test_one.t -T slow,gtest-shuffle=0
$ prove -r test/t/amd64_amdfam10/l4f/l4re-core/ :: -T slow,gtest-shuffle=0
$ TEST_RUN_TAGS=slow,gtest-shuffle=0 prove -r test/t/amd64_amdfam10/l4f/l4re-core/
```

For each test tag requirements defined in the corresponding `TEST_TAGS` Makefile variable are tested. If the requirements for tags do not match the test is skipped. The `SKIP` message will provide insight why the test was skipped:

```
$ make test
...
test/t/amd64_amdfam10/test_one.t .... ok
test/t/amd64_amdfam10/test_two.t .... skipped: Running this test requires tag slow to be set to 1.
test/t/amd64_amdfam10/test_three.t .. ok
```

When tags are provided, the tests requiring those tags are now also executed while the tests that forbid them are skipped:

```
$ TEST_RUN_TAGS=slow,gtest-shuffle
$ make test
...
test/t/amd64_amdfam10/test_one.t .... ok
test/t/amd64_amdfam10/test_two.t .... ok
test/t/amd64_amdfam10/test_three.t .. skipped: Running this test requires tag gtest-shuffle to be set to 0 or
```

For further details on how values in `TEST_TAGS` and `TEST_RUN_TAGS` interact, see the help text for `TEST_TAGS`.

## Running Tests in External Programs

You can hand-over test execution to an external program by setting the environment variable `EXTERNAL_TEST↵_STARTER` to the full path of that program:

```
export EXTERNAL_TEST_STARTER=/path/to/external/test-starter
make test
```

`EXTERNAL_TEST_STARTER`

This variable is evaluated by `tool/bin/run_test` (the backend behind `make test`) and contains the full path to the tool which actually starts the test instead of the test itself.

The `EXTERNAL_TEST_STARTER` can be any program instead of the default execution via `make qemu E=maketest`. Its output is taken by `run_test` as the test output.

Usually it is just a bridge to prepare the test execution, e.g., it could create the test as image and start that image via a simulator.

## Running Tests in a Simulator

Based on above mechanism there is a dedicated external test starter `tool/bin/teststarter-image-telnet.pl` shipped in BID which assumes an image to be started with another program which provides test execution output on a network port.

This can be used to execute tests in a simulator, like this:

```
export EXTERNAL_TEST_STARTER=$L4RE_SRC/tool/bin/teststarter-image-telnet.pl
export SIMULATOR_START=/path/to/configured/simulator-exe
make test
```

After building the image and starting the simulator it contacts the simulator via a network port (sometimes called "telnet" port) to pass-through its execution output as its own output so it gets captured by `run_test` as usual.

The following variables control `teststarter-image-telnet.pl`:

### `SIMULATOR_START`

This points to the full path of the program that actually starts the prepared test image. Most often this is the frontend script of your simulator environment which is pre-configured so that it actually works in the way that `teststarter-image-telnet.pl` expects from the following settings.

### `SIMULATOR_IMAGETYPE`

The image type to be generated via `make $SIMULATOR_IMAGETYPE E=maketest`. Default is `elfimage`.

### `SIMULATOR_HOST`

The simulator will be contacted via socket on that host to read its output. Default is `localhost`.

### `SIMULATOR_PORT`

The simulator will be contacted via socket on that port to read its output. Default is `11111`.

### `SIMULATOR_START_SLEEP_TIME`

After starting the simulator it waits that many seconds before reading from the port. Default is `1` (second).

## Running tests without taper-wrapper

In case you want to replace the taper-wrapper test starter, you can replace the default one by setting the environment variable `TEST_STARTER` to the path of your test starter. Then your test starter can use the same environment which is normally set up for the default starter, which includes environment variables provided by the build system as well as the test itself. Among these are `SEARCHPATH`, `MODE`, `ARCH`, `MOE_CFG`, `MOE_ARGS`, `TEST_TIMEOUT`, `TEST_TARGET`, `TEST_EXPECTED`, `QEMU_ARGS` and many more.

## Debugging Tests

The test script is only a thin wrapper that sets up the test environment as it was defined in the make file and then executes two scripts: `tapper-wrapper` and `run_test`.

The main work horse of the two is `tool/bin/run_test`. It collects the necessary files and starts qemu to execute the test. This script is always required.

There is then a second script wrapped around the test runner: `tool/bin/tapper-wrapper`. This tool inspects the output of the test runner and reformats it, so that it can be read by tools like `prove`. If the test produces tap output, then the script scans for this output and filters away all the debug output. If `TEST_EXPECTED` was defined, then the script scans the output for the expected lines and prints a suitable TAP message with success or failure. It also makes sure that qemu is killed as soon as the test is finished.

There are a number of command-line parameters that allow to quickly change test parameters for debugging purposes. Run the test with `'-help'` for more information about available parameters.

## 4.12 Kernel Factory

The kernel factory is a kernel object that provides the ability to create new kernel objects dynamically.

The kernel factory enforces a memory quota. This quota defines the maximum amount of kernel memory the factory service can use to construct the requested objects. When the quota is depleted, the factory refuses the creation of new objects.

The quota may be higher than the amount of available kernel memory; ultimately, the amount of available kernel memory is the strict limit for the factory to remain operational.

The kernel factory creates the following kinds of objects:

- [DMA space](#)
- [L4::Factory](#)
- [L4::lpc\\_gate](#) ([L4\\_PROTO\\_NONE](#), [L4\\_PROTO\\_KOBJECT](#))
- [L4::Irq](#) ([L4\\_PROTO\\_IRQ\\_SENDER](#))
- [L4::Semaphore](#)
- [L4::Task](#)
- [L4::Thread](#)
- [L4::Vm](#)

The protocol IDs for objects in this list are given in [L4\\_msgtag\\_protocol](#). Kernel objects whose protocol ID is not immediately clear from the documentation of [L4\\_msgtag\\_protocol](#) have their protocol IDs stated within parenthesis. As an exception, [L4::lpc\\_gate](#) can be identified by more than one protocol IDs. The protocol ID shall be used as the second argument for [L4::Factory.create](#)([Cap<void>](#), long, [l4\\_utcb\\_t](#) \*).

For the C++ interface see [L4::Factory](#), for the C interface see [Factory](#).



### 4.12.1 Passing parameters for the create stream

[L4::Factory.create\(\)](#) returns a [create stream](#) that allows arguments to be forwarded to the constructor of the object to be created.

Objects that support additional parameters on their creation are presented with a non-empty list of parameters. The parameters are listed in the order they should be provided to a create stream returned by [L4::Factory.create\(\)](#).

- [Dmar\\_space\(\)](#)
- [L4::Factory\(l4\\_umword\\_t\)](#)
  - Argument: factory quota (in bytes).
  - See [L4::Factory.create\\_factory\(\)](#) for details.
- [L4::lpc\\_gate\(\)](#)
  - Creates an unbound IPC gate.
  - Alternatively, an IPC gate can be immediately bound to a thread upon creation using [L4::Factory.create\\_gate\(\)](#).
- [L4::lirq\(\)](#)
- [L4::Semaphore\(\)](#)
- [L4::Task\(l4\\_fpage\\_t\)](#)
  - Argument: utcb\_area
  - See [L4::Factory.create\\_task\(\)](#) for details.
- [L4::Thread\(\)](#)
- [L4::Vm\(\)](#)



# Chapter 5

## L4Re Servers

Here you shall find a quick overview over the standard services running on the [L4Re](#) Microkernel.

### Sigma0, the Root Pager

Sigma0 is a special server running on [L4](#) because it is responsible of resolving page faults for the root task, the first useful task on [L4Re](#). Sigma0 can be seen as part of the kernel, however it runs in unprivileged mode. To run something useful on the [L4Re](#) Microkernel you usually need to run Sigma0, nevertheless it is possible to replace Sigma0 by a different implementation.

For more details see [Sigma0, the Root-Pager](#)

### Moe, the Root Task

Moe is our implementation of the [L4](#) root task that is responsible for bootstrapping the system, and to provide basic resource management services to the applications on top. Therefore Moe provides [L4Re](#) resource management and multiplexing services:

- **Memory** in the form of memory allocators ([L4Re::Mem\\_alloc](#), [L4::Factory](#)) and data spaces ([L4Re::Dataspace](#))
- **Cpu** in the form of basic scheduler objects ([L4::Scheduler](#))
- **Vcon** multiplexing for debug output (output only)
- **Virtual memory management** for applications, [L4Re::Rm](#)

Moe further provides an implementation of [L4Re](#) name spaces ([L4Re::Namespace](#)), which are for example used to provide a read only directory of all multi-boot modules. In the case of a boot loader, like grub that enables a VESA frame buffer, there is also a single instance of an [L4Re](#) graphics session ([L4Re::Goos](#)).

To start the system Moe starts a single ELF program, the init process. The init process (usually Ned, see the next section) gets access to all resources managed by Moe and to the Sigma0 root pager interface.

For more details see [Moe, the Root-Task](#).

## Ned, the Default Init Process

To keep the root task free from complicated scripting engines and to avoid circular dependencies in application startup (that could lead to dead locks) the configuration and startup of the real system is managed by an extra task, the init process.

Ned is such an init process that allows system configuration via Lua scripts.

For more information see [Ned](#).

## Io, the Platform and Device Resource Manager

Because all peripheral management in [L4Re](#) is done in user-level applications, there is the need to have a centralized management of the resources belonging to the platform and to peripheral devices.

This is the job of Io. Io provides portable abstractions for iterating and accessing devices and their resources (IRQ's, IO Memory...), as well as delegating access to those resources to other applications (e.g., device drivers).

For more details see [Io, the Io Server](#).

## Other Servers

The following additional server package are available on top of the core [L4Re](#) environment.

- [Rtc, the Real-Time Clock Server](#)  
is a simple multiplexer for real-time clock hardware on your platform.
- [fb-drv, the Low-Level Graphics Driver](#)  
provides low-level access and initialization of various graphics hardware. It has support for running VESA BIOS calls on Intel x86 platforms, as well as support for various ARM display controllers. `fb-drv` provides a single instance of the `L4Re::Goos` interface and can serve as a back end for the Mag server, in particular, if there is no graphics support in the boot loader.
- [AHCI driver](#)
- [Cons, the Console Multiplexer](#)  
An interactive multiplexer for console in- and output. It buffers the output from different L4 clients and allows to switch between them to redirect input.
- [Sigma0, the Root-Pager](#)
- [Mag, the GUI Multiplexer](#)  
Our default multiplexer for the graphics hardware is Mag. Mag is a Nitpicker (TODO: ref) derivate that allows secure multiplexing of the graphics and input hardware among multiple applications and multiple complete windowing environments.
- [NVMe server](#)
- [Uvmm, the virtual machine monitor](#)
- [l4vio\\_net\\_p2p, a virtual network point-to-point link](#)

## 5.1 Sigma0, the Root-Pager

Sigma0 is a special [L4](#) server that serves as the origin for mapping memory.

It is started by Fiasco.OC on the system boot and gets full access to all userland RAM and device memory. It functions as the pager (main memory provider) for Moe and as the provider for device memory for Io. Moe and Io are trusted and usually the only applications besides Ned that get a capability for Sigma0. Memory can be requested from Sigma0 directly via an IPC, or indirectly by causing page faults and having them resolved by Sigma0.

### 5.1.1 Factory

There is only one instance of Sigma0 in an [L4Re](#) system, which is made accessible to Moe via an IPC gate capability. Using this capability, Moe can request Sigma0 to create new communication channels to itself by creating additional IPC gate capabilities. This request is done using the [L4::Factory](#) interface. This is the only kind of object that can be created by the factory in Sigma0.

List of objects that the Sigma0 Factory can create:

- Sigma0 ()
  - Use protocol id [L4\\_PROTO\\_SIGMA0](#) for creation
  - No arguments supported

See also

[Sigma0 API](#)

## 5.2 Moe, the Root-Task

Moe is the default root-task implementation for L4Re-based systems.

*Moe* is the first task which is usually started in L4Re-based systems. The micro kernel starts *Moe* as the Root-Task.

### 5.2.1 Moe objects

Moe provides a default implementation for the basic [L4Re](#) abstractions, such as data spaces ([L4Re::Dataspace](#)), region maps ([L4Re::Rm](#)), memory allocators ([L4::Factory](#), [L4Re::Mem\\_alloc](#)), name spaces ([L4Re::Namespace](#)) and so on (see [L4Re Interface](#)). These are described in the following subsections.

### 5.2.1.1 Factory

The factory in Moe is responsible for all kinds of dynamic object allocation.

Moe's factory allows allocation of the following objects:

- [L4Re::Namespace](#)
- [L4Re::Dataspace](#), RAM allocation
- [L4Re::Dma\\_space](#), memory management for DMA-capable devices
- [L4Re::Rm](#), virtual memory management for application tasks
- [L4::Vcon](#) (output only)
- [L4::Scheduler](#), to provide a restricted priority / CPU range for clients
- [L4::Factory](#), to provide a quota limited allocation for clients

#### Note

[L4::Scheduler](#) objects can be only created through the user factory provided by Moe to the initial application. Other factory instances cannot create this object.

#### 5.2.1.1.1 Passing parameters to the create stream

[L4::Factory.create\(\)](#) returns a [create stream](#) that allows arguments to be forwarded to the the object creation in Moe.

Objects that support additional parameters on their creation are presented next with a non-empty list of parameters. The parameters are listed in the order they should be provided to a create stream. Optional parameters are identified by their default values. Multiple entries in the next list denote different ways of initializing an object.

- [L4Re::Namespace](#) ()
  - For more details see [Namespace](#)
- [L4Re::Dataspace](#) (l4\_mword\_t size, l4\_umword\_t flags = 0, l4\_umword\_t align = 0)
  - Argument `size`: size in bytes (mandatory)
  - Argument `flags`: special dataspace properties, see [L4Re::Mem\\_alloc::Mem\\_alloc\\_flags](#)
  - Argument `align`: Log2 alignment of dataspace if supported by allocator
  - See detailed description of the parameters in [L4Re::Mem\\_alloc::alloc\(\)](#)
  - For details on the types of dataspace provided by Moe, see [Dataspace](#)
- [L4Re::Dma\\_space](#) ()
  - For more details see [DMA Space](#)
- [L4Re::Rm](#) ()
  - For more details see [Region Map](#)
- [L4::Vcon](#) (char const \*label, l4\_mword\_t color = 7)
  - Argument `label`: label used as prefix for the console output
  - Argument `color`: color code 0..15

- For more details see [Log Subsystem](#)
- **L4::Vcon** (`char const *label, char const *color = "w"`)
  - Argument `label`: label used as prefix for the console output
  - Argument `color`: color code
    - \* The color is identified by a single character
    - \* Supported colors: N, n, R, r, G, g, Y, y, B, b, M, m, C, c, W, w
  - For more details see [Log Subsystem](#)
- **L4::Scheduler** (`l4_mword_t limit, l4_mword_t offset, l4_umword_t bitmap = ~0UL, ...`)
  - Argument `limit`: maximum priority
  - Argument `offset`: priority offset
  - Argument `bitmap`: bitmap of CPUs - can be repeated to address higher order CPUs
  - Argument `limit` must be greater than `offset`
  - For more details see [Scheduler subsystem](#)
- **L4::Factory** (`l4_mword_t quota`)
  - Argument `quota`: limit in bytes (not zero)
  - The limit is deducted from the limit of the factory that creates the new factory

### 5.2.1.2 Namespace

Moe provides a name space conforming to the [L4Re::Namespace](#) interface (see [Name-space API](#)). Per default Moe creates a single name space for the [Boot FS](#). That is available as `rom` in the initial objects of the init process.

#### 5.2.1.2.1 Boot FS

The Boot FS subsystem provides access to the files loaded during the platform boot (or available in ROM). These files are either linked into the boot image or loaded via a flexible boot loader, such as GRUB.

The subsystem provides an [L4Re::Namespace](#) object as directory and an [L4Re::Dataspace](#) object for each file.

By default all files are read only and visible in the namespace `rom`. As an option, files can be supplied with the argument `:rw` to mark them as writable modules. Moe will allow read and write access to these dataspace and make them visible in a different namespace called ``rwfs``.

An example entry in 'modules.list' would look like this:

```
module somemodule :rw
```

#### Note

In order for a client to receive write permissions to the dataspace, the corresponding cap also needs write permissions.

### 5.2.1.3 Dataspace

Dataspaces can be allocated with an arbitrary size. The granularity for memory allocation however is the machine page size ([L4\\_PAGESIZE](#)). A dataspace user must be aware that, as a result of this page-size granularity, there may be padding memory at the end of a dataspace which is accessible to each client. Moe currently allows most dataspace operations on this padding area. Nonetheless, the client must not make any assumptions about the size or content of the padding area, as this behaviour might change in the future.

The provided data spaces can have different characteristics:

- Physically contiguous and pre-allocated
- Non contiguous and on-demand allocated with possible copy on write (COW)

Dataspaces allocated via the Moe's factory allow mappings with any combination of the read-write-execute (RWX) rights, subject to a possible restriction of the writable right for client capabilities lacking the 'W' right.

### 5.2.1.4 Log Subsystem

The logging facility of Moe provides per application tagged and synchronized log output.

### 5.2.1.5 DMA Space

### 5.2.1.6 Scheduler subsystem

The scheduler subsystem provides a simple scheduler proxy for scheduling policy enforcement.

The priority offset provided on the creation of a scheduler proxy defines the minimum priority assigned to threads which are scheduled by that instance of the scheduler proxy. The offset is implicitly added to priorities provided to [L4::Scheduler.run\\_thread\(\)](#).

### 5.2.1.7 Region Map

## 5.2.2 Command Line Options

Moe's command-line syntax is:

```
moe [--debug=<flags>] [--init=<binary>] [--l4re-dbg=<flags>] [--ldr-flags=<flags>] [-- <init options>]
```

```
--debug=<debug flags>
```

This option enables debug messages from Moe itself, the `<debug flags>` values are a combination of `info`, `warn`, `boot`, `server`, `loader`, `exceptions`, and `ns` (or all for full verbosity).

```
--init=<init process>
```

This options allows to override the default init process binary, which is 'rom/ned'.



```
--l4re-dbg=<debug flags>
```

This option allows to set the debug options for the [L4Re](#) runtime environment of the init process. The flags are the same as for `--debug=`.

```
--ldr-flags=<loader flags>
```

This option allows setting some loader options for the [L4Re](#) runtime environment. The flags are `pre_alloc`, `all_segs_cow`, and `pinned_segs`.

```
--brk=<address>
```

This option is only present on systems without MMU. It restricts dynamic allocations to addresses equal or higher than `<address>`. The argument is parsed as hexadecimal number without any `0x` prefix. Use it to prevent moe from allocating memory in regions that shall later be used by other applications or virtual machines.

```
-- <init options>
```

All command-line parameters after the special `--` option are passed directly to the init process.

## 5.3 Ned, the Init Process

Ned's job is to bootstrap the system running on [L4Re](#).

The main thing to do here is to coordinate the startup of services and applications as well as to provide the communication channels for them. The central facility in Ned is the Lua ( <http://www.lua.org>) script interpreter with the [L4Re](#) and ELF-loader bindings.

The boot process is based on the execution of one or more Lua scripts that create communication channels (IPC gates), instantiate other [L4Re](#) objects, organize capabilities to these objects in sets, and start application processes with access to those objects (or based on those objects).

For starting applications, Ned depends on the services of [Moe, the Root-Task](#) or another *loader*, which must provide data spaces and region maps. Ned also uses the 'rom' capability as source for Lua scripts and at least the 'l4re' binary (the runtime environment core) running in each application.

Each application Ned starts is equipped with an [L4Re::Env](#) environment that provides information about all the initial objects made accessible to this application.

### 5.3.1 Lua Bindings for L4Re

Ned provides various bindings for [L4Re](#) abstractions. These bindings are located in the '[L4](#)' package (`require "L4"`).

### 5.3.1.1 Capabilities in Lua

Capabilities are handled as normal values in Lua. They can be stored in normal variables or Lua compound structures (tables). A capability in Lua possesses additional information about the access rights that shall be transferred to other tasks when the capability is transferred. To support implementation of the Principle of Least Privilege, minimal rights are assigned by default. Extended rights can be added using the method `mode("...")` (short `m("...")`) that returns a new reference to the capability with the given rights.

#### Note

It is generally impossible to elevate the real access rights to an object. This means that if Ned has only restricted rights to an object it is not possible to upgrade the access rights with the `mode` method.

The capabilities in Lua also carry dynamic type information about the referenced objects. They thereby provide type-specific operations on the objects, such as the `create` operation on a generic factory or the `query` and `register` operations on a name space.

### 5.3.1.2 Access to L4Re::Env Capabilities

The initial objects provided to Ned itself are accessible via the table `L4.Env`. The default (usually unnamed) capabilities are accessible as `factory`, `log`, `mem_alloc`, `parent`, `rm`, and `scheduler` in the `L4.Env` table.

### 5.3.1.3 Constants

#### Protocols

The protocol constants are defined by default in the [L4](#) package's table `L4.Proto`. The definition is not complete and only covers what is usually needed to configure and start applications. The protocols are for example used as first argument to the `Factory:create` method.

```
Proto = {
  Dataspace = 0x4000,
  Namespace = 0x4001,
  Goos      = 0x4003,
  Mem_alloc = 0x4004,
  Rm        = 0x4005,
  Event     = 0x4006,
  Inhibitor = 0x4007,
  Sigma0    = -6,
  Log       = -13,
  Scheduler = -14,
  Factory   = -15,
  Vm        = -16,
  Dma_space = -17,
  Irq_sender = -18,
  Semaphore = -20,
  Iommu     = -22,
  Ipc_gate  = 0,
}
```

#### Rights

The rights of a Lua capability can be defined in two ways via the `:mode()` interface. Either via a string representation of the rights or via an integer value. An example for the former is `:mode("rsnc")` while the latter equivalent is `:mode(L4.Rights.r | L4.Rights.s | L4.Rights.n | L4.Rights.c)`. The following listing shows the integer constants. The constant names can be used in the string parameter to `:mode()`.

```
Rights = {
  s = 2,
  w = 1,
  r = 4,
  d = 8,
  n = 16,
```

```

c    = 32,
ro   = 4,
rw   = 5,
rws  = 7,
}

```

## Debugging Flags

Debugging flags used for the applications [L4Re](#) core:

```

Dbg = {
    Info      = 1,
    Warn      = 2,
    Boot      = 4,
    Server     = 0x10,
    Exceptions = 0x20,
    Cmd_line   = 0x40,
    Loader     = 0x80,
    Name_space = 0x400,
    All        = 0xffffffff,
}

```

## Loader Flags

Flags for configuring the loading process of an application.

```

Ldr_flags = {
    eager_map      = 0x1, -- L4RE_AUX_LDR_FLAG_EAGER_MAP
    all_segs_cow  = 0x2, -- L4RE_AUX_LDR_FLAG_ALL_SEGS_COW
    pinned_segs   = 0x4, -- L4RE_AUX_LDR_FLAG_PINNED_SEGS
}

```

### 5.3.1.4 Application Startup Details

The central facility for starting a new task with Ned is the class `L4.Loader`. This class provides interfaces for conveniently configuring and starting programs. It provides three operations:

- `new_channel()` Returns a new IPC gate that can be used to connect two applications
- `start()` and `startv()` Start a new application process and return a process object

The `new_channel()` call is used to provide a service application with a communication channel to bind its initial service to. The concrete behavior of the object and the number of IPC gates required by a server depends on the server implementation. The channel can be passed to client applications as well or can be used for operations within the script itself.

`start()` and `startv()` always require at least two arguments. The first one is a table that contains information about the initial objects an application shall get. The second argument is a string, which for `start()` is the program name plus a white-space-separated list of program arguments (`argv`). For `startv()` the second argument is just the program binary name – which may contain spaces –, and the program arguments are provided as separate string arguments following the binary name (allowing spaces in arguments, too). The last optional argument is a table containing the POSIX environment variables for the program.

The Loader class uses reasonable defaults for most of the initial objects. However, you can override any initial object with some user-defined values. The main elements of the initial object table are:

- `factory` The factory used by the new process to create new kernel objects, such as threads etc. This must be a capability to an object implementing the [L4::Factory](#) protocol and defaults to the factory object provided to Ned.
- `mem` The memory allocator provided to the application and used by Ned allocates data spaces for the process. This defaults to Ned's memory allocator object (see [L4Re::Mem\\_alloc](#)).
- `rm_fab` The generic factory object used to allocate the region-map object for the process. (defaults to Ned's memory allocator).

- `log_fab` The generic factory to create the [L4Re::Log](#) object for the application's output (defaults to Ned's memory allocator). The `create` method of the `log_fab` object is called with `log_tag` and `log_color`, from this table, as arguments.
- `log_tag` The string used for tagging log output of this process (defaults to the program name) (see `log_fab`).
- `log_color` The color used for the log tag (defaults to "white").
- `scheduler` The scheduler object used for the process' threads (defaults to Ned's own scheduler).
- `caps` The table with application-specific named capabilities (default is an empty table). If the table does not contain a capability with the name 'rom', the 'rom' capability from Ned's initial caps is inserted into the table.

The `start()` and `startv()` calls return a task object that supports a number of operations.

- `state()` returns a string with the current task state: "running" or "zombie" if the task has terminated. If the task was already reaped (`wait()` returned) or if `kill()` was called, then the function will return `nil`.
- `exit_code()` returns the exit code if the task has terminated or `nil` if it was either killed or has been reaped.
- `kill()` forcefully terminates the task. Returns "killed" if the task was terminated or the exit code if the task was already gone. Returns `nil` if the task was already reaped.
- `exit_handler(function)` registers a Lua function that is invoked when the task terminates. If the task has already terminated, the function is called immediately. Returns `true` if the callback is pending, otherwise `false`. The callback function gets the exit code (`nil` if killed) of the task as its only parameter. The return value of the function is ignored. Only one callback can be registered.
- `wait()` suspends execution until the task has terminated. It's better to use `exit_handler()` instead. While the Lua code executes `wait()`, no `exit_handler()` will be dispatched.

### 5.3.1.5 Reacting on task termination

Ned can react on the termination of child tasks. The preferred mechanism is to register an `exit_handler()` for a task:

```
task = L4.default_loader:start(...)
task:exit_handler(function(exit_code)
  if exit_code == nil then
    print("Task was killed")
  else
    print("Task has terminated w/ code [" .. exit_code .. "]")
  end
end)
```

If the task did already terminate then the callback is invoked immediately. It is also possible to suspend execution of the Ned script until a task has terminated:

```
task = L4.default_loader:start(...)
task:wait()
```

This method should be used with caution, though. The Ned script will wait until the child task has terminated. Neither will any of the registered `exit_handler()` will be called during this time, nor will the [remote command interface](#) be able to execute commands either.

### 5.3.1.6 Access to the kernel debugger

Applications can enrich the kernel debugger with information using the API defined in [l4/sys/debugger](#). In order to do so, the developer has to assign access to the kernel debugger kernel object to the application. This can be done like this:

```
L4.default_loader:start({ caps = { jdb = L4.Env.jdb; }}, "rom/example")
```

### 5.3.1.7 Using the interactive ned prompt

Ned can be used in interactive mode by connecting the small ned-prompt helper tool to the command capability. Add the following code snippet at the end of your ned script:

```
local L4 = require("L4");
local l = L4.default_loader;

cmd = l:new_channel()

l:start({ log = L4.Env.log, caps = { svr = cmd } }, "rom/ned-prompt")

L4.server_loop(cmd)
```

The script hands in ned's own log capability to `ned-prompt`. This ensures that input and output of ned and the prompt appear on the same console.

`ned-prompt` needs to be added to your modules list.

## 5.3.2 Command Line Options

Ned's command line syntax is:

```
[--exit|--wait-and-exit] [--execute|-e STATEMENT] <lua script> [options passed to lua script]
```

Ned interprets the first non-option argument `<lua script>` as the Lua script which it should load and run. All arguments following the first non-option argument are passed as arguments to the Lua script via Lua's global `arg` table.

- Exit Options: **exit**, **wait-and-exit** (must be first if used)
  - `exit*`: terminates the application after the script has run through even if there are still tasks running. `wait` for tasks at the end of the script to ensure they do not die forcefully.
  - `exit-and-wait**`: terminates the application after the script has run through and all tasks started by ned have signaled their exit.
- Execute Statement Option: **execute**, **e**
  - Execute the Lua statement `STATEMENT`.

## 5.4 Io, the Io Server

The Io server handles all platform devices and resources such as I/O memory, ports (on x86) and interrupts, and grants access to those to clients.

Upon startup Io discovers all platform devices using available means on the system, e.g. on x86 the PCI bus is scanned and the ACPI subsystem initialised. Available I/O resource can also be configured via configuration scripts.

Io's configuration consists of two parts:

- the description of the real hardware
- the description of virtual buses

Both descriptions represent a hierarchical (tree) structure of device nodes. Where each device has a set of resources attached to it. And a device that has child devices can be considered a bus.

## Hardware Description

The hardware description represents the devices that are available on the particular platform including their resource descriptions, such as MMIO regions, IO-Port regions, IRQs, bus numbers etc.

The root of the hardware devices is formed by a system bus device (accessible in the configuration via `lo.system↔_bus()`). As mentioned before, platforms that support methods for device discovery may populate the hardware description automatically, for example from ACPI. On platforms that do not have support for such methods you have to specify the hardware description by hand. A simple example for this is `x86-legacy.devs`.

## Virtual Bus Description

Each lo server client is provided with its own virtual bus which it can iterate to find devices. A virtual PCI bus may be a part of this virtual bus.

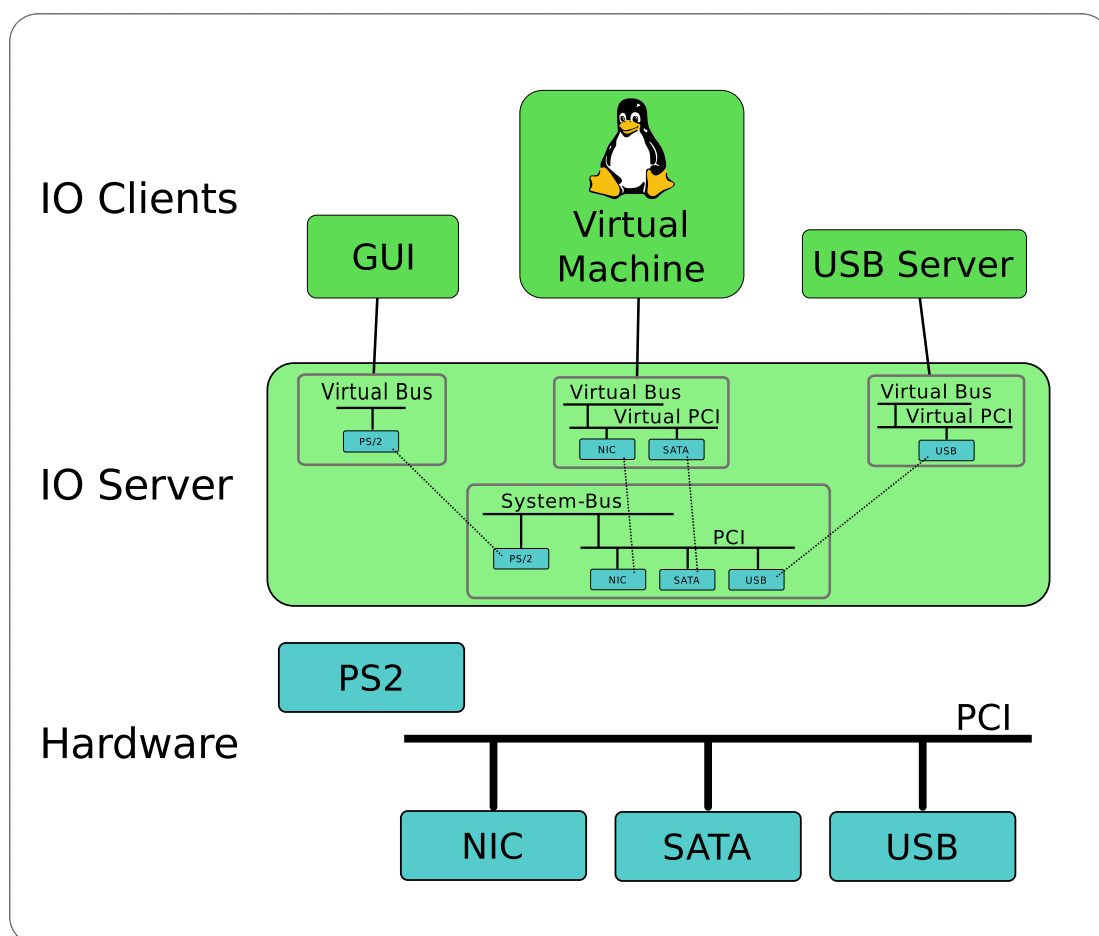


Figure 5.1 IO Service Architecture Overview

The lo server must be configured to create virtual buses for its clients.

This is done with at least one configuration file specifying static resources as well as virtual buses for clients. The configuration may be split across several configuration files passed to lo through the command line.

To allow clients access to available devices, a virtual system bus needs to be created that lists the devices and their resources that should be available to that client. The names of the buses correspond to the capabilities given to lo in its launch configuration.

A very simple configuration for Io could look like this:

```
-- vim:ft=lua
-- Example configuration for io

-- Configure two platform devices to be known to io
Io.Dt.add_children(Io.system_bus(), function()

    -- create a new hardware device called "FOODEVICE"
    FOODEVICE = Io.Hw.Device(function()
        -- set the compatibility IDs for this device
        -- a client tries to match against these IDs and configures
        -- itself accordingly
        -- the list should be sorted from specific to less specific IDs
        compatible = {"dev-foo,mmio", "dev-foo"};

        -- set the 'hid' property of the device, the hid can also be used
        -- as a compatible ID when matching clients
        Property.hid = "dev-foo,Example";

        -- note: names for resources are truncated to 4 letters and a client
        -- can determine the name from the ID field of a l4vbus_resource_t
        -- add two resources 'irq0' and 'reg0' to the device
        Resource.irq0 = Io.Res.irq(17);
        Resource.reg0 = Io.Res.mmio(0x6f000000, 0x6f007fff);
    end);

    -- create a new hardware device called "BARDEVICE"
    BARDEVICE = Io.Hw.Device(function()
        -- set the compatibility IDs for this device
        -- a client tries to match against these IDs and configures
        -- itself accordingly
        -- the list should be sorted from specific to less specific IDs
        compatible = {"dev-bar,mmio", "dev-bar"};

        -- set the 'hid' property of the device, the hid can also be used
        -- as a compatible ID when matching clients
        Property.hid = "dev-bar,Example";

        -- Specify that this device is able to use direct memory access (DMA).
        -- This is needed to allow clients to gain access to DMA addresses
        -- used by this device to directly access memory.
        Property.flags = Io.Hw_device_DF_dma_supported;

        -- note: names for resources are truncated to 4 letters and a client
        -- can determine the name from the ID field of a l4vbus_resource_t
        -- add three resources 'irq0', 'irq1', and 'reg0' to the device
        Resource.irq0 = Io.Res.irq(19);
        Resource.irq1 = Io.Res.irq(20);
        Resource.reg0 = Io.Res.mmio(0x6f100000, 0x6f100fff);
    end);
end);

Io.add_vbusses
{
    -- Create a virtual bus for a client and give access to FOODEVICE
    client1 = Io.Vi.System_bus(function ()
        dev = wrap(Io.system_bus():match("dev-foo,mmio"));
    end);

    -- Create a virtual bus for another client and give it access to BARDEVICE
    client2 = Io.Vi.System_bus(function ()
        dev = wrap(Io.system_bus():match("dev-bar,Example"));
    end);
}
```

Each device supports a 'compatible' property. It is a list of compatibility strings. A client matches itself against one (or multiple) compatibility IDs and configures itself accordingly. All other device members are handled according to their type. If the type is a resource (Io.Res) it is added as a named resource. Note that resource names are truncated to 4 letters and are stored in the ID field of a [l4vbus\\_resource\\_t](#). If the type is a device it is added as a child device to the current one. All other types are treated as a device property which can be used to configure a device driver. Right now, device properties are internal to Io only.

## Matching and Assigning PCI Devices

Assigning clients PCI devices could look like this:

```
-- This is a configuration snippet for PCI device selection

local hw = Io.system_bus();

Io.add_vbusses
```

```

{
  pciclient = Io.Vi.System_bus(function ()
    PCI = Io.Vi.PCI_bus(function ()
      pci_mm      = wrap(hw:match("PCI/CC_04"));
      pci_net     = wrap(hw:match("PCI/CC_02"));
      pci_storage = wrap(hw:match("PCI/CC_01"));
    end)
  end)
}

```

The "PCI/" is followed by a bus-specific ID string. The format of the PCI ID string may be one of the following:

- PCI/CC\_cc
- PCI/CC\_ccss
- PCI/CC\_ccssp
- PCI/VEN\_vvvv
- PCI/DEV\_dddd
- PCI/SUBSYS\_ssssssss
- PCI/REV\_rr
- PCI/ADR\_xxxx:xx:xx.x

Where:

- cc is the hexadecimal representation of the class code byte
- ss is the hexadecimal representation of the subclass code byte
- pp is the hexadecimal representation of the programming interface byte
- vvvv is the hexadecimal representation of the vendor ID
- dddd is the hexadecimal representation of the device ID
- ssssssss is the hexadecimal representation of the subsystem ID
- rr is the hexadecimal representation of the revision byte
- xxxx:xx:xx.x is the bus address in PCI nomenclature

As a special extension lo supports replacing the ID string with a human-readable common PCI class name. The following table gives an overview of the names known to lo and their respective PCI class and subclass.

Common Name	Description	PCI ID string
storage	Mass storage controller	CC_01
scsi	SCSI storage controller	CC_0100
ide	IDE interface	CC_0101
floppy	Floppy disk controller	CC_0102
raid	RAID bus controller	CC_0104
ata	ATA controller	CC_0105
sata	SATA controller	CC_0106
sas	Serial attached SCSI controller	CC_0107
nvm	Non-volatile memory controller	CC_0108
-	-	-
network	Network controller	CC_02
ethernet	Ethernet controller	CC_0200



Common Name	Description	PCI ID string
token_ring	Token ring network controller	CC_0201
fddi	FDDI network controller	CC_0202
atm	ATM network controller	CC_0203
isdn	ISDN controller	CC_0204
picmg	PICMG controller	CC_0206
net_infiniband	Infiniband controller	CC_0207
fabric	Fabric controller	CC_0208
network_nw	Network controller e.g. Wifi	CC_0280
-	-	-
display	Display controller	CC_03
vga	VGA compatible controller	CC_0300
xga	XGA compatible controller	CC_0301
-	-	-
media	Multimedia controller	CC_04
mm_video	Multimedia video controller	CC_0400
mm_audio	Multimedia audio controller	CC_0401
telephony	Computer telephony device	CC_0402
audio	Audio device	CC_0403
-	-	-
bridge	Bridge	CC_06
br_host	Host bridge	CC_0600
br_isa	ISA bridge	CC_0601
br_eisa	EISA bridge	CC_0602
br_microchannel	MicroChannel bridge	CC_0603
br_pci	PCI bridge	CC_0604
br_pcmcia	PCMCIA bridge	CC_0605
br_nubus	NuBus bridge	CC_0606
br_cardbus	CardBus bridge	CC_0607
br_raceway	RACEway bridge	CC_0608
br_semi_pci	Semi-transparent PCI-to-PCI bridge	CC_0609
br_infiniband_to_pci	InfiniBand to PCI host bridge	CC_060a
-	-	-
com	Communication controller	CC_07
com_serial	Serial controller	CC_0700
com_parallel	Parallel controller	CC_0701
com_multiport_ser	Multiport serial controller	CC_0702
com_modem	Modem	CC_0703
com_gpib	GPiB controller	CC_0704
com_smart_card	Smart card controller	CC_0705
-	-	-
serial_bus	Serial bus controller	CC_0c
firewire	FireWire (IEEE 1394)	CC_0c00
access_bus	ACCESS bus	CC_0c01
ssa	SSA	CC_0c02
usb	USB controller	CC_0c03
fibre_channel	Fibre channel	CC_0c04
smbus	SMBus	CC_0c05
bus_infiniband	InfiniBand	CC_0c06
ipmi_smic	IPMI SMIC interface	CC_0c07
sercos	SERCOS interface	CC_0c08

Common Name	Description	PCI ID string
canbus	CAN bus	CC_0c09
-	-	-
wireless	Wireless controller	CC_0d
bluetooth	Bluetooth	CC_0d11
w_8021a	802.1a controller	CC_0d20
w_8021b	802.1b controller	CC_0d21

### Strong Matching of PCI Devices

If more specific matching of PCI devices is required it is possible to concatenate multiple ID strings using `&`. An example where a specific device from a specific vendor at a fixed bus address is matched would use the string `PCI/VEN_vvvv&DEV_dddd&ADR_xxxx:xx:xx.x`.

### Isolation of PCIe devices

PCIe encodes device communication with a network-like protocol with destination headers and packet fragmentation allowing a devices to talk directly to other devices. This potentially works against security boundaries for a system. E.g. two network cards could exchange packets and thereby leak information from one security domain to the other without involvement of the OS.

PCIe introduced an optional capability named PCI Access Control Services (PCI/ACS) to control communication between PCIe devices.

With PCI/ACS it is possible to restrict inter-device communication between PCIe devices.

PCI/ACS is optional and for Intel chipsets, it is usually only implemented on high-end PCI platform controller hubs (PCHs), and is missing on low-end and mobile PCHs. On some Intel-PCHs there exist facilities that allow for similar isolation.

If IO encounters a supported PCH, it will enable those facilities in order to enforce device isolation.

### Command Line Options

The Io Server supports the following optional parameters:

```
[--verbose|v] [--transparent-msi] [--trace <trace_mask>] [--acpi-debug-level <debug_level>] [config_files]
```

- **verbose|v**

By default, error debug messages are enabled. This option increments the verbosity level, and can be applied multiple times to reach the desired debug level. The available debug levels are ordered as: `DBG_ERR` (default, level 1), `DBG_WARN`, `DBG_INFO`, `DBG_DEBUG`, `DBG_DEBUG2` and `DBG_ALL` (level 6).

- **transparent-msi**

Enable MSI on PCI devices which support this feature. This is transparent to clients, as there are no changes in the API used to interact with PCI device via interrupts.

- **acpi-debug-level <level\_mask>**

Set the ACPI debug level. The `<level_mask>` is a mask that selects components of interest for debugging. It can be constructed from the ACPI debug constants defined in the linux kernel, see [ACPI Debug Output](#) for details. By default, the ACPI debug level is set to `ACPI_LV_INIT | ACPI_LV_TABLES | ACPI_LV_VERBOSE_INFO`.

- **trace <trace\_mask>**

Enable tracing of events matching `trace_mask`. The only supported trace mask is 1 and this matches ACPI events.

- **config\_files**

Space separated list of Lua configuration files specifying real hardware and virtual buses. See example on [Virtual Bus Description](#).

## 5.5 AHCI driver

The AHCI driver is a driver for PCI serial ATA host controllers.

The AHCI driver is capable of exposing entire disks (by serial number) or individual partitions (by their partition UUID) of a hard drive to clients via the Virtio block interface.

The driver consists of two parts. The first one is the hardware driver itself that takes care of the communication with the underlying hardware and interrupt handling. The second part implements a virtual block device and is responsible to communicate with clients. The virtual block device translates commands it receives into AHCI requests and issues them to the hardware driver.

The AHCI driver allows both statically and dynamically configured clients. A static configuration is given priority over dynamically connecting clients and configured while the service starts. Dynamic clients can connect and disconnect during runtime of the AHCI driver.

### Building and Configuration

The AHCI driver can be built using the [L4Re](#) build system. Just place this project into your `pkg` directory. The resulting binary is called `ahci-drv`

### Starting the service

The AHCI driver can be started with Lua like this:

```
local ahci_bus = L4.default_loader:new_channel();
L4.default_loader:start({
  caps = {
    vbus = vbus_ahci,
    svr = ahci_bus:svr(),
  },
},
"rom/ahci-drv");
```

First an IPC gate (`ahci_bus`) is created which is used between the AHCI driver and a client to request access to a particular disk or partition. The server-side is assigned to the mandatory `svr` capability of the AHCI driver. See the section below on how to configure access to a disk or partition.

The ahci driver needs access to a virtual bus capability (`vbus`). On the virtual bus the AHCI driver searches for AHCI 1.0 compliant storage controllers. Please see io's documentation about how to setup a virtual bus.

### Options

In the example above the ahci driver is started in its default configuration. To customize the configuration of the ahci-driver it accepts the following command line options:

- `-A`  
Disable check for address width of the device. Only do this if all physical memory is guaranteed to be below 4GB.

- `-v`  
Enable verbose mode. You can repeat this option up to three times to increase verbosity up to trace level.
- `-q`  
This option enables the quiet mode. All output is silenced.
- `--client <cap_name>`  
This option starts a new static client option context. The following `device`, `ds-max` and `readonly` options belong to this context until a new client option context is created.  
The option parameter is the name of a local IPC gate capability with server rights.
- `--device <UUID | SN>`  
This option denotes the partition UUID or serial number of the preceding `client` option.
- `--ds-max <max>`  
This option sets the upper limit of the number of dataspaces the client is able to register with the AHCI driver for virtio DMA.
- `--slot-max <max>`  
This option defines the maximum number of requests a single client may have in parallel running on the device. If a positive number is given, then this is considered the absolute number of slots to be used. If a negative number is given, then the client may use all available slots except the number given. In any case, a client gets at least 1 slot and at most the number of slots available in hardware. This parameter is only valid when a client accesses a partition and ignored otherwise.
- `--readonly`  
This option sets the access to disks or partitions to read only for the preceding `client` option.

## Connecting a client

Prior to connecting a client to a virtual block session it has to be created using the following Lua function. It has to be called on the client side of the IPC gate capability whose server side is bound to the ahci driver.

```
create(obj_type, "device=<UUID | SN>", "ds-max=<max>"[, "slot-max=<max>"])
```

- `obj_type`  
The type of object that should be created by the driver. The type must be a positive integer. Currently the following objects are supported:
  - 0: Virtio block host
- `"device=<UUID | SN>"`  
This string denotes either a partition UUID or a disk serial number the client wants to be exported via the Virtio block interface.
- `"ds-max=<max>"`  
Specifies the upper limit of the number of dataspaces the client is allowed to register with the AHCI driver for virtio DMA.
- `"slot-max=<max>"`  
Specifies the maximum number of requests that will be processed in parallel by the AHCI device. See `--slot-max` option above for details.

If the `create()` call is successful a new capability which references an AHCI virtio driver is returned. A client uses this capability to communicate with the AHCI driver using the Virtio block protocol.

## Examples

A couple of examples on how to request different disks or partitions are listed below.

- Request a partition with the given UUID

```
vda1 = ahci_bus:create(0, "ds-max=5", "device=88E59675-4DC8-469A-98E4-B7B021DC7FBE")
```

- Request complete disk with the given serial number

```
vda = ahci_bus:create(0, "ds-max=4", "device=QM00005")
```

- A more elaborate example with a static client. The client uses the client side of the `ahci_cl1` capability to communicate with the AHCI driver.

```
local ahci_cl1 = L4.default_loader:new_channel();
local ahci_bus = L4.default_loader:new_channel();
L4.default_loader:start({
  caps = {
    vbus = vbus_ahci,
    svr = ahci_bus:svr(),
    cl1 = ahci_cl1:svr(),
  },
},
"rom/ahci-drv --client cl1 --device 88E59675-4DC8-469A-98E4-B7B021DC7FBE --ds-max 5");
```

## 5.6 Cons, the Console Multiplexer

`cons` allows to multiplex console output from different [L4](#) clients to different [L4::Vcon](#)-capable in/output servers.

### Multiplexers and Frontends

`cons` is able to connect multiple clients with multiple in/output servers.

Clients are handled by a *multiplexer*. Each multiplexer publishes a server capability that allows to create new client connections. The default multiplexer is normally known under the `cons` capability.

Actual in/output is handled by separate frontends. From the point-of-view of `cons`, a frontend consists of an IPC channel to a server that speaks an appropriate server protocol. By default the `L4.Env.log` capability is used.

For clients `cons` implements the [L4::Vcon](#) and the Virtio console interface. The supported frontends is limited to [L4::Vcon](#) only.

### Command Line Options

`cons` accepts the following command line switches:

- `-a, --show-all`

Initially show output from all clients.

- `-c <client>, --autoconnect <client>`

Automatically connect to the client with the given name. That means that output of this client will be visible and input will be routed to it.

- `-k, --keep`  
Keep the console buffer when a client disconnects.
- `-l, --no-line-buffering`  
By default, merge the client output to entire lines. If the client writes characters without a final newline, the following client output is merged with the current line content. Specifying this switch disables the line buffered mode by default.
- `--line-buffering-ms <timeout>`  
Timeout in milliseconds before buffered client output is written even without a newline. Default value is 50.
- `-n, --defaultname`  
Default multiplexer capability to use. Default: `cons`.
- `-B <size>, --defaultbufsize <size>`  
Default buffer size per client in bytes. Default: 40960
- `-m <prompt name>, --mux <prompt name>`  
Add a new multiplexer named `<prompt name>`. This is necessary if output should be sent to different frontends.
- `-f <cap>, --frontend <cap>`  
Set the frontend for the current multiplexer. Output for the multiplexer is then sent to the capability with the given name. The server connected to the capability needs to understand the [L4::Vcon](#) protocol.

## Connecting a client

```
create(backend_type, ["client_name"], ["color"], ["options"])
```

- `backend_type`  
The type of backend that should be created for the client. The type is a positive integer and currently the following types are supported:
  - `L4.Proto.Log`: [L4::Vcon](#) client
  - `1`: Virtio console client

## 5.7 Mag, the GUI Multiplexer

Mag is the default multiplexer for graphics hardware.

It is not, and does not attempt to be, a fully-fledged window manager.

### Command Line Options

As Mag's only command line option it supports loading additional plugins via the application's command line. Plugins must be either a Lua file or a shared library. Shared libraries must be named `libmag-$LIBNAME.so`.

## Mag Sessions

Mag provides two types of sessions which a client can create via the Factory interface.

### 1. Mag client session

A client with a mag client session gets access to the whole screen. The client has to allocate and manage its own buffers and has to position them on the screen on its own. Mag provides the factory to create client sessions via the capability named `mag`.

### 2. Client framebuffer session

For a client framebuffer session mag allocates a view of the requested size and displays it at the requested coordinates on the screen. Mag provides the factory to create framebuffer sessions via the capability named `svc`.

The options described below are options the client provides to the `L4::Factory::create()` call. These options influence the appearance and behaviour of the newly created session.

## Session Factory Options

As a simple nitpicker clone Mag supports the so-called Xray mode. This mode displays all session labels and draws a colored frame around them. The session that currently has the input focus is highlighted. The Xray mode is activated via the special keys *Scroll* or *NEXTSONG*.

Mag allows to define a text label and a color for all client session types. The label and the color are displayed when Mag enters the Xray mode.

- Label Option: **label**

`l=LABEL, label=LABEL` Set the session's text label to LABEL. The label is restricted to a length of 256 characters.

- Color Option: **col**

`col=COLOR` Set the session's color which is used in Xray mode to tint the session's screen area and the border drawn around it. The argument can be either one of the following letters or a hexadecimal representation of the RGB values.

- `r`, `R` Red color
- `g`, `G` Green color
- `b`, `B` Blue color
- `w`, `W` White color
- `y`, `Y` Yellow color
- `v`, `V` Magenta color

## Example

```
-- set label to "Linux" and use a light blue color
fb = mag_client:create(L4.Proto.Goos, "l=Linux", "col=98d9ff");
```

## Mag Client Session Options

These options only apply to Mag client sessions.

- Default Background Option: **default-background**  
`dfl-bg, default-background` Marks this session as the default background.

## Mag Client Framebuffer Session Options

These options only apply to Mag client framebuffer sessions.

- Geometry Option: **geometry**  
`g=GEOMETRY, geometry=GEOMETRY` Set the session's geometry and position on the screen. GEOMETRY is provided in an X11-style format, e.g. `g=WIDTHxHEIGHT+X_OFFSET+Y_OFFSET`.
- Focus Option: **focus**  
`focus` Set the focus to this session.
- Collapsed Option: **shaded**  
`shaded` The window is collapsed and only the title bar is visible. The window can be expanded by clicking into the title bar with the middle mouse button. Collapsing and expanding works also independently of this option.
- Fixed Option: **fixed** `fixed` The window cannot be moved on the screen.
- Barheight Option: **barheight**  
`barheight=X` Set the height of the title bar in pixels.

### Example

```
-- create a window of 640x480 pixels at position (100,100) on the screen.
fb = mag_fb:create(L4.Proto.Goos, "g=640x480+100+100");
```

## 5.8 NVMe server

The NVMe server is a driver for PCI Express NVMe controllers.

The NVMe server is capable of exposing entire disks (i.e. NVMe namespaces) (by serial number and namespace identifier) or individual partitions (by their partition UUID) of a hard drive to clients via the Virtio block interface.

The server consists of two parts. The first one is the hardware driver itself that takes care of the communication with the underlying hardware and interrupt handling. The second part implements a virtual block device and is responsible to communicate with clients. The virtual block device translates commands it receives into NVMe requests and issues them to the hardware driver.

The NVMe server allows both statically and dynamically configured clients. A static configuration is given priority over dynamically connecting clients and configured while the service starts. Dynamic clients can connect and disconnect during runtime of the NVMe server.



## Building and Configuration

The NVMe server can be built using the [L4Re](#) build system. Just place this project into your `pkg` directory. The resulting binary is called `nvme-drv`

## Starting the service

The NVMe server can be started with Lua like this:

```
local nvme_bus = L4.default_loader:new_channel();
L4.default_loader:start({
  caps = {
    vbus = vbus_nvme,
    svr = nvme_bus:svr(),
  },
},
"rom/nvme-drv");
```

First an IPC gate (`nvme_bus`) is created which is used between the NVMe server and a client to request access to a particular disk or partition. The server-side is assigned to the mandatory `svr` capability of the NVMe server. See the section below on how to configure access to a disk or partition.

The NVMe server needs access to a virtual bus capability (`vbus`). On the virtual bus the NVMe server searches for NVMe compliant storage controllers. Please see `io`'s documentation about how to setup a virtual bus.

## Options

In the example above the NVMe server is started in its default configuration. To customize the configuration of the NVMe-server it accepts the following command line options:

- `-v`  
Enable verbose mode. You can repeat this option up to three times to increase verbosity up to trace level.
- `-q`  
This option enables the quiet mode. All output is silenced.
- `--client <cap_name>`  
This option starts a new static client option context. The following `device`, `ds-max` and `readonly` options belong to this context until a new client option context is created.  
The option parameter is the name of a local IPC gate capability with server rights.
- `--device <UUID | SN:n<NAMESPACE_ID>>`  
This option denotes the partition UUID or serial number of the preceding `client` option followed by a colon, letter 'n' and the identifier of the requested NVMe namespace.
- `--ds-max <max>`  
This option sets the upper limit of the number of dataspace the client is able to register with the NVMe server for virtio DMA.
- `--readonly`  
This option sets the access to disks or partitions to read only for the preceding `client` option.

- `--nosgl`  
This option disables support for SGLs.
- `--noms`  
This option disables support for MSI interrupts.
- `--nomsix`  
This option disables support for MSI-X interrupts.
- `-d <cap_name>, --register-ds <cap_name>` This option registers a trusted dataspace capability. If this option gets used, it is not possible to communicate to the driver via dataspaces other than the registered ones. Can be used multiple times for multiple dataspaces.  
The option parameter is the name of a dataspace capability.

## Connecting a client

Prior to connecting a client to a virtual block session it has to be created using the following Lua function. It has to be called on the client side of the IPC gate capability whose server side is bound to the NVMe server.

```
create(obj_type, "device=<UUID | SN:n<NAMESPACE_ID>>", "ds-max=<max>", "read-only")
```

- `obj_type`  
The type of object that should be created by the server. The type must be a positive integer. Currently the following objects are supported:
  - 0: Virtio block host
- `"device=<UUID | SN>"`  
This string denotes either a partition UUID, or a disk serial number the client wants to be exported via the Virtio block interface followed by a colon, letter 'n' and the identifier of the requested NVMe namespace.
- `"ds-max=<max>"`  
Specifies the upper limit of the number of dataspaces the client is allowed to register with the NVMe server for virtio DMA.
- `"read-only"`  
This string sets the access to disks or partitions to read only for the client.

If the `create()` call is successful a new capability which references an NVMe virtio device is returned. A client uses this capability to communicate with the NVMe server using the Virtio block protocol.

## Examples

A couple of examples on how to request different disks or partitions are listed below.

- Request a partition with the given UUID  

```
vda1 = nvme_bus:create(0, "ds-max=5", "device=88E59675-4DC8-469A-98E4-B7B021DC7FBE")
```
- Request complete namespace with the given serial number  

```
vda = nvme_bus:create(0, "ds-max=4", "device=1234:n1")
```

- A more elaborate example with a static client. The client uses the client side of the `nvme_cll` capability to communicate with the NVMe server.

```
local nvme_cll = L4.default_loader:new_channel();
local nvme_bus = L4.default_loader:new_channel();
L4.default_loader:start({
  caps = {
    vbus = vbus_nvme,
    svr = nvme_bus:svr(),
    cll = nvme_cll:svr(),
  },
},
"rom/nvme-drv --client cll --device 88E59675-4DC8-469A-98E4-B7B021DC7FBE --ds-max 5");
```

## 5.9 Uvmm, the virtual machine monitor

Uvmm provides a virtual machine for running an unmodified guest in non-privileged mode.

### Command Line Options

uvmm provides the following command line options:

- `-c, --cmdline=<guest command line>`  
Command line that is passed to the guest on boot.
- `-k, --kernel=<kernel image name>`  
The name of the guest-kernel image file present in the ROM namespace.
- `-d, --dtb=<DTB overlay>`  
The name of the device tree file present in the ROM namespace. The device tree will be placed in the upmost region of guest memory. Optionally, a user may use an additional parameter in the form of "`<DTB overlay>:limit=0xffffffff`" to set an upper limit for the device tree location.
- `-r, --ramdisk=<RAM disk name>`  
The name of the RAM disk file present in the ROM namespace
- `-b, --rambase=<Base address of the guest RAM>`  
Physical start address for the guest RAM. This value is platform specific.
- `-D, --debug=[<component>=] [level]`  
Control the verbosity level of the uvmm. Possible `level` values are: `quiet`, `warn`, `info`, `trace`  
Using the `component` prefix, the verbosity level of each uvmm component is configurable. The component names are: `core`, `cpu`, `mmio`, `irq`, `dev`, `pm`, `vbus_event`  
For example, the following command line sets the verbosity of all uvmm components to `info` except for IRQ handling, which is set to `trace`.  

```
uvmm -D info -D irq=trace
```
- `-f, --fault-mode`  
Control the handling of guest reads/writes to non-existing memory. Possible values are:
  - `ignore` - Invalid writes are ignored. Invalid reads either return 0 or are skipped. The guest may experience undefined behaviour.
  - `halt` - Halt the VM on the first invalid memory access.

- `inject` - Try to forward the invalid access to the guest. This is not supported on all architectures. Falls back to `halt` if the error could not be forwarded to the guest.

Defaults to `ignore`.

- `-q, --quiet`  
Silence all uvmm output.
- `-v, --verbose`  
Increase the verbosity of the uvmm. Repeating the option increases the verbosity by another level.
- `-W, --wakeup-on-system-resume`  
When set, the uvmm resumes when the host system resumes after a suspend call.
- `-i`  
When set, the option forces the guest RAM to be mapped to its corresponding host-physical addresses.

### Setting up guest memory

In the most simple setup, memory for the guest can be provided via a simple dataspace. In your ned script, create a new dataspace of the required size and hand it into uvmm as the `ram` capability:

```
local ramds = L4.Env.user_factory:create(L4.Proto.Dataspace, 60 * 1024 * 1024)
L4.default_loader::startv({caps = {ram = ramds:m("rw")}}, "rom/uvmm")
```

The memory will be mapped to the most appropriate place and a memory node added to the device tree, so that the guest can find the memory.

For a more complex setup, the memory can be configured via the device tree. uvmm scans for memory nodes and tries to set up the memory from them. A memory device node should look like this:

```
memory@0 {
    device_type = "memory";
    reg = <0x00000000 0x00100000
          0x00200000 0xffffffff>;
    l4vmm,dscap = "memcap";
    dma-ranges = <>;
};
```

The `device_type` property is mandatory and needs to be set to `memory`.

`l4vmm,dscap` contains the name of the capability containing the dataspace to be used for the RAM. `reg` describe the memory regions to use for the memory. The regions will be filled up to the size of the supplied dataspace. If they are larger, then the remaining area will be cut.

If the optional `dma-ranges` property is given, the host-physical address ranges for the memory regions will be added here. Note that the property is not cleared first, so it should be left empty.

For more details see [RAM configuration](#).

## Memory layout

uvmm populates the RAM with the following data:

- kernel binary
- (optional) ramdisk
- (optional) device tree

The kernel binary is put at the predefined address. For ELF binaries, this is an absolute physical address. If the binary supports relative addressing, the binary is put to the requested offset relative to beginning of the first 'memory' region defined in the device tree.

The ramdisk and device tree are placed as far as possible to the end of the regions defined in the first 'memory' node.

If there is a part of RAM that must remain empty, then define an extra memory node for it in the device tree. uvmm only writes to memory in the first memory node it finds.

Warning: uvmm does not touch any unpopulated memory. In particular, it does not ensure that the memory is cleared. It is the responsibility of the provider of the RAM dataspace to make sure that no data leakage can happen. Normally this is not an issue because dataspaces are guaranteed to be cleaned when they are newly created but users should be careful when reusing memory or dataspaces, for example, when restarting the uvmm.

## Forwarding hardware resources to the guest

Hardware resources must be specified in two places: the device tree contains the description of all hardware devices the guest could see and the Vbus describes which resources are actually available to the uvmm.

The vbus allows the uvmm access to hardware resources in the same way as any other [L4](#) application. uvmm expects a capability named 'vbus' where it can access its hardware resources. It is possible to leave out the capability for purely virtual guests (Note that this is not actually practical on some architectures. On ARM, for example, the guest needs hardware access to the interrupt controller. Without a 'vbus' capability, interrupts will not work.) For information on how to configure a vbus, see the [IO documentation](#).

The device tree needs to contain the hardware description the guest should see. For hardware devices this usually means to use a device tree that would also be used when running the guest directly on hardware.

On startup, uvmm scans the device tree for any devices that require memory or interrupt resources and compares the required resources with the ones available from its vbus. When all resources are available, it sets up the appropriate forwarding, so that the guest now has direct access to the hardware. If the resources are not available, the device will be marked as 'disabled'. This mechanism allows to work with a standard device tree for all guests in the system while handling the actual resource allocation in a flexible manner via the vbus configuration.

The default mechanism assigns all resources 1:1, i.e. with the same memory address and interrupt number as on hardware. It is also possible to map a hardware device to a different location. In this case, the assignment between vbus device and device tree device must be known in advance and marked in the device tree using the `l4vmm, vbus-dev` property.

The following device will for example be bound with the vbus device with the HID 'l4-test,dev':

```
test@e0000000 {
    compatible = "memdev,bar";
    reg = <0 0xe0000000 0 0x50000>,
        <0 0xe1000000 0 0x50000>;
    l4vmm,vbus-dev = "l4-test,dev";
    interrupts-extended = <&gic 0 139 4>;
};
```

Resources are then matched by name. Memory resources in the vbus must be named `reg0` to `reg9` to match against the address ranges in the device tree `reg` property. Interrupts must be called `irq0` to `irq9` and will be matched against `interrupts` or `interrupts-extended` entries in the device tree. The vbus must expose resources for all resources defined in the device tree entry or the initialisation will fail.

An appropriate IO entry for the above device would thus be:

```
MEM = Io.Hw.Device(function()
    Property.hid = "l4-test,dev"
    Resource.reg0 = Io.Res.mmio(0x41000000, 0x4104ffff)
    Resource.reg1 = Io.Res.mmio(0x42000000, 0x4204ffff)
    Resource.irq0 = Io.Res.irq(134);
end)
```

Please note that HIDs on the vbus are not necessarily unique. If multiple devices with the HID given in `l4vmm,vbus-dev` are available on the vbus, then one device is chosen at random.

If no vbus device with the given HID is available, the device is disabled.

## How to enable guest suspend/resume

### Note

Currently only supported on ARM. It should work fine with Linux version 4.4 or newer.

Uvmm (partially) implements the power state coordination interface (PSCI), which is the standard ARM power management interface. To make use of this interface, you have to announce its availability to the guest operating system via the device tree like so:

```
psci {
    compatible = "arm,psci-0.2";
    method = "hvc";
};
```

The Linux guest must be configured with at least these options:

```
CONFIG_SUSPEND=y
CONFIG_ARM_PSCI=y
```

## How to communicate power management (PM) events

Uvmm can be instructed to inform a PM manager of PM events through the [L4::Platform\\_control](#) interface. To that end, uvmm may be equipped with a `pfc` cap. On suspend, uvmm will call [l4\\_platform\\_ctl\\_system\\_suspend\(\)](#).

The `pfc` cap can also be implemented by IO. In that case the guest can start a machine suspend/shutdown/reboot.

### Ram block device support

The example ramdisk works by loading a file system into RAM, which needs RAM block device support to work. In the Linux kernel configuration add: `CONFIG_BLK_DEV_RAM=y`

### Framebuffer support for uvmm/amd64 guests

Uvmm can be instructed to pass along a framebuffer to the Linux guest. To enable this three things need to be done:

1. Configure Linux to support a simple framebuffer by enabling `CONFIG_FB_SIMPLE=y` `CONFIG_X86_`  
`SYSFB=y`
2. Configure a simple framebuffer device in the device tree (currently only read by uvmm, linearer framebuffer at [0xf0000000 - 0xf1000000])  

```
simplefb { compatible = "simple-framebuffer"; reg = <0x0 0xf0000000 0x0 0x1000000>; l4vmm,fbcap = "fb";
};
```
3. Start a framebuffer instance and connect it to uvmm e.g. – Start fb-drv (but only if we need to) local `fbdrv_fb`  
`= L4.Env.vesa;` if (not `fbdrv_fb`) then `fbdrv_fb = l:new_channel(); l:start({ caps = { vbus = io_busses.fbdrv, fb`  
`= fbdrv_fb:svr(), }, log = { "fbdrv", "r" }, }, "rom/fb-drv"); end vmm.start_vm{ ext_caps = { fb = fbdrv_fb }, – ...`

### Requirements on the Fiasco.OC configuration on amd64

The kernel configuration must feature `CONFIG_SYNC_TSC=y` in order for the emulated timers to reach a sufficiently high resolution.

### Recommended Linux configuration options for uvmm/amd64 guests

The following options are recommended in addition to the amd64 defaults provided by a `make defconfig`:

Virtio support is required to access virtual devices provided by uvmm:

```
CONFIG_VIRTIO=y
CONFIG_VIRTIO_PCI=y
CONFIG_VIRTIO_BLK=y
CONFIG_BLK_MQ_VIRTIO=y
CONFIG_VIRTIO_CONSOLE=y
CONFIG_VIRTIO_INPUT=y
CONFIG_VIRTIO_NET=y
```

It is highly recommended to use the X2APIC, which needs virtualization awareness to work under uvmm:

```
CONFIG_X86_X2APIC=y
CONFIG_PARAVIRT=y
CONFIG_PARAVIRT_SPINLOCKS=y
```

### KVM clock for uvmm/amd64 guests

When executing L4Re + uvmm on QEMU, the PIT as clock source is not reliable. The paravirtualized KVM clock provides the guest with a stable clock source.

A KVM clock device is available to the guest, if the device tree contains the corresponding entry:

```
kvm_clock {
    compatible = "kvm-clock";
    reg = <0x0 0x0 0x0 0x0>;
};
```

To make use of this clock, the Linux guest must be built with the following configuration options:

```
CONFIG_HYPERVISOR_GUEST=y
CONFIG_KVM_GUEST=y
CONFIG_PTP_1588_CLOCK_KVM is not set
```

Note: KVM calls besides the KVM clock are unhandled and lead to failure in the uvmm, e.g. vmcall 0x9 for the PTP\_1588\_CLOCK\_KVM.

This is considered a development feature. The KVM clock is not required when running on physical hardware as TSC calibration via the PIT works as expected.

### Development notes for amd64

When you are developing on Linux using QEMU please note that nested virtualization support is necessary on your host system to run uvmm guests. Your host Linux version should be 4.12 or greater, **excluding 4.20**.

Check if your KVM module has nested virtualization enabled via:

```
> cat /sys/module/kvm_intel/parameters/nested
Y
```

In case it shows N instead of Y enable nested virtualization support via:

```
modprobe kvm_intel nested=1
```

On AMD platforms the module name is `kvm_amd`.



### QEMU network setup for a uvmm guest on amd64

```
qemu-system-x86_64 -M q35 -cpu host -enable-kvm -device intel-iommu -device e1000e,netdev=net0 -netdev bridge,id=net0,br=virbr0
```

where 'virbr0' is the name of the host's bridge device. The line 'allow virbr0' needs to be present in /etc/qemu/bridge.conf. The bridge can either be created via the network manager or via the command line↵:

```
brctl addbr virbr0
ip addr add 192.168.124.1/24 dev virbr0
ip link set up dev virbr0
```

In the guest linux with eth0 as network device:

```
ip a a 192.168.124.5/24 dev eth0
ip li se up dev eth0
```

Now the host and guest can ping each other using their respective IPs.

Of course, uvmm needs to be connected to io and io needs a vbus configuration for the uvmm client like this:

```
Io.add_vbusses
{
  vm_pci = Io.Vi.System_bus(function ()
    Property.num_msis = 6
    PCI = Io.Vi.PCI_bus(function ()
      pci_net = wrap(Io.system_bus():match("PCI/CC_0200"))
    end)
  end)
}
```

### QEMU emulated VirtIO devices and IO-MMU on amd64

QEMU does not route VirtIO devices through the IO-MMU per default. To use QEMU emulated VirtIO devices add the `disable-legacy=on,disable-modern=off,iommu_platform=on` flags to the option list of the device. The e1000e card in the network example above can be replaced with an virtio-net-pci card like this:

```
-device virtio-net-pci,disable-legacy=on,disable-modern=off,
      iommu_platform=on,netdev=net0
```

For more information on VirtIO devices and their options see <https://wiki.qemu.org/Features/VT-d>.

### Using the uvmm monitor interface

Uvmm implements an interface with which parts of the guest's state can be queried and manipulated at runtime. This monitor interface needs to be enabled during compilation as well as during startup of uvmm. This is described in detail below.

## Compiling uvmm with monitor interface support

To compile uvmm with monitor interface support pass the `CONFIG_MONITOR=y`, option during the `make` step (or set in in the `Makefile.config`). This option is available on all architectures but note that the set of available monitor interface features may vary significantly between them. Also note that the monitor interface will always be disabled in release mode, i.e. if `CONFIG_RELEASE_MODE=y`.

## Enabling the monitor interface at runtime

When starting a uvmm instance from inside a `ned` script using the `vmm.start_vm` function, the `mon` argument controls whether the monitor interface is enabled at runtime. There are three cases to distinguish:

- `mon=true` (default): The monitor interface is enabled but no server implementing the client side of the monitor interface is started. The monitor interface can still be utilized via `cons` but no readline functionality will be available.
- `'mon=some_binary'`: If a string is passed as the value of `mon`, the monitor interface is enabled and the string is interpreted as the name of a server binary which implements the client side of the monitor interface. This server is automatically started and has access to a `vcon` capability named `mon` at startup through which it can make use of the monitor interface. Unless you have written your own server you should specify `'uvmm_cli'` which is a server implementing a simple readline interface.
- `mon=false`: The monitor interface is disabled at runtime.

## Using the monitor interface

If the monitor interface was enabled you can connect to it via `cons` under the name `mon<n>` where `<n>` is a unique integer for every uvmm instance that is started with the monitor interface enabled (numbered starting from one in order of corresponding `vmm.start_vm` calls). If `'mon=uvmm_cli'` was specified, readline functionality such as command completion and history will be available. Enter a command followed by `enter` to run that command. To obtain a list of all available commands issue the `help` command, to obtain usage information for a specific command `foo` issue `help foo`.

### Note

Some commands will modify the guests state. Since it should be obvious to which ones this applies this is usually not specifically highlighted. Exercise reasonable caution.

## Using the guest debugger

The guest debugger provides monitoring functionality akin to a very bare-bone GDB interface, e.g. guest RAM and page table dumping, breakpointing and single stepping. Additional functionality might be added in the future.

### Note

The guest debugger is currently still under development. The guest debugger may also not be available on all architectures. To check whether the guest debugger is available check if `help dbg` returns usage information.

If the guest debugger is available, you have to manually load it at runtime using the monitor interface. This saves resources if the guest debugger is not used. To enable the guest debugger, issue the `dbg on` monitor command. Once enabled, the guest debugger can not be disabled again.

To list available guest debugger subcommands, issue `dbg help` after `dbg on`.

### Note

When using SMP, most guest debugger subcommands require you to explicitly specify a guest vcpu using an index starting from zero.

### 5.9.1 RAM configuration

#### RAM configuration for uvmm

##### Without a memory node in the device tree

- setup default RAM for guest VM.
- RAM starts either
  - at base-address which defaults to 0x0 or the base address value set via the -b cmdline option or
  - in case of identity mapping at the host-physical address of the dataspace allocated for the RAM

##### With a memory node in the device tree

The memory node needs at least the properties `device_type` and `l4vmm,dscap`:

```
memory@0 {  
    device_type = "memory";  
    l4vmm,dscap = "ram";  
}
```

Where the given `l4vmm,dscap` name is accessible in the capability namespace of the uvmm. If the capability is invalid, the memory node is disabled.

If memory nodes are given, but none provides valid RAM the configuration is invalid and uvmm refuses to boot.

Additional properties of the memory node are `reg` and `dma-ranges`.

The `reg` property describes the location in the guest's address space that should be backed by RAM.

The `dma-ranges` property describes the offset between guest-physical and host-physical addresses. The guest can evaluate this non-standard property to derive the correct DMA addresses to program into passed-through devices. Usage of this property **requires** modification of guest code.

##### Without `reg` and `dma-ranges` properties

The `reg` property is optional only in case the uvmm maps the guest's RAM into the VM under the host-physical addresses of the backing memory (`l4vmm,dscap`).

This case can be forced via the cmdline parameter `-i` and is the default for platforms without IOMMU, but with DMA capable devices on the configured vBus.

##### Without a `reg` property, but with a `dma-ranges` property

If the `-i` cmdline parameter is given, identity mapping is forced and the behavior is the same as in the case above. Additionally, the `dma-ranges` property is written

In case no `-i` cmdline parameter is given, the configuration is invalid and uvmm refuses to boot.

### With a reg property

uvmm parses the reg property of the memory node and maps the memory into the VM to the given range(s).

If the -i cmdline parameter is set, the reg property is ignored and the memory is mapped into the VM under the corresponding host-physical addresses of the backing memory (l4vmm,dscap)

### With a reg and dma-ranges property

uvmm parses the reg property of the memory node and maps the memory into the VM to the given range(s).

The dma-ranges property is filled with the corresponding host-physical addresses of the backing memory (l4vmm,dscap).

## 5.10 l4vio\_net\_p2p, a virtual network point-to-point link

The virtual network point-to-point server (p2p) connects two clients with a virtual network connection.

It uses virtio as the transport mechanism. Each virtual network p2p endpoint implements the device-side of a virtio network device. Each client can access its endpoint using the driver-side semantics of a virtio network device.

### Building and Configuration

The virtual network p2p server can be built using the [L4Re](#) build system by placing this project into the `pkg` directory.

### Starting the service

The virtual network p2p server can be started with Lua like this:

```
local p2p = L4.default_loader:new_channel();
L4.default_loader:start(
{
  caps = {
    svr = p2p:svr(),
  },
},
"rom/l4vio_net_p2p [<options>]");
```

First an IPC gate (p2p) is created which is used between the virtual network p2p server and a client to create new virtual ports. The server-side is assigned to the mandatory `svr` capability of the virtual network p2p server. See the section below on how to create a new virtual port and connect a client to it.

### Options

The following command line options are supported:

- `-p <num_usec>,--poll <num_usec>`  
Enable polling mode and set the poll interval. IRQ notification is disabled for queues while in polling mode. Must be a positive integer specified in microseconds.
- `-s <num>,--size <num>`  
Set the maximum queue size for the device-side virtio queues. Must be a power of 2 in the range of 1 to 32768 inclusive.

## Connecting a client

Prior to connecting a client to a virtual network p2p server port it has to be created using the following Lua function. It has to be called on the client side of the IPC gate capability whose server side is bound to the virtual network p2p server.

The "key=value" pairs passed to create() can be omitted and their order is not important.

```
create(obj_type, ["ds-max=<max>" , "mac-addr=<mac_address>"])
```

- obj\_type

The type of object that should be created by the server. The type must be a positive integer. Currently the following objects are supported:

- 0: Virtual p2p port

- "ds-max=<max>"

Specifies the upper limit of the number of dataspaces the client is allowed to register with the server for virtio DMA. Must be in the range of 1 to 80 inclusive. The default value is 2.

- "mac-addr=<mac\_address>"

Specify the MAC address of the endpoint where <mac\_address> is of the form X:XX:Xx:x:xx:xX.

If the create() call is successful a new capability which references a virtual network p2p server port is returned. A client uses this capability to talk to the virtual network p2p server using the virtio network protocol.

A couple of examples on how to create ports with different properties are listed below.

```
-- two normal ports with at most 4 data spaces
net0 = p2p:create(0, "ds-max=4")
net1 = p2p:create(0, "ds-max=4")
-- normal port with 4 data spaces and MAC address
net0 = p2p:create(0, "ds-max=4", "mac-addr=11:22:33:44:55:66")
```



## Chapter 6

# Bootstrap, the L4 kernel bootstrapper

### Bootstrap Command Line Options

`bootstrap` and the kernel can be configured through command line switches. `bootstrap` is responsible for parsing both command lines: `bootstrap` options are the ones directly given to `bootstrap`, whereas kernel options are those directly given to the kernel, respectively.

When using Multiboot boot, the first module directly after `bootstrap` is considered the kernel: An example using GRUB2 might look like this:

```
multiboot /path/to/bootstrap bootstrap -bs-boolean-opt -bs-opt-with-argument=foo
module /path/to/fiasco fiasco -kernel-opt-with-argument=bar -kernel-boolean-opt
```

#### Note

The exact way to provide the command line to `bootstrap` is platform-dependent. On platforms supporting booting via Multiboot Specification the command line can be specified in the boot configuration (currently x86/amd64 only, e.g. using GRUB) whereas other platforms need the command line to be compiled into the `bootstrap` binary.

Platforms utilising flattened device trees usually also allow specifying a command line through the DT. This mechanism is **not** currently supported in `bootstrap`!

#### Note

`bootstrap` will ignore options it does not understand. For most cases, this also holds true if an option's arguments cannot be understood.

## bootstrap options

Command line options directly understood by `bootstrap` itself are as follows (passed via `bootstrap` command line):

- `-comirq=<irqno>` (x86/amd64 only)

If serial logging is enabled (default on), `<irqno>` defines which IRQ to use for serial port communication. This option is ignored if `-noserial` is also specified (see below).

- `-comport=<portspec>` (x86/amd64 only)

If serial logging is enabled (default on), `<portspec>` defines which serial port to use, being one of:

- `<number>`  
Use legacy port `<number>`, e.g. use `-comport=1` for the port commonly known as *COM1*, or
- `pci:<card>:<port>`  
Use serial port number `<port>` at PCI card number `<card>`, e.g. use `-comport=pci:0:1` for the second port on the first PCI card.
- `pci:probe`  
Use this to have `bootstrap` autodiscover all PCI serial lines. On each discovered line a message will be displayed, telling the correct `<portspec>` to be given to use the respective line.

### Note

`bootstrap` does not support specifying the serial port's baudrate and always uses 115200 bps.

- `-noserial`

Disable serial logging.

- `-wait`

Wait for key press at early startup.

### Note

This is not to be confused with the kernel's `-wait` option, see below.

- `-maxmem=<mbytes>`

Limit the available memory to at most `<mbytes>` MiB.

- `-mem=<size>@<offset>`

Add a region of memory of `<size>` at `<offset>` to the system's memory map. Both `<size>` and `<offset>` may be suffixed by either G, M, or K/k to denote GiB, MiB, or KiB, respectively.

This option may be specified multiple times. If the option is not given, a platform-specific method for determining the memory layout will be used, if available.

- `-presetmem=<intval>`

Initialise memory regions with `<intval>` before starting the kernel.

- `-modaddr=<paddr>`

Relocate modules to the physical address `<paddr>`. Use this when utilising a version of GRUB that lacks support for the `modaddr` command.



## Kernel Options

Command line options for the kernel (passed on kernel command line, i.e. to first module) `bootstrap` understands are as follows.

### Note

Availability of individual options might depend on used platform and/or actual kernel configuration.

- `-wait`  
Enter debugger directly after startup, prior to executing any task.
- `-serial_esc`  
Enable entering the debugger over serial line by pressing `Esc`.
- `-noserial`  
Disable serial logging.

### Note

If this is given as kernel command line argument, it does not affect `bootstrap` but only the kernel.

- `-noscreen`  
Disable VGA console.
- `-esc`  
Enable entering the debugger by pressing `Esc` on attached keyboard.
- `-nojdb`  
Disable the kernel debugger.
- `-nohlt`  
Enable quirk for broken HLT instruction.
- `-apic`  
Use Advanced Programmable Interrupt Controller (APIC) if available and known to be well-behaving.
- `-loadcnt`  
Use load counter for performance counting.
- `-watchdog`  
Enable watchdog timer.
- `-irq0`  
Allow IRQ 0 to be used by userland. This enables some sanity checks to ensure IRQ 0 is not used by the kernel, e.g. for profiling or scheduling purposes. This only has an effect on x86.
- `-nosfn`  
Disable SFN (special fully nested) mode of interrupt controller. This only has an effect on x86 with PIC8259 interrupt controller.
- `-jdb_never_stop`  
Prevent system from stopping to enter JDB. This only has an effect on x86.
- `-kmemsize=<kbytes>`  
Reserve `<kbytes>` KiB of memory for the kernel.

- `-tbuf_entries=<number>`  
Specify the `<number>` of trace buffer entries.
- `-out_buf=<length>`  
Specify length of console buffer to be `<length>` bytes.
- `-jdb_cmd=<ctrlseq>`  
Execute JDB command sequence `<ctrlseq>` right after start-up. If `-wait` is also given, `<ctrlseq>` is executed right before entering JDB.

## Module options

Bootstrap supports module attributes for `sigma0` and the `roottask`. They need to be specified in `modules.list`, e.g.:

```
sigma0[attr:nodes=4-7] ...
```

Attributes are not supported when using multi-boot on platforms that support it. The following attributes are supported:

- `nodes`

This is a colon separated list of AMP node ranges. A range can also be a single number. Examples:

- `nodes=1` – Only node 1
- `nodes=1-3` – Nodes 1 to 3 (inclusive)
- `nodes=0:2-3` – Nodes 0, 2 and 3

If not present, the `sigma0/roottask` module is applicable to all AMP nodes.

- `reloc`

Normally the `sigma0` or `roottask` images are loaded at the preferred load address if the RAM is available at the desired location. If this is not possible, they will be relocated to some free RAM region. Setting the "reloc" module attribute to a non-empty string will always request the dynamic relocation.

This attribute can be used on no-MMU systems to maximize the size of contiguous free RAM regions.

# Chapter 7

## Deprecated List

### Struct [L4::Irq\\_mux](#)

IRQ muxer objects are no longer supported by the kernel.

### Global [L4\\_CAP\\_SIZE](#)

Superseded by [L4\\_CAP\\_OFFSET](#).

### Global [l4\\_irq\\_mux\\_chain](#) ([l4\\_cap\\_idx\\_t](#) irq, [l4\\_cap\\_idx\\_t](#) slave) [L4\\_NOTHROW](#)

IRQ muxer objects are no longer supported by the kernel.

### Global [l4\\_kip\\_clock\\_lw](#) ([l4\\_kernel\\_info\\_t](#) const \*kip) [L4\\_NOTHROW](#)

Use [l4\\_kip\\_clock\(\)](#) instead.

### Global [L4\\_SYSF\\_NONE](#)

Default flags (call to a kernel object).

Using this value as flags in the capability selector for an invocation indicates a call (send and wait for a reply).

### Global [L4Re::Util::Registry\\_server](#) < [LOOP\\_HOOKS](#) > ::[Registry\\_server](#) ([l4\\_utcb\\_t](#) \*, [L4::Cap](#) < [L4::Thread](#) > server, [L4::Cap](#) < [L4::Factory](#) > factory)

Note that this variant of the constructor is deprecated, please do not supply the UTCB pointer, it's not used.

### Global [l4util\\_kip\\_for\\_each\\_feature](#) (s)

Use [l4\\_kip\\_for\\_each\\_feature\(\)](#).

### Global [l4util\\_kip\\_kernel\\_has\\_feature](#) ([l4\\_kernel\\_info\\_t](#) const \*k, char const \*str)

Use [l4\\_kip\\_kernel\\_has\\_feature\(\)](#).

### Global [l4util\\_micros2l4to](#) ([l4\\_uint64\\_t](#) us) [L4\\_NOTHROW](#)

Use [l4\\_timeout\\_from\\_us\(\)](#).



# Chapter 8

## Topic Index

### 8.1 Topics

Here is a list of all topics with brief descriptions:

Base API	131
Basic Macros	134
C++ IPC Interface Definition.	138
Internal Helpers	138
Cache Consistency	139
Capabilities	142
Error codes	146
Fiasco extensions	148
Fiasco real time scheduling extensions	151
Kernel Debugger	152
Kernel Tracing	159
Flex pages	162
Integer Types	180
Kernel Interface Page	182
Fiasco-UX Virtual devices	188
Memory descriptors (C version)	189
Kernel Objects	194
DMA space	195
Factory	196
IPC-Gate API	205
IRQs	208
Interrupt controller	222
Kernel-provided semaphore	238
L4 kernel object type information	240
Platform Control C API	241
Scheduler	247
Task	255
Thread	265
Thread control	285
vCPU API	292
Virtual Console	296
Virtual Machines	311
VM API for SVM	311
VM API for TZ	312

VM API for VMX . . . . .	313
Memory operations. . . . .	330
Memory related . . . . .	332
Object Invocation . . . . .	340
Error Handling . . . . .	358
Message Items . . . . .	364
Message Tag . . . . .	367
Realtime API . . . . .	381
Timeouts . . . . .	381
Virtual Registers (UTCBS) . . . . .	389
ARM Virtual Registers (UTCB) . . . . .	394
Buffer Registers (BRs) . . . . .	394
Message Registers (MRs) . . . . .	395
Exception registers . . . . .	396
Thread Control Registers (TCRs) . . . . .	399
amd64 Virtual Registers (UTCB) . . . . .	400
x86 Virtual Registers (UTCB) . . . . .	400
EDID parsing functionality . . . . .	401
IO interface . . . . .	405
IPC Helpers . . . . .	412
IRQ handling library . . . . .	414
Interface for asynchronous ISR handlers. . . . .	414
Interface for asynchronous ISR handlers with a given IRQ capability. . . . .	416
Interface using direct functionality. . . . .	417
Interface using direct functionality. . . . .	421
L4 IPC Opcodes . . . . .	423
L4 VIRTIO Interface . . . . .	427
L4 VIRTIO Block Device . . . . .	428
L4 VIRTIO Input Device . . . . .	429
L4 VIRTIO Network Device . . . . .	430
L4 VIRTIO Transport Layer . . . . .	431
L4 Vbus functions . . . . .	439
L4vbus GPIO functions . . . . .	450
L4vbus PCI functions . . . . .	459
L4vbus power management functions . . . . .	465
L4Re C Interface . . . . .	467
Capability allocator . . . . .	469
DMA Space Interface . . . . .	470
Dataspace interface . . . . .	474
Debug interface . . . . .	479
Event interface . . . . .	480
Initial Environment . . . . .	483
Kumem allocator utility . . . . .	488
L4Re Util C Interface . . . . .	488
Log interface . . . . .	488
Memory allocator . . . . .	491
Namespace interface . . . . .	495
Parent interface . . . . .	499
Region map interface . . . . .	499
Video API . . . . .	513
L4Re C++ Interface . . . . .	520
Auxiliary data . . . . .	522
C++ Exceptions . . . . .	523
Console API . . . . .	524
Debugging API . . . . .	524
Event API . . . . .	525

L4Re ELF Auxiliary Information . . . . .	526
L4Re Protocol identifiers . . . . .	528
L4Re Util C++ Interface . . . . .	529
Kumem utilities . . . . .	530
L4Re Capability API . . . . .	531
Logging interface . . . . .	533
Name-space API . . . . .	534
Parent API . . . . .	535
Region map API . . . . .	535
Vbus API . . . . .	536
L4SHM-based ring buffer implementation . . . . .	538
Internal . . . . .	538
Receiver . . . . .	540
Sender . . . . .	540
Server-Side IPC framework . . . . .	540
Shared Memory Library . . . . .	542
Chunks . . . . .	548
Consumer . . . . .	551
Producer . . . . .	555
Signals . . . . .	558
Consumer . . . . .	562
Producer . . . . .	566
Sigma0 API . . . . .	566
Internal constants . . . . .	571
Small C++ Template Library . . . . .	572
Utility Functions . . . . .	576
Atomic Instructions . . . . .	583
Bit Manipulation . . . . .	600
Bitmap graphics and fonts . . . . .	607
Functions for rendering bitmap data in frame buffers . . . . .	608
Functions for rendering bitmap fonts to frame buffers . . . . .	608
CPU related functions . . . . .	608
Comfortable Command Line Parsing . . . . .	611
ELF binary format . . . . .	613
Functions to manipulate the local IDT . . . . .	636
IA32 Port I/O API . . . . .	637
Internal functions . . . . .	643
Kernel Interface Page API . . . . .	644
Low-Level Thread Functions . . . . .	646
Random number support . . . . .	646
Timestamp Counter . . . . .	647
vCPU Support Library . . . . .	654
Extended vCPU support . . . . .	662





## Chapter 9

# Namespace Index

### 9.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

<a href="#">cxx</a>	Our C++ library . . . . .	663
<a href="#">cxx::Bits</a>	Internal helpers for the cxx package . . . . .	667
<a href="#">L4</a>	<a href="#">L4</a> low-level kernel interface . . . . .	667
<a href="#">L4::lpc</a>	IPC related functionality . . . . .	677
<a href="#">L4::lpc::Msg</a>	IPC Message related functionality . . . . .	685
<a href="#">L4::lpc_svr</a>	Helper classes for <a href="#">L4::Server</a> instantiation . . . . .	690
<a href="#">L4::Typeid</a>	Definition of interface data-type helpers . . . . .	691
<a href="#">L4::Types</a>	<a href="#">L4</a> basic type helpers for C++ . . . . .	692
<a href="#">L4Re</a>	<a href="#">L4Re</a> C++ Interfaces . . . . .	693
<a href="#">L4Re::Util</a>	Documentation of the <a href="#">L4</a> Runtime Environment utility functionality in C++ . . . . .	709
<a href="#">L4Re::Vfs</a>	Virtual file system for interfaces in POSIX libc . . . . .	718
<a href="#">L4vbus</a>	C++ interface of the <a href="#">Vbus</a> API. . . . .	719
<a href="#">L4virtio</a>	L4-VIRTIO Transport C++ API . . . . .	720



# Chapter 10

## Hierarchical Index

### 10.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

L4::lpc::Array_ref< char const, unsigned long > . . . . .	1033
L4::lpc::Array_ref< ELEM_TYPE, Array_len_default > . . . . .	1033
cxx::Bits::Base_avl_set< ITEM_TYPE, Lt_functor< ITEM_TYPE >, New_allocator, Bits::Avl_set_get_↵ key< ITEM_TYPE > > . . . . .	807
cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, Lt_functor< KEY_TYPE >, New_allocator, Bits::Avl_map_get_key< KEY_TYPE > > . . . . .	807
cxx::Bits::Base_avl_set< Pair< Region, Hdlr >, cxx::Lt_functor< Region >, Alloc, Bits::Avl_map_get_↵ key< Region > > . . . . .	807
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > . . . . .	763
cxx::Base_slab< sizeof(Type), L4_PAGESIZE, 2, New_allocator > . . . . .	763
cxx::Base_slab_static< sizeof(Type), L4_PAGESIZE, 2, New_allocator > . . . . .	769
cxx::Bits::Basic_list< Bits::Basic_list_policy< cxx::Base_slab::Slab_i, H_list_item > > . . . . .	823
cxx::Bits::Basic_list< Bits::Basic_list_policy< Observer, H_list_item > > . . . . .	823
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item > > . . . . .	823
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item_t< T > > > . . . . .	823
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, S_list_item > > . . . . .	823
cxx::Bits::Basic_list< Bits::Basic_list_policy< Timeout, H_list_item > > . . . . .	823
cxx::Bits::Basic_list< Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_base > > > .	823
Block_device::Device_discard_feature . . . . .	721
Block_device::Device_mgr< DEV, FACTORY, SCHEDULER > . . . . .	722
Block_device::Device_with_notification_domain< DEV > . . . . .	723
Block_device::Dma_region_info . . . . .	724
Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, bool > . . . . .	731
Block_device::Partitioned_device< BASE_DEV > . . . . .	738
Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, true > . . . . .	732
Block_device::Inout_block . . . . .	733
Block_device::Inout_memory< DEV > . . . . .	734
Block_device::Mem_region_info . . . . .	735
Block_device::Notification_domain . . . . .	735
Block_device::Partition_info . . . . .	736
Block_device::Partition_reader< DEV > . . . . .	737
Block_device::Pending_request . . . . .	739
Block_device::Scheduler_base< DEV > . . . . .	742
Block_device::Rr_scheduler< DEV > . . . . .	740
L4::Types::Bool< __lface_conflict< I, I2 >::value  _lface_conflict< I, LIST::type >::value > . . . . .	1339

L4::Types::Bool< false > . . . . .	1339
L4::Types::False . . . . .	1340
L4::Types::Same< A, B > . . . . .	1352
L4::Types::Bool< I1::Proto !=PROTO_EMPTY &&I1::Proto==I2::Proto > . . . . .	1339
L4::Types::Bool< lface_conflict< I, L2::type >::value  _Conflict< L1::type, L2::type >::value > . . . . .	1339
L4::Types::Bool< true > . . . . .	1339
L4::Types::True . . . . .	1354
L4::lpc::Msg::ls_valid_rpc_type< A * > . . . . .	1082
L4::lpc::Msg::ls_valid_rpc_type< T > . . . . .	1082
L4::Types::Bool< Typeid::Conflict< L1::type, L2::type >::value  Conflict< L1, LIST... >::value  Conflict< L2, LIST... >::value > . . . . .	1339
L4::Types::Bool< Typeid::lface_conflict< I::type, L::type >::value  lface_conflict< I, LIST... >::value > . . . . .	1339
cxx::Bits::Bst< _Node, Bits::Avl_map_get_key< KEY_TYPE >, Lt_functor< KEY_TYPE > > . . . . .	825
cxx::Bits::Bst< _Node, Bits::Avl_map_get_key< Region >, cxx::Lt_functor< Region > > . . . . .	825
cxx::Bits::Bst< _Node, Bits::Avl_set_get_key< ITEM_TYPE >, Lt_functor< ITEM_TYPE > > . . . . .	825
cxx::Bits::Bst< _Node, GET_KEY, COMPARE > . . . . .	825
cxx::Bits::Bst< Entry, Names_get_key, Lt_functor< typename Get_key::Key_type > > . . . . .	825
cxx::Bits::Bst< Node, Get_key, Lt_functor< typename Get_key::Key_type > > . . . . .	825
L4::lpc::Msg::Cnt_val_ops< Detail::_Plain< T >::type, DIR, CLASS > . . . . .	1067
L4::lpc::Msg::Cnt_val_ops< Detail::_Plain< typename ELEM::arg_type >::type, typename Direction< A * >::type, typename Class< typename Detail::_Plain< A * >::type >::type > . . . . .	1067
L4::lpc::Msg::Cnt_val_ops< Detail::_Plain< typename ELEM::arg_type >::type, typename Direction< A const * >::type, typename Class< typename Detail::_Plain< A const * >::type >::type > . . . . .	1067
L4::lpc::Msg::Cnt_val_ops< Detail::_Plain< typename ELEM::arg_type >::type, typename Direction< Array< A, LEN > & >::type, typename Class< typename Detail::_Plain< Array< A, LEN > & >::type >::type > . . . . .	1067
L4::lpc::Msg::Cnt_val_ops< Detail::_Plain< typename ELEM::arg_type >::type, typename Direction< Array< A, LEN > >::type, typename Class< typename Detail::_Plain< Array< A, LEN > >::type >::type > . . . . .	1067
L4::lpc::Msg::Cnt_val_ops< Detail::_Plain< typename ELEM::arg_type >::type, typename Direction< String< A, LEN > & >::type, typename Class< typename Detail::_Plain< String< A, LEN > & >::type >::type > . . . . .	1067
L4::lpc::Msg::Cnt_val_ops< Detail::_Plain< typename ELEM::arg_type >::type, typename Direction< T >::type, typename Class< typename Detail::_Plain< T >::type >::type > . . . . .	1067
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > . . . . .	763
cxx::Slab< Type, Slab_size, Max_free, Alloc > . . . . .	890
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc > . . . . .	769
cxx::Slab_static< Type, Slab_size, Max_free, Alloc > . . . . .	894
cxx::Bitfield< T, LSB, MSB > . . . . .	774
cxx::Bitfield< T, LSB, MSB >::Value_base< TT > . . . . .	788
cxx::Bitfield< T, LSB, MSB >::Value< TT > . . . . .	786
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT > . . . . .	789
cxx::Bitmap_base . . . . .	795
cxx::Bitmap< BITS > . . . . .	790
cxx::Bitmap_base::Bit . . . . .	803
cxx::Bitmap_base::Char< BITS > . . . . .	804
cxx::Bitmap_base::Word< BITS > . . . . .	805
cxx::Bits::Avl_map_get_key< KEY_TYPE > . . . . .	806
cxx::Bits::Avl_set_get_key< KEY_TYPE > . . . . .	806
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > . . . . .	807
cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc > . . . . .	744
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC > . . . . .	744
cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC > . . . . .	750
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node . . . . .	820
cxx::Bits::Basic_list< POLICY > . . . . .	823
cxx::H_list< T, Bits::Basic_list_policy< T, H_list_item_t< T > > > . . . . .	848

cxx::H_list_t< T > . . . . .	859
cxx::H_list< Observer > . . . . .	848
cxx::H_list< cxx::Base_slab::Slab_i > . . . . .	848
cxx::H_list< Weak_ref_base, Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_base > > > . . . . .	848
cxx::H_list< Timeout > . . . . .	848
cxx::S_list< T, Bits::Basic_list_policy< T, S_list_item > > . . . . .	887
cxx::H_list< T, POLICY > . . . . .	848
cxx::H_list_t< Weak_ref_base > . . . . .	859
cxx::Weak_ref_base::List . . . . .	911
cxx::S_list< T, POLICY > . . . . .	887
cxx::Bits::Bst< Node, Get_key, Compare > . . . . .	825
cxx::Avl_tree< _Node, Bits::Avl_map_get_key< KEY_TYPE >, Lt_functor< KEY_TYPE > > . . . . .	754
cxx::Avl_tree< _Node, Bits::Avl_set_get_key< ITEM_TYPE >, Lt_functor< ITEM_TYPE > > . . . . .	754
cxx::Avl_tree< Entry, Names_get_key > . . . . .	754
cxx::Avl_tree< _Node, Bits::Avl_map_get_key< Region >, cxx::Lt_functor< Region > > . . . . .	754
cxx::Avl_tree< _Node, GET_KEY, COMPARE > . . . . .	754
cxx::Avl_tree< Node, Get_key, Compare > . . . . .	754
cxx::Bits::Bst_node . . . . .	839
cxx::Avl_tree_node . . . . .	761
cxx::Bits::Direction . . . . .	841
cxx::Bits::Smart_ptr_list< ITEM > . . . . .	843
cxx::Bits::Smart_ptr_list_item< T, STORE_T > . . . . .	846
cxx::Ref_obj_list_item< Connection > . . . . .	882
cxx::Ref_obj_list_item< T > . . . . .	882
cxx::H_list_item_t< ELEM_TYPE > . . . . .	856
L4::lpc_svr::Timeout . . . . .	1143
Block_device::Errand::Errand . . . . .	724
Block_device::Errand::Poll_errand . . . . .	728
cxx::List< D, Alloc > . . . . .	862
cxx::List< D, Alloc >::Iter . . . . .	864
cxx::List_alloc . . . . .	865
cxx::List_item . . . . .	868
cxx::List_item::Iter . . . . .	872
cxx::List_item::T_iter< E > . . . . .	873
cxx::List_item::T_iter< T, Poly > . . . . .	873
cxx::Lt_functor< Obj > . . . . .	875
cxx::New_allocator< _Type > . . . . .	875
cxx::Nothrow . . . . .	876
cxx::Pair< First, Second > . . . . .	877
cxx::Pair_first_compare< Cmp, Typ > . . . . .	880
cxx::Ref_ptr< T, CNT > . . . . .	883
cxx::static_vector< T, IDX > . . . . .	898
cxx::String . . . . .	899
L4Re::Util::Names::Name . . . . .	1599
L4virtio::Svr::Device_t< Ds_data > . . . . .	1919
L4virtio::Svr::Block_dev_base< Ds_data > . . . . .	1848
L4virtio::Svr::Device_t< Mem_region_info > . . . . .	1919
L4virtio::Svr::Driver_mem_list_t< Ds_data > . . . . .	1927
L4virtio::Svr::Driver_mem_list_t< Mem_region_info > . . . . .	1927
L4virtio::Svr::Driver_mem_region_t< Ds_data > . . . . .	1936
Elf32_Auxv . . . . .	915
Elf32_Dyn . . . . .	916
Elf32_Ehdr . . . . .	917
Elf32_Phdr . . . . .	920
Elf32_Rel . . . . .	921

Elf32_Rela . . . . .	922
Elf32_Shdr . . . . .	923
Elf32_Sym . . . . .	924
Elf64_Auxv . . . . .	925
Elf64_Dyn . . . . .	926
Elf64_Ehdr . . . . .	927
Elf64_Phdr . . . . .	930
Elf64_Rel . . . . .	931
Elf64_Rela . . . . .	932
Elf64_Shdr . . . . .	933
Elf64_Sym . . . . .	934
L4::Epiface_t0< IFACE, BASE > . . . . .	990
L4::Epiface_t0< void, BASE > . . . . .	990
L4Re::Event_buffer_t< PAYLOAD > . . . . .	1488
L4Re::Util::Event_buffer_t< PAYLOAD > . . . . .	1589
L4Re::Util::Event_buffer_consumer_t< PAYLOAD > . . . . .	1584
L4::Types::Flags_ops_t< Flags > . . . . .	1346
L4::Types::Flags_ops_t< Flags_t< DT, T > > . . . . .	1346
L4::Types::Flags_t< DT, T > . . . . .	1348
gfxbitmap_offset . . . . .	935
cxx::H_list_item_t< Weak_ref_base > . . . . .	856
cxx::Weak_ref_base . . . . .	908
cxx::Weak_ref< T > . . . . .	905
Block_device::Inout_memory< Device_type > . . . . .	734
L4::Kobject_2t< Console, Video::Goos, Event, L4::PROTO_EMPTY > . . . . .	1192
L4Re::Console . . . . .	1443
L4::Kobject_2t< Debug_obj_t< BASE >, BASE, Debug_obj, L4::PROTO_EMPTY > . . . . .	1192
L4::Kobject_x< lommu, Proto_t< L4_PROTO_IOMMU >, Type_info::Demand_t< 1 > > . . . . .	1205
L4::lommu . . . . .	1026
L4::Alloc_list . . . . .	936
L4::Basic_registry . . . . .	941
L4Re::Util::Object_registry . . . . .	1607
L4::Cap_base . . . . .	952
L4::Cap< void > . . . . .	946
L4::Cap< L4::Rcv_endpoint > . . . . .	946
L4::Cap< L4Re::Rm > . . . . .	946
L4::Cap< L4::Irq > . . . . .	946
L4::Cap< L4Re::Namespace > . . . . .	946
L4::Cap< L4Re::Dataspace > . . . . .	946
L4::Cap< L4::Vcon > . . . . .	946
L4::Cap< L4::Semaphore > . . . . .	946
L4::Cap< L4::Thread > . . . . .	946
L4::Cap< L4::Factory > . . . . .	946
L4::Cap< L4Re::Video::Goos > . . . . .	946
L4::Cap< L4vbus::Vbus > . . . . .	946
L4::Cap< L4virtio::Device > . . . . .	946
L4::Cap< T > . . . . .	946
L4::Smart_cap< T, SMART > . . . . .	1272
L4::Epiface . . . . .	982
L4::Epiface_t0< void, Epiface > . . . . .	990
L4::Epiface_t0< L4virtio::Device, L4::Epiface > . . . . .	990
L4::Epiface_t0< IFACE, L4::Epiface > . . . . .	990
L4::Epiface_t0< L4::Kobject, L4::Epiface > . . . . .	990
L4::Epiface_t0< RPC_IFACE, BASE > . . . . .	990
L4::Epiface_t< Virtio_client< DEV >, L4virtio::Device > . . . . .	987
L4::Epiface_t< Null_handler, L4::Kobject > . . . . .	987

L4::Epiface_t< Block_dev< Ds_data >, L4virtio::Device >	987
L4::Irqep_t< Irq_object >	1174
L4::Irqep_t< Host_irq >	1174
L4::Epiface_t< Derived, IFACE, BASE, bool >	987
L4::Irqep_t< Derived, BASE, bool >	1174
L4::Server_object	1259
L4::Server_object_t< Kobject >	1263
L4::Irq_handler_object	1167
L4::Server_object_t< IFACE, L4::Server_object >	1263
L4::Server_object_t< IFACE, BASE >	1263
L4::Server_object_x< Derived, IFACE, BASE >	1269
L4::Server_object_x< Derived, IFACE, BASE >	1269
L4::Exception_tracer	994
L4::Base_exception	938
L4::Invalid_capability	1021
L4::Runtime_error	1237
L4::Bounds_error	944
L4::Com_error	963
L4::Element_already_exists	976
L4::Element_not_found	979
L4::Out_of_memory	1211
L4::Unknown_error	1367
L4::Factory::Lstr	1005
L4::Factory::Nil	1006
L4::Factory::S	1007
L4::IOModifier	1028
L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >	1032
L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >	1033
L4::lpc::Array< char const, unsigned long >	1029
L4::lpc::Array< ELEM_TYPE, LEN_TYPE >	1029
L4::lpc::As_value< T >	1034
L4::lpc::Call	1035
L4::lpc::Call_t< RIGHTS >	1037
L4::lpc::Call_zero_send_timeout	1038
L4::lpc::Cap< T >	1039
L4::lpc::Gen_fpage	1042
L4::lpc::Rcv_fpage	1095
L4::lpc::Snd_fpage	1100
L4::lpc::In_out< T >	1045
L4::lpc::Istream	1056
L4::lpc::Iostream	1046
L4::lpc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS >	1067
L4::lpc::Msg::Cls_buffer	1068
L4::lpc::Msg::Do_rcv_buffers	1077
L4::lpc::Msg::Cls_data	1069
L4::lpc::Msg::Do_in_data	1073
L4::lpc::Msg::Do_out_data	1075
L4::lpc::Msg::Cls_item	1070
L4::lpc::Msg::Do_in_items	1074
L4::lpc::Msg::Do_out_items	1076
L4::lpc::Msg::Dir_in	1071
L4::lpc::Msg::Do_in_data	1073
L4::lpc::Msg::Do_in_items	1074
L4::lpc::Msg::Do_rcv_buffers	1077
L4::lpc::Msg::Dir_out	1072

L4::lpc::Msg::Do_out_data	1075
L4::lpc::Msg::Do_out_items	1076
L4::lpc::Msg::Elem< Array< A, LEN > & >	1079
L4::lpc::Msg::Elem< Array< A, LEN > >	1080
L4::lpc::Msg::Elem< Array_ref< A, LEN > & >	1081
L4::lpc::Msg::Svr_arg_pack< IPC_TYPE >	1084
L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >	1084
L4::lpc::Msg_ptr< T >	1085
L4::lpc::Opt< T >	1086
L4::lpc::Ostream	1088
L4::lpc::lostream	1046
L4::lpc::Out< T >	1094
L4::lpc::Ret_array< T >	1097
L4::lpc::Send_only	1098
L4::lpc::Small_buf	1098
L4::lpc::Str_cp_in< T >	1102
L4::lpc::Varg	1103
L4::lpc::Varg_list_ref	1111
L4::lpc::Varg_list< MAX >	1109
L4::lpc::Varg_list_ref::iterator	1113
L4::lpc_svr::Compound_reply	1123
L4::lpc_svr::Default_loop_hooks	1125
L4::Server< L4::lpc_svr::Default_loop_hooks >	1254
L4::Server< LOOP_HOOKS >	1254
L4Re::Util::Registry_server< LOOP_HOOKS >	1616
L4Re::Util::Br_manager_hooks	1562
L4::lpc_svr::Default_setup_wait	1128
L4::lpc_svr::Default_timeout	1129
L4::lpc_svr::Default_loop_hooks	1125
L4Re::Util::Br_manager_hooks	1562
L4::lpc_svr::Direct_dispatch< R >	1131
L4::lpc_svr::Exc_dispatch< R, Exc >	1134
L4::lpc_svr::Direct_dispatch< R * >	1133
L4::lpc_svr::Ignore_errors	1136
L4::lpc_svr::Default_loop_hooks	1125
L4Re::Util::Br_manager_hooks	1562
L4Re::Util::Br_manager_timeout_hooks	1564
L4::lpc_svr::Server_iface	1137
L4::lpc_svr::Timeout_queue_hooks< Loop_hooks, L4Re::Util::Br_manager >	1151
L4::lpc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager >	1151
L4Re::Util::Br_manager_timeout_hooks	1564
L4::lpc_svr::Br_manager_no_buffers	1119
L4::lpc_svr::Default_loop_hooks	1125
L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >	1151
L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >	1151
L4Re::Util::Br_manager	1556
L4::lpc_svr::Timeout_queue_hooks< Loop_hooks, L4Re::Util::Br_manager >	1151
L4Re::Util::Br_manager_hooks	1562
L4::lpc_svr::Timeout_queue	1146
L4::Kip::Mem_desc	1179
L4::Kobject	1189
L4::Kobject_t< Arm_smccc, L4::Kobject, PROTO, Type_info::Demand_t<> >	1200
L4::Kobject_t< Debugger, Kobject, L4_PROTO_DEBUGGER >	1200
L4::Debugger	967
L4::Kobject_t< Exception, L4::Kobject, PROTO, Type_info::Demand_t<> >	1200



L4::Kobject_t< Factory, Kobject, L4_PROTO_FACTORY > . . . . .	1200
L4::Factory . . . . .	996
L4::Kobject_t< Mem_alloc, L4::Factory, L4::PROTO_EMPTY > . . . . .	1200
L4Re::Mem_alloc . . . . .	1503
L4::Kobject_t< Io_pager, L4::Kobject, PROTO, Type_info::Demand_t<> > . . . . .	1200
L4::Kobject_t< Irq_eoi, L4::Kobject, PROTO, Type_info::Demand_t<> > . . . . .	1200
L4::Kobject_t< Derived, L4::Kobject, L4::PROTO_ANY, Type_info::Demand_t<> > . . . . .	1200
L4::Kobject_t< Meta, Kobject, L4_PROTO_META > . . . . .	1200
L4::Meta . . . . .	1206
L4::Kobject_t< Platform_control, Kobject, L4_PROTO_PLATFORM_CTL > . . . . .	1200
L4::Platform_control . . . . .	1218
L4::Kobject_t< Rcv_endpoint, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > > . . . . .	1200
L4::Rcv_endpoint . . . . .	1229
L4::Kobject_2t< Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER > . . . . .	1192
L4::Irq . . . . .	1157
L4::Kobject_t< Ipc_gate, Rcv_endpoint, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > > . . . . .	1200
L4::Ipc_gate . . . . .	1114
L4::Kobject_t< Task, Kobject, L4_PROTO_TASK, Type_info::Demand_t< 2 > > . . . . .	1200
L4::Task . . . . .	1277
L4::Kobject_t< Vm, Task, L4_PROTO_VM > . . . . .	1200
L4::Vm . . . . .	1378
L4::Vm . . . . .	1378
L4::Kobject_t< Thread, Kobject, L4_PROTO_THREAD, Type_info::Demand_t< 1 > > . . . . .	1200
L4::Thread . . . . .	1289
L4::Kobject_t< Vcpu_context, Kobject, L4_PROTO_VCPU_CONTEXT > . . . . .	1200
L4::Kobject_t< Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE, L4::Type_info::Demand_t< 1 > > . . . . .	1200
L4Re::Dataspace . . . . .	1446
L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event > . . . . .	1195
L4vbus::Vbus . . . . .	1765
L4::Kobject_t< Debug_obj, L4::Kobject, L4RE_PROTO_DEBUG > . . . . .	1200
L4Re::Debug_obj . . . . .	1459
L4::Kobject_t< Dma_space, L4::Kobject, PROTO, L4::Type_info::Demand_t< 1 > > . . . . .	1200
L4::Kobject_t< Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR > . . . . .	1200
L4Re::Inhibitor . . . . .	1493
L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event > . . . . .	1195
L4::Kobject_t< Mmio_space, L4::Kobject, L4RE_PROTO_MMIO_SPACE > . . . . .	1200
L4Re::Mmio_space . . . . .	1509
L4::Kobject_t< Namespace, L4::Kobject, L4RE_PROTO_NAMESPACE, L4::Type_info::Demand_t< 1 > > . . . . .	1200
L4Re::Namespace . . . . .	1513
L4::Kobject_t< Cmd_control, L4::Kobject, L4::PROTO_ANY, Type_info::Demand_t<> > . . . . .	1200
L4::Kobject_t< Parent, L4::Kobject, L4RE_PROTO_PARENT > . . . . .	1200
L4Re::Parent . . . . .	1522
L4::Kobject_t< Goos, L4::Kobject, L4RE_PROTO_GOOS > . . . . .	1200
L4Re::Video::Goos . . . . .	1668
L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND > . . . . .	1192
L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND > . . . . .	1195
L4::Kobject_demand< T > . . . . .	1199
L4::Kobject_t< Derived, Base, PROTO, S_DEMAND > . . . . .	1200
L4::Kobject_typeid< T > . . . . .	1201
L4::Kobject_typeid< void > . . . . .	1204
L4::Kobject_x< Derived, ARGS > . . . . .	1205
L4::Poll_timeout_counter . . . . .	1222
L4::Poll_timeout_kipclock . . . . .	1225
L4::Proto_t< P > . . . . .	1228

L4::Registry_iface . . . . .	1233
L4Re::Util::Object_registry . . . . .	1607
L4::String . . . . .	1276
L4::Thread::Attr . . . . .	1302
L4::Thread::Modify_senders . . . . .	1307
L4::Type_info . . . . .	1313
L4::Type_info::Demand . . . . .	1314
L4::Type_info::Demand_t< __l::Max< D1::Caps, D2::Caps >::Res, D1::Flags D2::Flags, __l::Max< D1::Mem, D2::Mem >::Res, __l::Max< D1::Ports, D2::Ports >::Res > . . . . .	1318
L4::Type_info::Demand_union_t< Kobject_typeid< T1 >::Demand, Kobject_demand< T2... > > . . . . .	1320
L4::Type_info::Demand_union_t< D1, D2 > . . . . .	1320
L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS > . . . . .	1318
L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y > . . . . .	1325
L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... > . . . . .	1325
L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y > . . . . .	1327
L4::Typeid::Detail::Rpc_end . . . . .	1327
L4::Typeid::Detail::_Rpc< void, 0, OPERATION > . . . . .	1323
L4::Typeid::Rpc_nocode< OPERATION > . . . . .	1330
L4::Typeid::Detail::_Rpc< L4::Opcode, 0, RPCS... > . . . . .	1323
L4::Typeid::Rpc< RPCS > . . . . .	1332
L4::Typeid::Detail::_Rpc< OPCODE_TYPE, 0, RPCS... > . . . . .	1323
L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS > . . . . .	1335
L4::Typeid::Detail::_Rpc< l4_umword_t, 0, ARG... > . . . . .	1323
L4::Typeid::Rpc_sys< ARG > . . . . .	1337
L4::Typeid::Detail::_Rpc< OPCODE, O, X > . . . . .	1323
L4::Typeid::P_dispatch< LIST > . . . . .	1328
L4::Typeid::Raw_ipc< CLASS > . . . . .	1329
L4::Typeid::Rpc_code< OPCODE_TYPE > . . . . .	1334
L4::Types::Bool< V > . . . . .	1339
L4::Types::Flags< BITS_ENUM, UNDERLYING > . . . . .	1342
L4::Types::Flags_ops_t< DT > . . . . .	1346
L4::Types::Int_for_size< SIZE, bool > . . . . .	1350
L4::Types::Int_for_type< T > . . . . .	1351
L4::Uart . . . . .	1356
L4::Uart_apb . . . . .	1361
l4_buf_regs_t . . . . .	1383
l4_exc_regs_t . . . . .	1384
l4_ext_vcpu_state_vmx_t . . . . .	1387
l4_fpage_t . . . . .	1388
l4_icu_info_t . . . . .	1389
L4::Icu::Info . . . . .	1019
l4_icu_msi_info_t . . . . .	1390
l4_kernel_info_mem_desc_t . . . . .	1391
l4_kernel_info_t . . . . .	1392
l4_msg_regs_t . . . . .	1394
l4_msgtag_t . . . . .	1395
l4_sched_cpu_set_t . . . . .	1397
l4_sched_param_t . . . . .	1401
l4_snd_fpage_t . . . . .	1402
l4_thread_regs_t . . . . .	1403
l4_timeout_s . . . . .	1404
l4_timeout_t . . . . .	1405
l4_vcon_attr_t . . . . .	1406
l4_vcpu_arch_state_t . . . . .	1407
l4_vcpu_ipc_regs_t . . . . .	1408
l4_vcpu_regs_t . . . . .	1409

<code>l4_vcpu_state_t</code>	1413
<code>L4vcpu::Vcpu</code>	1780
<code>l4_vhw_descriptor</code>	1416
<code>l4_vhw_entry</code>	1417
<code>l4_vm_svm_vmcb_control_area</code>	1418
<code>l4_vm_svm_vmcb_state_save_area</code>	1419
<code>l4_vm_svm_vmcb_state_save_area_seg</code>	1420
<code>l4_vm_svm_vmcb_t</code>	1420
<code>l4_vm_tz_state</code>	1421
<code>l4_vmx_offset_table_t</code>	1422
<code>L4drivers::Register_block&lt; MAX_BITS, BLOCK &gt;</code>	1425
<code>L4drivers::Register_block_base&lt; MAX_BITS &gt;</code>	1428
<code>L4drivers::Register_block_impl&lt; BASE, MAX_BITS &gt;</code>	1429
<code>L4drivers::Mmio_register_block&lt; MAX_BITS &gt;</code>	1424
<code>L4drivers::Register_block_tmpl&lt; BLOCK &gt;</code>	1431
<code>L4drivers::Ro_register_block&lt; MAX_BITS, BLOCK &gt;</code>	1436
<code>L4drivers::Ro_register_tmpl&lt; BITS, BLOCK &gt;</code>	1438
<code>L4drivers::Register_tmpl&lt; BITS, BLOCK &gt;</code>	1432
<code>L4Re::Cap_alloc</code>	1440
<code>L4Re::Dataspace::F</code>	1457
<code>L4Re::Dataspace::Stats</code>	1458
<code>L4Re::Default_event_payload</code>	1462
<code>L4Re::Env</code>	1469
<code>L4Re::Event_buffer_t&lt; PAYLOAD &gt;</code>	1488
<code>L4Re::Event_buffer_t&lt; PAYLOAD &gt;::Event</code>	1492
<code>L4Re::Rm::F</code>	1545
<code>L4Re::Rm::Range</code>	1547
<code>L4Re::Rm::Unique_region&lt; T &gt;</code>	1548
<code>L4Re::Smart_cap_auto&lt; Unmap_flags &gt;</code>	1554
<code>L4Re::Smart_count_cap&lt; Unmap_flags &gt;</code>	1555
<code>L4Re::Util::Cap_alloc_base</code>	1568
<code>L4Re::Util::Counter&lt; COUNTER &gt;</code>	1569
<code>L4Re::Util::Counter_atomic&lt; COUNTER &gt;</code>	1571
<code>L4Re::Util::Counting_cap_alloc&lt; COUNTERTYPE, Dbg &gt;</code>	1572
<code>L4Re::Util::Dataspace_svr</code>	1578
<code>L4Re::Util::Event_svr&lt; SVR &gt;</code>	1593
<code>L4Re::Util::Event_t&lt; PAYLOAD &gt;</code>	1595
<code>L4Re::Util::Item_alloc_base</code>	1599
<code>L4Re::Util::Names::Name_space</code>	1603
<code>L4Re::Util::Ref_cap&lt; T &gt;</code>	1614
<code>L4Re::Util::Ref_del_cap&lt; T &gt;</code>	1615
<code>L4Re::Util::Smart_cap_auto&lt; Unmap_flags &gt;</code>	1621
<code>L4Re::Util::Smart_count_cap&lt; Unmap_flags &gt;</code>	1622
<code>L4Re::Util::Vcon_svr&lt; SVR &gt;</code>	1623
<code>L4Re::Util::Video::Goos_svr</code>	1624
<code>L4Re::Vfs::Directory</code>	1634
<code>L4Re::Vfs::File</code>	1639
<code>L4Re::Vfs::Be_file</code>	1627
<code>L4Re::Vfs::File_system</code>	1642
<code>L4Re::Vfs::Be_file_system</code>	1631
<code>L4Re::Vfs::Fs</code>	1644
<code>L4Re::Vfs::Ops</code>	1653
<code>L4Re::Vfs::Generic_file</code>	1648
<code>L4Re::Vfs::File</code>	1639
<code>L4Re::Vfs::Mman</code>	1651
<code>L4Re::Vfs::Ops</code>	1653

L4Re::Vfs::Regular_file . . . . .	1656
L4Re::Vfs::File . . . . .	1639
L4Re::Vfs::Special_file . . . . .	1662
L4Re::Vfs::File . . . . .	1639
L4Re::Video::Color_component . . . . .	1664
L4Re::Video::Goos::Info . . . . .	1674
L4Re::Video::Pixel_info . . . . .	1676
L4Re::Video::View . . . . .	1684
L4Re::Video::View::Info . . . . .	1688
l4re_aux_t . . . . .	1691
l4re_ds_stats_t . . . . .	1692
l4re_elf_aux_mword_t . . . . .	1692
l4re_elf_aux_t . . . . .	1693
l4re_elf_aux_vma_t . . . . .	1693
l4re_env_cap_entry_t . . . . .	1694
l4re_env_t . . . . .	1696
l4re_event_t . . . . .	1698
l4re_video_color_component_t . . . . .	1699
l4re_video_goos_info_t . . . . .	1699
l4re_video_pixel_info_t . . . . .	1701
l4re_video_view_info_t . . . . .	1702
l4re_video_view_t . . . . .	1703
l4shmc_ringbuf_head_t . . . . .	1704
l4shmc_ringbuf_t . . . . .	1705
l4util_idt_desc_t . . . . .	1706
l4util_idt_header_t . . . . .	1707
l4util_l4mod_info . . . . .	1708
l4util_l4mod_mod . . . . .	1709
l4util_mb_addr_range_t . . . . .	1710
l4util_mb_apm_t . . . . .	1711
l4util_mb_drive_t . . . . .	1712
l4util_mb_info_t . . . . .	1714
l4util_mb_mod_t . . . . .	1715
l4util_mb_vbe_ctrl_t . . . . .	1716
l4util_mb_vbe_mode_t . . . . .	1717
L4vbus::Gpio_module::Pin_slice . . . . .	1737
L4vbus::Pm< DEC > . . . . .	1762
l4vbus_device_t . . . . .	1776
l4vbus_resource_t . . . . .	1777
L4vcpu::State . . . . .	1778
L4virtio::Driver::Block_device::Handle . . . . .	1808
L4virtio::Driver::Device . . . . .	1809
L4virtio::Driver::Block_device . . . . .	1799
L4virtio::Driver::Virtio_net_device . . . . .	1822
L4virtio::Driver::Virtio_net_device::Packet . . . . .	1831
L4virtio::Ptr< T > . . . . .	1842
L4virtio::Svr::Bad_descriptor . . . . .	1846
L4virtio::Svr::Block_request< Ds_data > . . . . .	1856
L4virtio::Svr::Console::Control_message . . . . .	1859
L4virtio::Svr::Console::Control_request . . . . .	1860
L4virtio::Svr::Console::Port . . . . .	1881
L4virtio::Svr::Console::Device_port . . . . .	1874
L4virtio::Svr::Data_buffer . . . . .	1899
L4virtio::Svr::Dev_config . . . . .	1903
L4virtio::Svr::Dev_features . . . . .	1916
L4virtio::Svr::Console::Features . . . . .	1878
L4virtio::Svr::Dev_status . . . . .	1918

L4virtio::Svr::Device_t< DATA > . . . . .	1919
L4virtio::Svr::Block_dev_base< Mem_region_info > . . . . .	1848
L4virtio::Svr::Console::Virtio_con . . . . .	1884
L4virtio::Svr::Console::Device . . . . .	1862
L4virtio::Svr::Scmi::Scmi_dev . . . . .	1960
L4virtio::Svr::Driver_mem_list_t< DATA > . . . . .	1927
L4virtio::Svr::Driver_mem_region_t< DATA > . . . . .	1936
L4virtio::Svr::Request_processor . . . . .	1943
L4virtio::Svr::Scmi::Base_attr_t . . . . .	1950
L4virtio::Svr::Scmi::Performance_attr_t . . . . .	1956
L4virtio::Svr::Scmi::Performance_describe_level_t . . . . .	1956
L4virtio::Svr::Scmi::Performance_describe_levels_n_t . . . . .	1957
L4virtio::Svr::Scmi::Performance_domain_attr_t . . . . .	1958
L4virtio::Svr::Scmi::Proto< OBSERV > . . . . .	1959
L4virtio::Svr::Scmi::Scmi_hdr_t . . . . .	1964
L4virtio::Svr::Virtqueue::Head_desc . . . . .	1975
L4virtio::Virtqueue . . . . .	1977
L4virtio::Driver::Virtqueue . . . . .	1831
L4virtio::Svr::Virtqueue . . . . .	1964
L4virtio::Virtqueue::Avail . . . . .	1995
L4virtio::Virtqueue::Avail::Flags . . . . .	1996
L4virtio::Virtqueue::Desc . . . . .	1997
L4virtio::Virtqueue::Desc::Flags . . . . .	1999
L4virtio::Virtqueue::Used . . . . .	2001
L4virtio::Virtqueue::Used::Flags . . . . .	2002
L4virtio::Virtqueue::Used_elem . . . . .	2003
l4virtio_block_config_t . . . . .	2004
l4virtio_block_discard_t . . . . .	2005
l4virtio_block_header_t . . . . .	2006
l4virtio_config_hdr_t . . . . .	2007
l4virtio_config_queue_t . . . . .	2008
l4virtio_input_absinfo_t . . . . .	2009
l4virtio_input_config_t . . . . .	2010
l4virtio_input_devids_t . . . . .	2010
l4virtio_input_event_t . . . . .	2011
l4virtio_net_config_t . . . . .	2011
l4virtio_net_header_t . . . . .	2012
cxx::New_allocator< _Node > . . . . .	875
cxx::New_allocator< Node > . . . . .	875
cxx::New_allocator< Slab_i > . . . . .	875
Block_device::Impl::Partitioned_device_discard_mixin< Partitioned_device< Device >, Device > . . . . .	731
L4vbus::Pm< Device > . . . . .	1762
L4vbus::Device . . . . .	1721
L4vbus::Gpio_module . . . . .	1730
L4vbus::Gpio_pin . . . . .	1737
L4vbus::Icu . . . . .	1746
L4vbus::Pci_dev . . . . .	1750
L4vbus::Pci_host_bridge . . . . .	1756
L4virtio::Svr::Scmi::Proto< Scmi_dev > . . . . .	1959
L4virtio::Svr::Scmi::Base_proto . . . . .	1951
L4virtio::Svr::Scmi::Perf_proto . . . . .	1953
L4virtio::Ptr< void > . . . . .	1842
cxx::Ref_ptr< Device_type > . . . . .	883
cxx::Ref_ptr< L4Re::Vfs::File > . . . . .	883
cxx::Ref_ptr< Mount_tree > . . . . .	883
L4drivers::Register_block_base< 16 > . . . . .	1428
L4drivers::Register_block_base< 32 > . . . . .	1428

L4drivers::Register_block_base< 64 > . . . . .	1428
L4drivers::Register_block_base< 8 > . . . . .	1428
L4drivers::Register_block_base< MAX_BITS > . . . . .	1428
L4drivers::Register_block_impl< Mmio_register_block< 32 >, 32 > . . . . .	1429
L4drivers::Register_block_tmpl< Register_block_base< MAX_BITS > > . . . . .	1431
L4drivers::Register_block_tmpl< Register_block_base< MAX_BITS > const > . . . . .	1431
cxx::Bits::Smart_ptr_list< Connection > . . . . .	843
L4::lpc::Msg::Svr_val_ops< Array_ref< A, LEN >, Dir_in, CLASS > . . . . .	1084
L4::lpc::Msg::Svr_val_ops< Array_ref< A, LEN >, Dir_out, CLASS > . . . . .	1084
L4::lpc::Msg::Svr_val_ops< L4::lpc::Snd_fpage, Dir_in, CLASS > . . . . .	1084
L4::lpc::Msg::Svr_val_ops< typename ELEM::svr_type, typename Direction< A * >::type, typename Class< typename Detail::_Plain< A * >::type >::type > . . . . .	1084
L4::lpc::Msg::Svr_val_ops< typename ELEM::svr_type, typename Direction< A >::type, typename Class< typename Detail::_Plain< A >::type >::type > . . . . .	1084
L4::lpc::Msg::Svr_val_ops< typename ELEM::svr_type, typename Direction< A const * >::type, typename Class< typename Detail::_Plain< A const * >::type >::type > . . . . .	1084
L4::lpc::Msg::Svr_val_ops< typename ELEM::svr_type, typename Direction< Array_ref< A, LEN > & >::type, typename Class< typename Detail::_Plain< Array_ref< A, LEN > & >::type >::type > . . . . .	1084
L4::lpc::Msg::Svr_val_ops< typename ELEM::svr_type, typename Direction< Array_ref< A, LEN > >::type, typename Class< typename Detail::_Plain< Array_ref< A, LEN > >::type >::type > . . . . .	1084
L4::lpc::Msg::Svr_val_ops< typename ELEM::svr_type, typename Direction< T >::type, typename Class< typename Detail::_Plain< T >::type >::type > . . . . .	1084
L4Re::Rm::Unique_region< char * > . . . . .	1548
L4Re::Rm::Unique_region< l4_addr_t > . . . . .	1548
L4Re::Rm::Unique_region< l4_uint8_t * > . . . . .	1548
L4Re::Rm::Unique_region< L4virtio::Device::Config_hdr * > . . . . .	1548
L4Re::Rm::Unique_region< l4virtio_config_hdr_t * > . . . . .	1548
L4Re::Rm::Unique_region< unsigned char * > . . . . .	1548
L4Re::Rm::Unique_region< void * > . . . . .	1548
cxx::Bitmap_base::Word< Bits > . . . . .	805
cxx::Bitmap_base::Word< COUNT > . . . . .	805
cxx::Bitmap_base::Word< Size > . . . . .	805

# Chapter 11

## Data Structure Index

### 11.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">Block_device::Device_discard_feature</a>	721
Partial interface for devices that offer discard functionality . . . . .	
<a href="#">Block_device::Device_mgr&lt; DEV, FACTORY, SCHEDULER &gt;</a>	722
Basic class that scans devices and handles client connections . . . . .	
<a href="#">Block_device::Device_with_notification_domain&lt; DEV &gt;</a>	723
Device with a per-device notification domain . . . . .	
<a href="#">Block_device::Dma_region_info</a>	724
Base class used by the driver implementation to derive its own DMA mapping tracking structure	
<a href="#">Block_device::Errand::Errand</a>	724
Wrapper for a small task executed asynchronously in the server loop . . . . .	
<a href="#">Block_device::Errand::Poll_errand</a>	728
Wrapper for a regularly repeated task . . . . .	
<a href="#">Block_device::Impl::Partitioned_device_discard_mixin&lt; PART_DEV, BASE_DEV, bool &gt;</a>	731
Dummy class used when the device class is not derived from <a href="#">Device_discard_feature</a> . . . . .	
<a href="#">Block_device::Impl::Partitioned_device_discard_mixin&lt; PART_DEV, BASE_DEV, true &gt;</a>	732
Mixin implementing discard for partition devices . . . . .	
<a href="#">Block_device::Inout_block</a>	733
Description of an inout block to be sent to the device . . . . .	
<a href="#">Block_device::Inout_memory&lt; DEV &gt;</a>	734
Helper class that temporarily allocates memory that can be used for in/out operations with the device . . . . .	
<a href="#">Block_device::Mem_region_info</a>	735
Additional info stored in each <a href="#">L4virtio::Svr::Driver_mem_region_t</a> used for tracking dataspace-wide DMA mappings . . . . .	
<a href="#">Block_device::Notification_domain</a>	735
Opaque type for representing a notification domain . . . . .	
<a href="#">Block_device::Partition_info</a>	736
Information about a single partition . . . . .	
<a href="#">Block_device::Partition_reader&lt; DEV &gt;</a>	737
Partition table reader for block devices . . . . .	
<a href="#">Block_device::Partitioned_device&lt; BASE_DEV &gt;</a>	738
A partition device for the given device interface . . . . .	
<a href="#">Block_device::Pending_request</a>	739
Interface for pending requests . . . . .	
<a href="#">Block_device::Rr_scheduler&lt; DEV &gt;</a>	740
Round Robin scheduler class . . . . .	

<a href="#">Block_device::Scheduler_base&lt; DEV &gt;</a>	
Scheduler base class	742
<a href="#">cxx::Avl_map&lt; KEY_TYPE, DATA_TYPE, COMPARE, ALLOC &gt;</a>	
AVL tree based associative container	744
<a href="#">cxx::Avl_set&lt; ITEM_TYPE, COMPARE, ALLOC &gt;</a>	
AVL set for simple compareable items	750
<a href="#">cxx::Avl_tree&lt; Node, Get_key, Compare &gt;</a>	
A generic AVL tree	754
<a href="#">cxx::Avl_tree_node</a>	
Node of an AVL tree	761
<a href="#">cxx::Base_slab&lt; Obj_size, Slab_size, Max_free, Alloc &gt;</a>	
Basic slab allocator	763
<a href="#">cxx::Base_slab&lt; Obj_size, Slab_size, Max_free, Alloc &gt;::Slab_i</a>	
Type of a slab	769
<a href="#">cxx::Base_slab_static&lt; Obj_size, Slab_size, Max_free, Alloc &gt;</a>	
Merged slab allocator (allocators for objects of the same size are merged together)	769
<a href="#">cxx::Bitfield&lt; T, LSB, MSB &gt;</a>	
Definition for a member (part) of a bit field	774
<a href="#">cxx::Bitfield&lt; T, LSB, MSB &gt;::Value&lt; TT &gt;</a>	
Internal helper type	786
<a href="#">cxx::Bitfield&lt; T, LSB, MSB &gt;::Value_base&lt; TT &gt;</a>	
Internal helper type	788
<a href="#">cxx::Bitfield&lt; T, LSB, MSB &gt;::Value_unshifted&lt; TT &gt;</a>	
Internal helper type	789
<a href="#">cxx::Bitmap&lt; BITS &gt;</a>	
A static bitmap	790
<a href="#">cxx::Bitmap_base</a>	
Basic bitmap abstraction	795
<a href="#">cxx::Bitmap_base::Bit</a>	
A writeable bit in a bitmap	803
<a href="#">cxx::Bitmap_base::Char&lt; BITS &gt;</a>	
Helper abstraction for a byte contained in the bitmap	804
<a href="#">cxx::Bitmap_base::Word&lt; BITS &gt;</a>	
Helper abstraction for a word contained in the bitmap	805
<a href="#">cxx::Bits::Avl_map_get_key&lt; KEY_TYPE &gt;</a>	
Key-getter for <a href="#">Avl_map</a>	806
<a href="#">cxx::Bits::Avl_set_get_key&lt; KEY_TYPE &gt;</a>	
Internal, key-getter for <a href="#">Avl_set</a> nodes	806
<a href="#">cxx::Bits::Base_avl_set&lt; ITEM_TYPE, COMPARE, ALLOC, GET_KEY &gt;</a>	
Internal: AVL set with internally managed nodes	807
<a href="#">cxx::Bits::Base_avl_set&lt; ITEM_TYPE, COMPARE, ALLOC, GET_KEY &gt;::Node</a>	
A smart pointer to a tree item	820
<a href="#">cxx::Bits::Basic_list&lt; POLICY &gt;</a>	
Internal: Common functions for all head-based list implementations	823
<a href="#">cxx::Bits::Bst&lt; Node, Get_key, Compare &gt;</a>	
Basic binary search tree (BST)	825
<a href="#">cxx::Bits::Bst_node</a>	
Basic type of a node in a binary search tree (BST)	839
<a href="#">cxx::Bits::Direction</a>	
The direction to go in a binary search tree	841
<a href="#">cxx::Bits::Smart_ptr_list&lt; ITEM &gt;</a>	
List of smart-pointer-managed objects	843
<a href="#">cxx::Bits::Smart_ptr_list_item&lt; T, STORE_T &gt;</a>	
List item for an arbitrary item in a <a href="#">Smart_ptr_list</a>	846
<a href="#">cxx::H_list&lt; T, POLICY &gt;</a>	
General double-linked list of unspecified <a href="#">cxx::H_list_item</a> elements	848
<a href="#">cxx::H_list_item_t&lt; ELEM_TYPE &gt;</a>	
Basic element type for a double-linked <a href="#">H_list</a>	856



<a href="#">cxx::H_list_t&lt; T &gt;</a>	Double-linked list of typed <a href="#">H_list_item_t</a> elements . . . . .	859
<a href="#">cxx::List&lt; D, Alloc &gt;</a>	Doubly linked list, with internal allocation . . . . .	862
<a href="#">cxx::List&lt; D, Alloc &gt;::Iter</a>	Iterator . . . . .	864
<a href="#">cxx::List_alloc</a>	Standard list-based allocator . . . . .	865
<a href="#">cxx::List_item</a>	Basic list item . . . . .	868
<a href="#">cxx::List_item::Iter</a>	Iterator for a list of <a href="#">ListItem</a> -s . . . . .	872
<a href="#">cxx::List_item::T_iter&lt; T, Poly &gt;</a>	Iterator for derived classes from <a href="#">ListItem</a> . . . . .	873
<a href="#">cxx::Lt_functor&lt; Obj &gt;</a>	Generic comparator class that defaults to the less-than operator . . . . .	875
<a href="#">cxx::New_allocator&lt; _Type &gt;</a>	Standard allocator based on <code>operator new ()</code> . . . . .	875
<a href="#">cxx::Nothrow</a>	Helper type to distinguish the <code>operator new</code> version that does not throw exceptions . . .	876
<a href="#">cxx::Pair&lt; First, Second &gt;</a>	Pair of two values . . . . .	877
<a href="#">cxx::Pair_first_compare&lt; Cmp, Typ &gt;</a>	Comparison functor for <a href="#">Pair</a> . . . . .	880
<a href="#">cxx::Ref_obj_list_item&lt; T &gt;</a>	Item for list linked via <a href="#">cxx::Ref_ptr</a> with default reference counting . . . . .	882
<a href="#">cxx::Ref_ptr&lt; T, CNT &gt;</a>	A reference-counting pointer with automatic cleanup . . . . .	883
<a href="#">cxx::S_list&lt; T, POLICY &gt;</a>	Simple single-linked list . . . . .	887
<a href="#">cxx::Slab&lt; Type, Slab_size, Max_free, Alloc &gt;</a>	Slab allocator for object of type <code>Type</code> . . . . .	890
<a href="#">cxx::Slab_static&lt; Type, Slab_size, Max_free, Alloc &gt;</a>	Merged slab allocator (allocators for objects of the same size are merged together) . . . . .	894
<a href="#">cxx::static_vector&lt; T, IDX &gt;</a>	Simple encapsulation for a dynamically allocated array . . . . .	898
<a href="#">cxx::String</a>	Allocation free string class with explicit length field . . . . .	899
<a href="#">cxx::Weak_ref&lt; T &gt;</a>	Typed weak reference to an object of type <code>T</code> . . . . .	905
<a href="#">cxx::Weak_ref_base</a>	Generic (base) weak reference to some object . . . . .	908
<a href="#">cxx::Weak_ref_base::List</a>	The list type for keeping all weak references to an object . . . . .	911
<a href="#">Elf32_Auxv</a>	Auxiliary vector (32-bit) . . . . .	915
<a href="#">Elf32_Dyn</a>	ELF32 dynamic entry . . . . .	916
<a href="#">Elf32_Ehdr</a>	ELF32 header . . . . .	917
<a href="#">Elf32_Phdr</a>	ELF32 program header . . . . .	920
<a href="#">Elf32_Rel</a>	ELF32 relocation entry w/o addend . . . . .	921
<a href="#">Elf32_Rela</a>	ELF32 relocation entry w/ addend . . . . .	922
<a href="#">Elf32_Shdr</a>	ELF32 section header . . . . .	923

<a href="#">Elf32_Sym</a>	ELF32 symbol table entry . . . . .	924
<a href="#">Elf64_Auxv</a>	Auxiliary vector (64-bit) . . . . .	925
<a href="#">Elf64_Dyn</a>	ELF64 dynamic entry . . . . .	926
<a href="#">Elf64_Ehdr</a>	ELF64 header . . . . .	927
<a href="#">Elf64_Phdr</a>	ELF64 program header . . . . .	930
<a href="#">Elf64_Rel</a>	ELF64 relocation entry w/o addend . . . . .	931
<a href="#">Elf64_Rela</a>	ELF64 relocation entry w/ addend . . . . .	932
<a href="#">Elf64_Shdr</a>	ELF64 section header . . . . .	933
<a href="#">Elf64_Sym</a>	ELF64 symbol table entry . . . . .	934
<a href="#">gfxbitmap_offset</a>	Offsets in pmap[] and bmap[] . . . . .	935
<a href="#">L4::Alloc_list</a>	A simple list-based allocator . . . . .	936
<a href="#">L4::Arm_smccc</a>	Wrapper for function calls that follow the ARM SMC/HVC calling convention . . . . .	937
<a href="#">L4::Base_exception</a>	Base class for all exceptions, thrown by the <a href="#">L4Re</a> framework . . . . .	938
<a href="#">L4::Basic_registry</a>	This registry returns the corresponding server object based on the label of an <a href="#">lpc_gate</a> . . . . .	941
<a href="#">L4::Bounds_error</a>	Access out of bounds . . . . .	944
<a href="#">L4::Cap&lt; T &gt;</a>	C++ interface for capabilities . . . . .	946
<a href="#">L4::Cap_base</a>	Base class for all kinds of capabilities . . . . .	952
<a href="#">L4::Com_error</a>	Error conditions during IPC . . . . .	963
<a href="#">L4::Debugger</a>	C++ kernel debugger API . . . . .	967
<a href="#">L4::Element_already_exists</a>	<a href="#">Exception</a> for duplicate element insertions . . . . .	976
<a href="#">L4::Element_not_found</a>	<a href="#">Exception</a> for a failed lookup (element not found) . . . . .	979
<a href="#">L4::Epiface</a>	Base class for interface implementations . . . . .	982
<a href="#">L4::Epiface_t&lt; Derived, IFACE, BASE, bool &gt;</a>	<a href="#">Epiface</a> implementation for Kobject-based interface implementations . . . . .	987
<a href="#">L4::Epiface_t0&lt; RPC_IFACE, BASE &gt;</a>	<a href="#">Epiface</a> mixin for generic Kobject-based interfaces . . . . .	990
<a href="#">L4::Exception</a>	<a href="#">Exception</a> interface . . . . .	993
<a href="#">L4::Exception_tracer</a>	Back-trace support for exceptions . . . . .	994
<a href="#">L4::Factory</a>	C++ Factory interface, see <a href="#">Factory</a> for the C interface . . . . .	996
<a href="#">L4::Factory::Lstr</a>	Special type to add a pascal string into the factory create stream . . . . .	1005
<a href="#">L4::Factory::Nil</a>	Special type to add a void argument into the factory create stream . . . . .	1006

<a href="#">L4::Factory::S</a>	Stream class for the <a href="#">create()</a> argument stream . . . . .	1007
<a href="#">L4::lcu</a>	C++ <a href="#">lcu</a> interface, see <a href="#">Interrupt controller</a> for the C interface . . . . .	1011
<a href="#">L4::lcu::Info</a>	This class encapsulates information about an ICU . . . . .	1019
<a href="#">L4::Invalid_capability</a>	Indicates that an invalid object was invoked . . . . .	1021
<a href="#">L4::lo_pager</a>	<a href="#">lo_pager</a> interface . . . . .	1024
<a href="#">L4::lomu</a>	Interface for IO-MMUs used for DMA remapping . . . . .	1026
<a href="#">L4::IOModifier</a>	Modifier class for the IO stream . . . . .	1028
<a href="#">L4::lpc::Array&lt; ELEM_TYPE, LEN_TYPE &gt;</a>	Array data type for dynamically sized arrays in RPCs . . . . .	1029
<a href="#">L4::lpc::Array_in_buf&lt; ELEM_TYPE, LEN_TYPE, MAX &gt;</a>	Server-side copy in buffer for <a href="#">Array</a> . . . . .	1032
<a href="#">L4::lpc::Array_ref&lt; ELEM_TYPE, LEN_TYPE &gt;</a>	Array reference data type for arrays located in the message . . . . .	1033
<a href="#">L4::lpc::As_value&lt; T &gt;</a>	Pass the argument as plain data value . . . . .	1034
<a href="#">L4::lpc::Call</a>	RPC attribute for a standard RPC call . . . . .	1035
<a href="#">L4::lpc::Call_t&lt; RIGHTS &gt;</a>	RPC attribute for an RPC call with required rights . . . . .	1037
<a href="#">L4::lpc::Call_zero_send_timeout</a>	RPC attribute for an RPC call, with zero send timeout . . . . .	1038
<a href="#">L4::lpc::Cap&lt; T &gt;</a>	Capability type for RPC interfaces (see <a href="#">L4::Cap&lt;T&gt;</a> ) . . . . .	1039
<a href="#">L4::lpc::Gen_fpage</a>	Generic RPC base for <a href="#">L4</a> flex-pages . . . . .	1042
<a href="#">L4::lpc::In_out&lt; T &gt;</a>	Mark an argument as in-out argument . . . . .	1045
<a href="#">L4::lpc::lostream</a>	Input/Output stream for IPC [un]marshalling . . . . .	1046
<a href="#">L4::lpc::lstream</a>	Input stream for IPC unmarshalling . . . . .	1056
<a href="#">L4::lpc::Msg::CInt_val_ops&lt; MTYPE, DIR, CLASS &gt;</a>	Defines client-side handling of 'MTYPE' as RPC argument . . . . .	1067
<a href="#">L4::lpc::Msg::Cls_buffer</a>	Marker type for receive buffer values . . . . .	1068
<a href="#">L4::lpc::Msg::Cls_data</a>	Marker type for data values . . . . .	1069
<a href="#">L4::lpc::Msg::Cls_item</a>	Marker type for item values . . . . .	1070
<a href="#">L4::lpc::Msg::Dir_in</a>	Marker type for input values . . . . .	1071
<a href="#">L4::lpc::Msg::Dir_out</a>	Marker type for output values . . . . .	1072
<a href="#">L4::lpc::Msg::Do_in_data</a>	Marker for Input data . . . . .	1073
<a href="#">L4::lpc::Msg::Do_in_items</a>	Marker for Input items . . . . .	1074
<a href="#">L4::lpc::Msg::Do_out_data</a>	Marker for Output data . . . . .	1075
<a href="#">L4::lpc::Msg::Do_out_items</a>	Marker for Output items . . . . .	1076

<a href="#">L4::lpc::Msg::Do_rcv_buffers</a>	
Marker for receive buffers	1077
<a href="#">L4::lpc::Msg::Elem&lt; Array&lt; A, LEN &gt; &amp; &gt;</a>	
Array as output argument	1079
<a href="#">L4::lpc::Msg::Elem&lt; Array&lt; A, LEN &gt; &gt;</a>	
Array as input arguments	1080
<a href="#">L4::lpc::Msg::Elem&lt; Array_ref&lt; A, LEN &gt; &amp; &gt;</a>	
Array_ref as output argument	1081
<a href="#">L4::lpc::Msg::Is_valid_rpc_type&lt; T &gt;</a>	
Type trait defining a valid RPC parameter type	1082
<a href="#">L4::lpc::Msg::Svr_arg_pack&lt; IPC_TYPE &gt;</a>	
Server-side RPC arguments data structure used to provide arguments to the server-side implementation of an RPC function	1084
<a href="#">L4::lpc::Msg::Svr_val_ops&lt; MTYPE, DIR, CLASS &gt;</a>	
Defines server-side handling for MTYPE server arguments	1084
<a href="#">L4::lpc::Msg_ptr&lt; T &gt;</a>	
Pointer to an element of type T in an <a href="#">lpc::lstream</a>	1085
<a href="#">L4::lpc::Opt&lt; T &gt;</a>	
Attribute for defining an optional RPC argument	1086
<a href="#">L4::lpc::Ostream</a>	
Output stream for IPC marshalling	1088
<a href="#">L4::lpc::Out&lt; T &gt;</a>	
Mark an argument as a output value in an RPC signature	1094
<a href="#">L4::lpc::Rcv_fpage</a>	
Rcv flex-page	1095
<a href="#">L4::lpc::Ret_array&lt; T &gt;</a>	
Dynamically sized output array of type T	1097
<a href="#">L4::lpc::Send_only</a>	
RPC attribute for a send-only RPC	1098
<a href="#">L4::lpc::Small_buf</a>	
A receive item for receiving a single object capability	1098
<a href="#">L4::lpc::Snd_fpage</a>	
Send flex-page	1100
<a href="#">L4::lpc::Str_cp_in&lt; T &gt;</a>	
Abstraction for extracting a zero-terminated string from an <a href="#">lpc::lstream</a>	1102
<a href="#">L4::lpc::Varg</a>	
Variably sized RPC argument	1103
<a href="#">L4::lpc::Varg_list&lt; MAX &gt;</a>	
Self-contained list of variable-sized RPC parameters	1109
<a href="#">L4::lpc::Varg_list_ref</a>	
List of variable-sized RPC parameters as received by the server	1111
<a href="#">L4::lpc::Varg_list_ref::Iterator</a>	
Iterator for Valists	1113
<a href="#">L4::lpc_gate</a>	
The C++ IPC gate interface, see <a href="#">IPC-Gate API</a> for the C interface	1114
<a href="#">L4::lpc_svr::Br_manager_no_buffers</a>	
Empty implementation of <a href="#">Server_iface</a>	1119
<a href="#">L4::lpc_svr::Compound_reply</a>	
Mix in for LOOP_HOOKS to always use compound reply and wait	1123
<a href="#">L4::lpc_svr::Default_loop_hooks</a>	
Default LOOP_HOOKS	1125
<a href="#">L4::lpc_svr::Default_setup_wait</a>	
Mix in for LOOP_HOOKS for setup_wait no op	1128
<a href="#">L4::lpc_svr::Default_timeout</a>	
Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout	1129
<a href="#">L4::lpc_svr::Direct_dispatch&lt; R &gt;</a>	
Direct dispatch helper, for forwarding dispatch calls to a registry <i>R</i>	1131

<a href="#">L4::lpc_svr::Direct_dispatch&lt; R * &gt;</a>	Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry <i>R</i> . . . . .	1133
<a href="#">L4::lpc_svr::Exc_dispatch&lt; R, Exc &gt;</a>	Dispatch helper wrapping try {} catch {} around the dispatch call . . . . .	1134
<a href="#">L4::lpc_svr::Ignore_errors</a>	Mix in for LOOP_HOOKS to ignore IPC errors . . . . .	1136
<a href="#">L4::lpc_svr::Server_iface</a>	Interface for server-loop related functions . . . . .	1137
<a href="#">L4::lpc_svr::Timeout</a>	Callback interface for <a href="#">Timeout_queue</a> . . . . .	1143
<a href="#">L4::lpc_svr::Timeout_queue</a>	<a href="#">Timeout</a> queue to be used in <a href="#">L4re</a> server loop . . . . .	1146
<a href="#">L4::lpc_svr::Timeout_queue_hooks&lt; HOOKS, BR_MAN &gt;</a>	Loop hooks mixin for integrating a timeout queue into the server loop . . . . .	1151
<a href="#">L4::lrq</a>	C++ <a href="#">lrq</a> interface, see <a href="#">IRQs</a> for the C interface . . . . .	1157
<a href="#">L4::lrq_eoi</a>	Interface for sending an unmask message to an object . . . . .	1165
<a href="#">L4::lrq_handler_object</a>	<a href="#">Server</a> object base class for handling <a href="#">IRQ</a> messages . . . . .	1167
<a href="#">L4::lrq_mux</a>	<a href="#">IRQ</a> multiplexer for shared <a href="#">IRQs</a> . . . . .	1170
<a href="#">L4::lrqep_t&lt; Derived, BASE, bool &gt;</a>	<a href="#">Epiface</a> implementation for interrupt handlers . . . . .	1174
<a href="#">L4::Kip::Mem_desc</a>	Memory descriptors stored in the kernel interface page . . . . .	1179
<a href="#">L4::Kobject</a>	Base class for all kinds of kernel objects and remote objects, referenced by capabilities . . . . .	1189
<a href="#">L4::Kobject_2t&lt; Derived, Base1, Base2, PROTO, S_DEMAND &gt;</a>	Helper class to create an <a href="#">L4Re</a> interface class that is derived from two base classes (see <a href="#">L4::Kobject_t</a> ) . . . . .	1192
<a href="#">L4::Kobject_3t&lt; Derived, Base1, Base2, Base3, PROTO, S_DEMAND &gt;</a>	Helper class to create an <a href="#">L4Re</a> interface class that is derived from three base classes (see <a href="#">L4::Kobject_t</a> ) . . . . .	1195
<a href="#">L4::Kobject_demand&lt; T &gt;</a>	Get the combined server-side resource requirements for all type <i>T</i> . . . . .	1199
<a href="#">L4::Kobject_t&lt; Derived, Base, PROTO, S_DEMAND &gt;</a>	Helper class to create an <a href="#">L4Re</a> interface class that is derived from a single base class . . . . .	1200
<a href="#">L4::Kobject_typeid&lt; T &gt;</a>	<a href="#">Meta</a> object for handling access to type information of <a href="#">Kobjects</a> . . . . .	1201
<a href="#">L4::Kobject_typeid&lt; void &gt;</a>	Minimalistic ID for <code>void</code> interface . . . . .	1204
<a href="#">L4::Kobject_x&lt; Derived, ARGS &gt;</a>	Generic <a href="#">Kobject</a> inheritance template . . . . .	1205
<a href="#">L4::Meta</a>	<a href="#">Meta</a> interface that shall be implemented by each <a href="#">L4Re</a> object and gives access to the dynamic type information for <a href="#">L4Re</a> objects . . . . .	1206
<a href="#">L4::Out_of_memory</a>	<a href="#">Exception</a> signalling insufficient memory . . . . .	1211
<a href="#">L4::Pager</a>	<a href="#">Pager</a> interface including the <a href="#">lo_pager</a> interface . . . . .	1214
<a href="#">L4::Platform_control</a>	<a href="#">L4</a> C++ interface for controlling platform-wide properties, see <a href="#">Platform Control C API</a> for the C interface . . . . .	1218
<a href="#">L4::Poll_timeout_counter</a>	Evaluate an expression for a maximum number of times . . . . .	1222
<a href="#">L4::Poll_timeout_kipclock</a>	A polling timeout based on the <a href="#">L4Re</a> clock . . . . .	1225

<a href="#">L4::Proto_t&lt; P &gt;</a>	Data type for defining protocol numbers . . . . .	1228
<a href="#">L4::Rcv_endpoint</a>	Interface for kernel objects that allow to receive IPC from them . . . . .	1229
<a href="#">L4::Registry_iface</a>	Abstract interface for object registries . . . . .	1233
<a href="#">L4::Runtime_error</a>	<a href="#">Exception</a> for an abstract runtime error . . . . .	1237
<a href="#">L4::Scheduler</a>	C++ interface of the <a href="#">Scheduler</a> kernel object, see <a href="#">Scheduler</a> for the C interface . . . . .	1241
<a href="#">L4::Semaphore</a>	C++ Kernel-provided semaphore interface, see <a href="#">Kernel-provided semaphore</a> for the C interface . . . . .	1248
<a href="#">L4::Server&lt; LOOP_HOOKS &gt;</a>	Basic server loop for handling client requests . . . . .	1254
<a href="#">L4::Server_object</a>	Abstract server object to be used with <a href="#">L4::Server</a> and <a href="#">L4::Basic_registry</a> . . . . .	1259
<a href="#">L4::Server_object_t&lt; IFACE, BASE &gt;</a>	Base class (template) for server implementing server objects . . . . .	1263
<a href="#">L4::Server_object_x&lt; Derived, IFACE, BASE &gt;</a>	Helper class to implement p_dispatch based server objects . . . . .	1269
<a href="#">L4::Smart_cap&lt; T, SMART &gt;</a>	Smart capability class . . . . .	1272
<a href="#">L4::String</a>	A null-terminated string container class . . . . .	1276
<a href="#">L4::Task</a>	C++ interface of the <a href="#">Task</a> kernel object, see <a href="#">Task</a> for the C interface . . . . .	1277
<a href="#">L4::Thread</a>	C++ <a href="#">L4</a> kernel thread interface, see <a href="#">Thread</a> for the C interface . . . . .	1289
<a href="#">L4::Thread::Attr</a>	<a href="#">Thread</a> attributes used for <a href="#">control()</a> . . . . .	1302
<a href="#">L4::Thread::Modify_senders</a>	Class wrapping a list of rules which modify the sender label of IPC messages inbound to this thread . . . . .	1307
<a href="#">L4::Triggerable</a>	Interface that allows an object to be triggered by some source . . . . .	1309
<a href="#">L4::Type_info</a>	Dynamic Type Information for <a href="#">L4Re</a> Interfaces . . . . .	1313
<a href="#">L4::Type_info::Demand</a>	Data type for expressing the needed receive buffers at the server-side of an interface . . . . .	1314
<a href="#">L4::Type_info::Demand_t&lt; CAPS, FLAGS, MEM, PORTS &gt;</a>	Template type statically describing demand of receive buffers . . . . .	1318
<a href="#">L4::Type_info::Demand_union_t&lt; D1, D2 &gt;</a>	Template type statically describing the combination of two <a href="#">Demand</a> object . . . . .	1320
<a href="#">L4::Typeid::Detail::_Rpc&lt; OPCODE, O, X &gt;</a>	Empty list of RPCs . . . . .	1323
<a href="#">L4::Typeid::Detail::_Rpc&lt; OPCODE, O, Default_op&lt; R &gt; &gt;::Rpc&lt; Y &gt;</a>	Find the given RPC in the list . . . . .	1325
<a href="#">L4::Typeid::Detail::_Rpc&lt; OPCODE, O, R, X... &gt;</a>	Non-empty list of RPCs . . . . .	1325
<a href="#">L4::Typeid::Detail::_Rpc&lt; OPCODE, O, R, X... &gt;::Rpc&lt; Y &gt;</a>	Find the given RPC in the list . . . . .	1327
<a href="#">L4::Typeid::Detail::Rpc_end</a>	Internal end-of-list marker . . . . .	1327
<a href="#">L4::Typeid::P_dispatch&lt; LIST &gt;</a>	Use for protocol based dispatch stage . . . . .	1328
<a href="#">L4::Typeid::Raw_ipc&lt; CLASS &gt;</a>	RPCs list for passing raw incoming IPC to the server object . . . . .	1329

<a href="#">L4::Typeid::Rpc_nocode&lt; OPERATION &gt;</a>	1330
List of RPCs of an interface using a single operation without an opcode	
<a href="#">L4::Typeid::Rpcs&lt; RPCS &gt;</a>	1332
Standard list of RPCs of an interface	
<a href="#">L4::Typeid::Rpcs_code&lt; OPCODE_TYPE &gt;</a>	1334
List of RPCs of an interface using a special opcode type	
<a href="#">L4::Typeid::Rpcs_code&lt; OPCODE_TYPE &gt;::F&lt; RPCS &gt;</a>	1335
<a href="#">L4::Typeid::Rpcs_sys&lt; ARG &gt;</a>	1337
List of RPCs typically used for kernel interfaces	
<a href="#">L4::Types::Bool&lt; V &gt;</a>	1339
Boolean meta type	
<a href="#">L4::Types::False</a>	1340
False meta value	
<a href="#">L4::Types::Flags&lt; BITS_ENUM, UNDERLYING &gt;</a>	1342
Template for defining typical <a href="#">Flags</a> bitmaps	
<a href="#">L4::Types::Flags_ops_t&lt; DT &gt;</a>	1346
Mixin class to define a set of friend bitwise operators on DT	
<a href="#">L4::Types::Flags_t&lt; DT, T &gt;</a>	1348
Template type to define a flags type with bitwise operations	
<a href="#">L4::Types::Int_for_size&lt; SIZE, bool &gt;</a>	1350
Metafunction to get an unsigned integral type for the given size	
<a href="#">L4::Types::Int_for_type&lt; T &gt;</a>	1351
Metafunction to get an integral type of the same size as T	
<a href="#">L4::Types::Same&lt; A, B &gt;</a>	1352
Compare two data types for equality	
<a href="#">L4::Types::True</a>	1354
True meta value	
<a href="#">L4::Uart</a>	1356
Uart driver abstraction	
<a href="#">L4::Uart_apb</a>	1361
Driver for the Advanced Peripheral Bus (APB) UART from the Cortex-M System Design Kit (apb)	
<a href="#">L4::Unknown_error</a>	1367
Exception for an unknown condition	
<a href="#">L4::Vcon</a>	1369
C++ <a href="#">L4 Vcon</a> interface, see <a href="#">Virtual Console</a> for the C interface	
<a href="#">L4::Vm</a>	1378
Virtual machine host address space	
<a href="#">l4_buf_regs_t</a>	1383
Encapsulation of the buffer-registers block in the UTCB	
<a href="#">l4_exc_regs_t</a>	1384
UTCB structure for exceptions	
<a href="#">l4_ext_vcpu_state_vmx_t</a>	1387
VMX extended vCPU state	
<a href="#">l4_fpage_t</a>	1388
L4 flexpage type	
<a href="#">l4_icu_info_t</a>	1389
Info structure for an ICU	
<a href="#">l4_icu_msi_info_t</a>	1390
Info to use for a specific MSI	
<a href="#">l4_kernel_info_mem_desc_t</a>	1391
Memory descriptor data structure	
<a href="#">l4_kernel_info_t</a>	1392
L4 Kernel Interface Page	
<a href="#">l4_msg_regs_t</a>	1394
Encapsulation of the message-register block in the UTCB	
<a href="#">l4_msgtag_t</a>	1395
Message tag data structure	

<a href="#">l4_sched_cpu_set_t</a>	
CPU sets	1397
<a href="#">l4_sched_param_t</a>	
Scheduler parameter set	1401
<a href="#">l4_snd_fpage_t</a>	
Send-flex-page types	1402
<a href="#">l4_thread_regs_t</a>	
Encapsulation of the thread-control-register block of the UTCB	1403
<a href="#">l4_timeout_s</a>	
Basic timeout specification	1404
<a href="#">l4_timeout_t</a>	
Timeout pair	1405
<a href="#">l4_vcon_attr_t</a>	
Vcon attribute structure	1406
<a href="#">l4_vcpu_arch_state_t</a>	
Architecture-specific vCPU state	1407
<a href="#">l4_vcpu_ipc_regs_t</a>	
VCPU message registers	1408
<a href="#">l4_vcpu_regs_t</a>	
VCPU registers	1409
<a href="#">l4_vcpu_state_t</a>	
State of a vCPU	1413
<a href="#">l4_vhw_descriptor</a>	
Virtual hardware devices description	1416
<a href="#">l4_vhw_entry</a>	
Description of a device	1417
<a href="#">l4_vm_svm_vmcb_control_area</a>	
VMCB structure for SVM VMs	1418
<a href="#">l4_vm_svm_vmcb_state_save_area</a>	
State save area structure for SVM VMs	1419
<a href="#">l4_vm_svm_vmcb_state_save_area_seg</a>	
State save area segment selector struct	1420
<a href="#">l4_vm_svm_vmcb_t</a>	
Control structure for SVM VMs	1420
<a href="#">l4_vm_tz_state</a>	
State structure for TrustZone VMs	1421
<a href="#">l4_vmx_offset_table_t</a>	
Software VMCS field offset table	1422
<a href="#">L4drivers::Mmio_register_block&lt; MAX_BITS &gt;</a>	
An MMIO block with up to 64-bit register access (32-bit default) and little endian byte order	1424
<a href="#">L4drivers::Register_block&lt; MAX_BITS, BLOCK &gt;</a>	
Handles a reference to a register block of the given maximum access width	1425
<a href="#">L4drivers::Register_block_base&lt; MAX_BITS &gt;</a>	
Abstract register block interface	1428
<a href="#">L4drivers::Register_block_impl&lt; BASE, MAX_BITS &gt;</a>	
Implementation helper for register blocks	1429
<a href="#">L4drivers::Register_block_tmpl&lt; BLOCK &gt;</a>	
Helper template that translates to the <a href="#">Register_block_base</a> interface	1431
<a href="#">L4drivers::Register_tmpl&lt; BITS, BLOCK &gt;</a>	
Single hardware register inside a <a href="#">Register_block_base</a> interface	1432
<a href="#">L4drivers::Ro_register_block&lt; MAX_BITS, BLOCK &gt;</a>	
Handles a reference to a read only register block of the given maximum access width	1436
<a href="#">L4drivers::Ro_register_tmpl&lt; BITS, BLOCK &gt;</a>	
Single read only register inside a <a href="#">Register_block_base</a> interface	1438
<a href="#">L4Re::Cap_alloc</a>	
Capability allocator interface	1440
<a href="#">L4Re::Console</a>	
Console class	1443



<a href="#">L4Re::Dataspace</a>	
Interface for memory-like objects	1446
<a href="#">L4Re::Dataspace::F</a>	
<a href="#">Dataspace</a> flags definitions	1457
<a href="#">L4Re::Dataspace::Stats</a>	
Information about the dataspace	1458
<a href="#">L4Re::Debug_obj</a>	
Debug interface	1459
<a href="#">L4Re::Default_event_payload</a>	
Default event stream payload	1462
<a href="#">L4Re::Dma_space</a>	
Managed DMA Address Space	1463
<a href="#">L4Re::Env</a>	
C++ interface of the initial environment that is provided to an <a href="#">L4</a> task	1469
<a href="#">L4Re::Event</a>	
Event class	1481
<a href="#">L4Re::Event_buffer_t&lt; PAYLOAD &gt;</a>	
Event buffer class	1488
<a href="#">L4Re::Event_buffer_t&lt; PAYLOAD &gt;::Event</a>	
Event structure used in buffer	1492
<a href="#">L4Re::Inhibitor</a>	
Set of inhibitor locks, which inhibit specific actions when held	1493
<a href="#">L4Re::Log</a>	
Log interface class	1498
<a href="#">L4Re::Mem_alloc</a>	
Memory allocation interface	1503
<a href="#">L4Re::Mmio_space</a>	
Interface for memory-like address space accessible via IPC	1509
<a href="#">L4Re::Namespace</a>	
Name-space interface	1513
<a href="#">L4Re::Ned::Cmd_control</a>	
Direct control interface for Ned	1520
<a href="#">L4Re::Parent</a>	
Parent interface	1522
<a href="#">L4Re::Random</a>	
Low-bandwidth interface for random number generators	1526
<a href="#">L4Re::Rm</a>	
Region map	1531
<a href="#">L4Re::Rm::F</a>	
Rm flags definitions	1545
<a href="#">L4Re::Rm::Range</a>	
A range of virtual addresses	1547
<a href="#">L4Re::Rm::Unique_region&lt; T &gt;</a>	
Unique region	1548
<a href="#">L4Re::Smart_cap_auto&lt; Unmap_flags &gt;</a>	
Helper for Unique_cap and Unique_del_cap	1554
<a href="#">L4Re::Smart_count_cap&lt; Unmap_flags &gt;</a>	
Helper for Ref_cap and Ref_del_cap	1555
<a href="#">L4Re::Util::Br_manager</a>	
Buffer-register (BR) manager for <a href="#">L4::Server</a>	1556
<a href="#">L4Re::Util::Br_manager_hooks</a>	
Predefined server-loop hooks for a server loop using the <a href="#">Br_manager</a>	1562
<a href="#">L4Re::Util::Br_manager_timeout_hooks</a>	
Predefined server-loop hooks for a server with using the <a href="#">Br_manager</a> and a timeout queue	1564
<a href="#">L4Re::Util::Cap_alloc_base</a>	
Capability allocator	1568
<a href="#">L4Re::Util::Counter&lt; COUNTER &gt;</a>	
Counter for <a href="#">Counting_cap_alloc</a> with variable data width	1569

<a href="#">L4Re::Util::Counter_atomic&lt; COUNTER &gt;</a>	
Thread safe version of counter for <a href="#">Counting_cap_alloc</a>	1571
<a href="#">L4Re::Util::Counting_cap_alloc&lt; COUNTERTYPE, Dbg &gt;</a>	
Internal reference-counting cap allocator	1572
<a href="#">L4Re::Util::Dataspace_svr</a>	
Dataspace server class	1578
<a href="#">L4Re::Util::Event_buffer_consumer_t&lt; PAYLOAD &gt;</a>	
An event buffer consumer	1584
<a href="#">L4Re::Util::Event_buffer_t&lt; PAYLOAD &gt;</a>	
Event_buffer utility class	1589
<a href="#">L4Re::Util::Event_svr&lt; SVR &gt;</a>	
Convenience wrapper for implementing an event server	1593
<a href="#">L4Re::Util::Event_t&lt; PAYLOAD &gt;</a>	
Convenience wrapper for getting access to an event object	1595
<a href="#">L4Re::Util::Item_alloc_base</a>	
Item allocator	1599
<a href="#">L4Re::Util::Names::Name</a>	
Name class	1599
<a href="#">L4Re::Util::Names::Name_space</a>	
Abstract server-side implementation of the <a href="#">L4::Namespace</a> interface	1603
<a href="#">L4Re::Util::Object_registry</a>	
A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread	1607
<a href="#">L4Re::Util::Ref_cap&lt; T &gt;</a>	
Automatic capability that implements automatic free and unmap of the capability selector	1614
<a href="#">L4Re::Util::Ref_del_cap&lt; T &gt;</a>	
Automatic capability that implements automatic free and unmap+delete of the capability selector	1615
<a href="#">L4Re::Util::Registry_server&lt; LOOP_HOOKS &gt;</a>	
A server loop object which has a <a href="#">Object_registry</a> included	1616
<a href="#">L4Re::Util::Smart_cap_auto&lt; Unmap_flags &gt;</a>	
Helper for <a href="#">Unique_cap</a> and <a href="#">Unique_del_cap</a>	1621
<a href="#">L4Re::Util::Smart_count_cap&lt; Unmap_flags &gt;</a>	
Helper for <a href="#">Ref_cap</a> and <a href="#">Ref_del_cap</a>	1622
<a href="#">L4Re::Util::Vcon_svr&lt; SVR &gt;</a>	
Console server template class	1623
<a href="#">L4Re::Util::Video::Goos_svr</a>	
Goos server class	1624
<a href="#">L4Re::Vfs::Be_file</a>	
Boiler plate class for implementing an open file for <a href="#">L4Re::Vfs</a>	1627
<a href="#">L4Re::Vfs::Be_file_system</a>	
Boilerplate class for implementing a <a href="#">L4Re::Vfs::File_system</a>	1631
<a href="#">L4Re::Vfs::Directory</a>	
Interface for a POSIX file that is a directory	1634
<a href="#">L4Re::Vfs::File</a>	
The basic interface for an open POSIX file	1639
<a href="#">L4Re::Vfs::File_system</a>	
Basic interface for an <a href="#">L4Re::Vfs</a> file system	1642
<a href="#">L4Re::Vfs::Fs</a>	
POSIX File-system related functionality	1644
<a href="#">L4Re::Vfs::Generic_file</a>	
The common interface for an open POSIX file	1648
<a href="#">L4Re::Vfs::Mman</a>	
Interface for POSIX memory management	1651
<a href="#">L4Re::Vfs::Ops</a>	
Interface for the POSIX backends of an application	1653
<a href="#">L4Re::Vfs::Regular_file</a>	
Interface for a POSIX file that provides regular file semantics	1656

<a href="#">L4Re::Vfs::Special_file</a>	
Interface for a POSIX file that provides special file semantics . . . . .	1662
<a href="#">L4Re::Video::Color_component</a>	
A color component . . . . .	1664
<a href="#">L4Re::Video::Goos</a>	
Class that abstracts framebuffers . . . . .	1668
<a href="#">L4Re::Video::Goos::Info</a>	
Information structure of a <a href="#">Goos</a> . . . . .	1674
<a href="#">L4Re::Video::Pixel_info</a>	
Pixel information . . . . .	1676
<a href="#">L4Re::Video::View</a>	
<a href="#">View</a> of a framebuffer . . . . .	1684
<a href="#">L4Re::Video::View::Info</a>	
Information structure of a view . . . . .	1688
<a href="#">l4re_aux_t</a>	
Auxiliary descriptor . . . . .	1691
<a href="#">l4re_ds_stats_t</a>	
Information about the data space . . . . .	1692
<a href="#">l4re_elf_aux_mword_t</a>	
Auxiliary vector element for a single unsigned data word . . . . .	1692
<a href="#">l4re_elf_aux_t</a>	
Generic header for each auxiliary vector element . . . . .	1693
<a href="#">l4re_elf_aux_vma_t</a>	
Auxiliary vector element for a reserved virtual memory area . . . . .	1693
<a href="#">l4re_env_cap_entry_t</a>	
Entry in the <a href="#">L4Re</a> environment array for the named initial objects . . . . .	1694
<a href="#">l4re_env_t</a>	
Initial environment data structure . . . . .	1696
<a href="#">l4re_event_t</a>	
Event structure used in buffer . . . . .	1698
<a href="#">l4re_video_color_component_t</a>	
Color component structure . . . . .	1699
<a href="#">l4re_video_goos_info_t</a>	
Goos information structure . . . . .	1699
<a href="#">l4re_video_pixel_info_t</a>	
Pixel_info structure . . . . .	1701
<a href="#">l4re_video_view_info_t</a>	
View information structure . . . . .	1702
<a href="#">l4re_video_view_t</a>	
C representation of a goos view . . . . .	1703
<a href="#">l4shmc_ringbuf_head_t</a>	
Head field of a ring buffer . . . . .	1704
<a href="#">l4shmc_ringbuf_t</a>	
Ring buffer . . . . .	1705
<a href="#">l4util_idt_desc_t</a>	
IDT entry . . . . .	1706
<a href="#">l4util_idt_header_t</a>	
Header of an IDT table . . . . .	1707
<a href="#">l4util_l4mod_info</a>	
Base module structure . . . . .	1708
<a href="#">l4util_l4mod_mod</a>	
A single module . . . . .	1709
<a href="#">l4util_mb_addr_range_t</a>	
INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached . . . . .	1710
<a href="#">l4util_mb_apm_t</a>	
APM BIOS info . . . . .	1711

<a href="#">L4util_mb_drive_t</a>	
Drive Info structure	1712
<a href="#">L4util_mb_info_t</a>	
MultiBoot Info description	1714
<a href="#">L4util_mb_mod_t</a>	
The structure type "mod_list" is used by the <a href="#">multiboot_info</a> structure	1715
<a href="#">L4util_mb_vbe_ctrl_t</a>	
VBE controller information	1716
<a href="#">L4util_mb_vbe_mode_t</a>	
VBE mode information	1717
<a href="#">L4vbus::Device</a>	
Device on a <a href="#">L4vbus::Vbus</a>	1721
<a href="#">L4vbus::Gpio_module</a>	
A <a href="#">Gpio_module</a> groups multiple GPIO pins together	1730
<a href="#">L4vbus::Gpio_module::Pin_slice</a>	
A slice of the pins provided by this module	1737
<a href="#">L4vbus::Gpio_pin</a>	
A GPIO pin	1737
<a href="#">L4vbus::Icu</a>	
Vbus Interrupt controller API	1746
<a href="#">L4vbus::Pci_dev</a>	
A PCI device	1750
<a href="#">L4vbus::Pci_host_bridge</a>	
A Pci host bridge	1756
<a href="#">L4vbus::Pm&lt; DEC &gt;</a>	
Power-management API mixin	1762
<a href="#">L4vbus::Vbus</a>	
The virtual bus ( <a href="#">Vbus</a> ) interface	1765
<a href="#">L4vbus_device_t</a>	
Detailed information about a vbus device	1776
<a href="#">L4vbus_resource_t</a>	
Description of a single vbus resource	1777
<a href="#">L4vcpu::State</a>	
C++ implementation of state word in the vCPU area	1778
<a href="#">L4vcpu::Vcpu</a>	
C++ implementation of the vCPU save state area	1780
<a href="#">L4virtio::Device</a>	
IPC interface for virtio over <a href="#">L4 IPC</a>	1791
<a href="#">L4virtio::Driver::Block_device</a>	
Simple class for accessing a virtio block device synchronously	1799
<a href="#">L4virtio::Driver::Block_device::Handle</a>	
Handle to an ongoing request	1808
<a href="#">L4virtio::Driver::Device</a>	
Client-side implementation for a general virtio device	1809
<a href="#">L4virtio::Driver::Virtio_net_device</a>	
Simple class for accessing a virtio net device	1822
<a href="#">L4virtio::Driver::Virtio_net_device::Packet</a>	
Structure for a network packet (header including data) with maximum size, assuming that no extra features have been negotiated	1831
<a href="#">L4virtio::Driver::Virtqueue</a>	
Driver-side implementation of a <a href="#">Virtqueue</a>	1831
<a href="#">L4virtio::Ptr&lt; T &gt;</a>	
Pointer used in virtio descriptors	1842
<a href="#">L4virtio::Svr::Bad_descriptor</a>	
Exception used by Queue to indicate descriptor errors	1846
<a href="#">L4virtio::Svr::Block_dev_base&lt; Ds_data &gt;</a>	
Base class for virtio block devices	1848

<a href="#">L4virtio::Svr::Block_request&lt; Ds_data &gt;</a>	
A request to read or write data . . . . .	1856
<a href="#">L4virtio::Svr::Console::Control_message</a>	
Virtio console control message . . . . .	1859
<a href="#">L4virtio::Svr::Console::Control_request</a>	
Specialised <code>Virtqueue::Request</code> providing access to control message payload . . . . .	1860
<a href="#">L4virtio::Svr::Console::Device</a>	
Base class implementing a virtio console device with L4Re-based notification handling . . . . .	1862
<a href="#">L4virtio::Svr::Console::Device_port</a>	
A console port with associated read/write state . . . . .	1874
<a href="#">L4virtio::Svr::Console::Features</a>	
Virtio console specific feature bits . . . . .	1878
<a href="#">L4virtio::Svr::Console::Port</a>	
Representation of a Virtio console port . . . . .	1881
<a href="#">L4virtio::Svr::Console::Virtio_con</a>	
Base class implementing a virtio console functionality . . . . .	1884
<a href="#">L4virtio::Svr::Data_buffer</a>	
Abstract data buffer . . . . .	1899
<a href="#">L4virtio::Svr::Dev_config</a>	
Abstraction for L4-Virtio device config memory . . . . .	1903
<a href="#">L4virtio::Svr::Dev_features</a>	
Type for device feature bitmap . . . . .	1916
<a href="#">L4virtio::Svr::Dev_status</a>	
Type of the device status register . . . . .	1918
<a href="#">L4virtio::Svr::Device_t&lt; DATA &gt;</a>	
Server-side L4-VIRTIO device stub . . . . .	1919
<a href="#">L4virtio::Svr::Driver_mem_list_t&lt; DATA &gt;</a>	
List of driver memory regions assigned to a single L4-VIRTIO transport instance . . . . .	1927
<a href="#">L4virtio::Svr::Driver_mem_region_t&lt; DATA &gt;</a>	
Region of driver memory, that shall be managed locally . . . . .	1936
<a href="#">L4virtio::Svr::Request_processor</a>	
Encapsulate the state for processing a VIRTIO request . . . . .	1943
<a href="#">L4virtio::Svr::Scmi::Base_attr_t</a>	
SCMI base protocol attributes . . . . .	1950
<a href="#">L4virtio::Svr::Scmi::Base_proto</a>	
Base class for the SCMI base protocol . . . . .	1951
<a href="#">L4virtio::Svr::Scmi::Perf_proto</a>	
Base class for the SCMI performance protocol . . . . .	1953
<a href="#">L4virtio::Svr::Scmi::Performance_attr_t</a>	
SCMI performance protocol attributes . . . . .	1956
<a href="#">L4virtio::Svr::Scmi::Performance_describe_level_t</a>	
SCMI performance describe level . . . . .	1956
<a href="#">L4virtio::Svr::Scmi::Performance_describe_levels_n_t</a>	
SCMI performance describe levels numbers . . . . .	1957
<a href="#">L4virtio::Svr::Scmi::Performance_domain_attr_t</a>	
SCMI performance domain protocol attributes . . . . .	1958
<a href="#">L4virtio::Svr::Scmi::Proto&lt; OBSERV &gt;</a>	
Base class for all protocols . . . . .	1959
<a href="#">L4virtio::Svr::Scmi::Scmi_dev</a>	
A server implementation of the virtio-scmi protocol . . . . .	1960
<a href="#">L4virtio::Svr::Scmi::Scmi_hdr_t</a>	
SCMI header . . . . .	1964
<a href="#">L4virtio::Svr::Virtqueue</a>	
<code>Virtqueue</code> implementation for the device . . . . .	1964
<a href="#">L4virtio::Svr::Virtqueue::Head_desc</a>	
VIRTIO request, essentially a descriptor from the available ring . . . . .	1975
<a href="#">L4virtio::Virtqueue</a>	
Low-level <code>Virtqueue</code> . . . . .	1977

<a href="#">L4virtio::Virtqueue::Avail</a>	Type of available ring, this is read-only for the host . . . . .	1995
<a href="#">L4virtio::Virtqueue::Avail::Flags</a>	Flags of the available ring . . . . .	1996
<a href="#">L4virtio::Virtqueue::Desc</a>	Descriptor in the descriptor table . . . . .	1997
<a href="#">L4virtio::Virtqueue::Desc::Flags</a>	Type for descriptor flags . . . . .	1999
<a href="#">L4virtio::Virtqueue::Used</a>	Used ring . . . . .	2001
<a href="#">L4virtio::Virtqueue::Used::Flags</a>	Flags for the used ring . . . . .	2002
<a href="#">L4virtio::Virtqueue::Used_elem</a>	Type of an element of the used ring . . . . .	2003
<a href="#">l4virtio_block_config_t</a>	Device configuration for block devices . . . . .	2004
<a href="#">l4virtio_block_discard_t</a>	Structure used for the write zeroes and discard commands . . . . .	2005
<a href="#">l4virtio_block_header_t</a>	Header structure of a request for a block device . . . . .	2006
<a href="#">l4virtio_config_hdr_t</a>	L4-VIRTIO config header, provided in shared data space . . . . .	2007
<a href="#">l4virtio_config_queue_t</a>	Queue configuration entry . . . . .	2008
<a href="#">l4virtio_input_absinfo_t</a>	Information about the absolute axis in the underlying evdev implementation . . . . .	2009
<a href="#">l4virtio_input_config_t</a>	Device configuration for input devices . . . . .	2010
<a href="#">l4virtio_input_devids_t</a>	Device ID information for the device . . . . .	2010
<a href="#">l4virtio_input_event_t</a>	Single event in event or status queue . . . . .	2011
<a href="#">l4virtio_net_config_t</a>	Device configuration for network devices . . . . .	2011
<a href="#">l4virtio_net_header_t</a>	Header structure of a request for a network device . . . . .	2012

# Chapter 12

## File Index

### 12.1 File List

Here is a list of all documented files with brief descriptions:

pkg/drivers-frst/include/asm_access_gen.h	2018
pkg/drivers-frst/include/hw_mmio_register_block	2018
pkg/drivers-frst/include/hw_register_block	2019
pkg/drivers-frst/include/io_regblock.h	2025
pkg/drivers-frst/include/io_regblock_port.h	2027
pkg/drivers-frst/include/Makefile	2028
pkg/drivers-frst/include/poll_timeout_counter.h	2028
pkg/drivers-frst/include/ARCH-amd64/asm_access.h	2013
pkg/drivers-frst/include/ARCH-arm/asm_access.h	2013
pkg/drivers-frst/include/ARCH-arm64/asm_access.h	2014
pkg/drivers-frst/include/ARCH-mips/asm_access.h	2015
pkg/drivers-frst/include/ARCH-ppc32/asm_access.h	2015
pkg/drivers-frst/include/ARCH-riscv/asm_access.h	2016
pkg/drivers-frst/include/ARCH-sparc/asm_access.h	2017
pkg/drivers-frst/include/ARCH-x86/asm_access.h	2017
pkg/drivers-frst/uart/include/Makefile	2028
pkg/drivers-frst/uart/include/uart_16550.h	2029
pkg/drivers-frst/uart/include/uart_16550_dw.h	2030
pkg/drivers-frst/uart/include/uart_apb.h	2030
pkg/drivers-frst/uart/include/uart_base.h	2031
pkg/drivers-frst/uart/include/uart_cadence.h	2032
pkg/drivers-frst/uart/include/uart_dcc-v6.h	2032
pkg/drivers-frst/uart/include/uart_dm.h	2033
pkg/drivers-frst/uart/include/uart_dummy.h	2033
pkg/drivers-frst/uart/include/uart_geni.h	2034
pkg/drivers-frst/uart/include/uart_imx.h	2034
pkg/drivers-frst/uart/include/uart_leon3.h	2035
pkg/drivers-frst/uart/include/uart_linflex.h	2036
pkg/drivers-frst/uart/include/uart_lpuart.h	2036
pkg/drivers-frst/uart/include/uart_mvebu.h	2037
pkg/drivers-frst/uart/include/uart_of.h	2037
pkg/drivers-frst/uart/include/uart_omap35x.h	2038
pkg/drivers-frst/uart/include/uart_pl011.h	2038
pkg/drivers-frst/uart/include/uart_s3c2410.h	2039
pkg/drivers-frst/uart/include/uart_sa1000.h	2040

pkg/drivers-frst/uart/include/uart_sbi.h	2041
pkg/drivers-frst/uart/include/uart_sh.h	2041
pkg/l4re-core/ned/lib/include/cmd_control	2041
pkg/l4re-core/ned/lib/include/Makefile	2028
amd64/l4/sys/__kip-arch.h	2091
amd64/l4/sys/__vcpu-arch.h	
AMD64-specific vCPU interface	2092
amd64/l4/sys/cache.h	
Cache functions	2697
amd64/l4/sys/consts.h	
Common L4 constants, amd64 version	2480
amd64/l4/sys/ktrace_events.h	2101
amd64/l4/sys/l4int.h	
Fixed sized integer types, amd64 version	2857
amd64/l4/sys/linkage.h	
Linkage	2111
amd64/l4/sys/segment.h	
Segment handling	2069
amd64/l4/sys/utcb.h	
UTCB definitions for amd64	2919
amd64/l4/sys/vm.h	2116
amd64/l4/util/bitops_arch.h	
Amd64 bit manipulation functions	2118
amd64/l4/util/cpu.h	
CPU related functions	2125
amd64/l4/util/idt.h	
IDT related functions	2042
amd64/l4/util/irq.h	
Some PIC and hardware interrupt related functions	2814
amd64/l4/util/l4_macros.h	
Main function	2130
amd64/l4/util/mbi_argv.h	
Command line handling	2133
amd64/l4/util/perform.h	
Performance Monitoring using P5/P6 Measurement Counters	2046
amd64/l4/util/port_io.h	
Port I/O functions	2083
amd64/l4/util/rdtsc.h	
Timestamp counter related functions	2058
amd64/l4/util/spin.h	
Spinning for amd64	2067
amd64/l4f/l4/sys/ipc.h	2795
amd64/l4f/l4/sys/segment.h	
L4f specific fs/gs manipulation	2075
amd64/l4f/l4/util/port_io.h	
Port I/O functions	2084
arm/l4/sys/__kip-arch.h	2092
arm/l4/sys/__vcpu-arch.h	
ARM-specific vCPU interface	2095
arm/l4/sys/atomic.h	
Atomic memory modifications	2150
arm/l4/sys/cache.h	
Cache functions	2698
arm/l4/sys/consts.h	
Common L4 constants, arm version	2481
arm/l4/sys/ktrace_events.h	2104
arm/l4/sys/l4int.h	
Fixed sized integer types, arm version	2858



arm/l4/sys/linkage.h	
Linkage	2112
arm/l4/sys/mem_op.h	
Memory access functions (ARM specific)	2113
arm/l4/sys/platform_control.h	2869
arm/l4/sys/task.h	2899
arm/l4/sys/thread.h	
ARM-specific thread related definitions	2903
arm/l4/sys/utcb.h	
UTCB definitions for ARM	2922
arm/l4/sys/vm	2949
arm/l4/sys/vm.h	
ARM virtualization interface	2116
arm/l4/util/bitops_arch.h	
ARM specific implementation of bitops functions	2121
arm/l4/util/cpu.h	
CPU related functions	2128
arm/l4/util/irq.h	
ARM specific implementation of irq functions	2817
arm/l4/util/l4_macros.h	
Main function	2131
arm/l4/util/mbi_argv.h	
Multiboot	2135
arm/l4f/l4/sys/ipc.h	
L4 IPC System Calls, ARM	2796
arm/l4f/l4/sys/syscall_defs.h	
Syscall entry definitions	2138
contrib/libio-io/l4/io/io.h	2139
contrib/libio-io/l4/io/types.h	2142
l4/cxx/alloc.h	
Alloc list	2149
l4/cxx/arith	2150
l4/cxx/atomic.h	
Atomic template	2152
l4/cxx/avl_map	
AVL map	2160
l4/cxx/avl_set	
AVL set	2163
l4/cxx/avl_tree	
AVL tree	2168
l4/cxx/basic_ostream	
Basic IO stream	2174
l4/cxx/basic_vector.h	
Basic vector	2178
l4/cxx/bitfield	2179
l4/cxx/bitmap	2182
l4/cxx/dlist	2204
l4/cxx/exceptions	
Base exceptions	2207
l4/cxx/hlist	2211
l4/cxx/iostream	
IO Stream	2214
l4/cxx/ipc_helper	
IPC helper	2215
l4/cxx/ipc_server	
IPC server loop	2217
l4/cxx/ipc_stream	
IPC stream	2223

l4/cxx/ <a href="#">ipc_timeout_queue</a>	2241
l4/cxx/ <a href="#">l4iostream</a>	
L4 IO stream	2243
l4/cxx/ <a href="#">l4types.h</a>	
L4 Types	2245
l4/cxx/ <a href="#">list</a>	2246
l4/cxx/ <a href="#">list_alloc</a>	2250
l4/cxx/ <a href="#">main_thread</a>	
Main thread	2256
l4/cxx/ <a href="#">minmax</a>	2258
l4/cxx/ <a href="#">observer</a>	2259
l4/cxx/ <a href="#">pair</a>	
Pair implementation	2259
l4/cxx/ <a href="#">ref_ptr</a>	2262
l4/cxx/ <a href="#">ref_ptr_list</a>	
Implementation of a list of ref-ptr-managed objects	2265
l4/cxx/ <a href="#">slab_alloc</a>	2267
l4/cxx/ <a href="#">slist</a>	2271
l4/cxx/ <a href="#">static_container</a>	2274
l4/cxx/ <a href="#">static_vector</a>	2274
l4/cxx/ <a href="#">std_alloc</a>	2275
l4/cxx/ <a href="#">std_exc_io</a>	
Base exceptions std stream operator	2276
l4/cxx/ <a href="#">std_ops</a>	2277
l4/cxx/ <a href="#">string</a>	2278
l4/cxx/ <a href="#">string.h</a>	
String	2281
l4/cxx/ <a href="#">thread</a>	
Thread implementation	2283
l4/cxx/ <a href="#">type_list</a>	2288
l4/cxx/ <a href="#">type_traits</a>	2288
l4/cxx/ <a href="#">unique_ptr</a>	2293
l4/cxx/ <a href="#">unique_ptr_list</a>	
Implementation of a list of unique-ptr-managed objects	2295
l4/cxx/ <a href="#">utils</a>	2297
l4/cxx/ <a href="#">weak_ref</a>	2297
l4/cxx/bits/ <a href="#">bst.h</a>	
AVL tree	2185
l4/cxx/bits/ <a href="#">bst_base.h</a>	
AVL tree	2189
l4/cxx/bits/ <a href="#">bst_iter.h</a>	
AVL tree	2192
l4/cxx/bits/ <a href="#">list_basics.h</a>	2195
l4/cxx/bits/ <a href="#">smart_ptr_list.h</a>	
Implementation of a list of smart-pointer-managed objects	2197
l4/cxx/bits/ <a href="#">type_traits.h</a>	2201
l4/irq/ <a href="#">irq.h</a>	
IRQ handling routines	2818
l4/l4re_vfs/ <a href="#">backend</a>	2299
l4/l4re_vfs/ <a href="#">vfs.h</a>	2328
l4/l4re_vfs/impl/ <a href="#">default_ops_impl.h</a>	2302
l4/l4re_vfs/impl/ <a href="#">fd_store.h</a>	2303
l4/l4re_vfs/impl/ <a href="#">fd_store_impl.h</a>	2304
l4/l4re_vfs/impl/ <a href="#">ns_fs.h</a>	2304
l4/l4re_vfs/impl/ <a href="#">ns_fs_impl.h</a>	2305
l4/l4re_vfs/impl/ <a href="#">ro_file.h</a>	2310
l4/l4re_vfs/impl/ <a href="#">ro_file_impl.h</a>	2311
l4/l4re_vfs/impl/ <a href="#">vcon_stream.h</a>	2312

l4/l4re_vfs/impl/vcon_stream_impl.h	2313
l4/l4re_vfs/impl/vfs_impl.h	2315
l4/l4virtio/l4virtio	2351
l4/l4virtio/virtio.h	2386
l4/l4virtio/virtio_block.h	2389
l4/l4virtio/virtio_input.h	2390
l4/l4virtio/virtio_net.h	2391
l4/l4virtio/virtqueue	2392
l4/l4virtio/client/l4virtio	2349
l4/l4virtio/client/virtio-block	2337
l4/l4virtio/client/virtio-net	2346
l4/l4virtio/server/l4virtio	2353
l4/l4virtio/server/virtio	2364
l4/l4virtio/server/virtio-block	2340
l4/l4virtio/server/virtio-console	2368
l4/l4virtio/server/virtio-console-device	2373
l4/l4virtio/server/virtio-scmi-device	2377
l4/libblock-device/block_device_mgr.h	2396
l4/libblock-device/debug.h	2437
l4/libblock-device/device.h	2401
l4/libblock-device/errand.h	2402
l4/libblock-device/gpt.h	2404
l4/libblock-device/inout_memory.h	2405
l4/libblock-device/part_device.h	2406
l4/libblock-device/partition.h	2409
l4/libblock-device/request.h	2412
l4/libblock-device/scheduler.h	2880
l4/libblock-device/types.h	2143
l4/libblock-device/virtio_client.h	2412
l4/libedid/edid.h	2420
l4/libgfxbitmap/bitmap.h	
Bitmap renderer header file	2422
l4/libgfxbitmap/font.h	
Bitmap font renderer header file	2427
l4/libgfxbitmap/support	
Terminal support functionality	2433
l4/re/cap_alloc	
Abstract capability-allocator interface	2470
l4/re/console	2477
l4/re/consts	
Constants	2477
l4/re/consts.h	
Constants	2482
l4/re/dataspace	
Dataspace interface	2489
l4/re/dataspace-sys.h	
Dataspace protocol definition	2492
l4/re/debug	
Debug interface	2493
l4/re/dma_space	2496
l4/re/elf_aux.h	
Auxiliary information for binaries	2499
l4/re/env	
Environment interface	2501
l4/re/env.h	
Environment interface	2504
l4/re/error_helper	
Error helper	2507

<a href="#">l4/re/event</a> . . . . .	2510
<a href="#">l4/re/event-sys.h</a> . . . . .	2515
<a href="#">l4/re/event.h</a>	
Events . . . . .	2518
<a href="#">l4/re/event_enums.h</a> . . . . .	2520
<a href="#">l4/re/inhibitor</a> . . . . .	2536
<a href="#">l4/re/inhibitor-sys.h</a> . . . . .	2536
<a href="#">l4/re/l4aux.h</a>	
Auxiliary definitions . . . . .	2536
<a href="#">l4/re/log</a>	
Log interface . . . . .	2538
<a href="#">l4/re/log-sys.h</a>	
Log protocol definition . . . . .	2540
<a href="#">l4/re/mem_alloc</a>	
Memory allocator interface . . . . .	2541
<a href="#">l4/re/mem_alloc-sys.h</a>	
Memory allocator protocol definitions . . . . .	2543
<a href="#">l4/re/mmio_space</a>	
Interface definition to emit MMIO-like accesses via IPC . . . . .	2545
<a href="#">l4/re/namespace</a>	
Namespace interface . . . . .	2546
<a href="#">l4/re/namespace-sys.h</a>	
Namespace protocol definitions . . . . .	2549
<a href="#">l4/re/parent</a>	
Parent interface . . . . .	2550
<a href="#">l4/re/parent-sys.h</a>	
Parent protocol definition . . . . .	2552
<a href="#">l4/re/protocols.h</a>	
L4Re Protocol Constants (C version) . . . . .	2553
<a href="#">l4/re/random</a>	
Random number generator interface definition . . . . .	2555
<a href="#">l4/re/rm</a>	
Region mapper interface . . . . .	2557
<a href="#">l4/re/rm-sys.h</a>	
Region mapper protocol definitions . . . . .	2562
<a href="#">l4/re/shared_cap</a>	
Shared_cap / Shared_del_cap . . . . .	2563
<a href="#">l4/re/unique_cap</a>	
Unique_cap / Unique_del_cap . . . . .	2569
<a href="#">l4/re/c/dataspace.h</a>	
Data space C interface . . . . .	2434
<a href="#">l4/re/c/debug.h</a>	
Debug C interface . . . . .	2437
<a href="#">l4/re/c/dma_space.h</a>	
DMA space C interface . . . . .	2439
<a href="#">l4/re/c/event.h</a>	
Event C interface . . . . .	2516
<a href="#">l4/re/c/event_buffer.h</a> . . . . .	2442
<a href="#">l4/re/c/inhibitor.h</a>	
Inhibitor C interface . . . . .	2442
<a href="#">l4/re/c/log.h</a>	
Log C interface . . . . .	2445
<a href="#">l4/re/c/mem_alloc.h</a>	
Memory allocator C interface . . . . .	2447
<a href="#">l4/re/c/namespace.h</a>	
Namespace functions, C interface . . . . .	2449
<a href="#">l4/re/c/parent.h</a>	
Parent C interface . . . . .	2451

l4/re/c/ <a href="#">rm.h</a>	
Region map interface, C interface	2453
l4/re/c/util/ <a href="#">cap_alloc.h</a>	
Capability allocator C interface	2457
l4/re/c/util/ <a href="#">kumem_alloc.h</a>	
Kumem allocator utility C interface	2458
l4/re/c/util/video/ <a href="#">goos_fb.h</a>	
Framebuffer utility functionality	2460
l4/re/c/video/ <a href="#">colors.h</a>	2462
l4/re/c/video/ <a href="#">goos.h</a>	2464
l4/re/c/video/ <a href="#">view.h</a>	2467
l4/re/impl/ <a href="#">dataspace_impl.h</a>	
Dataspace client stub implementation	2527
l4/re/impl/ <a href="#">mem_alloc_impl.h</a>	
Memory allocator client stub implementation	2529
l4/re/impl/ <a href="#">namespace_impl.h</a>	
Namespace client stub implementation	2531
l4/re/impl/ <a href="#">rm_impl.h</a>	
Region map client stub implementation	2533
l4/re/util/ <a href="#">bitmap_cap_alloc</a>	
Bitmap capability allocator	2573
l4/re/util/ <a href="#">br_manager</a>	2576
l4/re/util/ <a href="#">cap</a>	
Capability utility functions	2578
l4/re/util/ <a href="#">cap_alloc</a>	
Capability allocator	2474
l4/re/util/ <a href="#">cap_alloc_impl.h</a>	
Capability allocator implementation	2580
l4/re/util/ <a href="#">counting_cap_alloc</a>	
Reference-counting capability allocator	2582
l4/re/util/ <a href="#">dataspace_svr</a>	2587
l4/re/util/ <a href="#">debug</a>	2494
l4/re/util/ <a href="#">env_ns</a>	2590
l4/re/util/ <a href="#">event</a>	2513
l4/re/util/ <a href="#">event_buffer</a>	2591
l4/re/util/ <a href="#">event_svr</a>	2592
l4/re/util/ <a href="#">icu_svr</a>	2594
l4/re/util/ <a href="#">item_alloc</a>	
Item allocator	2596
l4/re/util/ <a href="#">kumem_alloc</a>	
Kumem allocator helper	2599
l4/re/util/ <a href="#">meta</a>	2600
l4/re/util/ <a href="#">name_space_svr</a>	2602
l4/re/util/ <a href="#">object_registry</a>	2608
l4/re/util/ <a href="#">poll_timeout_kipclock</a>	2611
l4/re/util/ <a href="#">region_mapping</a>	
Region handling	2611
l4/re/util/ <a href="#">region_mapping_svr_2</a>	2618
l4/re/util/ <a href="#">shared_cap</a>	
Shared_cap / Shared_del_cap	2566
l4/re/util/ <a href="#">unique_cap</a>	
Unique_cap / Unique_del_cap	2571
l4/re/util/ <a href="#">vcon_svr</a>	2621
l4/re/util/video/ <a href="#">goos_fb</a>	2622
l4/re/util/video/ <a href="#">goos_svr</a>	2624
l4/re/video/ <a href="#">colors</a>	2626
l4/re/video/ <a href="#">goos</a>	2627

<a href="#">l4/re/video/goos-sys.h</a>	
Goos protocol definition	2630
<a href="#">l4/re/video/view</a>	2632
<a href="#">l4/shmc/ringbuf.h</a>	2632
<a href="#">l4/shmc/shmc.h</a>	
Shared memory library header file	2640
<a href="#">l4/sigma0/sigma0.h</a>	
Sigma0 interface	2645
<a href="#">l4/sys/__kernel_object_impl.h</a>	
Low-level kernel debugger functions	2649
<a href="#">l4/sys/__kip-32bit.h</a>	2650
<a href="#">l4/sys/__kip-64bit.h</a>	2651
<a href="#">l4/sys/__ktrace-impl.h</a>	
L4 kernel event tracing	2652
<a href="#">l4/sys/__l4_fpage.h</a>	2655
<a href="#">l4/sys/__platform_control-arm.h</a>	2659
<a href="#">l4/sys/__task-arm.h</a>	2659
<a href="#">l4/sys/__timeout.h</a>	2660
<a href="#">l4/sys/__typeinfo.h</a>	
Type information handling	2662
<a href="#">l4/sys/__vcpu-arm.h</a>	2675
<a href="#">l4/sys/__vm-arm.h</a>	
Virtualization interface	2676
<a href="#">l4/sys/__vm-svm.h</a>	2678
<a href="#">l4/sys/__vm-vmx.h</a>	2680
<a href="#">l4/sys/arm_smccc</a>	
ARM secure monitor call functions	2686
<a href="#">l4/sys/arm_smccc.h</a>	
ARM secure monitor call functions	2688
<a href="#">l4/sys/assert.h</a>	
Low-level assert implementation	2691
<a href="#">l4/sys/cache.h</a>	
Cache-consistency functions	2701
<a href="#">l4/sys/capability</a>	
L4::Cap related definitions	2704
<a href="#">l4/sys/compiler.h</a>	
L4 compiler related defines	2708
<a href="#">l4/sys/consts.h</a>	
Common constants	2484
<a href="#">l4/sys/debugger</a>	
The debugger interface specifies common debugging related definitions	2767
<a href="#">l4/sys/debugger.h</a>	
Debugger related definitions	2769
<a href="#">l4/sys/err.h</a>	
Error codes	2773
<a href="#">l4/sys/exception</a>	
Exception C++ interface	2775
<a href="#">l4/sys/factory</a>	
Common factory related definitions	2777
<a href="#">l4/sys/factory.h</a>	
Common factory related definitions	2781
<a href="#">l4/sys/icu</a>	
Interrupt controller	2787
<a href="#">l4/sys/icu.h</a>	
Interrupt controller	2789
<a href="#">l4/sys/iommu</a>	2795
<a href="#">l4/sys/ipc.h</a>	
Common IPC interface	2798

<a href="#">l4/sys/ipc_gate</a>	
The C++ IPC gate interface . . . . .	2805
<a href="#">l4/sys/ipc_gate.h</a>	
The C IPC gate interface, see <a href="#">L4::ipc_gate</a> for the C++ interface . . . . .	2807
<a href="#">l4/sys/irq</a>	
C++ Irq interface . . . . .	2811
<a href="#">l4/sys/irq.h</a>	
C Irq interface . . . . .	2821
<a href="#">l4/sys/kdebug.h</a>	
Functionality for invoking the kernel debugger . . . . .	2827
<a href="#">l4/sys/kernel_object.h</a>	
Kernel object system calls . . . . .	2845
<a href="#">l4/sys/kip</a>	
Kernel Info Page access functions . . . . .	2850
<a href="#">l4/sys/kip.h</a>	
Kernel Info Page access functions . . . . .	2850
<a href="#">l4/sys/kobject</a>	
Kernel object system calls . . . . .	2845
<a href="#">l4/sys/ktrace.h</a>	
L4 kernel event tracing . . . . .	2856
<a href="#">l4/sys/l4int.h</a>	
Fixed sized integer types, generic version . . . . .	2859
<a href="#">l4/sys/memdesc.h</a>	
Memory description functions . . . . .	2862
<a href="#">l4/sys/meta</a>	
Meta interface for getting dynamic type information about objects behind capabilities . . . . .	2600
<a href="#">l4/sys/pager</a>	
Pager and lo_pager C++ interface . . . . .	2865
<a href="#">l4/sys/platform_control</a>	
Platform control object . . . . .	2867
<a href="#">l4/sys/platform_control.h</a>	
Platform control object . . . . .	2869
<a href="#">l4/sys/rcv_endpoint</a>	
The C++ Receive endpoint interface . . . . .	2873
<a href="#">l4/sys/rcv_endpoint.h</a>	
Receive endpoint C interface . . . . .	2875
<a href="#">l4/sys/scheduler</a>	
Scheduler object functions . . . . .	2878
<a href="#">l4/sys/scheduler.h</a>	
Scheduler object functions . . . . .	2883
<a href="#">l4/sys/semaphore</a>	
Semaphore class definition . . . . .	2888
<a href="#">l4/sys/semaphore.h</a>	
C semaphore interface . . . . .	2890
<a href="#">l4/sys/smart_capability</a>	
L4::Capability class . . . . .	2892
<a href="#">l4/sys/task</a>	
Common task related definitions . . . . .	2896
<a href="#">l4/sys/task.h</a>	
Common task related definitions . . . . .	2899
<a href="#">l4/sys/thread</a>	
Common thread related definitions . . . . .	2285
<a href="#">l4/sys/thread.h</a>	
Common thread related definitions . . . . .	2905
<a href="#">l4/sys/typeinfo_svr</a>	
Type information server template . . . . .	2916
<a href="#">l4/sys/types.h</a>	
Common <a href="#">L4</a> ABI Data Types . . . . .	2144
<a href="#">l4/sys/utcb.h</a>	
UTCB definitions . . . . .	2924

l4/sys/vcon	
C++ Virtual console interface	2931
l4/sys/vcon.h	
Virtual console interface	2933
l4/sys/vcpu.h	
VCPU API	2939
l4/sys/vcpu_context	
Hardware vCPU context interface	2946
l4/sys/vcpu_context.h	2947
l4/sys/vhw.h	
Descriptors for virtual hardware (under UX)	2947
l4/sys/vm	
Virtualization interface	2949
l4/sys/cxx/capability.h	2712
l4/sys/cxx/consts	2479
l4/sys/cxx/ipc_array	2714
l4/sys/cxx/ipc_basics	2718
l4/sys/cxx/ipc_client	2722
l4/sys/cxx/ipc_epiface	2724
l4/sys/cxx/ipc_iface	
Interface Definition Language	2728
l4/sys/cxx/ipc_legacy	2739
l4/sys/cxx/ipc_ret_array	2740
l4/sys/cxx/ipc_server	2220
l4/sys/cxx/ipc_server_loop	2741
l4/sys/cxx/ipc_string	2744
l4/sys/cxx/ipc_types	2745
l4/sys/cxx/ipc_varg	2754
l4/sys/cxx/smart_capability_1x	2760
l4/sys/cxx/types	2763
l4/util/assert.h	
Some useful assert-style macros	2694
l4/util/atomic.h	
Atomic operations header and generic implementations	2153
l4/util/backtrace.h	
Backtrace	2951
l4/util/base64.h	
Base 64 encoding and decoding functions adapted from Bob Trower 08/04/01	2953
l4/util/bitops.h	
Bit manipulation functions	2954
l4/util/elf.h	
ELF definition	2960
l4/util/getopt.h	
Getopt	2976
l4/util/keymap.h	
Event to ASCII key mapping	2978
l4/util/kip.h	2854
l4/util/kprintf.h	
Printf using the kernel debugger	2979
l4/util/l4_macros.h	
Some useful generic macros, L4f version	2132
l4/util/l4mod.h	
L4mod structures and constants	2980
l4/util/list_alloc.h	
Simple list-based allocator	2982
l4/util/lock.h	
Simple lock implementation	2984



l4/util/ <a href="#">mb_info.h</a>	
Multiboot info structure as defined by GRUB	2986
l4/util/ <a href="#">parse_cmd.h</a>	
Comfortable command-line parsing	2994
l4/util/ <a href="#">rand.h</a>	
Simple Pseudo-Random Number Generator	2995
l4/util/ <a href="#">splitlog2.h</a>	
Split a range in log2 aligned and size-aligned chunks	2997
l4/util/ <a href="#">thread.h</a>	
Low-level Thread Functions	2914
l4/util/ <a href="#">util.h</a>	2999
l4/vbus/ <a href="#">vbus</a>	2999
l4/vbus/ <a href="#">vbus.h</a>	
Description of the vbus C API	3001
l4/vbus/ <a href="#">vbus_generic</a>	3005
l4/vbus/ <a href="#">vbus_gpio</a>	3005
l4/vbus/ <a href="#">vbus_gpio-ops.h</a>	3007
l4/vbus/ <a href="#">vbus_gpio.h</a>	3007
l4/vbus/ <a href="#">vbus_i2c.h</a>	3008
l4/vbus/ <a href="#">vbus_inhibitor.h</a>	3008
l4/vbus/ <a href="#">vbus_interfaces.h</a>	
This header contains the definition of VBUS sub-interfaces and convenience functions to work with the interface IDs	3008
l4/vbus/ <a href="#">vbus_mcspi.h</a>	3012
l4/vbus/ <a href="#">vbus_pci</a>	3012
l4/vbus/ <a href="#">vbus_pci-ops.h</a>	3013
l4/vbus/ <a href="#">vbus_pci.h</a>	3014
l4/vbus/ <a href="#">vbus_pm-ops.h</a>	3014
l4/vbus/ <a href="#">vbus_pm.h</a>	3015
l4/vbus/ <a href="#">vbus_types.h</a>	
This header file contains descriptions of vbus related data types and constants	3015
l4/vbus/ <a href="#">vdevice-ops.h</a>	3018
l4/vcpu/ <a href="#">vcpu</a>	
VCPU support library (C++ interface)	3019
l4/vcpu/ <a href="#">vcpu.h</a>	
VCPU support library (C interface)	2942
x86/l4/sys/ <a href="#">__kip-arch.h</a>	2092
x86/l4/sys/ <a href="#">__vcpu-arch.h</a>	
X86-specific vCPU interface	2099
x86/l4/sys/ <a href="#">cache.h</a>	
Cache functions	2703
x86/l4/sys/ <a href="#">consts.h</a>	
Common L4 constants, x86 version	2488
x86/l4/sys/ <a href="#">ipc-invoke.h</a>	3021
x86/l4/sys/ <a href="#">ktrace_events.h</a>	2107
x86/l4/sys/ <a href="#">l4int.h</a>	
Fixed sized integer types, x86 version	2861
x86/l4/sys/ <a href="#">linkage.h</a>	
Linkage	2112
x86/l4/sys/ <a href="#">segment.h</a>	
Segment handling	2079
x86/l4/sys/ <a href="#">utcb.h</a>	
UTCB definitions for X86	2928
x86/l4/sys/ <a href="#">vm.h</a>	2118
x86/l4/util/ <a href="#">bitops_arch.h</a>	
X86 bit manipulation functions	2122
x86/l4/util/ <a href="#">cpu.h</a>	
CPU related functions	2128

x86/l4/util/ <a href="#">idt.h</a>	
IDT related functions . . . . .	2044
x86/l4/util/ <a href="#">irq.h</a>	
Some PIC and hardware interrupt related functions . . . . .	2825
x86/l4/util/ <a href="#">l4_macros.h</a>	
Main function . . . . .	2132
x86/l4/util/ <a href="#">mbi_argv.h</a>	
Command line handling . . . . .	2136
x86/l4/util/ <a href="#">perform.h</a>	
Performance Monitoring using P5/P6 Measurement Counters . . . . .	2052
x86/l4/util/ <a href="#">port_io.h</a>	
X86 port I/O . . . . .	2085
x86/l4/util/ <a href="#">rdtsc.h</a>	
Timestamp counter related functions . . . . .	2062
x86/l4/util/ <a href="#">spin.h</a>	
Spinning for x86 . . . . .	2068
x86/l4f/l4/sys/ <a href="#">ipc-l42-gcc3-nopic.h</a>	3022
x86/l4f/l4/sys/ <a href="#">ipc.h</a>	
L4 IPC System Calls, x86 . . . . .	2803
x86/l4f/l4/sys/ <a href="#">segment.h</a>	
L4f specific segment manipulation . . . . .	2081
x86/l4f/l4/util/ <a href="#">port_io.h</a>	
Port I/O functions . . . . .	2088

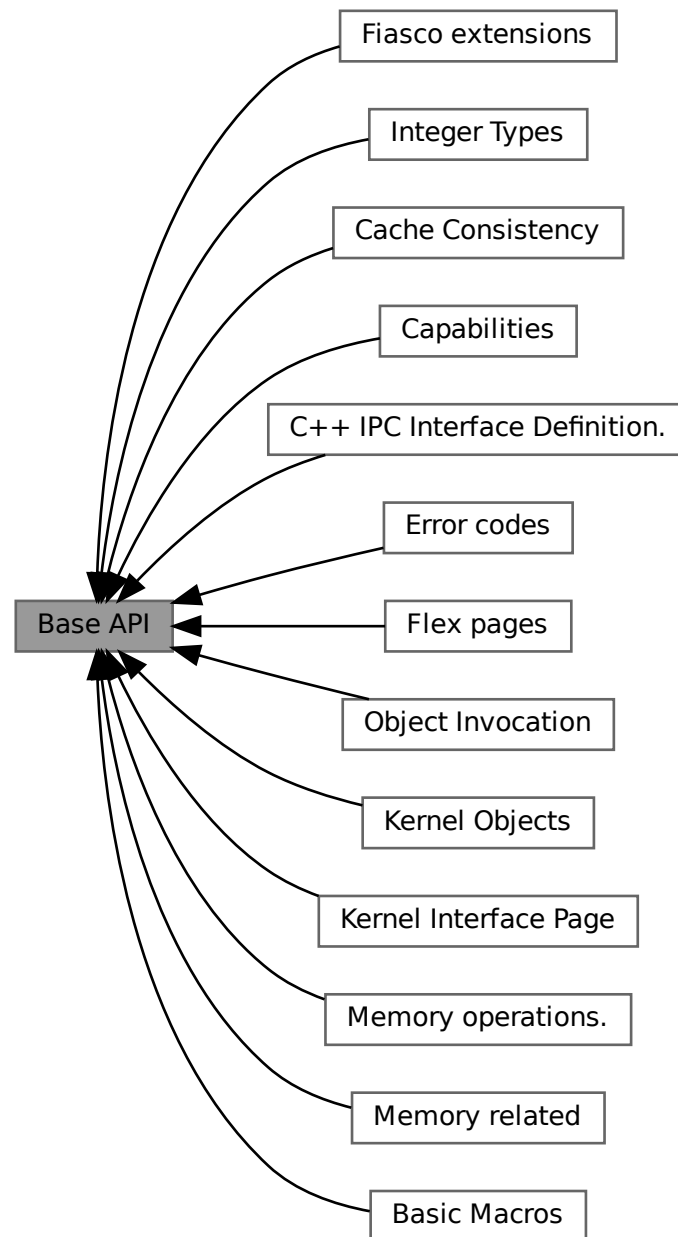
## Chapter 13

# Topic Documentation

### 13.1 Base API

Interfaces for all kinds of base functionality.

Collaboration diagram for Base API:



## Modules

- [Basic Macros](#)  
*L4 standard macros for header files, function definitions, and public APIs etc.*
- [C++ IPC Interface Definition.](#)  
*APIs for defining IPC interfaces using C++ as language.*
- [Cache Consistency](#)

- Various functions for cache consistency.*
- [Capabilities](#)
  - C interface for capabilities.*
- [Error codes](#)
  - Common error codes.*
- [Fiasco extensions](#)
  - Extensions of the Fiasco [L4](#) implementation.*
- [Flex pages](#)
  - Flex-page related API.*
- [Integer Types](#)
- [Kernel Interface Page](#)
  - Kernel Interface Page.*
- [Kernel Objects](#)
  - API of kernel objects.*
- [Memory operations.](#)
  - Operations for memory access.*
- [Memory related](#)
  - Memory related constants, data types and functions.*
- [Object Invocation](#)
  - API for [L4](#) object invocation.*

## Files

- file [cache.h](#)
  - Cache-consistency functions.*
- file [compiler.h](#)
  - [L4](#) compiler related defines.*
- file [consts.h](#)
  - Common constants.*
- file [debugger.h](#)
  - Debugger related definitions.*
- file [factory.h](#)
  - Common factory related definitions.*
- file [icu](#)
  - Interrupt controller.*
- file [icu.h](#)
  - Interrupt controller.*
- file [ipc.h](#)
  - Common IPC interface.*
- file [irq.h](#)
  - C Irq interface.*
- file [kip](#)
- file [kip.h](#)
  - Kernel Info Page access functions.*
- file [memdesc.h](#)
  - Memory description functions.*
- file [semaphore.h](#)
  - C semaphore interface.*
- file [types.h](#)
  - Common [L4](#) ABI Data Types.*

- file [vhw.h](#)  
*Descriptors for virtual hardware (under UX).*
- file [consts.h](#)  
*Common L4 constants, arm version.*
- file [consts.h](#)  
*Common L4 constants, amd64 version.*
- file [ipc.h](#)  
*L4 IPC System Calls, x86.*
- file [consts.h](#)  
*Common L4 constants, x86 version.*

### 13.1.1 Detailed Description

Interfaces for all kinds of base functionality.

Some notes on Inter Process Communication (IPC)

IPC in L4 is always synchronous and unbuffered: a message is transferred from the sender to the recipient if and only if the recipient has invoked a corresponding IPC operation. The sender blocks until this happens or a timeout specified by the sender elapsed without the destination becoming ready to receive.

### 13.1.2 Basic Macros

L4 standard macros for header files, function definitions, and public APIs etc.

Collaboration diagram for Basic Macros:



#### Macros

- `#define L4_INLINE`  
*L4 Inline function attribute.*
- `#define L4_ALWAYS_INLINE`  
*Always inline a function.*
- `#define __BEGIN_DECLS`  
*Start section with C types and functions.*
- `#define __END_DECLS`  
*End section with C types and functions.*
- `#define EXTERN_C_BEGIN`  
*Start section with C types and functions.*
- `#define EXTERN_C_END`

- End section with C types and functions.*

  - **#define EXTERN\_C**  
*Mark C types and functions.*
  - **#define L4\_NOTHROW**  
*Mark a function declaration and definition as never throwing an exception.*
  - **#define L4\_EXPORT**  
*Attribute to mark functions, variables, and data types as being exported from a library.*
  - **#define L4\_HIDDEN**  
*Attribute to mark functions, variables, and data types as being explicitly hidden from users of a library.*
  - **#define L4\_CONSTEXPR**  
*Constexpr function attribute.*
  - **#define L4\_NORETURN**  
*Noreturn function attribute.*
  - **#define L4\_NOINSTRUMENT**  
*No instrumentation function attribute.*
  - **#define L4\_LIKELY(x)**  
*Expression is likely to execute.*
  - **#define L4\_UNLIKELY(x)**  
*Expression is unlikely to execute.*
  - **#define L4\_STICKY(x)**  
*Mark symbol sticky (even not there)*
  - **#define L4\_DEPRECATED(s)**  
*Mark symbol deprecated.*
  - **#define L4\_stringify\_helper(x)**  
*stringify helper.*
  - **#define L4\_stringify(x)**  
*stringify.*
  - **#define L4\_CV**  
*Define calling convention.*
  - **#define L4\_CV**  
*Define calling convention.*
  - **#define L4\_CV**  
*Define calling convention.*

## Functions

- unsigned long **l4\_align\_stack\_for\_direct\_fncall** (unsigned long stack)  
*Specify the desired alignment of the stack pointer.*
- void **l4\_barrier** (void)  
*Memory barrier.*
- void **l4\_mb** (void)  
*Memory barrier.*
- void **l4\_wmb** (void)  
*Write memory barrier.*
- **L4\_NORETURN** void **l4\_infinite\_loop** (void)  
*Infinite loop.*

### 13.1.2.1 Detailed Description

L4 standard macros for header files, function definitions, and public APIs etc.

#### Include File

```
#include <l4/sys/compiler.h>
```

### 13.1.2.2 Macro Definition Documentation

#### 13.1.2.2.1 L4\_EXPORT

```
#define L4_EXPORT
```

Attribute to mark functions, variables, and data types as being exported from a library.

All data types, functions, and global variables that shall be exported from a library shall be marked with this attribute. The default may become to hide everything that is not marked as L4\_EXPORT from the users of a library and provide the possibility for aggressive optimization of all those internal functionality of a library.

#### Usage:

```
class L4_EXPORT My_class
{
    ...
};

int L4_EXPORT function(void);

int L4_EXPORT global_data; // global data is not recommended
```

Definition at line 231 of file [compiler.h](#).

#### 13.1.2.2.2 L4\_HIDDEN

```
#define L4_HIDDEN
```

Attribute to mark functions, variables, and data types as being explicitly hidden from users of a library.

This attribute is intended for functions, data, and data types that shall never be visible outside of a library. In particular, for shared libraries this may result in much faster code within the library and short linking times.

```
class L4_HIDDEN My_class
{
    ...
};

int L4_HIDDEN function(void);

int L4_HIDDEN global_data; // global data is not recommended
```

Definition at line 228 of file [compiler.h](#).



### 13.1.2.2.3 L4\_NOTHROW

```
#define L4_NOTHROW
```

Mark a function declaration and definition as never throwing an exception.

(Also for C code).

This macro shall be used to mark C and C++ functions that never throw any exception. Note that also C functions may throw exceptions according to the compilers ABI and shall be marked with L4\_NOTHROW if they never do. In C++ this is equivalent to `throw()`.

```
int foo() L4_NOTHROW;
...
int foo() L4_NOTHROW
{
    ...
    return result;
}
```

Definition at line 188 of file [compiler.h](#).

### 13.1.2.3 Function Documentation

#### 13.1.2.3.1 l4\_align\_stack\_for\_direct\_fncall()

```
unsigned long l4_align_stack_for_direct_fncall (
    unsigned long stack ) [inline]
```

Specify the desired alignment of the stack pointer.

**BIGGEST\_ALIGNMENT** provides the largest alignment ever used for any data type on the target machine. This is normally identical to desired stack alignment. Align stack pointer for directly invoked functions.

The stack needs to be aligned to L4\_STACK\_ALIGN for being able to access certain data on the stack. On x86/AMD64, a function call is performed using the 'call' instruction decrementing the stack pointer and writing the return address onto the stack. The called function considers this when adapting the stack pointer after function entry. If the called function was not invoked by a 'call' instruction, the stack pointer is actually off by a machine word leading to stack alignment issues when executing SSE instructions.

This function fixes the stack pointer for directly invoked functions. For architectures not automatically pushing the stack pointer during a function call, just enforce the L4\_STACK\_ALIGN alignment.

Definition at line 285 of file [compiler.h](#).

#### 13.1.2.3.2 l4\_infinite\_loop()

```
L4_NORETURN void l4_infinite_loop (
    void ) [inline]
```

Infinite loop.

Will never return. Use [l4\\_sleep\\_forever\(\)](#) if at all possible.

Definition at line 359 of file [compiler.h](#).

References [l4\\_barrier\(\)](#).

Here is the call graph for this function:



### 13.1.3 C++ IPC Interface Definition.

APIs for defining IPC interfaces using C++ as language.

Collaboration diagram for C++ IPC Interface Definition.:



#### Modules

- [Internal Helpers](#)

#### Namespaces

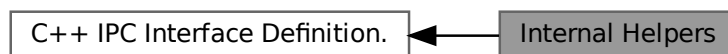
- namespace [L4::Typeid](#)  
*Definition of interface data-type helpers.*

#### 13.1.3.1 Detailed Description

APIs for defining IPC interfaces using C++ as language.

#### 13.1.3.2 Internal Helpers

Collaboration diagram for Internal Helpers:



#### Data Structures

- struct [L4::Types::Bool< V >](#)  
*Boolean meta type.*
- struct [L4::Types::False](#)  
*False meta value.*
- struct [L4::Types::True](#)  
*True meta value.*
- struct [L4::Types::Same< A, B >](#)  
*Compare two data types for equality.*

### 13.1.3.2.1 Detailed Description

## 13.1.4 Cache Consistency

Various functions for cache consistency.

Collaboration diagram for Cache Consistency:



### Functions

- `int l4_cache_clean_data` (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache clean a range in D-cache; writes back to PoC.*
- `int l4_cache_flush_data` (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache flush a range; writes back to PoC.*
- `int l4_cache_inv_data` (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache invalidate a range; might write back to PoC.*
- `int l4_cache_coherent` (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent between I-cache and D-cache; writes back to PoU.*
- `int l4_cache_dma_coherent` (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory; writes back to PoC.*
- `int l4_cache_dma_coherent_full` (void) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory; writes back to PoC.*

### 13.1.4.1 Detailed Description

Various functions for cache consistency.

These functions shall be used to ensure that

- all blocks (e.g. CPU cores, devices, DMA engines) are guaranteed to see the same copy of a memory location (Point of Coherency – PoC),
- instruction and data caches of a core are guaranteed to see the same copy of a memory location (Point of Unification – PoU).

Certain functions are NOPs on certain architectures, for example on Intel it's not necessary to explicitly make caches coherent to PoU.

### 13.1.4.2 Function Documentation

#### 13.1.4.2.1 `l4_cache_clean_data()`

```
int l4_cache_clean_data (
    unsigned long start,
    unsigned long end ) [inline]
```

Cache clean a range in D-cache; writes back to PoC.

**Parameters**

<i>start</i>	Start of range (inclusive)
<i>end</i>	End of range (exclusive)

**Return values**

<i>0</i>	on success
<i>-EFAULT</i>	in the case of an unresolved page fault in the given area

Writes back any dirty cache lines in the range but leaves them in the cache and marks the cached copies clean.

**Examples**

[examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#).

Definition at line [81](#) of file [cache.h](#).

**13.1.4.2.2 l4\_cache\_coherent()**

```
int l4_cache_coherent (
    unsigned long start,
    unsigned long end ) [inline]
```

Make memory coherent between I-cache and D-cache; writes back to PoU.

**Parameters**

<i>start</i>	Start of range (inclusive)
<i>end</i>	End of range (exclusive)

**Return values**

<i>0</i>	on success
<i>-EFAULT</i>	in the case of an unresolved page fault in the given area

Definition at line [105](#) of file [cache.h](#).

**13.1.4.2.3 l4\_cache\_dma\_coherent()**

```
int l4_cache_dma_coherent (
    unsigned long start,
    unsigned long end ) [inline]
```

Make memory coherent for use with external memory; writes back to PoC.

## Parameters

<i>start</i>	Start of range (inclusive)
<i>end</i>	End of range (exclusive)

## Return values

<i>0</i>	on success
<i>-EFAULT</i>	in the case of an unresolved page fault in the given area

Definition at line 113 of file [cache.h](#).

**13.1.4.2.4 l4\_cache\_flush\_data()**

```
int l4_cache_flush_data (
    unsigned long start,
    unsigned long end ) [inline]
```

Cache flush a range; writes back to PoC.

## Parameters

<i>start</i>	Start of range (inclusive)
<i>end</i>	End of range (exclusive)

## Return values

<i>0</i>	on success
<i>-EFAULT</i>	in the case of an unresolved page fault in the given area

Writes back any dirty cache lines and invalidates all cache entries in the range.

Definition at line 89 of file [cache.h](#).

**13.1.4.2.5 l4\_cache\_inv\_data()**

```
int l4_cache_inv_data (
    unsigned long start,
    unsigned long end ) [inline]
```

Cache invalidate a range; might write back to PoC.

## Parameters

<i>start</i>	Start of range (inclusive)
<i>end</i>	End of range (exclusive)

## Return values

0	on success
-EFAULT	in the case of an unresolved page fault in the given area

Invalidates all cache entries in the range but does not necessarily write back dirty cache lines.

## Note

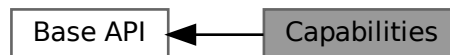
Implementations may choose to write back dirty lines nonetheless if this is more efficient.

Definition at line 97 of file [cache.h](#).

### 13.1.5 Capabilities

C interface for capabilities.

Collaboration diagram for Capabilities:



#### Typedefs

- typedef unsigned long [l4\\_cap\\_idx\\_t](#)  
*Capability selector type.*

#### Enumerations

- enum [l4\\_cap\\_consts\\_t](#) {  
[L4\\_CAP\\_SHIFT](#) , [L4\\_CAP\\_SIZE](#) = 1UL << [L4\\_CAP\\_SHIFT](#) , [L4\\_CAP\\_OFFSET](#) , [L4\\_CAP\\_MASK](#) ,  
[L4\\_INVALID\\_CAP](#) , [L4\\_INVALID\\_CAP\\_BIT](#) = 1UL << ([L4\\_CAP\\_SHIFT](#) - 1) }  
*Constants related to capability selectors.*
- enum [l4\\_default\\_caps\\_t](#) {  
[L4\\_BASE\\_TASK\\_CAP](#) , [L4\\_BASE\\_FACTORY\\_CAP](#) , [L4\\_BASE\\_THREAD\\_CAP](#) , [L4\\_BASE\\_PAGER\\_CAP](#) ,  
[L4\\_BASE\\_LOG\\_CAP](#) , [L4\\_BASE\\_ICU\\_CAP](#) , [L4\\_BASE\\_SCHEDULER\\_CAP](#) , [L4\\_BASE\\_IOMMU\\_CAP](#) ,  
[L4\\_BASE\\_DEBUGGER\\_CAP](#) , [L4\\_BASE\\_ARM\\_SMCCC\\_CAP](#) , [L4\\_BASE\\_CAPS\\_LAST\\_P1](#) , [L4\\_BASE\\_CAPS\\_LAST](#)  
= [L4\\_BASE\\_CAPS\\_LAST\\_P1](#) - 1 }  
*Default capabilities setup for the initial tasks.*

## Functions

- unsigned [l4\\_is\\_invalid\\_cap](#) ([l4\\_cap\\_idx\\_t](#) c) [L4\\_NOTHROW](#)  
*Test if a capability selector is the invalid capability.*
- unsigned [l4\\_is\\_valid\\_cap](#) ([l4\\_cap\\_idx\\_t](#) c) [L4\\_NOTHROW](#)  
*Test if a capability selector is a valid selector.*
- unsigned [l4\\_capability\\_equal](#) ([l4\\_cap\\_idx\\_t](#) c1, [l4\\_cap\\_idx\\_t](#) c2) [L4\\_NOTHROW](#)  
*Test if the capability indices of two capability selectors are equal.*

### 13.1.5.1 Detailed Description

C interface for capabilities.

Add

```
#include <l4/sys/types.h>
#include <l4/sys/consts.h>
```

to your code to use the functions and definitions explained here.

### 13.1.5.2 Typedef Documentation

#### 13.1.5.2.1 [l4\\_cap\\_idx\\_t](#)

```
typedef unsigned long l4\_cap\_idx\_t
```

Capability selector type.

A capability selector is either a (shifted) capability index or the invalid capability selector [L4\\_INVALID\\_CAP](#).

Usage of the invalid capability selector is defined only for invoking IPC (see [Object Invocation](#)): When IPC is invoked on [L4\\_INVALID\\_CAP](#), then it is resolved to a capability for the current thread with full permissions.

Otherwise, the API assumes that each argument of type [l4\\_cap\\_idx\\_t](#) is a capability index, i.e., `idx << L4\_CAP\_SHIFT` for arbitrary `idx`. The behavior for other arguments is then undefined.

Definition at line [359](#) of file [types.h](#).

### 13.1.5.3 Enumeration Type Documentation

#### 13.1.5.3.1 [l4\\_cap\\_consts\\_t](#)

```
enum l4\_cap\_consts\_t
```

Constants related to capability selectors.

#### Enumerator

<a href="#">L4_CAP_SHIFT</a>	Capability index shift.
<a href="#">L4_CAP_SIZE</a>	
Generated for L4Re by Doxygen	<b>Deprecated</b> Superseded by <a href="#">L4_CAP_OFFSET</a> .
<a href="#">L4_CAP_OFFSET</a>	Offset of two consecutive capability selectors.
<a href="#">L4_CAP_MASK</a>	Mask to get only the relevant bits of an <a href="#">l4_cap_idx_t</a> .

Definition at line 154 of file [consts.h](#).

### 13.1.5.3.2 l4\_default\_caps\_t

```
enum l4_default_caps_t
```

Default capabilities setup for the initial tasks.

These capability selectors are setup per default by the micro kernel for the two initial tasks, the Root-Pager (Sigma0) and the Root-Task (Moe).

#### Attention

These constants do not have any particular meaning for applications started by Moe, see [Initial Environment](#) for this kind of information.

#### See also

[Initial Environment](#) for information useful for normal user applications.

#### Enumerator

L4_BASE_TASK_CAP	Capability selector for the current task.
L4_BASE_FACTORY_CAP	Capability selector for the factory.
L4_BASE_THREAD_CAP	Capability selector for the first thread.
L4_BASE_PAGER_CAP	Capability selector for the pager gate. For Sigma0, the pager is not present since it never raises page faults. For Moe, the pager is set to Sigma0.
L4_BASE_LOG_CAP	Capability selector for the log object. Present if the corresponding feature is turned on in the microkernel configuration.
L4_BASE_ICU_CAP	Capability selector for the base icu object.
L4_BASE_SCHEDULER_CAP	Capability selector for the scheduler cap.
L4_BASE_IOMMU_CAP	Capability selector for the IO-MMU cap. Present if the microkernel detected an IO-MMU.
L4_BASE_DEBUGGER_CAP	Capability selector for the debugger cap. Present if the corresponding feature is turned on in the microkernel configuration.
L4_BASE_ARM_SMCCC_CAP	Capability selector for the ARM SMCCC cap. Present if the microkernel detected an ARM SMC capable trusted execution environment.
L4_BASE_CAPS_LAST	Last capability index used for base capabilities.

Definition at line 313 of file [consts.h](#).

### 13.1.5.4 Function Documentation

#### 13.1.5.4.1 l4\_capability\_equal()

```
unsigned l4_capability_equal (
    l4_cap_idx_t c1,
    l4_cap_idx_t c2 ) [inline]
```

Test if the capability indices of two capability selectors are equal.



**Parameters**

<i>c1</i>	Capability selector.
<i>c2</i>	Capability selector.

**Return values**

0	The index parts of the capability selectors differ.
1	The index parts of the capability selectors are equal.

**Precondition**

Both capability selectors must be valid (cf. [l4\\_is\\_valid\\_cap\(\)](#)) otherwise the return value is undefined.

Definition at line 420 of file [types.h](#).

References [L4\\_CAP\\_SHIFT](#).

**13.1.5.4.2 l4\_is\_invalid\_cap()**

```
unsigned l4_is_invalid_cap (
    l4_cap_idx_t c ) [inline]
```

Test if a capability selector is the invalid capability.

**Parameters**

<i>c</i>	Capability selector
----------	---------------------

**Return values**

0	The capability selector is not the invalid capability.
>0	The capability selector is the invalid capability.

**Examples**

[examples/libs/l4re/c/ma+rm.c](#), [examples/sys/aliens/main.c](#), [examples/sys/isr/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 412 of file [types.h](#).

**13.1.5.4.3 l4\_is\_valid\_cap()**

```
unsigned l4_is_valid_cap (
    l4_cap_idx_t c ) [inline]
```

Test if a capability selector is a valid selector.

## Parameters

<code>c</code>	Capability selector
----------------	---------------------

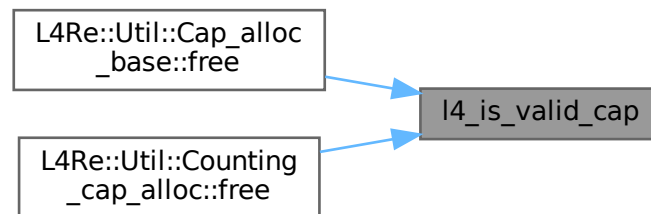
## Return values

<code>0</code>	The capability selector is not valid.
<code>&gt;0</code>	The capability selector is valid.

Definition at line 416 of file [types.h](#).

Referenced by [L4Re::Util::Cap\\_alloc\\_base::free\(\)](#), and [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE, Dbg >::free\(\)](#).

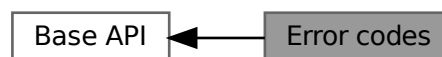
Here is the caller graph for this function:



### 13.1.6 Error codes

Common error codes.

Collaboration diagram for Error codes:



## Enumerations

- enum [l4\\_error\\_code\\_t](#) {  
[L4\\_EOK](#) = 0 , [L4\\_EPERM](#) = 1 , [L4\\_ENOENT](#) = 2 , [L4\\_EIO](#) = 5 ,  
[L4\\_ENXIO](#) = 6 , [L4\\_E2BIG](#) = 7 , [L4\\_EAGAIN](#) = 11 , [L4\\_ENOMEM](#) = 12 ,  
[L4\\_EACCESS](#) = 13 , [L4\\_EFAULT](#) = 14 , [L4\\_EBUSY](#) = 16 , [L4\\_EEXIST](#) = 17 ,

```

L4_ENODEV = 19 , L4_ENOTDIR = 20 , L4_EINVAL = 22 , L4_ENOSPC = 28 ,
L4_ERANGE = 34 , L4_ENAMETOOLONG = 36 , L4_ENOSYS = 38 , L4_EBADPROTO = 39 ,
L4_EADDRNOTAVAIL = 99 , L4_ERRNOMAX = 100 , L4_ENOREPLY = 1000 , L4_MSGTOOSHORT =
1001 ,
L4_MSGTOOLONG = 1002 , L4_MSGMISSARG = 1003 , L4_EIPC_LO = 2000 , L4_EIPC_HI = 2000 +
0x1f }

```

*L4 error codes.*

### 13.1.6.1 Detailed Description

Common error codes.

#### Include File

```
#include <l4/sys/err.h>
```

### 13.1.6.2 Enumeration Type Documentation

#### 13.1.6.2.1 l4\_error\_code\_t

```
enum l4_error_code_t
```

*L4 error codes.*

Those error codes are used by both the kernel and the user programs.

#### Enumerator

L4_EOK	Ok.
L4_EPERM	No permission.
L4_ENOENT	No such entity.
L4_EIO	I/O error.
L4_ENXIO	No such device or address.
L4_E2BIG	Argument value too big.
L4_EAGAIN	Try again.
L4_ENOMEM	No memory.
L4_EACCESS	Permission denied.
L4_EFAULT	Invalid memory address.
L4_EBUSY	Object currently busy, try later.
L4_EEXIST	Already exists.
L4_ENODEV	No such thing.
L4_ENOTDIR	Not a directory.
L4_EINVAL	Invalid argument.
L4_ENOSPC	No space left on device.
L4_ERANGE	Range error.
L4_ENAMETOOLONG	Name too long.
L4_ENOSYS	No sys.
L4_EBADPROTO	Unsupported protocol.
L4_EADDRNOTAVAIL	Address not available.
L4_ERRNOMAX	Maximum error value.

## Enumerator

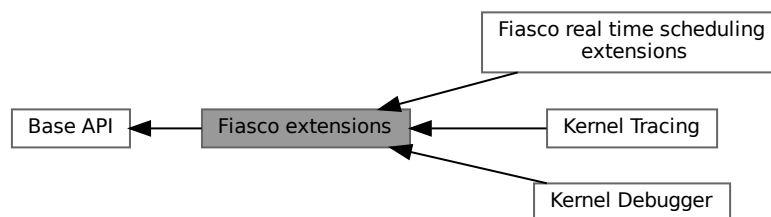
L4_ENOREPLY	No reply.
L4_MSGTOOSHORT	Message too short.
L4_MSGTOOLONG	Message too long.
L4_MSGMISSARG	Message has invalid capability.
L4_EIPC_LO	Communication error-range low.
L4_EIPC_HI	Communication error-range high.

Definition at line 41 of file [err.h](#).

### 13.1.7 Fiasco extensions

Extensions of the Fiasco [L4](#) implementation.

Collaboration diagram for Fiasco extensions:



### Modules

- [Fiasco real time scheduling extensions](#)  
*Real time scheduling extension for the Fiasco L4 implementation.*
- [Kernel Debugger](#)  
*Kernel debugger related functionality.*
- [Kernel Tracing](#)  
*Kernel tracing related functionality.*

### Files

- file [segment.h](#)  
*l4f specific fs/gs manipulation*
- file [segment.h](#)  
*l4f specific segment manipulation*

## Functions

- long `fiasco_ldt_set` (`l4_cap_idx_t` task, void \*ldt, unsigned int num\_desc, unsigned int entry\_number\_start, `l4_utcb_t` \*utcb)  
*Set LDT segments descriptors.*
- long `fiasco_gdt_set` (`l4_cap_idx_t` thread, void \*desc, unsigned int size, unsigned int entry\_number\_start, `l4_utcb_t` \*utcb)  
*Set GDT segment descriptors.*
- unsigned `fiasco_gdt_get_entry_offset` (`l4_cap_idx_t` thread, `l4_utcb_t` \*utcb)  
*Return the offset of the entry in the GDT.*

### 13.1.7.1 Detailed Description

Extensions of the Fiasco L4 implementation.

### 13.1.7.2 Function Documentation

#### 13.1.7.2.1 `fiasco_gdt_get_entry_offset()`

```
unsigned fiasco_gdt_get_entry_offset (
    l4_cap_idx_t thread,
    l4_utcb_t * utcb ) [inline]
```

Return the offset of the entry in the GDT.

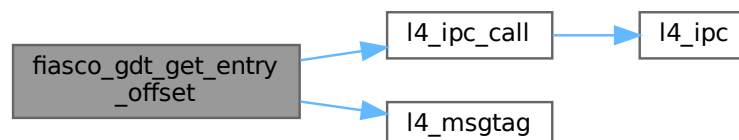
#### Parameters

<i>thread</i>	Thread to get info from.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

Definition at line 177 of file [segment.h](#).

References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_THREAD](#), [L4\\_THREAD\\_X86\\_GDT\\_OP](#), and [l4\\_msg\\_regs\\_t::mr](#).

Here is the call graph for this function:



### 13.1.7.2.2 `fiasco_gdt_set()`

```
long fiasco_gdt_set (
    l4_cap_idx_t thread,
    void * desc,
    unsigned int size,
    unsigned int entry_number_start,
    l4_utcb_t * utcb ) [inline]
```

Set GDT segment descriptors.

Fiasco supports 4 consecutive entries, starting at the value returned by `fiasco_gdt_get_entry_offset()`.

#### Parameters

<i>thread</i>	Thread to set the GDT entry for.
<i>desc</i>	Pointer to GDT descriptors.
<i>size</i>	Size of the descriptors in bytes (multiple of 8).
<i>entry_number_start</i>	Entry number to start (valid values: 0-3).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

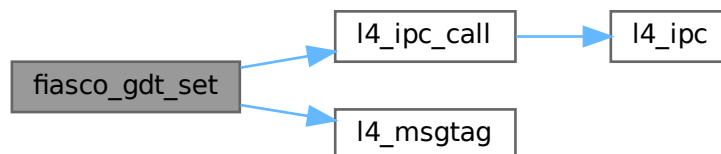
#### Return values

<code>&lt; 0</code>	At least one provided GDT descriptor is considered unsafe by the kernel, and not all selected GDT descriptors have been updated.
<code>L4_EOK</code>	Success.

Definition at line 52 of file [segment.h](#).

References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_THREAD](#), [L4\\_THREAD\\_X86\\_GDT\\_OP](#), and [l4\\_msg\\_regs\\_t::mr](#).

Here is the call graph for this function:



### 13.1.7.2.3 `fiasco_ldt_set()`

```
long fiasco_ldt_set (
    l4_cap_idx_t task,
```

```
void * ldt,
unsigned int num_desc,
unsigned int entry_number_start,
l4_utcb_t * utcb ) [inline]
```

Set LDT segments descriptors.

#### Parameters

<i>task</i>	Task to set the segment for.
<i>ldt</i>	Pointer to LDT hardware descriptors.
<i>num_desc</i>	Number of descriptors.
<i>entry_number_start</i>	Entry number to start.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

#### Return values

<i>-L4_ENOSYS</i>	The kernel configuration doesn't support this feature.
<i>-L4_EINVAL</i>	Invalid descriptor or invalid entry number.
<i>L4_EOK</i>	Success.

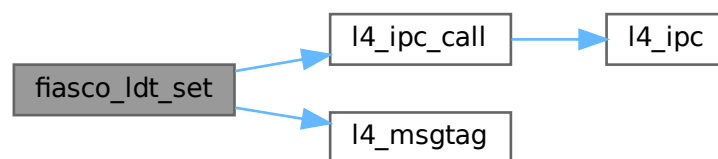
#### Note

This feature is not available if the kernel is configured with page table isolation.

Definition at line 164 of file [segment.h](#).

References [L4\\_EINVAL](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_TASK](#), [L4\\_TASK\\_LDT\\_SET\\_X86\\_OP](#), [L4\\_TASK\\_LDT\\_X86\\_ENTRY\\_SIZE](#), [L4\\_TASK\\_LDT\\_X86\\_MAX\\_ENTRIES](#), and [l4\\_msg\\_regs\\_t::mr](#).

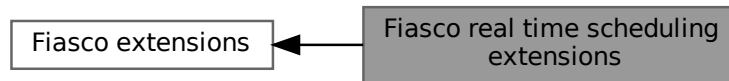
Here is the call graph for this function:



#### 13.1.7.3 Fiasco real time scheduling extensions

Real time scheduling extension for the Fiasco L4 implementation.

Collaboration diagram for Fiasco real time scheduling extensions:



Real time scheduling extension for the Fiasco L4 implementation.

#### 13.1.7.4 Kernel Debugger

Kernel debugger related functionality.

Collaboration diagram for Kernel Debugger:



#### Files

- file [kdebug.h](#)

*Functionality for invoking the kernel debugger.*

#### Functions

- [l4\\_msgtag\\_t l4\\_debugger\\_set\\_object\\_name](#) ([l4\\_cap\\_idx\\_t](#) cap, const char \*name) [L4\\_NOTHROW](#)  
*Set the name of a kernel object.*
- [l4\\_msgtag\\_t l4\\_debugger\\_get\\_object\\_name](#) ([l4\\_cap\\_idx\\_t](#) cap, unsigned id, char \*name, unsigned size) [L4\\_NOTHROW](#)  
*Get name of the kernel object with Id id.*
- unsigned long [l4\\_debugger\\_global\\_id](#) ([l4\\_cap\\_idx\\_t](#) cap) [L4\\_NOTHROW](#)  
*Get the globally unique ID of the object behind a capability.*
- unsigned long [l4\\_debugger\\_kobj\\_to\\_id](#) ([l4\\_cap\\_idx\\_t](#) cap, [l4\\_addr\\_t](#) kobjp) [L4\\_NOTHROW](#)  
*Get the globally unique ID of the object behind the kobject pointer.*
- long [l4\\_debugger\\_query\\_log\\_typeid](#) ([l4\\_cap\\_idx\\_t](#) cap, const char \*name, unsigned idx) [L4\\_NOTHROW](#)  
*Query the log-id for a log type.*
- long [l4\\_debugger\\_query\\_log\\_name](#) ([l4\\_cap\\_idx\\_t](#) cap, unsigned idx, char \*name, unsigned namelen, char \*shortname, unsigned shortnamelen) [L4\\_NOTHROW](#)  
*Query the name of a log type given the ID.*
- [l4\\_msgtag\\_t l4\\_debugger\\_switch\\_log](#) ([l4\\_cap\\_idx\\_t](#) cap, const char \*name, int on\_off) [L4\\_NOTHROW](#)  
*Set or unset log.*
- [l4\\_msgtag\\_t l4\\_debugger\\_add\\_image\\_info](#) ([l4\\_cap\\_idx\\_t](#) cap, [l4\\_addr\\_t](#) base, const char \*name) [L4\\_NOTHROW](#)  
*Add loaded image information for a task.*



#### 13.1.7.4.1 Detailed Description

Kernel debugger related functionality.

##### Attention

This API is subject to change!

This is a debugging facility, any call to any function might be invalid. Do not rely on it in any real code.

##### Include File

```
#include <l4/sys/debugger.h>
```

#### 13.1.7.4.2 Function Documentation

##### 13.1.7.4.2.1 l4\_debugger\_add\_image\_info()

```
l4_msgtag_t l4_debugger_add_image_info (
    l4_cap_idx_t cap,
    l4_addr_t base,
    const char * name ) [inline]
```

Add loaded image information for a task.

##### Parameters

<i>cap</i>	Capability which refers to the task object.
<i>base</i>	Load base address of image.
<i>name</i>	Image base name.

This is a debugging facility, the call might be invalid.

Definition at line 428 of file [debugger.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 13.1.7.4.2.2 l4\_debugger\_get\_object\_name()

```
l4_msgtag_t l4_debugger_get_object_name (
    l4_cap_idx_t cap,
    unsigned id,
    char * name,
    unsigned size ) [inline]
```

Get name of the kernel object with Id `id`.

#### Parameters

	<i>cap</i>	Capability of the debugger object.
	<i>id</i>	Global id of the object whose name is asked.
out	<i>name</i>	Buffer to copy the name into. The buffer must be allocated by the caller.
	<i>size</i>	Length of the <code>name</code> buffer.

#### Returns

Syscall return tag

Definition at line 421 of file [debugger.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 13.1.7.4.2.3 l4\_debugger\_global\_id()

```
unsigned long l4_debugger_global_id (
    l4_cap_idx_t cap ) [inline]
```

Get the globally unique ID of the object behind a capability.

#### Parameters

<i>cap</i>	Capability
------------	------------

#### Return values

<code>~0UL</code>	Capability is not valid.
-------------------	--------------------------

## Return values

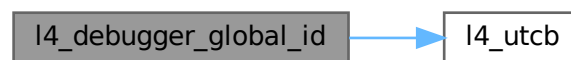
<i>otherwise</i>	Global debugger id.
------------------	---------------------

This is a debugging facility, the call might be invalid.

Definition at line 386 of file [debugger.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



## 13.1.7.4.2.4 l4\_debugger\_kobj\_to\_id()

```

unsigned long l4_debugger_kobj_to_id (
    l4_cap_idx_t cap,
    l4_addr_t kobjp ) [inline]
  
```

Get the globally unique ID of the object behind the kobject pointer.

## Parameters

<i>cap</i>	Capability
<i>kobjp</i>	Kobject pointer

## Return values

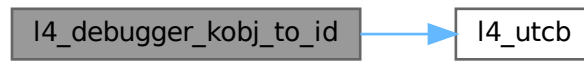
$\sim 0UL$	The capability or the kobject pointer are invalid.
<i>otherwise</i>	The globally unique id.

This is a debugging facility, the call might be invalid.

Definition at line 392 of file [debugger.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.7.4.2.5 l4\_debugger\_query\_log\_name()

```
long l4_debugger_query_log_name (
    l4_cap_idx_t cap,
    unsigned idx,
    char * name,
    unsigned namelen,
    char * shortname,
    unsigned shortnamelen ) [inline]
```

Query the name of a log type given the ID.

##### Parameters

<i>cap</i>	Debugger capability.
<i>idx</i>	ID to query.
<i>name</i>	Buffer to copy name to.
<i>namelen</i>	Buffer length of name.
<i>shortname</i>	Buffer to copy shortname to.
<i>shortnamelen</i>	Buffer length of shortname.

##### Return values

0	Success
<0	Error

This is a debugging facility, the call might be invalid.

Definition at line 405 of file [debugger.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.7.4.2.6 l4\_debugger\_query\_log\_typeid()

```
long l4_debugger_query_log_typeid (
    l4_cap_idx_t cap,
    const char * name,
    unsigned idx ) [inline]
```

Query the log-id for a log type.

##### Parameters

<i>cap</i>	Debugger capability
<i>name</i>	Name to query for.
<i>idx</i>	Idx to start searching, start with 0

##### Returns

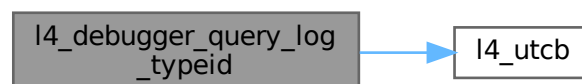
positive ID, or negative error code

This is a debugging facility, the call might be invalid.

Definition at line 398 of file [debugger.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.7.4.2.7 l4\_debugger\_set\_object\_name()

```
l4_msgtag_t l4_debugger_set_object_name (
    l4_cap_idx_t cap,
    const char * name ) [inline]
```

Set the name of a kernel object.

##### Parameters

<i>cap</i>	Capability which refers to the kernel object.
<i>name</i>	Name of the kernel object that is e.g. displayed in the kernel debugger.

This is a debugging facility, the call might be invalid.

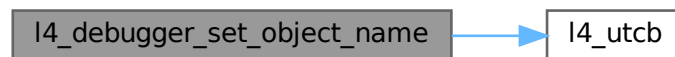
##### Examples

[examples/libs/shmc/prodcons.c](#), and [examples/sys/aliens/main.c](#).

Definition at line 379 of file [debugger.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.7.4.2.8 l4\_debugger\_switch\_log()

```
l4_msgtag_t l4_debugger_switch_log (
    l4_cap_idx_t cap,
    const char * name,
    int on_off ) [inline]
```

Set or unset log.

##### Parameters

<i>cap</i>	Debugger object.
<i>name</i>	Name of the log type.
<i>on_off</i>	1: turn log on, 0: turn log off

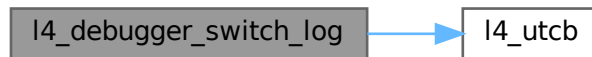
**Returns**

Syscall return tag

Definition at line 414 of file [debugger.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**13.1.7.5 Kernel Tracing**

Kernel tracing related functionality.

Collaboration diagram for Kernel Tracing:

**Functions**

- [l4\\_umword\\_t fiasco\\_tbuf\\_log](#) (const char \*text)  
*Create new trace-buffer entry with describing <text>.*
- [l4\\_umword\\_t fiasco\\_tbuf\\_log\\_3val](#) (const char \*text, [l4\\_umword\\_t](#) v1, [l4\\_umword\\_t](#) v2, [l4\\_umword\\_t](#) v3)  
*Create new trace-buffer entry with describing <text> and three additional values.*
- [l4\\_umword\\_t fiasco\\_tbuf\\_log\\_binary](#) (const unsigned char \*data)  
*Create new trace-buffer entry with binary data.*
- void **fiasco\_tbuf\_clear** (void)  
*Clear trace-buffer.*
- void **fiasco\_tbuf\_dump** (void)  
*Dump trace-buffer to kernel console.*

### 13.1.7.5.1 Detailed Description

Kernel tracing related functionality.

#### Attention

This API is subject to change!

This is a tracing facility for the Fiasco kernel trace buffer. Any call to any function might be invalid. Do not rely on it in any real code.

#### Include File

```
#include <l4/sys/ktrace.h>
```

### 13.1.7.5.2 Function Documentation

#### 13.1.7.5.2.1 fiasco\_tbuf\_log()

```
l4_umword_t fiasco_tbuf_log (
    const char * text ) [inline]
```

Create new trace-buffer entry with describing <text>.

#### Parameters

<i>text</i>	Logging text
-------------	--------------

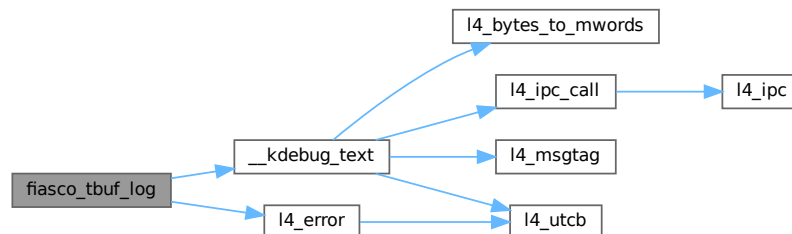
#### Returns

Pointer to trace-buffer entry

Definition at line 35 of file [\\_\\_ktrace-impl.h](#).

References [\\_\\_kdebug\\_text\(\)](#), and [l4\\_error\(\)](#).

Here is the call graph for this function:





### 13.1.7.5.2.2 fiasco\_tbuf\_log\_3val()

```
l4_umword_t fiasco_tbuf_log_3val (
    const char * text,
    l4_umword_t v1,
    l4_umword_t v2,
    l4_umword_t v3 ) [inline]
```

Create new trace-buffer entry with describing <text> and three additional values.

#### Parameters

<i>text</i>	Logging text
<i>v1</i>	first value
<i>v2</i>	second value
<i>v3</i>	third value

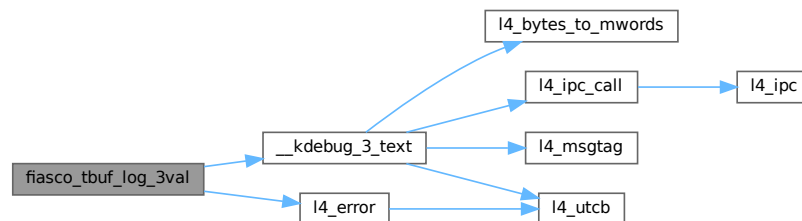
#### Returns

Pointer to trace-buffer entry

Definition at line 42 of file [\\_\\_ktrace-impl.h](#).

References [\\_\\_kdebug\\_3\\_text\(\)](#), and [l4\\_error\(\)](#).

Here is the call graph for this function:



### 13.1.7.5.2.3 fiasco\_tbuf\_log\_binary()

```
l4_umword_t fiasco_tbuf_log_binary (
    const unsigned char * data ) [inline]
```

Create new trace-buffer entry with binary data.

#### Parameters

<i>data</i>	binary data
-------------	-------------

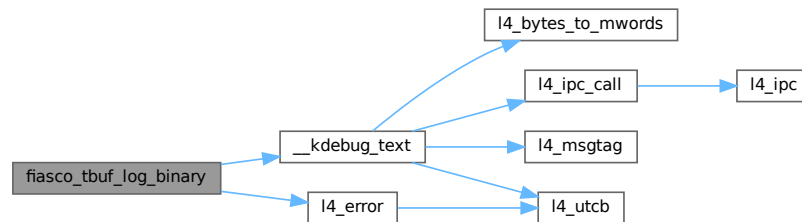
**Returns**

Pointer to trace-buffer entry

Definition at line 65 of file [\\_\\_ktrace-impl.h](#).

References [\\_\\_kdebug\\_text\(\)](#), and [l4\\_error\(\)](#).

Here is the call graph for this function:

**13.1.8 Flex pages**

Flex-page related API.

Collaboration diagram for Flex pages:

**Data Structures**

- union [l4\\_fpage\\_t](#)  
*L4 flexpage type.*
- struct [l4\\_snd\\_fpage\\_t](#)  
*Send-flex-page types.*

## Enumerations

- enum `L4_fpage_consts` {  
`L4_FPAGE_RIGHTS_SHIFT` = 0 , `L4_FPAGE_TYPE_SHIFT` = 4 , `L4_FPAGE_SIZE_SHIFT` = 6 ,  
`L4_FPAGE_ADDR_SHIFT` = 12 ,  
`L4_FPAGE_RIGHTS_BITS` = 4 , `L4_FPAGE_TYPE_BITS` = 2 , `L4_FPAGE_SIZE_BITS` = 6 , `L4_FPAGE_ADDR_BITS`  
= `L4_MWORD_BITS` - `L4_FPAGE_ADDR_SHIFT` ,  
`L4_FPAGE_RIGHTS_MASK` , `L4_FPAGE_TYPE_MASK` , `L4_FPAGE_SIZE_MASK` , `L4_FPAGE_ADDR_`  
`_MASK` = `~0UL << L4_FPAGE_ADDR_SHIFT` ,  
`L4_FPAGE_RIGHTS_ALL` = `L4_FPAGE_RIGHTS_MASK` }  
*L4 flexpage structure.*
- enum { `L4_WHOLE_ADDRESS_SPACE` = 63 }  
*Constants for flexpages.*
- enum `L4_fpage_rights` {  
`L4_FPAGE_X` = 1 , `L4_FPAGE_W` = 2 , `L4_FPAGE_RO` = 4 , `L4_FPAGE_RW` = `L4_FPAGE_RO` | `L4_`  
`FPAGE_W` ,  
`L4_FPAGE_RX` = `L4_FPAGE_RO` | `L4_FPAGE_X` , `L4_FPAGE_RWX` = `L4_FPAGE_RW` | `L4_FPAGE_X` }  
*Memory and IO port flex-page rights.*
- enum `L4_cap_fpage_rights` {  
`L4_CAP_FPAGE_W` = 0x1 , `L4_CAP_FPAGE_S` = 0x2 , `L4_CAP_FPAGE_R` = 0x4 , `L4_CAP_FPAGE_RO` =  
0x4 ,  
`L4_CAP_FPAGE_D` = 0x8 , `L4_CAP_FPAGE_RW` = `L4_CAP_FPAGE_R` | `L4_CAP_FPAGE_W` ,  
`L4_CAP_FPAGE_RS` = `L4_CAP_FPAGE_R` | `L4_CAP_FPAGE_S` , `L4_CAP_FPAGE_RWS` = `L4_CAP_`  
`_FPAGE_RW` | `L4_CAP_FPAGE_S` ,  
`L4_CAP_FPAGE_RWSD` = `L4_CAP_FPAGE_RWS` | `L4_CAP_FPAGE_D` , `L4_CAP_FPAGE_RWD` = `L4_`  
`_CAP_FPAGE_RW` | `L4_CAP_FPAGE_D` , `L4_CAP_FPAGE_RSD` = `L4_CAP_FPAGE_RS` | `L4_CAP_`  
`FPAGE_D` }  
*Cap-flex-page rights.*
- enum `L4_fpage_type` { `L4_FPAGE_SPECIAL` = 0 , `L4_FPAGE_MEMORY` = 1 , `L4_FPAGE_IO` = 2 ,  
`L4_FPAGE_OBJ` = 3 }  
*Flex-page type.*
- enum `L4_fpage_control` { `L4_FPAGE_CONTROL_OFFSET_SHIFT` = 12 , `L4_FPAGE_CONTROL_MASK` =  
`~0UL << L4_FPAGE_CONTROL_OFFSET_SHIFT` }  
*Flex-page map control flags.*
- enum `L4_obj_fpage_ctl` {  
`L4_FPAGE_C_REF_CNT` = 0x00 , `L4_FPAGE_C_NO_REF_CNT` = 0x10 , `L4_FPAGE_C_OBJ_RIGHT1` =  
0x20 , `L4_FPAGE_C_OBJ_RIGHT2` = 0x40 ,  
`L4_FPAGE_C_OBJ_RIGHT3` = 0x80 , `L4_FPAGE_C_OBJ_RIGHTS` = 0xe0 , `L4_FPAGE_C_IPCGATE_SVR`  
= `L4_FPAGE_C_OBJ_RIGHT1` }  
*Flex-page map control for capabilities (snd\_base)*
- enum `L4_fpage_cacheability_opt_t` { `L4_FPAGE_CACHE_OPT` = 0x1 , `L4_FPAGE_CACHEABLE` = 0x3 ,  
`L4_FPAGE_BUFFERABLE` = 0x5 , `L4_FPAGE_UNCACHEABLE` = 0x1 }  
*Flex-page cacheability option.*
- enum { `L4_WHOLE_IOADDRESS_SPACE` = 16 , `L4_IOPORT_MAX` = (`1L << L4_WHOLE_IOADDRESS_`  
`_SPACE`) }

*Special constants for IO flex pages.*

## Functions

- `l4_fpage_t l4_fpage(l4_addr_t address, unsigned int size, unsigned char rights)` `L4_NOTHROW`  
*Create a memory flex page.*
- `l4_fpage_t l4_fpage_all(void)` `L4_NOTHROW`  
*Get a flex page, describing all address spaces at once.*
- `l4_fpage_t l4_fpage_invalid(void)` `L4_NOTHROW`

- Get an invalid flex page.*
- [l4\\_fpage\\_t l4\\_iofpage](#) (unsigned long port, unsigned int size) [L4\\_NOTHROW](#)
- Create an IO-port flex page.*
- [l4\\_fpage\\_t l4\\_obj\\_fpage](#) ([l4\\_cap\\_idx\\_t](#) obj, unsigned int order, unsigned char rights) [L4\\_NOTHROW](#)
- Create a kernel-object flex page.*
- [int l4\\_is\\_fpage\\_writable](#) ([l4\\_fpage\\_t](#) fp) [L4\\_NOTHROW](#)
- Test if the flex page is writable.*
- [unsigned l4\\_fpage\\_rights](#) ([l4\\_fpage\\_t](#) f) [L4\\_NOTHROW](#)
- Return rights from a flex page.*
- [unsigned l4\\_fpage\\_type](#) ([l4\\_fpage\\_t](#) f) [L4\\_NOTHROW](#)
- Return type from a flex page.*
- [unsigned l4\\_fpage\\_size](#) ([l4\\_fpage\\_t](#) f) [L4\\_NOTHROW](#)
- Return size from a flex page.*
- [unsigned long l4\\_fpage\\_page](#) ([l4\\_fpage\\_t](#) f) [L4\\_NOTHROW](#)
- Return the page part from a flex page.*
- [l4\\_addr\\_t l4\\_fpage\\_memaddr](#) ([l4\\_fpage\\_t](#) f) [L4\\_NOTHROW](#)
- Return the memory address from the memory flex page.*
- [l4\\_cap\\_idx\\_t l4\\_fpage\\_obj](#) ([l4\\_fpage\\_t](#) f) [L4\\_NOTHROW](#)
- Return the capability index from the object flex page.*
- [unsigned long l4\\_fpage\\_ioport](#) ([l4\\_fpage\\_t](#) f) [L4\\_NOTHROW](#)
- Return the IO port number from the IO flex page.*
- [l4\\_fpage\\_t l4\\_fpage\\_set\\_rights](#) ([l4\\_fpage\\_t](#) src, unsigned char new\_rights) [L4\\_NOTHROW](#)
- Set new right in a flex page.*
- [int l4\\_fpage\\_contains](#) ([l4\\_fpage\\_t](#) fpage, [l4\\_addr\\_t](#) addr, unsigned size) [L4\\_NOTHROW](#)
- Test whether a given range is completely within an fpage.*
- [unsigned char l4\\_fpage\\_max\\_order](#) (unsigned char order, [l4\\_addr\\_t](#) addr, [l4\\_addr\\_t](#) min\_addr, [l4\\_addr\\_t](#) max\_addr, [l4\\_addr\\_t](#) hotspot=0)
- Determine maximum flex page size of a region.*

### 13.1.8.1 Detailed Description

Flex-page related API.

A flex page is a page with a variable size, that can describe memory, IO-Ports (IA32 only), and sets of kernel objects.

A flex page describes an always size aligned region of an address space. The size is given in a log2 scale. This means the size in elements (bytes for memory, ports for IO-Ports, and capabilities for kernel objects) is always a power of two.

A flex page also carries type and access right information for the described region. The type information selects the address space in which the flex page is valid. Access rights have a meaning depending on the specific address space (type).

There exists a special type for defining *receive windows* or for the [l4\\_task\\_unmap\(\)](#) method, that can be used to describe all address spaces (all types) with a single flex page.

### 13.1.8.2 Enumeration Type Documentation

#### 13.1.8.2.1 anonymous enum

`anonymous enum`

Constants for flexpages.

## Enumerator

L4_WHOLE_ADDRESS_SPACE	Whole address space size. This value does not only specify the log2 size of the biggest possible memory flex page. It can be also used as size for a special flex page to define a flex page which completely covers all spaces.
------------------------	--

Definition at line 93 of file [\\_\\_l4\\_fpage.h](#).

## 13.1.8.2.2 anonymous enum

anonymous enum

Special constants for IO flex pages.

## Enumerator

L4_WHOLE_IOADDRESS_SPACE	Whole I/O address space size. In contrast to <a href="#">L4_WHOLE_ADDRESS_SPACE</a> , this value forms the log2 size of the biggest possible I/O flex page.
L4_IOPORT_MAX	Maximum I/O port address plus 1.

Definition at line 304 of file [\\_\\_l4\\_fpage.h](#).

## 13.1.8.2.3 L4\_cap\_fpage\_rights

enum [L4\\_cap\\_fpage\\_rights](#)

Cap-flex-page rights.

Capabilities are modified or transfered with map and unmap operations. For that capabilities are wrapped into flex-page objects. The flex-page carries a set of rights the sender wants to hand over to the receiver along with the capability.

For the user only the 'S' and the 'W' right are visible. Other rights such as the 'D' right are internal to the corresponding kernel object and cannot be evaluated by the receiver.

## Note

A thread can also map a capability from its task's capability table with a reduced set of rights into another slot of its own capability table.

## Enumerator

L4_CAP_FPAGE_W	Interface specific 'W' right for capability flex-pages. The semantics of the 'W' right is defined by the protocol. For example in case of a dataspace cap, the 'W' right is needed to get a writable dataspace.
L4_CAP_FPAGE_S	Interface specific 'S' right for capability flex-pages. The semantics of the 'S' right is defined by the interface. When transferring object capabilities via IPC, the kernel masks this right with the 'S' right of the capability used to address the IPC partner. Thus, the 'S' right of sent capabilities is only transferred if both the flex-page and the IPC gate or thread capability specifying the IPC partner have the 'S' right. For <a href="#">L4::Task::map()</a> , the 'S' right is only transferred if the flex-page, the source and destination task capabilities have the 'S' right.
Generated for L4Re by Doxygen	

## Enumerator

L4_CAP_FPAGE_R	Read right for capability flex-pages. This is always required, otherwise no capability is mapped.
L4_CAP_FPAGE_RO	Read right for capability flex-pages. This is always required, otherwise no capability is mapped.
L4_CAP_FPAGE_D	Delete right for capability flex-pages. This allows the receiver to delete the corresponding kernel object using <code>unmap()</code> regardless of other tasks still holding a capability to the kernel object. Such capabilities are set to an empty capability if the object is deleted.
L4_CAP_FPAGE_RW	Read and interface specific 'W' right for capability flex-pages. The semantics of the 'W' right is defined by the interface.  See also <a href="#">L4_CAP_FPAGE_W</a>
L4_CAP_FPAGE_RS	Read and interface specific 'S' right for capability flex-pages. The semantics of the 'S' right is defined by the interface.  See also <a href="#">L4_CAP_FPAGE_S</a>
L4_CAP_FPAGE_RWS	Read, interface specific 'W', and 'S' rights for capability flex-pages. The semantics of the 'W' and 'S' right are defined by the interface.  See also <a href="#">L4_CAP_FPAGE_R</a> , <a href="#">L4_CAP_FPAGE_W</a> , and <a href="#">L4_CAP_FPAGE_S</a>
L4_CAP_FPAGE_RWSD	Full rights for capability flex-pages.  See also <a href="#">L4_CAP_FPAGE_R</a> , <a href="#">L4_CAP_FPAGE_W</a> , <a href="#">L4_CAP_FPAGE_S</a> , and <a href="#">L4_CAP_FPAGE_D</a>
L4_CAP_FPAGE_RWD	Read, write, and delete right for capability flex-pages.  See also <a href="#">L4_CAP_FPAGE_R</a> , <a href="#">L4_CAP_FPAGE_W</a> , and <a href="#">L4_CAP_FPAGE_D</a>
L4_CAP_FPAGE_RSD	Read, 'S', and delete right for capability flex-pages.  See also <a href="#">L4_CAP_FPAGE_R</a> , <a href="#">L4_CAP_FPAGE_S</a> , and <a href="#">L4_CAP_FPAGE_D</a>

Definition at line 151 of file [\\_\\_l4\\_fpage.h](#).

#### 13.1.8.2.4 l4\_fpage\_cacheability\_opt\_t

```
enum l4_fpage_cacheability_opt_t
```

Flex-page cacheability option.

## Enumerator

L4_FPAGE_CACHE_OPT	Enable the cacheability option in a send flex page.
L4_FPAGE_CACHEABLE	Cacheability option to enable caches for the mapping.
L4_FPAGE_BUFFERABLE	Cacheability option to enable buffered writes for the mapping.
L4_FPAGE_UNCACHEABLE	Cacheability option to disable caching for the mapping.

Definition at line 285 of file [\\_\\_l4\\_fpage.h](#).

## 13.1.8.2.5 L4\_fpage\_consts

enum [L4\\_fpage\\_consts](#)

[L4](#) flexpage structure.

## Enumerator

L4_FPAGE_RIGHTS_SHIFT	Access permissions shift.
L4_FPAGE_TYPE_SHIFT	Flexpage type shift (memory, IO port, obj...)
L4_FPAGE_SIZE_SHIFT	Flexpage size shift (log2-based)
L4_FPAGE_ADDR_SHIFT	Page address shift.
L4_FPAGE_RIGHTS_BITS	Access permissions size.
L4_FPAGE_TYPE_BITS	Flexpage type size (memory, IO port, obj...)
L4_FPAGE_SIZE_BITS	Flexpage size size (log2-based)
L4_FPAGE_ADDR_BITS	Page address size.
L4_FPAGE_RIGHTS_MASK	Mask to get the flexpage rights.
L4_FPAGE_RIGHTS_ALL	Specify as flexpage rights during grant.

Definition at line 57 of file [\\_\\_l4\\_fpage.h](#).

## 13.1.8.2.6 L4\_fpage\_control

enum [L4\\_fpage\\_control](#)

Flex-page map control flags.

## Enumerator

L4_FPAGE_CONTROL_OFFSET_SHIFT	Number of bits an index must be shifted or an address must be aligned to in the control word.
L4_FPAGE_CONTROL_MASK	Mask for truncating the lower bits of the send base or the index of the control word.

Definition at line 244 of file [\\_\\_l4\\_fpage.h](#).

### 13.1.8.2.7 L4\_fpage\_rights

enum [L4\\_fpage\\_rights](#)

Memory and IO port flex-page rights.

For IO flexpages, bit 1 and bit 2 are a combined read/write right. In a map operation, the receiver receives the IO port capability when the sender possesses it and at least one of these bits is present. For an unmap operation, the absence of one of those bits is sufficient to unmap the IO port capability.

#### Enumerator

L4_FPAGE_X	Executable flex page.
L4_FPAGE_W	Writable flex page.
L4_FPAGE_RO	Read-only flex page
L4_FPAGE_RW	Read-write flex page.
L4_FPAGE_RX	Read-execute flex page.
L4_FPAGE_RWX	Read-write-execute flex page.

Definition at line [124](#) of file [\\_\\_l4\\_fpage.h](#).

### 13.1.8.2.8 L4\_fpage\_type

enum [L4\\_fpage\\_type](#)

Flex-page type.

#### Enumerator

L4_FPAGE_SPECIAL	Special flex page, either invalid or all spaces.
L4_FPAGE_MEMORY	Memory flex page.
L4_FPAGE_IO	IO-port flex page.
L4_FPAGE_OBJ	Object flex page (capabilities).

Definition at line [233](#) of file [\\_\\_l4\\_fpage.h](#).

### 13.1.8.2.9 L4\_obj\_fpage\_ctl

enum [L4\\_obj\\_fpage\\_ctl](#)

Flex-page map control for capabilities (snd\_base)

These rights need to be added to the snd\_base when mapping and control internal behavior. The exact meaning depends on the type of capability (currently used only with IPC gates).

#### Enumerator

L4_FPAGE_C_REF_CNT	Mapping is reference-counted (default).
--------------------	---



## Enumerator

L4_FPAGE_C_NO_REF_CNT	Don't increase the reference counter.
L4_FPAGE_C_OBJ_RIGHT1	Object-type specific right.
L4_FPAGE_C_OBJ_RIGHT2	Object-type specific right.
L4_FPAGE_C_OBJ_RIGHT3	Object-type specific right.
L4_FPAGE_C_OBJ_RIGHTS	All Object-type specific right bits.
L4_FPAGE_C_IPCGATE_SVR	The receiver may invoke IPC-gate-specific functions on the capability, e.g. bind a thread to the gate and modify the label. Needed if the receiver implements the server side of an IPC gate.

Definition at line 263 of file [\\_\\_l4\\_fpage.h](#).

## 13.1.8.3 Function Documentation

## 13.1.8.3.1 l4\_fpage()

```
l4_fpage_t l4_fpage (
    l4_addr_t address,
    unsigned int size,
    unsigned char rights ) [inline]
```

Create a memory flex page.

## Parameters

<i>address</i>	Flex-page start address
<i>size</i>	Flex-page size (log2), <a href="#">L4_WHOLE_ADDRESS_SPACE</a> to specify the whole address space (with <code>address</code> 0). The minimum log2 size of a memory flex page is defined by <a href="#">L4_LOG2_PAGESIZE</a> according to the size of the smallest virtual page supported by the MMU.
<i>rights</i>	Access rights, see <a href="#">L4_fpage_rights</a>

## Returns

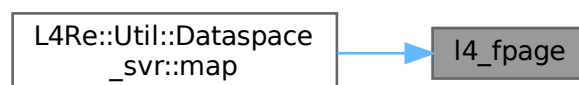
Memory flex page

Definition at line 668 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_FPAGE\\_MEMORY](#).

Referenced by [L4Re::Util::Dataspace\\_svr::map\(\)](#).

Here is the caller graph for this function:



### 13.1.8.3.2 l4\_fpage\_all()

```
l4_fpage_t l4_fpage_all (
    void ) [inline]
```

Get a flex page, describing all address spaces at once.

#### Returns

Special *all-spaces* flex page.

#### Note

This flex page can be used to define a receive window where the sender can send objects of any type, or for an unmap item completely covering all spaces of the target task. It does not make sense to use this flex page as send item.

Definition at line 688 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_FPAGE\\_SPECIAL](#), and [L4\\_WHOLE\\_ADDRESS\\_SPACE](#).

### 13.1.8.3.3 l4\_fpage\_contains()

```
int l4_fpage_contains (
    l4_fpage_t fpage,
    l4_addr_t addr,
    unsigned size ) [inline]
```

Test whether a given range is completely within an fpage.

#### Parameters

<i>fpage</i>	Flex page
<i>addr</i>	Address
<i>size</i>	Size of range in log2.

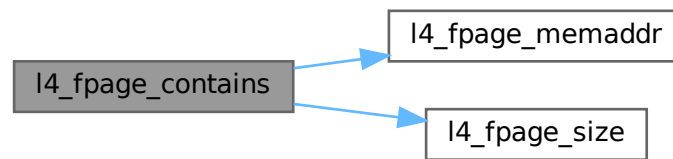
#### Return values

<i>==0</i>	The range is not completely in the fpage.
<i>!=0</i>	The range is within the fpage.

Definition at line 720 of file [\\_\\_l4\\_fpage.h](#).

References [l4\\_fpage\\_memaddr\(\)](#), and [l4\\_fpage\\_size\(\)](#).

Here is the call graph for this function:



#### 13.1.8.3.4 `l4_fpage_invalid()`

```
l4_fpage_t l4_fpage_invalid (
    void ) [inline]
```

Get an invalid flex page.

##### Returns

Special *invalid* flex page.

Definition at line 694 of file `__l4_fpage.h`.

References `L4_FPAGE_SPECIAL`.

#### 13.1.8.3.5 `l4_fpage_ioport()`

```
unsigned long l4_fpage_ioport (
    l4_fpage_t f ) [inline]
```

Return the IO port number from the IO flex page.

##### Parameters

<i>f</i>	Flex page
----------	-----------

##### Returns

IO port number from the given IO flex page.

##### Precondition

*f* must be an IO flex page (`l4_fpage_type(f) == L4_FPAGE_IO`) and

The function does not enforce size alignment of the read memory address. The caller must ensure the input fpage is correct.

Definition at line 624 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_FPAGE\\_ADDR\\_SHIFT](#).

### 13.1.8.3.6 l4\_fpage\_max\_order()

```
unsigned char l4_fpage_max_order (
    unsigned char order,
    l4_addr_t addr,
    l4_addr_t min_addr,
    l4_addr_t max_addr,
    l4_addr_t hotspot = 0 ) [inline]
```

Determine maximum flex page size of a region.

#### Parameters

<i>order</i>	Order value to start with (e.g. for memory L4_LOG2_PAGESIZE would be used)
<i>addr</i>	Address to be covered by the flex page.
<i>min_addr</i>	Start of region / minimal address (including).
<i>max_addr</i>	End of region / maximal address (excluding).
<i>hotspot</i>	(Optional) hot spot.

#### Returns

Maximum order (log2-size) possible.

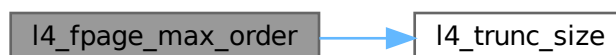
#### Note

The start address of the flex-page can be determined with `l4_trunc_size(addr, returnvalue)`

Definition at line 728 of file [\\_\\_l4\\_fpage.h](#).

References [l4\\_trunc\\_size\(\)](#).

Here is the call graph for this function:



#### 13.1.8.3.7 l4\_fpage\_memaddr()

```
l4_addr_t l4_fpage_memaddr (
    l4_fpage_t f ) [inline]
```

Return the memory address from the memory flex page.

**Parameters**

<i>f</i>	Flex page
----------	-----------

**Returns**

Page address from the given memory flex page.

**Precondition**

*f* must be a memory flex page (`l4_fpage_type(f) == L4_FPAGE_MEMORY`).

The function does not enforce size alignment of the read memory address. The caller must ensure the input fpage is correct.

Definition at line 630 of file [\\_\\_l4\\_fpage.h](#).

Referenced by [l4\\_fpage\\_contains\(\)](#).

Here is the caller graph for this function:

**13.1.8.3.8 l4\_fpage\_obj()**

```
l4_cap_idx_t l4_fpage_obj (
    l4_fpage_t f ) [inline]
```

Return the capability index from the object flex page.

**Parameters**

<i>f</i>	Flex page
----------	-----------

**Returns**

Capability index from the given object flex page.

**Precondition**

`f` must be an object flex page (`l4_fpage_type(f) == L4_FPAGE_OBJ`)

The function does not enforce size alignment of the read memory address. The caller must ensure the input fpage is correct.

Definition at line 636 of file [\\_\\_l4\\_fpage.h](#).

**13.1.8.3.9 l4\_fpage\_page()**

```
unsigned long l4_fpage_page (
    l4_fpage_t f ) [inline]
```

Return the page part from a flex page.

**Parameters**

<code>f</code>	Flex page
----------------	-----------

**Returns**

Page part of the given flex page.

**Note**

The meaning of the page part depends on the flex-page type.

Definition at line 618 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_FPAGE\\_ADDR\\_SHIFT](#).

**13.1.8.3.10 l4\_fpage\_rights()**

```
unsigned l4_fpage_rights (
    l4_fpage_t f ) [inline]
```

Return rights from a flex page.

**Parameters**

<code>f</code>	Flex page
----------------	-----------

**Returns**

Size part of the given flex page.

Definition at line 600 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_FPAGE\\_RIGHTS\\_MASK](#), and [L4\\_FPAGE\\_RIGHTS\\_SHIFT](#).

Referenced by [l4\\_is\\_fpage\\_writable\(\)](#).

Here is the caller graph for this function:



#### 13.1.8.3.11 l4\_fpage\_set\_rights()

```
l4_fpage_t l4_fpage_set_rights (
    l4_fpage_t src,
    unsigned char new_rights ) [inline]
```

Set new right in a flex page.

##### Parameters

<i>src</i>	Flex page
<i>new_rights</i>	New rights

##### Returns

Modified flex page with new rights.

Definition at line 659 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_FPAGE\\_RIGHTS\\_MASK](#), [L4\\_FPAGE\\_RIGHTS\\_SHIFT](#), and [l4\\_fpage\\_t::raw](#).

#### 13.1.8.3.12 l4\_fpage\_size()

```
unsigned l4_fpage_size (
    l4_fpage_t f ) [inline]
```

Return size from a flex page.

##### Parameters

<i>f</i>	Flex page
----------	-----------



**Returns**

Size part of the given flex page.

**See also**

[l4\\_fpage\\_memaddr\(\)](#), [l4\\_fpage\\_obj\(\)](#), [l4\\_fpage\\_ioport\(\)](#)

Definition at line 612 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_FPAGE\\_SIZE\\_SHIFT](#).

Referenced by [l4\\_fpage\\_contains\(\)](#).

Here is the caller graph for this function:

**13.1.8.3.13 l4\_fpage\_type()**

```
unsigned l4_fpage_type (
    l4_fpage_t f ) [inline]
```

Return type from a flex page.

**Parameters**

<i>f</i>	Flex page
----------	-----------

**Returns**

Type part of the given flex page.

Definition at line 606 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_FPAGE\\_TYPE\\_SHIFT](#).

**13.1.8.3.14 l4\_iofpage()**

```
l4_fpage_t l4_iofpage (
    unsigned long port,
    unsigned int size ) [inline]
```

Create an IO-port flex page.

## Parameters

<i>port</i>	I/O-flex-page port base
<i>size</i>	I/O-flex-page size (log2), <a href="#">L4_WHOLE_IOADDRESS_SPACE</a> to specify the whole I/O address space (with <code>port 0</code> )

## Returns

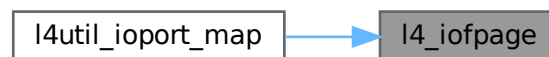
I/O flex page

Definition at line 674 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_FPAGE\\_ADDR\\_SHIFT](#), [L4\\_FPAGE\\_IO](#), and [L4\\_FPAGE\\_RW](#).

Referenced by [l4util\\_ioport\\_map\(\)](#).

Here is the caller graph for this function:



### 13.1.8.3.15 l4\_is\_fpage\_writable()

```
int l4_is_fpage_writable (
    l4\_fpage\_t fp ) [inline]
```

Test if the flex page is writable.

## Parameters

<i>fp</i>	Flex page.
-----------	------------

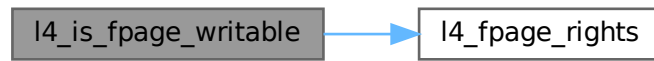
## Return values

<i>!=0</i>	if flex page is writable.
<i>==0</i>	if flex pags is not writable.

Definition at line 701 of file [\\_\\_l4\\_fpage.h](#).

References [l4\\_fpage\\_rights\(\)](#), and [L4\\_FPAGE\\_W](#).

Here is the call graph for this function:



### 13.1.8.3.16 l4\_obj\_fpage()

```

l4_fpage_t l4_obj_fpage (
    l4_cap_idx_t obj,
    unsigned int order,
    unsigned char rights ) [inline]
  
```

Create a kernel-object flex page.

#### Parameters

<i>obj</i>	Base capability selector.
<i>order</i>	Log2 size (number of capabilities).
<i>rights</i>	Access rights, see <a href="#">L4_cap_fpage_rights</a>

#### Returns

Flex page for a set of kernel objects.

#### Note

[L4\\_CAP\\_FPAGE\\_R](#) is always required, otherwise no capability is mapped.

#### Examples

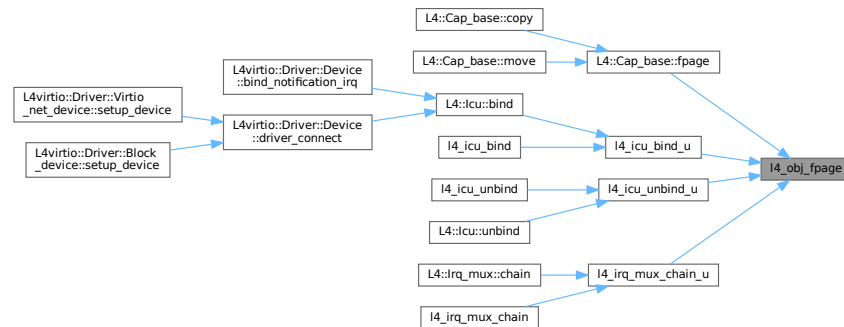
[examples/sys/utcb-ipc/main.c](#).

Definition at line 680 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_CAP\\_SHIFT](#), [L4\\_FPAGE\\_ADDR\\_SHIFT](#), and [L4\\_FPAGE\\_OBJ](#).

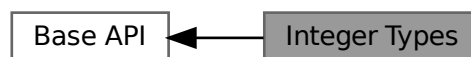
Referenced by [L4::Cap\\_base::fpage\(\)](#), [l4\\_icu\\_bind\\_u\(\)](#), [l4\\_icu\\_unbind\\_u\(\)](#), and [l4\\_irq\\_mux\\_chain\\_u\(\)](#).

Here is the caller graph for this function:



### 13.1.9 Integer Types

Collaboration diagram for Integer Types:



#### Files

- file [l4int.h](#)  
*Fixed sized integer types, generic version.*
- file [l4int.h](#)  
*Fixed sized integer types, arm version.*
- file [l4int.h](#)  
*Fixed sized integer types, amd64 version.*
- file [l4int.h](#)  
*Fixed sized integer types, x86 version.*

#### Macros

- **#define L4\_MWORD\_BITS 32**  
*Size of machine words in bits.*
- **#define L4\_MWORD\_BITS 64**  
*Size of machine words in bits.*
- **#define L4\_MWORD\_BITS 32**  
*Size of machine words in bits.*

## Typedefs

- typedef signed char **l4\_int8\_t**  
*Signed 8bit value.*
- typedef unsigned char **l4\_uint8\_t**  
*Unsigned 8bit value.*
- typedef signed short int **l4\_int16\_t**  
*Signed 16bit value.*
- typedef unsigned short int **l4\_uint16\_t**  
*Unsigned 16bit value.*
- typedef signed int **l4\_int32\_t**  
*Signed 32bit value.*
- typedef unsigned int **l4\_uint32\_t**  
*Unsigned 32bit value.*
- typedef signed long long **l4\_int64\_t**  
*Signed 64bit value.*
- typedef unsigned long long **l4\_uint64\_t**  
*Unsigned 64bit value.*
- typedef unsigned long **l4\_addr\_t**  
*Address type.*
- typedef signed long **l4\_mword\_t**  
*Signed machine word.*
- typedef unsigned long **l4\_umword\_t**  
*Unsigned machine word.*
- typedef [l4\\_uint64\\_t](#) **l4\_cpu\_time\_t**  
*CPU clock type.*
- typedef [l4\\_uint64\\_t](#) **l4\_kernel\_clock\_t**  
*Kernel clock type.*
- typedef unsigned int **l4\_size\_t**  
*Unsigned size type.*
- typedef signed int **l4\_ssize\_t**  
*Signed size type.*
- typedef unsigned long **l4\_size\_t**  
*Unsigned size type.*
- typedef signed long **l4\_ssize\_t**  
*Signed size type.*
- typedef unsigned int **l4\_size\_t**  
*Unsigned size type.*
- typedef signed int **l4\_ssize\_t**  
*Signed size type.*

### 13.1.9.1 Detailed Description

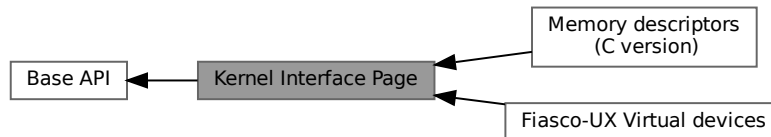
#### Include File

```
#include <l4/sys/l4int.h>
```

### 13.1.10 Kernel Interface Page

Kernel Interface Page.

Collaboration diagram for Kernel Interface Page:



#### Modules

- [Fiasco-UX Virtual devices](#)  
*Virtual hardware devices, provided by Fiasco-UX.*
- [Memory descriptors \(C version\)](#)  
*C Interface for KIP memory descriptors.*

#### Data Structures

- struct [l4\\_kernel\\_info\\_t](#)  
*L4 Kernel Interface Page.*
- class [L4::Kip::Mem\\_desc](#)  
*Memory descriptors stored in the kernel interface page.*

#### Macros

- `#define L4_KERNEL_INFO_MAGIC (0x4BE6344CL) /* "L4μK" */`  
*Kernel Info Page identifier ("L4μK").*

#### Typedefs

- typedef struct [l4\\_kernel\\_info\\_t](#) [l4\\_kernel\\_info\\_t](#)  
*L4 Kernel Interface Page.*
- typedef struct [l4\\_kernel\\_info\\_t](#) [l4\\_kernel\\_info\\_t](#)  
*L4 Kernel Interface Page.*

#### Enumerations

- enum { [L4\\_KIP\\_OFFSETS\\_READ\\_US](#) = 0x900 , [L4\\_KIP\\_OFFSETS\\_READ\\_NS](#) = 0x980 }

## Functions

- `l4_kernel_info_t` const \* `l4_kip` (void) `L4_NOTHROW`  
*Get Kernel Info Page.*
- `l4_umword_t` `l4_kip_version` (`l4_kernel_info_t` const \*kip) `L4_NOTHROW`  
*Get the kernel version.*
- const char \* `l4_kip_version_string` (`l4_kernel_info_t` const \*kip) `L4_NOTHROW`  
*Get the kernel version string.*
- int `l4_kernel_info_version_offset` (`l4_kernel_info_t` const \*kip) `L4_NOTHROW`  
*Return offset in bytes of version\_strings relative to the KIP base.*
- `l4_cpu_time_t` `l4_kip_clock` (`l4_kernel_info_t` const \*kip) `L4_NOTHROW`  
*Return clock value from the KIP.*
- `l4_umword_t` `l4_kip_clock_lw` (`l4_kernel_info_t` const \*kip) `L4_NOTHROW`  
*Return least significant machine word of clock value from the KIP.*
- `l4_uint64_t` `l4_kip_clock_ns` (`l4_kernel_info_t` const \*kip) `L4_NOTHROW`  
*Return current clock using the KIP in nanoseconds.*

### 13.1.10.1 Detailed Description

Kernel Interface Page.

C interface for the Kernel Interface Page:

C++ interface for the Kernel Interface Page:

#### Include File

```
#include <l4/sys/kip>
```

#### Include File

```
#include <l4/sys/kip.h>
```

### 13.1.10.2 Enumeration Type Documentation

#### 13.1.10.2.1 anonymous enum

anonymous enum

#### Enumerator

L4_KIP_OFFS_READ_US	Offset of KIP code (provided by the kernel) for reading the KIP clock in microseconds. If the kernel is configured for a fine-grained KIP clock (CONFIG_SYNC_TSC enabled for IA32, ARM_SYNC_CLOCK for ARM), this code provides the KIP clock with microseconds granularity and accuracy by reading the hardware clock used by the kernel and transforming this value into microseconds. Otherwise this code just reads the KIP clock value.
L4_KIP_OFFS_READ_NS	Offset of KIP code (provided by the kernel) for reading the time stamp counter and transforming this value into nanoseconds. If the kernel is configured for fine-grained KIP clock (CONFIG_SYNC enabled for IA32, ARM_SYNC_CLOCK for ARM), this code provides the KIP clock with nanoseconds granularity and accuracy by reading the hardware clock used by the kernel and transforming this value into nanoseconds. Otherwise this code just reads the KIP clock value and multiplies it by 1000.
Generated for L4Re by Doxygen	

Definition at line 64 of file [kip.h](#).

### 13.1.10.3 Function Documentation

#### 13.1.10.3.1 l4\_kernel\_info\_version\_offset()

```
int l4_kernel_info_version_offset (
    l4_kernel_info_t const * kip ) [inline]
```

Return offset in bytes of version\_strings relative to the KIP base.

##### Parameters

<i>kip</i>	Pointer to the kernel info page (KIP).
------------	--

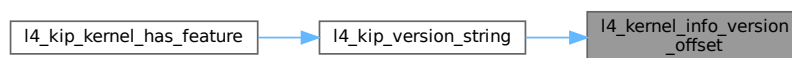
##### Returns

offset of version\_strings relative to the KIP base address, in bytes.

Definition at line 207 of file [kip.h](#).

Referenced by [l4\\_kip\\_version\\_string\(\)](#).

Here is the caller graph for this function:



#### 13.1.10.3.2 l4\_kip()

```
l4_kernel_info_t const * l4_kip (
    void ) [inline]
```

Get Kernel Info Page.

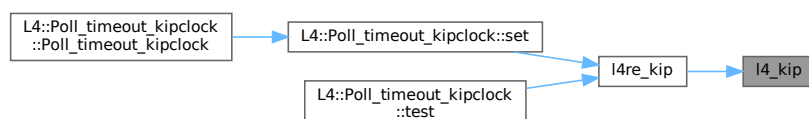
##### Returns

Pointer to Kernel Info Page (KIP) structure.

Definition at line 195 of file [kip.h](#).

Referenced by [l4re\\_kip\(\)](#).

Here is the caller graph for this function:





### 13.1.10.3.3 l4\_kip\_clock()

```
l4_cpu_time_t l4_kip_clock (
    l4_kernel_info_t const * kip ) [inline]
```

Return clock value from the KIP.

#### Parameters

<i>kip</i>	Pointer to the kernel info page (KIP).
------------	--

#### Returns

Value of the clock field in the KIP.

The KIP clock always contains the current (relative) time in micro seconds independently of the CPU frequency. The clock is only guaranteed to be accurate within the scheduling granularity announced in the KIP.

This function basically calls the KIP code for reading the KIP clock with microseconds resolution. The accuracy depends on the platform and the kernel configuration.

#### See also

[L4\\_KIP\\_OFFS\\_READ\\_US](#).

#### Examples

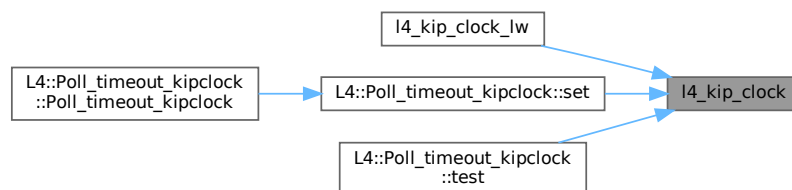
[examples/libs/shmc/prodcons.c](#).

Definition at line 211 of file [kip.h](#).

References [L4\\_KIP\\_OFFS\\_READ\\_US](#).

Referenced by [l4\\_kip\\_clock\\_lw\(\)](#), [L4::Poll\\_timeout\\_kipclock::set\(\)](#), and [L4::Poll\\_timeout\\_kipclock::test\(\)](#).

Here is the caller graph for this function:



### 13.1.10.3.4 l4\_kip\_clock\_lw()

```
l4_umword_t l4_kip_clock_lw (
    l4_kernel_info_t const * kip ) [inline]
```

Return least significant machine word of clock value from the KIP.

**Parameters**

<i>kip</i>	Pointer to the kernel info page (KIP).
------------	--

**Returns**

Lower machine word of clock value from the KIP.

**Deprecated** Use [l4\\_kip\\_clock\(\)](#) instead.

This function will always provide the least significant machine word of the clock value from the KIP, regardless of the kernel configuration.

Definition at line 230 of file [kip.h](#).

References [l4\\_kip\\_clock\(\)](#).

Here is the call graph for this function:

**13.1.10.3.5 l4\_kip\_clock\_ns()**

```
l4_cpu_time_t l4_kip_clock_ns (  
    l4_kernel_info_t const * kip ) [inline]
```

Return current clock using the KIP in nanoseconds.

**Parameters**

<i>kip</i>	Pointer to the kernel info page (KIP).
------------	--

**Returns**

Value of the current clock in nanoseconds.

This function basically calls the KIP code for reading the KIP clock with nanoseconds resolution. The accuracy depends on the platform and the kernel configuration.

See also

[L4\\_KIP\\_OFFS\\_READ\\_NS](#).

Definition at line 221 of file [kip.h](#).

References [L4\\_KIP\\_OFFS\\_READ\\_NS](#).

#### 13.1.10.3.6 l4\_kip\_version()

```
l4_umword_t l4_kip_version (
    l4_kernel_info_t const * kip ) [inline]
```

Get the kernel version.

Parameters

<i>kip</i>	Kernel Info Page.
------------	-------------------

Returns

Kernel version string. 0 if KIP could not be mapped.

Definition at line 199 of file [kip.h](#).

References [l4\\_kernel\\_info\\_t::version](#).

#### 13.1.10.3.7 l4\_kip\_version\_string()

```
const char * l4_kip_version_string (
    l4_kernel_info_t const * kip ) [inline]
```

Get the kernel version string.

Parameters

<i>kip</i>	Kernel Info Page.
------------	-------------------

Returns

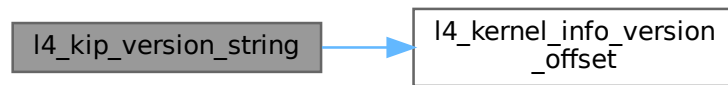
Kernel version string.

Definition at line 203 of file [kip.h](#).

References [l4\\_kernel\\_info\\_version\\_offset\(\)](#).

Referenced by [l4\\_kip\\_kernel\\_has\\_feature\(\)](#).

Here is the call graph for this function:



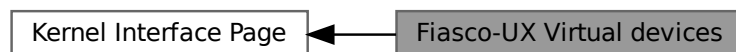
Here is the caller graph for this function:



#### 13.1.10.4 Fiasco-UX Virtual devices

Virtual hardware devices, provided by Fiasco-UX.

Collaboration diagram for Fiasco-UX Virtual devices:



#### Data Structures

- struct `l4_vhw_entry`  
*Description of a device.*
- struct `l4_vhw_descriptor`  
*Virtual hardware devices description.*

#### Enumerations

- enum `l4_vhw_entry_type` { `L4_TYPE_VHW_NONE`, `L4_TYPE_VHW_FRAMEBUFFER`, `L4_TYPE_VHW_INPUT`, `L4_TYPE_VHW_NET` }  
*Type of device.*

#### 13.1.10.4.1 Detailed Description

Virtual hardware devices, provided by Fiasco-UX.

##### Include File

```
#include <l4/sys/vhw.h>
```

#### 13.1.10.4.2 Enumeration Type Documentation

##### 13.1.10.4.2.1 l4\_vhw\_entry\_type

```
enum l4_vhw_entry_type
```

Type of device.

##### Enumerator

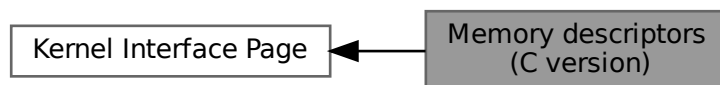
L4_TYPE_VHW_NONE	None entry.
L4_TYPE_VHW_FRAMEBUFFER	Framebuffer device.
L4_TYPE_VHW_INPUT	Input device.
L4_TYPE_VHW_NET	Network device.

Definition at line 44 of file [vhw.h](#).

#### 13.1.10.5 Memory descriptors (C version)

C Interface for KIP memory descriptors.

Collaboration diagram for Memory descriptors (C version):



#### Data Structures

- struct [l4\\_kernel\\_info\\_mem\\_desc\\_t](#)  
*Memory descriptor data structure.*

#### Typedefs

- typedef struct [l4\\_kernel\\_info\\_mem\\_desc\\_t](#) [l4\\_kernel\\_info\\_mem\\_desc\\_t](#)  
*Memory descriptor data structure.*

## Enumerations

- enum `l4_mem_type_t` {  
`l4_mem_type_undefined` = 0x0 , `l4_mem_type_conventional` = 0x1 , `l4_mem_type_reserved` = 0x2 ,  
`l4_mem_type_dedicated` = 0x3 ,  
`l4_mem_type_shared` = 0x4 , `l4_mem_type_info` = 0xd , `l4_mem_type_bootloader` = 0xe , `l4_mem_type_archspecific`  
= 0xf }  
*Type of a memory descriptor.*
- enum `l4_mem_info_sub_type_t` { `l4_mem_info_acpi_rsdp` = 0 }  
*Memory sub types for l4\_mem\_type\_info descriptors.*
- enum `l4_mem_archspecific_sub_type_common_t` { `l4_mem_archspecific_acpi_tables` = 3 , `l4_mem_archspecific_acpi_nvs`  
= 4 }  
*Memory sub types for l4\_mem\_type\_archspecific descriptors.*

## Functions

- `l4_kernel_info_mem_desc_t * l4_kernel_info_get_mem_descs (l4_kernel_info_t *kip) L4_NOTHROW`  
*Get pointer to memory descriptors from KIP.*
- unsigned `l4_kernel_info_get_num_mem_descs (l4_kernel_info_t *kip) L4_NOTHROW`  
*Get number of memory descriptors in KIP.*
- void `l4_kernel_info_set_mem_desc (l4_kernel_info_mem_desc_t *md, l4_addr_t start, l4_addr_t end, unsigned type, unsigned virt, unsigned sub_type) L4_NOTHROW`  
*Populate a memory descriptor.*
- `l4_umword_t l4_kernel_info_get_mem_desc_start (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`  
*Get start address of the region described by the memory descriptor.*
- `l4_umword_t l4_kernel_info_get_mem_desc_end (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`  
*Get end address of the region described by the memory descriptor.*
- `l4_umword_t l4_kernel_info_get_mem_desc_type (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`  
*Get type of the memory region.*
- `l4_umword_t l4_kernel_info_get_mem_desc_subtype (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`  
*Get sub-type of memory region.*
- `l4_umword_t l4_kernel_info_get_mem_desc_is_virtual (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`  
*Get virtual flag of the memory descriptor.*

### 13.1.10.5.1 Detailed Description

C Interface for KIP memory descriptors.

#### Include File

```
#include <l4/sys/memdesc.h>
```

This module contains the C functions to access the memory descriptor in the kernel interface page (KIP).

### 13.1.10.5.2 Typedef Documentation

#### 13.1.10.5.2.1 l4\_kernel\_info\_mem\_desc\_t

```
typedef struct l4_kernel_info_mem_desc_t l4_kernel_info_mem_desc_t
```

Memory descriptor data structure.

#### Note

This data type is opaque, and must be accessed by the accessor functions defined in this module.

### 13.1.10.5.3 Enumeration Type Documentation

#### 13.1.10.5.3.1 I4\_mem\_archspecific\_sub\_type\_common\_t

```
enum I4_mem_archspecific_sub_type_common_t
```

Memory sub types for I4\_mem\_type\_archspecific descriptors.

Enumerator

I4_mem_archspecific_acpi_tables	Firmware ACPI tables.
I4_mem_archspecific_acpi_nvs	Firmware reserved address space.

Definition at line 70 of file [memdesc.h](#).

#### 13.1.10.5.3.2 I4\_mem\_info\_sub\_type\_t

```
enum I4_mem_info_sub_type_t
```

Memory sub types for I4\_mem\_type\_info descriptors.

Enumerator

I4_mem_info_acpi_rsdp	Physical address of the ACPI root pointer.
-----------------------	--

Definition at line 61 of file [memdesc.h](#).

#### 13.1.10.5.3.3 I4\_mem\_type\_t

```
enum I4_mem_type_t
```

Type of a memory descriptor.

Enumerator

I4_mem_type_undefined	Undefined, unused descriptor.
I4_mem_type_conventional	Conventional memory.
I4_mem_type_reserved	Reserved memory for kernel etc.
I4_mem_type_dedicated	Dedicated memory (some device memory)
I4_mem_type_shared	Shared memory (not implemented)
I4_mem_type_info	Info from the boot loader.
I4_mem_type_bootloader	Memory owned by the boot loader.
I4_mem_type_archspecific	Architecture specific memory (e.g., ACPI memory)

Definition at line 44 of file [memdesc.h](#).

### 13.1.10.5.4 Function Documentation

#### 13.1.10.5.4.1 `l4_kernel_info_get_mem_desc_end()`

```
l4_umword_t l4_kernel_info_get_mem_desc_end (
    l4_kernel_info_mem_desc_t * md ) [inline]
```

Get end address of the region described by the memory descriptor.

##### Returns

End address.

Definition at line 227 of file [memdesc.h](#).

#### 13.1.10.5.4.2 `l4_kernel_info_get_mem_desc_is_virtual()`

```
l4_umword_t l4_kernel_info_get_mem_desc_is_virtual (
    l4_kernel_info_mem_desc_t * md ) [inline]
```

Get virtual flag of the memory descriptor.

##### Returns

1 if region is virtual memory, 0 if region is physical memory

Definition at line 248 of file [memdesc.h](#).

#### 13.1.10.5.4.3 `l4_kernel_info_get_mem_desc_start()`

```
l4_umword_t l4_kernel_info_get_mem_desc_start (
    l4_kernel_info_mem_desc_t * md ) [inline]
```

Get start address of the region described by the memory descriptor.

##### Returns

Start address.

Definition at line 220 of file [memdesc.h](#).

#### 13.1.10.5.4.4 `l4_kernel_info_get_mem_desc_subtype()`

```
l4_umword_t l4_kernel_info_get_mem_desc_subtype (
    l4_kernel_info_mem_desc_t * md ) [inline]
```

Get sub-type of memory region.

##### Returns

Sub-type.

The sub type is defined for architecture specific memory descriptors (see [l4\\_mem\\_type\\_archspecific](#)) and has architecture specific meaning.

Definition at line 241 of file [memdesc.h](#).



#### 13.1.10.5.4.5 l4\_kernel\_info\_get\_mem\_desc\_type()

```
l4_umword_t l4_kernel_info_get_mem_desc_type (
    l4_kernel_info_mem_desc_t * md ) [inline]
```

Get type of the memory region.

##### Returns

Type of the region (see [l4\\_mem\\_type\\_t](#)).

Definition at line 234 of file [memdesc.h](#).

#### 13.1.10.5.4.6 l4\_kernel\_info\_get\_num\_mem\_descs()

```
unsigned l4_kernel_info_get_num_mem_descs (
    l4_kernel_info_t * kip ) [inline]
```

Get number of memory descriptors in KIP.

##### Returns

Number of memory descriptors.

Definition at line 198 of file [memdesc.h](#).

#### 13.1.10.5.4.7 l4\_kernel\_info\_set\_mem\_desc()

```
void l4_kernel_info_set_mem_desc (
    l4_kernel_info_mem_desc_t * md,
    l4_addr_t start,
    l4_addr_t end,
    unsigned type,
    unsigned virt,
    unsigned sub_type ) [inline]
```

Populate a memory descriptor.

##### Parameters

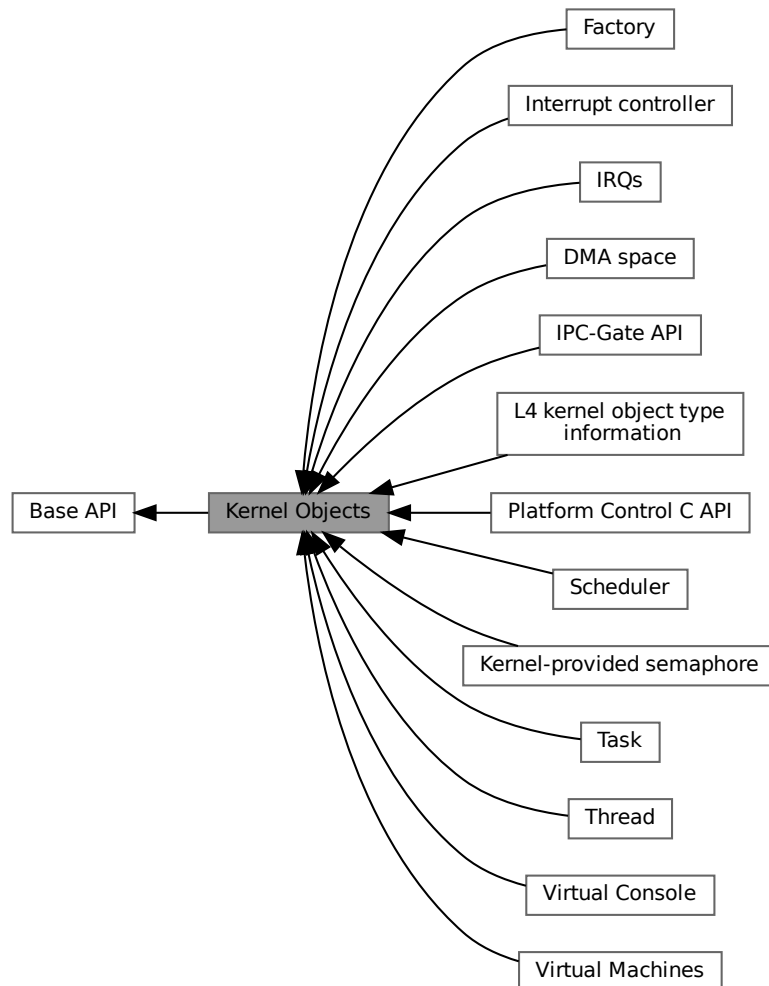
<i>md</i>	Pointer to memory descriptor
<i>start</i>	Start of region
<i>end</i>	End of region
<i>type</i>	Type of region
<i>virt</i>	1 if virtual region, 0 if physical region
<i>sub_type</i>	Sub type.

Definition at line 205 of file [memdesc.h](#).

### 13.1.11 Kernel Objects

API of kernel objects.

Collaboration diagram for Kernel Objects:



#### Modules

- [DMA space](#)

A DMA space represents a device memory address space managed by an IOMMU.

- [Factory](#)

C factory interface to create objects, see [L4::Factory](#) for the C++ interface.

- [IPC-Gate API](#)

The C IPC gate interface, see [L4::lpc\\_gate](#) for the C++ interface.

- [IRQs](#)

C IRQ interface, see [L4::irq](#) for the C++ interface.

- [Interrupt controller](#)

- The C Icu interface, see [L4::Icu](#) for the C++ interface.
- [Kernel-provided semaphore](#)

C semaphore interface, see [L4::Semaphore](#) for the C++ interface.
- [L4 kernel object type information](#)

Type information for [L4](#) server objects that can be called via IPC.
- [Platform Control C API](#)

C interface for controlling platform-wide properties, see [L4::Platform\\_control](#) for the C++ interface.
- [Scheduler](#)

C interface of the Scheduler kernel object, see [L4::Scheduler](#) for the C++ interface.
- [Task](#)

C interface of the Task kernel object, see [L4::Task](#) for the C++ interface.
- [Thread](#)

C Thread object interface, see [L4::Thread](#) for the C++ interface.
- [Virtual Console](#)

C Virtual console interface for simple character based input and output, see [L4::Vcon](#) for the C++ interface.
- [Virtual Machines](#)

Virtual Machine API.

## Data Structures

- class [L4::Kobject](#)

Base class for all kinds of kernel objects and remote objects, referenced by capabilities.
- class [L4::Vm](#)

Virtual machine host address space.

### 13.1.11.1 Detailed Description

API of kernel objects.

#### Include File

```
#include <l4/sys/kernel_object.h>
```

### 13.1.11.2 DMA space

A DMA space represents a device memory address space managed by an IOMMU.

Collaboration diagram for DMA space:



A DMA space represents a device memory address space managed by an IOMMU.

That is, it manages the translation of virtual addresses used by devices to physical addresses. It is accessed via the [L4::Task](#) interface, but with the following caveats:

- No threads can be bound to it.
- No objects (and IO ports on IA32) can be mapped to it.
- No kernel-user memory can be added to it.
- It must be constructed by passing the `L4_PROTO_DMA_SPACE` protocol constant to the kernel factory's `L4::Factory.create()` call.

A DMA space must be bound to an `L4::iommu` to enable the address translation for specific devices.

The kernel factory allows to create DMA spaces only if the kernel has been configured with IOMMU support and if an IOMMU was detected.

### 13.1.11.3 Factory

C factory interface to create objects, see `L4::Factory` for the C++ interface.

Collaboration diagram for Factory:



### Functions

- `l4_msgtag_t l4_factory_create_task (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_fpage_t *utcb_area) L4_NOTHROW`  
*Create a new task.*
- `l4_msgtag_t l4_factory_create_thread (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW`  
*Create a new thread.*
- `l4_msgtag_t l4_factory_create_factory (l4_cap_idx_t factory, l4_cap_idx_t target_cap, unsigned long limit) L4_NOTHROW`  
*Create a new factory.*
- `l4_msgtag_t l4_factory_create_gate (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_cap_idx_t thread_cap, l4_umword_t label) L4_NOTHROW`  
*Create a new IPC gate.*
- `l4_msgtag_t l4_factory_create_irq (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW`  
*Create a new IRQ sender.*
- `l4_msgtag_t l4_factory_create_vm (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW`  
*Create a new virtual machine.*
- `l4_msgtag_t l4_factory_create_vcpu_context (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW`  
*Create a new hardware vCPU context.*
- `l4_msgtag_t l4_factory_create (l4_cap_idx_t factory, long obj, l4_cap_idx_t target) L4_NOTHROW`  
*Create a new object.*

### 13.1.11.3.1 Detailed Description

C factory interface to create objects, see [L4::Factory](#) for the C++ interface.

A factory is used to create all kinds of kernel objects:

- [Task](#)
- [Thread](#)
- [Factory](#)
- [IPC-Gate API](#)
- [IRQs](#)
- [Virtual Machines](#)

To create a new kernel object the caller has to specify the factory to use for creation. The caller has to allocate a capability slot where the kernel stores the new object's capability.

The factory is equipped with a limit that limits the amount of kernel memory available for that factory.

#### Note

The limit does not give any guarantee for the amount of available kernel memory.

#### Include File

```
#include <l4/sys/factory.h>
```

For the C++ interface refer to [L4::Factory](#).

### 13.1.11.3.2 Function Documentation

#### 13.1.11.3.2.1 l4\_factory\_create()

```
l4_msgtag_t l4_factory_create (
    l4_cap_idx_t factory,
    long obj,
    l4_cap_idx_t target ) [inline]
```

Create a new object.

#### Parameters

	<i>factory</i>	Factory to use for creation.
	<i>obj</i>	Protocol ID to describe the type of the object to create.
out	<i>target</i>	The kernel stores the new objects's capability into this slot.

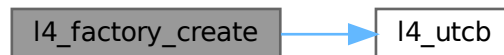
## Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	The factory instance requires <a href="#">L4_CAP_FPAGE_S</a> rights on <code>factory</code> and <a href="#">L4_CAP_FPAGE_S</a> is not present.
<i>&lt;0</i>	Error code.

Definition at line 648 of file [factory.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 13.1.11.3.2.2 l4\_factory\_create\_factory()

```

l4_msgtag_t l4_factory_create_factory (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    unsigned long limit ) [inline]
  
```

Create a new factory.

## Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new factory's capability into this slot.
	<i>limit</i>	Limit for the new factory in bytes.

## Returns

Syscall return tag

## Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	The factory instance requires <a href="#">L4_CAP_FPAGE_S</a> rights on <code>factory</code> and <a href="#">L4_CAP_FPAGE_S</a> is not present.
<i>&lt;0</i>	Error code.

**Note**

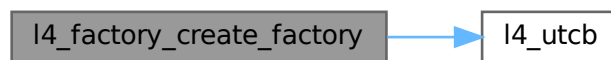
The limit of the new factory is subtracted from the available amount of the factory used for creation.

This method is only guaranteed to work with the [Kernel Factory](#). For other services, use the generic [L4::Factory::create\(\)](#) method and consult the service documentation for information on the arguments that need to be passed to the create stream.

Definition at line 495 of file [factory.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**13.1.11.3.2.3 l4\_factory\_create\_gate()**

```

l4_msgtag_t l4_factory_create_gate (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_cap_idx_t thread_cap,
    l4_umword_t label ) [inline]
  
```

Create a new IPC gate.

**Parameters**

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new IPC gate's capability into this slot.
	<i>thread_cap</i>	Optional capability selector of a thread to bind the gate to. Use <a href="#">L4_INVALID_CAP</a> to create an unbound IPC gate.
	<i>label</i>	Optional label of the gate (precisely used if <i>thread_cap</i> is valid). If <i>thread_cap</i> is valid, <i>label</i> must be present.

**Returns**

Syscall return tag containing one of the following return codes.

**Return values**

<i>L4_EOK</i>	No error occurred.
<i>-L4_ENOMEM</i>	Out-of-memory during allocation of the <i>lpc_gate</i> object.
<i>-L4_EINVAL</i>	<i>thread_cap</i> is void or points to something that is not a thread.
<i>-L4_EPERM</i>	No <a href="#">L4_CAP_FPAGE_S</a> rights on <i>factory</i> or <i>thread_cap</i> .

An unbound IPC gate can be bound to a thread using [l4\\_rcv\\_ep\\_bind\\_thread\(\)](#).

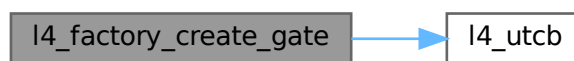
See also

[IPC-Gate API](#)

Definition at line 503 of file [factory.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.11.3.2.4 l4\_factory\_create\_irq()

```

l4_msgtag_t l4_factory_create_irq (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap ) [inline]
  
```

Create a new IRQ sender.

Parameters

	<i>factory</i>	Factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new IRQ's capability into this slot.

Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	The factory instance requires <a href="#">L4_CAP_FPAGE_S</a> rights on <i>factory</i> and <a href="#">L4_CAP_FPAGE_S</a> is not present.
<i>&lt;0</i>	Error code.

See also

[IRQs](#)

Examples

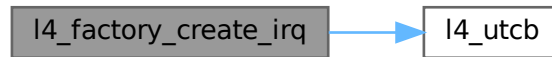
[examples/sys/isr/main.c](#).



Definition at line 511 of file [factory.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.11.3.2.5 l4\_factory\_create\_task()

```

l4_msgtag_t l4_factory_create_task (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_fpage_t * utcb_area ) [inline]
  
```

Create a new task.

##### Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new task's capability into this slot.
in, out	<i>utcb_area</i>	Pointer to flexpage that describes an area of kernel-user memory that can be used for UTCBs and vCPU state-save-areas of the new task.

On systems without MMU, the flexpage is adjusted to reflect the actually allocated physical address.

##### Returns

Syscall return tag.

##### Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	The factory instance requires <a href="#">L4_CAP_FPAGE_S</a> rights on <i>factory</i> and <a href="#">L4_CAP_FPAGE_S</a> is not present.
<i>&lt;0</i>	Error code.

##### Note

The size of the UTCB area specifies indirectly the number of UTCBs available for this task. Refer to [l4\\_task\\_add\\_ku\\_mem\(\)](#) for adding more of this type of memory.

See also

[Task](#)

Definition at line 481 of file [factory.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.11.3.2.6 l4\_factory\_create\_thread()

```

l4_msgtag_t l4_factory_create_thread (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap ) [inline]
  
```

Create a new thread.

Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new thread's capability into this slot.

Returns

Syscall return tag

Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	The factory instance requires <a href="#">L4_CAP_FPAGE_S</a> rights on <i>factory</i> and <a href="#">L4_CAP_FPAGE_S</a> is not present.
<i>&lt;0</i>	Error code.

See also

[Thread](#)

Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 488 of file [factory.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.11.3.2.7 l4\_factory\_create\_vcpu\_context()

```

l4_msgtag_t l4_factory_create_vcpu_context (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap ) [inline]
  
```

Create a new hardware vCPU context.

A hardware vCPU context typically represents a hardware vCPU control structure (e.g. VMX VMCS).

##### Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new hardware vCPU context's capability into this slot.

##### Returns

Syscall return tag

##### Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	The factory instance requires <a href="#">L4_CAP_FPAGE_S</a> rights on <i>factory</i> and <a href="#">L4_CAP_FPAGE_S</a> is not present.
<i>&lt;0</i>	Error code.

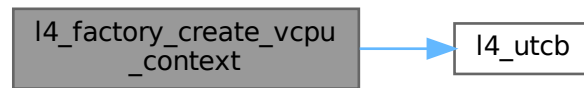
##### See also

[Virtual Machines](#)

Definition at line 525 of file [factory.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 13.1.11.3.2.8 l4\_factory\_create\_vm()

```
l4_msgtag_t l4_factory_create_vm (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap ) [inline]
```

Create a new virtual machine.

#### Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new VM's capability into this slot.

#### Returns

Syscall return tag

#### Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	The factory instance requires <a href="#">L4_CAP_FPAGE_S</a> rights on <i>factory</i> and <a href="#">L4_CAP_FPAGE_S</a> is not present.
<i>&lt;0</i>	Error code.

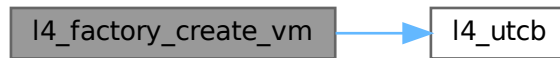
#### See also

[Virtual Machines](#)

Definition at line 518 of file [factory.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.11.4 IPC-Gate API

The C IPC gate interface, see [L4::lpc\\_gate](#) for the C++ interface.

Collaboration diagram for IPC-Gate API:



#### Functions

- [l4\\_msgtag\\_t l4\\_ipc\\_gate\\_get\\_infos](#) ([l4\\_cap\\_idx\\_t](#) gate, [l4\\_umword\\_t](#) \*label)  
*Get information about the IPC-gate.*
- [l4\\_msgtag\\_t l4\\_rcv\\_ep\\_bind\\_thread](#) ([l4\\_cap\\_idx\\_t](#) ep, [l4\\_cap\\_idx\\_t](#) thread, [l4\\_umword\\_t](#) label)  
*Bind the IPC receive endpoint to a thread.*

##### 13.1.11.4.1 Detailed Description

The C IPC gate interface, see [L4::lpc\\_gate](#) for the C++ interface.

IPC gates are used to create secure communication channels between protection domains. An IPC gate can be created using the [Factory](#) interface.

Depending on the permissions of the capability used, an IPC gate forwards IPC to the [Thread](#) that is *bound* to the IPC gate (cf. [l4\\_rcv\\_ep\\_bind\\_thread\(\)](#)). If the capability has the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission, only IPC using a protocol different from the [L4\\_PROTO\\_KOBJECT](#) protocol is forwarded. Without the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission, all IPC is forwarded. The latter is the usual case for a client in a client/server scenario. When no thread is bound yet, the forwarded IPC blocks until a thread is bound or the IPC times out.

Forwarded IPC is always forwarded to the userland of the bound thread. That means, the [Thread](#) interface of the bound thread is not accessible via an IPC gate. The [IPC-Gate API](#) of an IPC gate is only accessible if the capability used has the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission (cf. previous paragraph). Conversely that means, if the capability used lacks the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission, [IPC-Gate API](#) calls are forwarded to the bound

thread instead of being processed by the IPC gate itself. In a client/server scenario, a client should only get IPC gate capabilities without [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission so the client cannot tamper with the IPC gate.

When binding a thread to an IPC gate, a user-defined, kernel protected, machine-word sized payload called the IPC gate's *label* is assigned to the IPC gate (cf. [l4\\_rcv\\_ep\\_bind\\_thread\(\)](#)). When a send-only IPC or call IPC is forwarded via an IPC gate, the label provided by the sender is ignored and replaced by the IPC gate's label where the two least significant bits are the result of bitwise disjunction of the corresponding label bits with the [L4\\_CAP\\_FPAGE\\_S](#) and [L4\\_CAP\\_FPAGE\\_W](#) permissions of the capability used. Hence, the label provided via [l4\\_rcv\\_ep\\_bind\\_thread\(\)](#) should usually have its two least significant bits set to zero. The replaced label is only visible to the bound thread upon receive. However, the configured label of an IPC gate can also be queried via [l4\\_ipc\\_gate\\_get\\_infos\(\)](#) if the capability used has the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission.

When deleting an IPC gate or when unbinding it from a thread, the label of IPC already in flight won't be changed. To ensure that no IPC from this IPC gate is received by a thread with an unexpected label, [l4\\_thread\\_modify\\_sender\\_start\(\)](#) shall be used to change the labels of every pending IPC to that gate. This is also required if the label of an already bound IPC gate is changed. It is not necessary after binding the IPC gate to a thread for the first time.

When binding a new thread to an IPC gate that is currently bound, the same label should be used that was used with the old thread. Otherwise the old and the new thread need to synchronize to avoid IPC messages with unexpected labels.

#### Include File

```
#include <l4/sys/ipc_gate.h>
```

For the C++ interface refer to the [L4::ipc\\_gate](#) documentation.

#### See also

[Object Invocation](#)

### 13.1.11.4.2 Function Documentation

#### 13.1.11.4.2.1 l4\_ipc\_gate\_get\_infos()

```
l4_msgtag_t l4_ipc_gate_get_infos (
    l4_cap_idx_t gate,
    l4_umword_t * label ) [inline]
```

Get information about the IPC-gate.

#### Parameters

	<i>gate</i>	The IPC gate object to get information about.
out	<i>label</i>	The label of the IPC gate is returned here.

#### Returns

System call return tag.

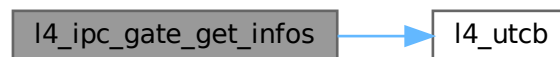
**Precondition**

If `gate` does not possess the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) right, the kernel will not perform this operation. Instead, the underlying IPC message will be forwarded to the thread bound to the IPC gate, blocking the caller if no thread is bound yet.

Definition at line 159 of file [ipc\\_gate.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**13.1.11.4.2.2 l4\_rcv\_ep\_bind\_thread()**

```

l4_msgtag_t l4_rcv_ep_bind_thread (
    l4_cap_idx_t ep,
    l4_cap_idx_t thread,
    l4_umword_t label ) [inline]
  
```

Bind the IPC receive endpoint to a thread.

**Parameters**

<i>ep</i>	The IPC receive endpoint object.
<i>thread</i>	The thread object that shall be bound to <i>ep</i> .
<i>label</i>	Label to assign to <i>ep</i> . The two least significant bits should usually be set to zero.

**Returns**

Syscall return tag containing one of the following return codes.

**Return values**

<i>L4_EOK</i>	Operation successful.
<i>-L4_EINVAL</i>	<i>thread</i> is not a thread object or other arguments were malformed.
<i>-L4_EPERM</i>	No <a href="#">L4_CAP_FPAGE_S</a> right on <i>ep</i> or <i>thread</i> .

**Precondition**

If `ep` is an IPC gate capability without the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) right, the kernel will not perform this operation. Instead, the underlying IPC message will be forwarded to the thread bound to the IPC gate, blocking the caller if no thread is bound yet.

The specified `label` is passed to the receiver of the incoming IPC. It is possible to re-bind a receive endpoint to the same or a different thread. In this case, IPC already in flight will be delivered with the old label to the previously bound thread unless [l4\\_thread\\_modify\\_sender\\_start\(\)](#) is used to change these labels.

**Examples**

[examples/sys/isr/main.c](#).

Definition at line 91 of file [rcv\\_endpoint.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**13.1.11.5 IRQs**

C IRQ interface, see [L4::Irq](#) for the C++ interface.

Collaboration diagram for IRQs:

**Enumerations**

- enum [L4\\_irq\\_mode](#) {  
[L4\\_IRQ\\_F\\_NONE](#) = 0 , [L4\\_IRQ\\_F\\_LEVEL](#) = 0x2 , [L4\\_IRQ\\_F\\_EDGE](#) = 0x0 , [L4\\_IRQ\\_F\\_POS](#) = 0x0 ,  
[L4\\_IRQ\\_F\\_NEG](#) = 0x4 , [L4\\_IRQ\\_F\\_BOTH](#) = 0x8 , [L4\\_IRQ\\_F\\_LEVEL\\_HIGH](#) = 0x3 , [L4\\_IRQ\\_F\\_LEVEL\\_LOW](#)  
= 0x7 ,  
[L4\\_IRQ\\_F\\_POS\\_EDGE](#) = 0x1 , [L4\\_IRQ\\_F\\_NEG\\_EDGE](#) = 0x5 , [L4\\_IRQ\\_F\\_BOTH\\_EDGE](#) = 0x9 ,  
[L4\\_IRQ\\_F\\_MASK](#) = 0xf ,  
[L4\\_IRQ\\_F\\_SET\\_WAKEUP](#) = 0x10 , [L4\\_IRQ\\_F\\_CLEAR\\_WAKEUP](#) = 0x20 }

*Interrupt attributes.*



## Functions

- [l4\\_msgtag\\_t l4\\_irq\\_mux\\_chain](#) ([l4\\_cap\\_idx\\_t](#) irq, [l4\\_cap\\_idx\\_t](#) slave) [L4\\_NOTHROW](#)  
*Chain an IRQ to another master IRQ source.*
- [l4\\_msgtag\\_t l4\\_irq\\_mux\\_chain\\_u](#) ([l4\\_cap\\_idx\\_t](#) irq, [l4\\_cap\\_idx\\_t](#) slave, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Attach an IRQ to this multiplexer.*
- [l4\\_msgtag\\_t l4\\_irq\\_detach](#) ([l4\\_cap\\_idx\\_t](#) irq) [L4\\_NOTHROW](#)  
*Detach from an interrupt source.*
- [l4\\_msgtag\\_t l4\\_irq\\_detach\\_u](#) ([l4\\_cap\\_idx\\_t](#) irq, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Detach from this interrupt.*
- [l4\\_msgtag\\_t l4\\_irq\\_trigger](#) ([l4\\_cap\\_idx\\_t](#) irq) [L4\\_NOTHROW](#)  
*Trigger an IRQ.*
- [l4\\_msgtag\\_t l4\\_irq\\_trigger\\_u](#) ([l4\\_cap\\_idx\\_t](#) irq, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Trigger the object.*
- [l4\\_msgtag\\_t l4\\_irq\\_receive](#) ([l4\\_cap\\_idx\\_t](#) irq, [l4\\_timeout\\_t](#) to) [L4\\_NOTHROW](#)  
*Unmask and wait for specified IRQ.*
- [l4\\_msgtag\\_t l4\\_irq\\_receive\\_u](#) ([l4\\_cap\\_idx\\_t](#) irq, [l4\\_timeout\\_t](#) timeout, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Unmask and wait for this IRQ.*
- [l4\\_msgtag\\_t l4\\_irq\\_wait](#) ([l4\\_cap\\_idx\\_t](#) irq, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) to) [L4\\_NOTHROW](#)  
*Unmask IRQ and wait for any message.*
- [l4\\_msgtag\\_t l4\\_irq\\_wait\\_u](#) ([l4\\_cap\\_idx\\_t](#) irq, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) timeout, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Unmask IRQ and (open) wait for any message.*
- [l4\\_msgtag\\_t l4\\_irq\\_unmask](#) ([l4\\_cap\\_idx\\_t](#) irq) [L4\\_NOTHROW](#)  
*Unmask IRQ.*
- [l4\\_msgtag\\_t l4\\_irq\\_unmask\\_u](#) ([l4\\_cap\\_idx\\_t](#) irq, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Unmask this IRQ.*

### 13.1.11.5.1 Detailed Description

C IRQ interface, see [L4::Irq](#) for the C++ interface.

The IRQ interface provides access to abstract interrupts provided by the microkernel. Interrupts may be

- hardware interrupts provided by the platform interrupt controller,
- virtual device interrupts provided by the microkernel's virtual devices (virtual serial or trace buffer) or
- virtual interrupts that can be triggered by user programs (IRQs) via [l4\\_irq\\_trigger\(\)](#).

For hardware and virtual device interrupts the `Irq` object must be bound to an interrupt source, see [Interrupt controller](#). To receive interrupts, the `Irq` object must be bound to a thread, see [l4\\_rcv\\_ep\\_bind\\_thread\(\)](#).

IRQ objects can be created using a factory, see the [Factory](#) API (use [l4\\_factory\\_create\\_irq\(\)](#)).

#### Include File

```
#include <l4/sys/irq.h>
```

For the C++ interface refer to the [L4::Irq](#) API for an overview.

### 13.1.11.5.2 Enumeration Type Documentation

#### 13.1.11.5.2.1 L4\_irq\_mode

```
enum L4_irq_mode
```

Interrupt attributes.

## Enumerator

L4_IRQ_F_NONE	Flow types. None
L4_IRQ_F_LEVEL	Level triggered.
L4_IRQ_F_EDGE	Edge triggered.
L4_IRQ_F_POS	Positive trigger.
L4_IRQ_F_NEG	Negative trigger.
L4_IRQ_F_BOTH	Both edges trigger.
L4_IRQ_F_LEVEL_HIGH	Level high trigger.
L4_IRQ_F_LEVEL_LOW	Level low trigger.
L4_IRQ_F_POS_EDGE	Positive edge trigger.
L4_IRQ_F_NEG_EDGE	Negative edge trigger.
L4_IRQ_F_BOTH_EDGE	Both edges trigger.
L4_IRQ_F_MASK	Mask.
L4_IRQ_F_SET_WAKEUP	Wakeup source? Use irq as wakeup source
L4_IRQ_F_CLEAR_WAKEUP	Do not use irq as wakeup source.

Definition at line 80 of file [icu.h](#).

## 13.1.11.5.3 Function Documentation

## 13.1.11.5.3.1 l4\_irq\_detach()

```
l4_msgtag_t l4_irq_detach (
    l4_cap_idx_t irq ) [inline]
```

Detach from an interrupt source.

## Parameters

<i>irq</i>	The IRQ object that shall be detached.
------------	--

## Returns

Syscall return tag

## Return values

0	Successfully detached, there was no interrupt pending.
1	Successfully detached, there was an interrupt pending.
-L4_EPERM	No <a href="#">L4_CAP_FPAGE_S</a> rights on the capability used to invoke this operation.

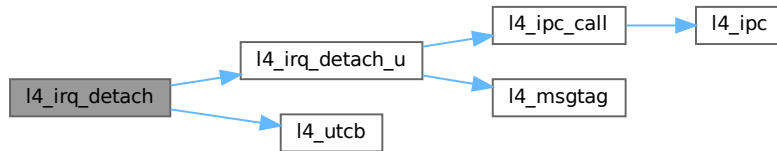
## Examples

[examples/sys/isr/main.c](#).

Definition at line 299 of file [irq.h](#).

References [l4\\_irq\\_detach\\_u\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 13.1.11.5.3.2 l4\_irq\_detach\_u()

```

l4_msgtag_t l4_irq_detach_u (
    l4_cap_idx_t irq,
    l4_utcb_t * utcb ) [inline]
  
```

Detach from this interrupt.

#### Parameters

<i>irq</i>	The IRQ object that shall be detached.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

#### Returns

Syscall return tag

#### Return values

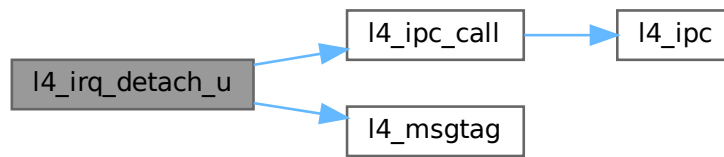
0	Successfully detached, there was no interrupt pending.
1	Successfully detached, there was an interrupt pending.
-L4_EPERM	No <a href="#">L4_CAP_FPAGE_S</a> rights on the capability used to invoke this operation.

Definition at line 254 of file [irq.h](#).

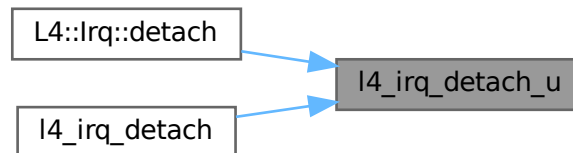
References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_IRQ\\_SENDER](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [L4::Irq::detach\(\)](#), and [l4\\_irq\\_detach\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.11.5.3.3 l4\_irq\_mux\_chain()

```

l4_msgtag_t l4_irq_mux_chain (
    l4_cap_idx_t irq,
    l4_cap_idx_t slave ) [inline]
  
```

Chain an IRQ to another master IRQ source.

**Deprecated** IRQ muxer objects are no longer supported by the kernel.

##### Parameters

<i>irq</i>	The master IRQ object.
<i>slave</i>	The slave that shall be attached to the master.

##### Returns

Syscall return tag

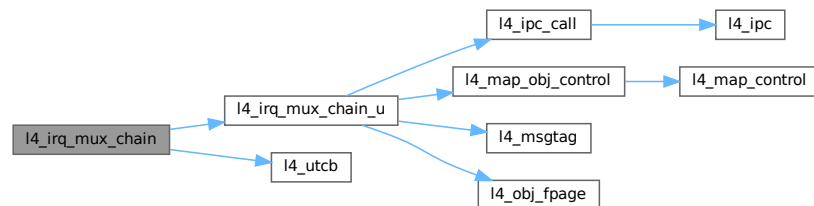
The chaining feature of IRQ objects allows to deal with shared IRQs. For chaining IRQs there must be a master IRQ object, bound to the real IRQ source. Note, the master IRQ must not have a thread bound to it.

This function allows to add a limited number of slave IRQs to this master IRQ, with the semantics that each of the slave IRQs is triggered whenever the master IRQ is triggered. The master IRQ will be masked automatically when an IRQ is delivered and shall be unmasked when all attached slave IRQs are unmasked.

Definition at line 293 of file [irq.h](#).

References [l4\\_irq\\_mux\\_chain\\_u\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.11.5.3.4 l4\_irq\_mux\_chain\_u()

```

l4_msgtag_t l4_irq_mux_chain_u (
    l4_cap_idx_t irq,
    l4_cap_idx_t slave,
    l4_utcb_t * utcb ) [inline]
  
```

Attach an IRQ to this multiplexer.

##### Parameters

<i>irq</i>	The master IRQ object.
<i>slave</i>	The slave that shall be attached to the master.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

##### Returns

Syscall return tag

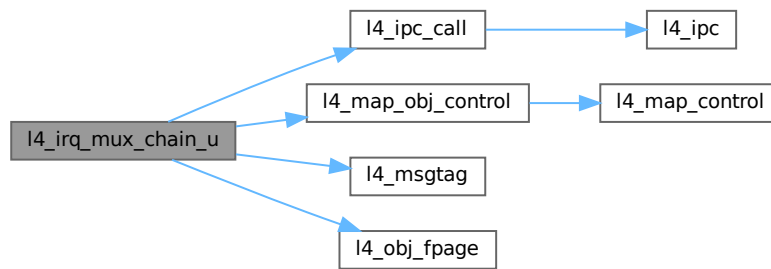
The chaining feature of IRQ objects allows to deal with shared IRQs. For chaining IRQs there must be an IRQ multiplexer ([l4\\_irq\\_mux](#)) bound to the real IRQ source. This function allows to add slave IRQs to this multiplexer.

Definition at line 242 of file [irq.h](#).

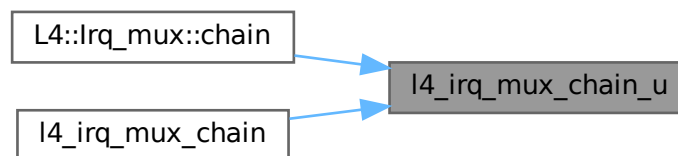
References [L4\\_CAP\\_FPAGE\\_RWS](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_map\\_obj\\_control\(\)](#), [l4\\_msgtag\(\)](#), [l4\\_obj\\_fpage\(\)](#), [L4\\_PROTO\\_IRQ\\_MUX](#), [l4\\_msg\\_regs\\_t::mr](#), and [l4\\_fpage\\_t::raw](#).

Referenced by [L4::l4\\_irq\\_mux::chain\(\)](#), and [l4\\_irq\\_mux\\_chain\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.11.5.3.5 l4\_irq\_receive()

```

l4_msgtag_t l4_irq_receive (
    l4_cap_idx_t irq,
    l4_timeout_t to ) [inline]
  
```

Unmask and wait for specified IRQ.

##### Parameters

<i>irq</i>	The IRQ object that shall be unmasked.
<i>to</i>	Timeout.

##### Returns

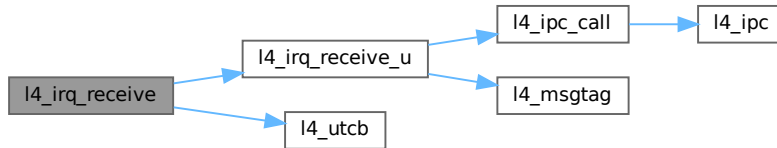
Syscall return tag

Definition at line 311 of file `irq.h`.

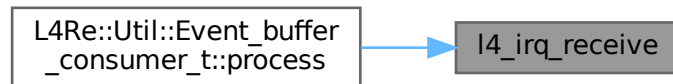
References `l4_irq_receive_u()`, and `l4_utcb()`.

Referenced by [L4Re::Util::Event\\_buffer\\_consumer\\_t< PAYLOAD >::process\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.11.5.3.6 l4\_irq\_receive\_u()

```

l4_msgtag_t l4_irq_receive_u (
    l4_cap_idx_t irq,
    l4_timeout_t timeout,
    l4_utcb_t * utcb ) [inline]
  
```

Unmask and wait for this IRQ.

##### Parameters

<i>irq</i>	The IRQ object that shall be unmasked.
<i>timeout</i>	Timeout.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

##### Returns

Syscall return tag

##### Note

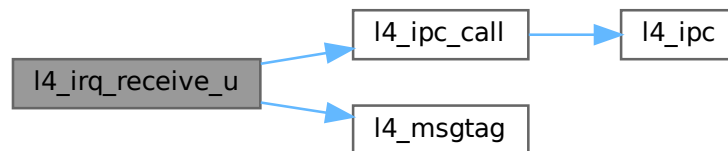
If this is the function normally used for your IRQs consider using [L4::Semaphore](#) instead of [L4::Irq](#).

Definition at line 269 of file [irq.h](#).

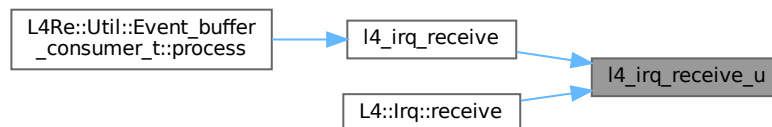
References [l4\\_ipc\\_call\(\)](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_IRQ](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [l4\\_irq\\_receive\(\)](#), and [L4::Irq::receive\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.11.5.3.7 l4\_irq\_trigger()

```
l4_msgtag_t l4_irq_trigger (
    l4_cap_idx_t irq ) [inline]
```

Trigger an IRQ.

##### Parameters

<i>irq</i>	The IRQ object that shall be triggered.
------------	---

##### Returns

Syscall return tag.

Note that this function is a send only operation, i.e. there is no return value except for a failed send operation. Especially [l4\\_error\(\)](#) will return an error value from the message tag which still contains the IRQ protocol used for the send operation.



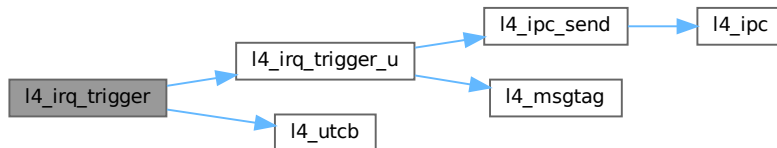
Use [l4\\_ipc\\_error\(\)](#) to check for (send) errors.

Definition at line 305 of file [irq.h](#).

References [l4\\_irq\\_trigger\\_u\(\)](#), and [l4\\_utcb\(\)](#).

Referenced by [l4\\_semaphore\\_up\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.11.5.3.8 l4\_irq\_trigger\_u()

```

l4_msgtag_t l4_irq_trigger_u (
    l4_cap_idx_t irq,
    l4_utcb_t * utcb ) [inline]
  
```

Trigger the object.

##### Parameters

<i>irq</i>	The IRQ object that shall be triggered.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

##### Returns

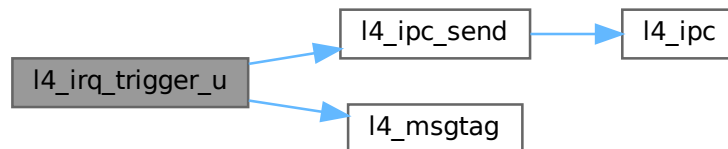
Syscall return tag for a send-only operation, this means there is no return value except [L4\\_MSGTAG\\_ERROR](#) indicating success or failure of the send operation. Use [l4\\_ipc\\_error\(\)](#) to check for errors and **do not** use [l4\\_error\(\)](#).

Definition at line 262 of file [irq.h](#).

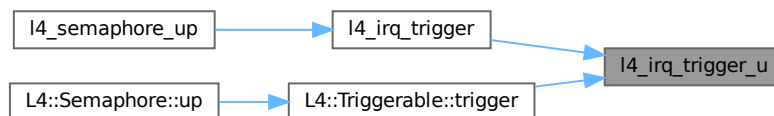
References [L4\\_IPC\\_BOTH\\_TIMEOUT\\_0](#), [l4\\_ipc\\_send\(\)](#), [l4\\_msgtag\(\)](#), and [L4\\_PROTO\\_IRQ](#).

Referenced by [l4\\_irq\\_trigger\(\)](#), and [L4::Triggerable::trigger\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.11.5.3.9 l4\_irq\_unmask()

```
l4_msgtag_t l4_irq_unmask (
    l4_cap_idx_t irq ) [inline]
```

Unmask IRQ.

##### Parameters

<i>irq</i>	The IRQ object that shall be unmasked.
------------	--

##### Returns

Syscall return tag

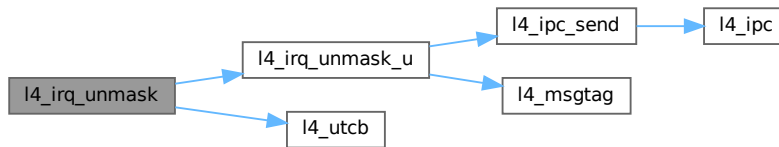
##### Note

[l4\\_irq\\_wait\(\)](#) and [l4\\_irq\\_receive\(\)](#) are doing the unmask themselves.

Definition at line 324 of file [irq.h](#).

References [l4\\_irq\\_unmask\\_u\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.11.5.3.10 l4\_irq\_unmask\_u()

```

l4_msgtag_t l4_irq_unmask_u (
    l4_cap_idx_t irq,
    l4_utcb_t * utcb ) [inline]
  
```

Unmask this IRQ.

##### Parameters

<i>irq</i>	The IRQ object that shall be unmasked.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

##### Returns

Syscall return tag for a send-only operation, this means there is no return value except [L4\\_MSGTAG\\_ERROR](#) indicating success or failure of the send operation. Use [l4\\_ipc\\_error\(\)](#) to check for errors and **do not** use [l4\\_error\(\)](#).

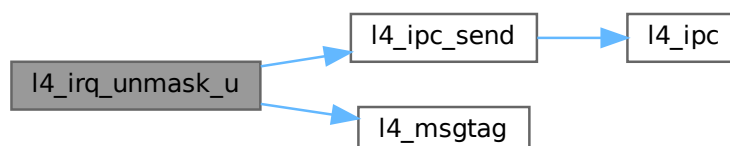
[Irq::wait\(\)](#) and [Irq::receive\(\)](#) operations already include an [unmask\(\)](#), do not use an extra [unmask\(\)](#) in these cases.

Definition at line 285 of file [irq.h](#).

References [L4\\_IPC\\_NEVER](#), [l4\\_ipc\\_send\(\)](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_IRQ](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [l4\\_irq\\_unmask\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 13.1.11.5.3.11 l4\_irq\_wait()

```

l4_msgtag_t l4_irq_wait (
    l4_cap_idx_t irq,
    l4_umword_t * label,
    l4_timeout_t to ) [inline]
  
```

Unmask IRQ and wait for any message.

#### Parameters

<i>irq</i>	The IRQ object that shall be unmasked.
<i>label</i>	Receive label.
<i>to</i>	Timeout.

#### Returns

Syscall return tag

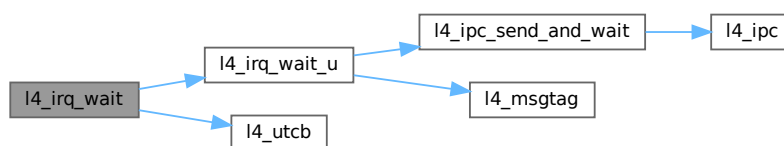
#### Examples

[examples/sys/isr/main.c](#).

Definition at line 317 of file [irq.h](#).

References [l4\\_irq\\_wait\\_u\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



## 13.1.11.5.3.12 l4\_irq\_wait\_u()

```
l4_msgtag_t l4_irq_wait_u (
    l4_cap_idx_t irq,
    l4_umword_t * label,
    l4_timeout_t timeout,
    l4_utcb_t * utcb ) [inline]
```

Unmask IRQ and (open) wait for any message.

## Parameters

<i>irq</i>	The IRQ object that shall be unmasked.
<i>label</i>	The <i>protected label</i> shall be received here.
<i>timeout</i>	Timeout.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

## Returns

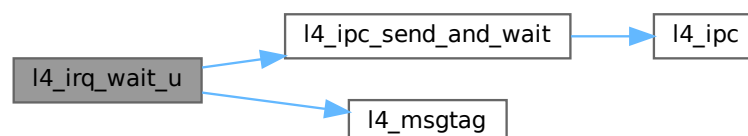
Syscall return tag

Definition at line 276 of file [irq.h](#).

References [l4\\_ipc\\_send\\_and\\_wait\(\)](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_IRQ](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [l4\\_irq\\_wait\(\)](#).

Here is the call graph for this function:



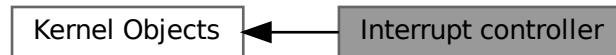
Here is the caller graph for this function:



### 13.1.11.6 Interrupt controller

The C Icu interface, see [L4::Icu](#) for the C++ interface.

Collaboration diagram for Interrupt controller:



#### Data Structures

- struct [l4\\_icu\\_info\\_t](#)  
*Info structure for an ICU.*

#### Typedefs

- typedef struct [l4\\_icu\\_info\\_t](#) [l4\\_icu\\_info\\_t](#)  
*Info structure for an ICU.*

#### Enumerations

- enum [L4\\_icu\\_flags](#) { [L4\\_ICU\\_FLAG\\_MSI](#) }  
*Flags for IRQ numbers used for the ICU.*

#### Functions

- [l4\\_msgtag\\_t](#) [l4\\_icu\\_bind](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_cap\\_idx\\_t](#) irq) [L4\\_NOTHROW](#)  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- [l4\\_msgtag\\_t](#) [l4\\_icu\\_bind\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_cap\\_idx\\_t](#) irq, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- [l4\\_msgtag\\_t](#) [l4\\_icu\\_unbind](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_cap\\_idx\\_t](#) irq) [L4\\_NOTHROW](#)  
*Remove binding of an interrupt line from the interrupt controller object.*
- [l4\\_msgtag\\_t](#) [l4\\_icu\\_unbind\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_cap\\_idx\\_t](#) irq, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Remove binding of an interrupt line from the interrupt controller object.*
- [l4\\_msgtag\\_t](#) [l4\\_icu\\_set\\_mode](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) mode) [L4\\_NOTHROW](#)  
*Set interrupt mode.*
- [l4\\_msgtag\\_t](#) [l4\\_icu\\_set\\_mode\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) mode, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Set interrupt mode.*
- [l4\\_msgtag\\_t](#) [l4\\_icu\\_info](#) ([l4\\_cap\\_idx\\_t](#) icu, [l4\\_icu\\_info\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Get information about the ICU features.*

- [l4\\_msgtag\\_t l4\\_icu\\_info\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, [l4\\_icu\\_info\\_t](#) \*info, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Get information about the ICU features.*
- [l4\\_msgtag\\_t l4\\_icu\\_msi\\_info](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_uint64\\_t](#) source, [l4\\_icu\\_msi\\_info\\_t](#) \*msi\_info) [L4\\_NOTHROW](#)  
*Get MSI info about IRQ.*
- [l4\\_msgtag\\_t l4\\_icu\\_msi\\_info\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_uint64\\_t](#) source, [l4\\_icu\\_msi\\_info\\_t](#) \*msi\_info, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Get MSI info about IRQ.*
- [l4\\_msgtag\\_t l4\\_icu\\_unmask](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) to) [L4\\_NOTHROW](#)  
*Unmask an IRQ line.*
- [l4\\_msgtag\\_t l4\\_icu\\_unmask\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) to, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Unmask the given interrupt line.*
- [l4\\_msgtag\\_t l4\\_icu\\_mask](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) to) [L4\\_NOTHROW](#)  
*Mask an IRQ line.*
- [l4\\_msgtag\\_t l4\\_icu\\_mask\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) to, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Mask an IRQ line.*

### 13.1.11.6.1 Detailed Description

The C lcu interface, see [L4::lcu](#) for the C++ interface.

#### Note

"ICU" is short for "interrupt control unit".

These functions define the interface for interrupt controllers, for binding IRQ objects to interrupt lines and other interrupt sources, as well as functions for masking and unmasking of interrupts.

To setup an IRQ line the following steps are required:

1. [l4\\_icu\\_set\\_mode\(\)](#) (optional if IRQ has a default mode)
2. [l4\\_rcv\\_ep\\_bind\\_thread\(\)](#) to attach the IRQ object to a thread
3. [l4\\_icu\\_bind\(\)](#)
4. [l4\\_icu\\_unmask\(\)](#) to receive the first IRQ

For certain interrupt sources only some of these steps are necessary and supported, see [Scheduler](#) and [Virtual Console](#).

At most one [IRQs](#) object can be bound to a certain interrupt source and a certain [IRQs](#) object can be bound to at most one interrupt source.

#### Include File

```
#include <l4/sys/icu.h>
```

### 13.1.11.6.2 Typedef Documentation

#### 13.1.11.6.2.1 l4\_icu\_info\_t

```
typedef struct l4_icu_info_t l4_icu_info_t
```

Info structure for an ICU.

This structure contains information about the features of an ICU.

See also

[l4\\_icu\\_info\(\)](#).

### 13.1.11.6.3 Enumeration Type Documentation

#### 13.1.11.6.3.1 L4\_icu\_flags

```
enum L4_icu_flags
```

Flags for IRQ numbers used for the ICU.

Enumerator

L4_ICU_FLAG_MSI	Flag to denote that the IRQ is actually an MSI. This flag may be used for <a href="#">l4_icu_bind()</a> and <a href="#">l4_icu_unbind()</a> functions to denote that the IRQ number is meant to be an MSI.
-----------------	--

Definition at line 63 of file [icu.h](#).

### 13.1.11.6.4 Function Documentation

#### 13.1.11.6.4.1 l4\_icu\_bind()

```
l4_msgtag_t l4_icu_bind (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_cap_idx_t irq ) [inline]
```

Bind an interrupt line of an interrupt controller to an interrupt object.

Parameters

<i>icu</i>	ICU object to bind <i>irq</i> to.
<i>irqnum</i>	IRQ line at the ICU.
<i>irq</i>	IRQ object to bind to this ICU.



## Returns

Syscall return tag. The caller should check the return value using [l4\\_error\(\)](#) to check for errors and to identify the correct method for unmasking the interrupt. Return values  $< 0$  indicate an error. A return value of 0 means a direct unmask via the IRQ object using [l4\\_irq\\_unmask\(\)](#). A return value of 1 means that the interrupt has to be unmasked via the ICU using [l4\\_icu\\_unmask\(\)](#).

## Return values

-L4_EINVAL	<code>irq</code> is bound to an interrupt source.
-L4_EPERM	The ICU instance requires <a href="#">L4_CAP_FPAGE_W</a> on <code>irq</code> and <a href="#">L4_CAP_FPAGE_W</a> was not present.

In case the `irq` is already bound to an interrupt source, it is unbound first. In case the `irq` is bound and the interrupt source is bound to a different IRQ object, only the unbinding happens. An IRQ object that is bound to an interrupt source will get unbound if the IRQ object is deleted.

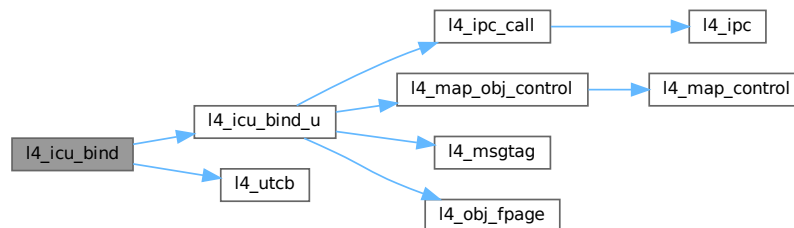
## Examples

[examples/sys/isr/main.c](#).

Definition at line 504 of file [icu.h](#).

References [l4\\_icu\\_bind\\_u\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



## 13.1.11.6.4.2 l4\_icu\_bind\_u()

```

l4_msgtag_t l4_icu_bind_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_cap_idx_t irq,
    l4_utcb_t * utcb ) [inline]
  
```

Bind an interrupt line of an interrupt controller to an interrupt object.

## Parameters

<i>icu</i>	The ICU object to bind <code>irq</code> to.
<i>irqnum</i>	IRQ line at the ICU.
<i>irq</i>	IRQ object for the given IRQ line to bind to this ICU.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .



#### 13.1.11.6.4.3 l4\_icu\_info()

```
l4_msgtag_t l4_icu_info (
    l4_cap_idx_t icu,
    l4_icu_info_t * info ) [inline]
```

Get information about the ICU features.

##### Parameters

	<i>icu</i>	The ICU object from which information shall be retrieved.
out	<i>info</i>	Info structure to be filled with information.

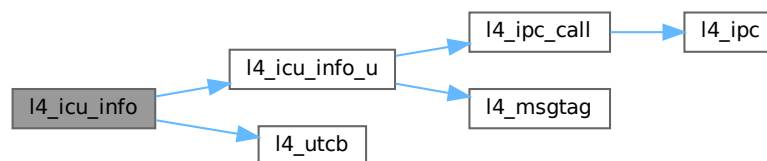
##### Returns

Syscall return tag

Definition at line 512 of file [icu.h](#).

References [l4\\_icu\\_info\\_u\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.11.6.4.4 l4\_icu\_info\_u()

```
l4_msgtag_t l4_icu_info_u (
    l4_cap_idx_t icu,
    l4_icu_info_t * info,
    l4_utcb_t * utcb ) [inline]
```

Get information about the ICU features.

##### Parameters

	<i>icu</i>	The ICU object from which MSI information shall be retrieved.
out	<i>info</i>	Info structure to be filled with information.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

## Returns

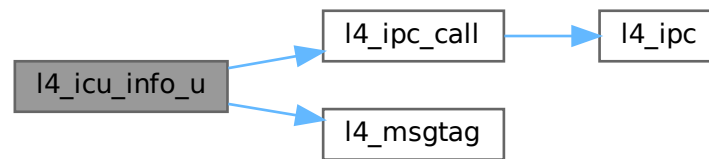
Syscall return tag

Definition at line 428 of file [icu.h](#).

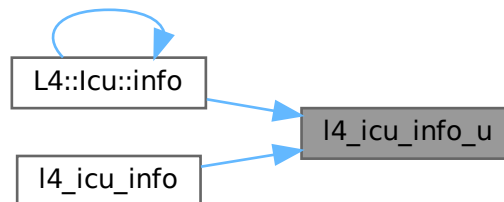
References [L4\\_ICU\\_OP\\_INFO](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_IRQ](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [L4::l4u::info\(\)](#), and [l4\\_icu\\_info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.11.6.4.5 l4\_icu\_mask()

```

l4_msgtag_t l4_icu_mask (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t * label,
    l4_timeout_t to ) [inline]
  
```

Mask an IRQ line.

## Parameters

<i>icu</i>	The ICU object where the IRQ line shall be masked.
<i>irqnum</i>	IRQ line at the ICU.
<i>label</i>	If non-NULL, the function also performs an open wait IPC operation waiting for the next message, and the received label is returned here.
<i>to</i>	IPC timeout, if unsure use L4_IPC_NEVER.

## Returns

Syscall return tag. If *label* is NULL, this function performs an IPC send-only operation and there is no return value except [L4\\_MSGTAG\\_ERROR](#) indicating success or failure of the send operation. In this case use [l4\\_ipc\\_error\(\)](#) to check for errors and **do not** use [l4\\_error\(\)](#).

Definition at line [526](#) of file [icu.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



## 13.1.11.6.4.6 l4\_icu\_mask\_u()

```

l4_msgtag_t l4_icu_mask_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t * label,
    l4_timeout_t to,
    l4_utcb_t * utcb ) [inline]
  
```

Mask an IRQ line.

## Parameters

<i>icu</i>	The ICU object where the IRQ line shall be masked.
<i>irqnum</i>	IRQ line at the ICU.
<i>label</i>	If NULL, this function is a send-only message to the ICU. If not NULL, this function will enter an open wait after sending the mask message and the received label is returned here.
<i>to</i>	The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non-NULL <i>label</i> only.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

## Returns

Syscall return tag. If `label` is `NULL`, this function performs an IPC send-only operation and there is no return value except `L4_MSGTAG_ERROR` indicating success or failure of the send operation. In this case use `l4_ipc_error()` to check for errors and **do not** use `l4_error()`.

Definition at line 491 of file `icu.h`.

Referenced by `L4::Icu::mask()`.

Here is the caller graph for this function:



## 13.1.11.6.4.7 l4\_icu\_msi\_info()

```

l4_msgtag_t l4_icu_msi_info (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_uint64_t source,
    l4_icu_msi_info_t * msi_info ) [inline]
  
```

Get MSI info about IRQ.

## Parameters

	<i>icu</i>	The ICU object from which MSI information shall be retrieved.
	<i>irqnum</i>	IRQ line at the ICU.
	<i>source</i>	Platform dependent requester ID for MSIs. On IA32 we use a 20bit source filter value as described in the Intel IRQ remapping specification.
out	<i>msi_info</i>	A <code>l4_icu_msi_info_t</code> structure receiving the address and the data value to trigger this MSI.

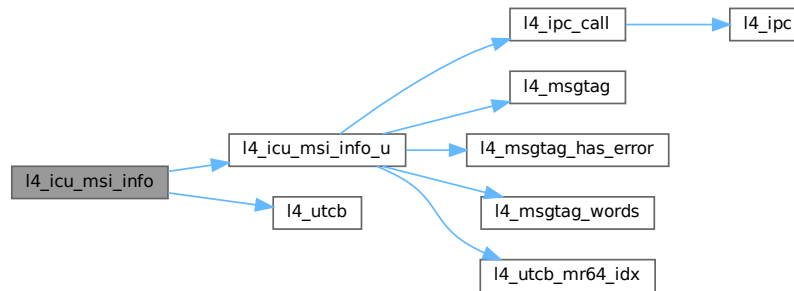
## Returns

Syscall return tag

Definition at line 516 of file `icu.h`.

References `l4_icu_msi_info_u()`, and `l4_utcb()`.

Here is the call graph for this function:



#### 13.1.11.6.4.8 l4\_icu\_msi\_info\_u()

```

l4_msgtag_t l4_icu_msi_info_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_uint64_t source,
    l4_icu_msi_info_t * msi_info,
    l4_utcb_t * utcb ) [inline]
  
```

Get MSI info about IRQ.

##### Parameters

	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
	<i>icu</i>	The ICU object from which MSI information shall be retrieved.
	<i>irqnum</i>	IRQ line at the ICU.
	<i>source</i>	Platform dependent requester ID for MSIs. On IA32 we use a 20bit source filter value as described in the Intel IRQ remapping specification.
out	<i>msi_info</i>	A <a href="#">l4_icu_msi_info_t</a> structure receiving the address and the data value to trigger this MSI.

##### Returns

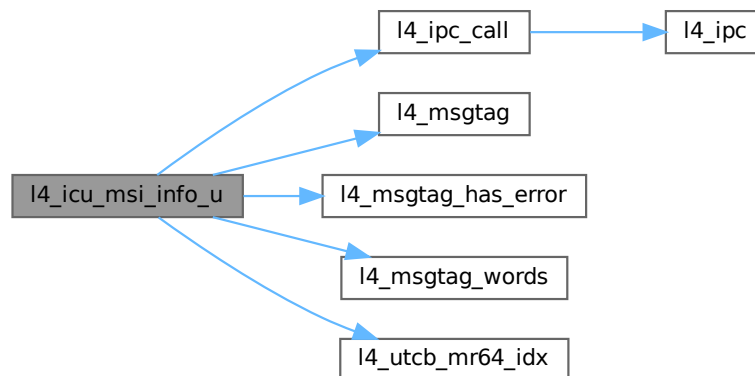
Syscall return tag

Definition at line 442 of file [icu.h](#).

References [L4\\_ICU\\_OP\\_MSI\\_INFO](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [l4\\_msgtag\\_has\\_error\(\)](#), [l4\\_msgtag\\_words\(\)](#), [L4\\_PROTO\\_IRQ](#), [L4\\_UNLIKELY](#), [l4\\_utcb\\_mr64\\_idx\(\)](#), [l4\\_msg\\_regs\\_t::mr](#), and [l4\\_msg\\_regs\\_t::mr64](#).

Referenced by [l4\\_icu\\_msi\\_info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.11.6.4.9 l4\_icu\_set\_mode()

```

l4_msgtag_t l4_icu_set_mode (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t mode ) [inline]
  
```

Set interrupt mode.

##### Parameters

<i>icu</i>	The ICU object.
<i>irqnum</i>	IRQ line at the ICU.
<i>mode</i>	Mode, see <a href="#">L4_irq_mode</a> .

##### Returns

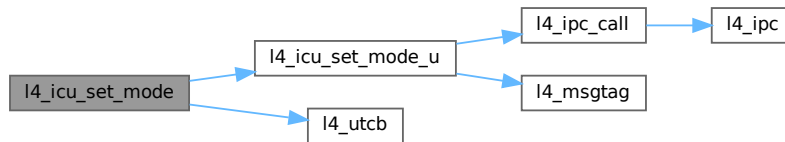
Syscall return tag



Definition at line 531 of file [icu.h](#).

References [l4\\_icu\\_set\\_mode\\_u\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.11.6.4.10 l4\_icu\_set\_mode\_u()

```

l4_msgtag_t l4_icu_set_mode_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t mode,
    l4_utcb_t * utcb ) [inline]
  
```

Set interrupt mode.

##### Parameters

<i>icu</i>	The ICU object.
<i>irqnum</i>	IRQ line at the ICU.
<i>mode</i>	Mode, see <a href="#">L4_irq_mode</a> .
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

##### Returns

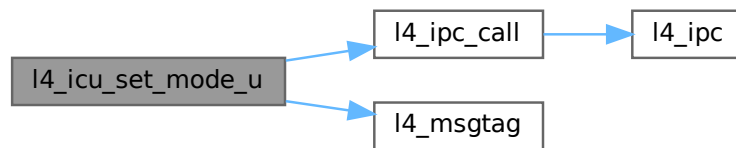
Syscall return tag

Definition at line 465 of file [icu.h](#).

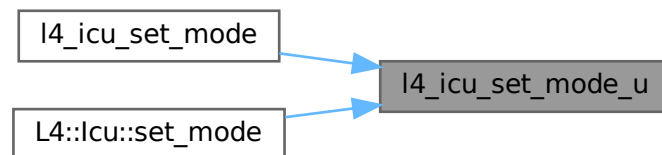
References [L4\\_ICU\\_OP\\_SET\\_MODE](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_IRQ](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [l4\\_icu\\_set\\_mode\(\)](#), and [L4::l4u::set\\_mode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.11.6.4.11 l4\_icu\_unbind()

```

l4_msgtag_t l4_icu_unbind (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_cap_idx_t irq ) [inline]
  
```

Remove binding of an interrupt line from the interrupt controller object.

##### Parameters

<i>icu</i>	The ICU object from where the binding shall be removed.
<i>irqnum</i>	IRQ line at the ICU.
<i>irq</i>	IRQ object to remove from the ICU.

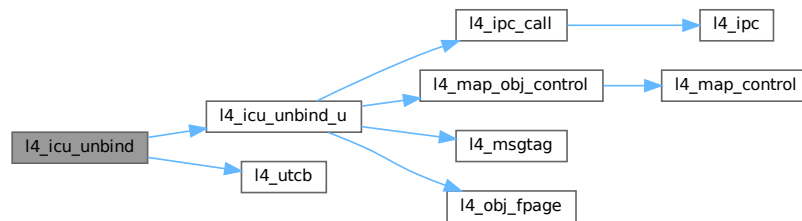
##### Returns

Syscall return tag

Definition at line 508 of file `icu.h`.

References `l4_icu_unbind_u()`, and `l4_utcb()`.

Here is the call graph for this function:



#### 13.1.11.6.4.12 l4\_icu\_unbind\_u()

```

l4_msgtag_t l4_icu_unbind_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_cap_idx_t irq,
    l4_utcb_t * utcb ) [inline]
  
```

Remove binding of an interrupt line from the interrupt controller object.

##### Parameters

<i>icu</i>	The ICU object from where the binding shall be removed.
<i>irqnum</i>	IRQ line at the ICU.
<i>irq</i>	IRQ object to remove from the ICU.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

##### Returns

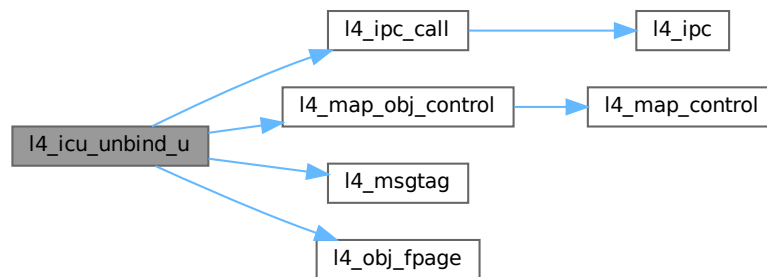
Syscall return tag

Definition at line 416 of file [icu.h](#).

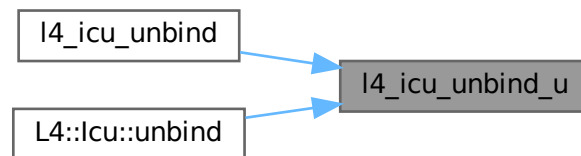
References [L4\\_CAP\\_FPAGE\\_RWS](#), [L4\\_ICU\\_OP\\_UNBIND](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_map\\_obj\\_control\(\)](#), [l4\\_msgtag\(\)](#), [l4\\_obj\\_fpage\(\)](#), [L4\\_PROTO\\_IRQ](#), [l4\\_msg\\_regs\\_t::mr](#), and [l4\\_fpage\\_t::raw](#).

Referenced by [l4\\_icu\\_unbind\(\)](#), and [L4::l4icu::unbind\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.11.6.4.13 l4\_icu\_unmask()

```

l4_msgtag_t l4_icu_unmask (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t * label,
    l4_timeout_t to ) [inline]
  
```

Unmask an IRQ line.

##### Parameters

<i>icu</i>	The ICU object where the IRQ line shall be unmasked.
<i>irqnum</i>	IRQ line at the ICU.
<i>label</i>	If non-NULL, the function also performs an open wait IPC operation waiting for the next message, and the received label is returned here.
<i>to</i>	IPC timeout, if unsure use <code>L4_IPC_NEVER</code> .

## Returns

Syscall return tag. If `label` is `NULL`, this function performs an IPC send-only operation and there is no return value except `L4_MSGTAG_ERROR` indicating success or failure of the send operation. In this case use `l4_ipc_error()` to check for errors and **do not** use `l4_error()`.

Definition at line 521 of file `icu.h`.

References `l4_utcb()`.

Here is the call graph for this function:



## 13.1.11.6.4.14 l4\_icu\_unmask\_u()

```

l4_msgtag_t l4_icu_unmask_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t * label,
    l4_timeout_t to,
    l4_utcb_t * utcb ) [inline]
  
```

Unmask the given interrupt line.

## Parameters

<i>icu</i>	The ICU object where the IRQ line shall be unmasked.
------------	--

When the object is an IRQ, the given interrupt line is ignored and instead the line which the IRQ is bound to (if any) is unmasked.

Its counterpart for explicitly masking an interrupt line is `L4::Icu::mask()`.

## Parameters

	<i>irqnum</i>	The interrupt line that shall be unmasked. Ignored if the object is an IRQ.
out	<i>label</i>	If <code>NULL</code> , this is a send-only unmask. If not <code>NULL</code> , this operation enters an open wait and the <i>protected label</i> shall be received here.
	<i>to</i>	The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non- <code>NULL label</code> only.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <code>l4_utcb</code> .

### Returns

Syscall return tag. If `label` is `NULL`, this function performs an IPC send-only operation and there is no return value except `L4_MSGTAG_ERROR` indicating success or failure of the send operation. In this case use `l4_ipc_error()` to check for errors and **do not** use `l4_error()`.

Definition at line 496 of file `icu.h`.

### 13.1.11.7 Kernel-provided semaphore

C semaphore interface, see `L4::Semaphore` for the C++ interface.

Collaboration diagram for Kernel-provided semaphore:



### Functions

- `l4_msgtag_t l4_semaphore_up (l4_cap_idx_t sem) L4_NOTHROW`  
*Semaphore up operation (wrapper for trigger()).*
- `l4_msgtag_t l4_semaphore_down (l4_cap_idx_t sem, l4_timeout_t timeout) L4_NOTHROW`  
*Semaphore down operation.*

#### 13.1.11.7.1 Detailed Description

C semaphore interface, see `L4::Semaphore` for the C++ interface.

#### Include File

```
#include <l4/sys/semaphore.h>
```

#### 13.1.11.7.2 Function Documentation

##### 13.1.11.7.2.1 l4\_semaphore\_down()

```
l4_msgtag_t l4_semaphore_down (
    l4_cap_idx_t sem,
    l4_timeout_t timeout ) [inline]
```

Semaphore down operation.

## Parameters

<i>sem</i>	Semaphore object.
<i>timeout</i>	Timeout for blocking the semaphore down operation. Note: The receive timeout of this timeout-pair is significant for blocking, the send part is usually non-blocking.

## Returns

Syscall return tag. Use [l4\\_error\(\)](#) to check for errors.

## Return values

<code>-L4_EPERM</code>	No <a href="#">L4_CAP_FPAGE_S</a> right on invoked semaphore capability.
------------------------	--

This method decrements the semaphore counter by one, or blocks if the counter is already zero, until either a timeout or cancel condition hits or the counter is increased by an `up()` operation.

Definition at line 110 of file [semaphore.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:

13.1.11.7.2.2 `l4_semaphore_up()`

```
l4_msgtag_t l4_semaphore_up (
    l4_cap_idx_t sem ) [inline]
```

Semaphore up operation (wrapper for `trigger()`).

## Parameters

<i>sem</i>	Semaphore object.
------------	-------------------

## Returns

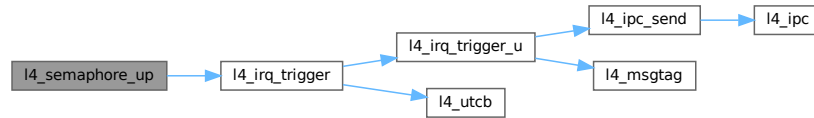
Send-only IPC message return tag. Use [l4\\_ipc\\_error\(\)](#) to check for errors, do **not** use [l4\\_error\(\)](#).

Increases the semaphore counter by one if it is smaller than an unspecified limit. The unspecified limit is guaranteed to be at least  $2^{31}-1$ .

Definition at line 56 of file [semaphore.h](#).

References [l4\\_irq\\_trigger\(\)](#).

Here is the call graph for this function:



### 13.1.11.8 L4 kernel object type information

Type information for [L4](#) server objects that can be called via IPC.

Collaboration diagram for L4 kernel object type information:



## Data Structures

- struct [L4::Type\\_info](#)  
*Dynamic Type Information for [L4Re](#) Interfaces.*
- struct [L4::Kobject\\_typeid< T >](#)  
*Meta object for handling access to type information of Kobjects.*
- class [L4::Kobject\\_t< Derived, Base, PROTO, S\\_DEMAND >](#)  
*Helper class to create an [L4Re](#) interface class that is derived from a single base class.*
- class [L4::Kobject\\_2t< Derived, Base1, Base2, PROTO, S\\_DEMAND >](#)  
*Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject\\_t](#)).*
- struct [L4::Kobject\\_3t< Derived, Base1, Base2, Base3, PROTO, S\\_DEMAND >](#)  
*Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject\\_t](#)).*
- struct [L4::Kobject\\_x< Derived, ARGS >](#)  
*Generic [Kobject](#) inheritance template.*

## Functions

- template<typename T >  
[Type\\_info](#) const \* [L4::kobject\\_typeid](#) () noexcept  
*Get the [L4::Type\\_info](#) for the [L4Re](#) interface given in T.*



### 13.1.11.8.1 Detailed Description

Type information for [L4](#) server objects that can be called via IPC.

This type information consists of inheritance information, the protocol number assigned to an interface as well as the demand on server-side resources.

### 13.1.11.8.2 Function Documentation

#### 13.1.11.8.2.1 kobject\_typeid()

```
template<typename T >
Type_info const * L4::kobject_typeid ( ) [inline], [noexcept]
```

Get the [L4::Type\\_info](#) for the [L4Re](#) interface given in T.

#### Template Parameters

<a href="#">T</a>	The type ( <a href="#">L4Re</a> interface) for which the information shall be returned.
-------------------	---

#### Returns

A pointer to the [L4::Type\\_info](#) structure for T.

Definition at line 693 of file [\\_\\_typeinfo.h](#).

References [L4::Kobject\\_typeid< T >::id\(\)](#).

Here is the call graph for this function:



### 13.1.11.9 Platform Control C API

C interface for controlling platform-wide properties, see [L4::Platform\\_control](#) for the C++ interface.

Collaboration diagram for Platform Control C API:



## Functions

- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_set\\_task\\_asid](#) ([l4\\_cap\\_idx\\_t](#) pfc, [l4\\_cap\\_idx\\_t](#) task, [l4\\_umword\\_t](#) asid) [L4\\_NOTHROW](#)  
*Set ASID of task.*
- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_system\\_suspend](#) ([l4\\_cap\\_idx\\_t](#) pfc, [l4\\_umword\\_t](#) extras) [L4\\_NOTHROW](#)  
*Enter suspend to RAM.*
- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_system\\_shutdown](#) ([l4\\_cap\\_idx\\_t](#) pfc, [l4\\_umword\\_t](#) reboot) [L4\\_NOTHROW](#)  
*Shutdown or reboot the system.*
- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_cpu\\_allow\\_shutdown](#) ([l4\\_cap\\_idx\\_t](#) pfc, [l4\\_umword\\_t](#) phys\_id, [l4\\_umword\\_t](#) enable) [L4\\_NOTHROW](#)  
*Allow a CPU to be shut down.*
- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_cpu\\_enable](#) ([l4\\_cap\\_idx\\_t](#) pfc, [l4\\_umword\\_t](#) phys\_id) [L4\\_NOTHROW](#)  
*Enable an offline CPU.*
- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_cpu\\_disable](#) ([l4\\_cap\\_idx\\_t](#) pfc, [l4\\_umword\\_t](#) phys\_id) [L4\\_NOTHROW](#)  
*Disable an online CPU.*

### 13.1.11.9.1 Detailed Description

C interface for controlling platform-wide properties, see [L4::Platform\\_control](#) for the C++ interface.

#### Include File

```
#include <l4/sys/platform_control.h>
```

The API allows a client to suspend, reboot or shutdown the system.

For the C++ interface refer to [L4::Platform\\_control](#)

### 13.1.11.9.2 Function Documentation

#### 13.1.11.9.2.1 l4\_platform\_ctl\_cpu\_allow\_shutdown()

```
l4\_msgtag\_t l4_platform_ctl_cpu_allow_shutdown (
    l4\_cap\_idx\_t pfc,
    l4\_umword\_t phys_id,
    l4\_umword\_t enable ) [inline]
```

Allow a CPU to be shut down.

#### Parameters

<i>pfc</i>	Capability selector for the platform-control object.
<i>phys_id</i>	Physical CPU id of CPU (e.g. local APIC id) to enable.
<i>enable</i>	Allow shutdown when 1, disallow when 0.

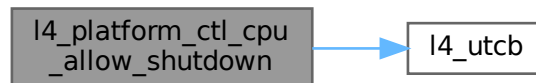
**Returns**

Syscall return tag

Definition at line 253 of file [platform\\_control.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**13.1.11.9.2.2 l4\_platform\_ctl\_cpu\_disable()**

```

l4_msgtag_t l4_platform_ctl_cpu_disable (
    l4_cap_idx_t pfc,
    l4_umword_t phys_id ) [inline]
  
```

Disable an online CPU.

**Parameters**

<i>pfc</i>	Capability to the platform control object.
<i>phys_id</i>	Physical CPU id of CPU (e.g. local APIC id) to disable.

**Returns**

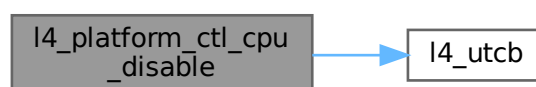
System call message tag

This function is currently only supported on the ARM EXYNOS platform.

Definition at line 292 of file [platform\\_control.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 13.1.11.9.2.3 l4\_platform\_ctl\_cpu\_enable()

```
l4_msgtag_t l4_platform_ctl_cpu_enable (
    l4_cap_idx_t pfc,
    l4_umword_t phys_id ) [inline]
```

Enable an offline CPU.

#### Parameters

<i>pfc</i>	Capability to the platform control object.
<i>phys_id</i>	Physical CPU id of CPU (e.g. local APIC id) to enable.

#### Returns

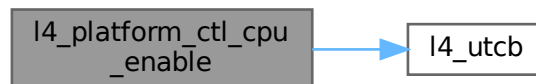
System call message tag

This function is currently only supported on the ARM EXYNOS platform.

Definition at line 285 of file [platform\\_control.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 13.1.11.9.2.4 l4\_platform\_ctl\_set\_task\_asid()

```
l4_msgtag_t l4_platform_ctl_set_task_asid (
    l4_cap_idx_t pfc,
    l4_cap_idx_t task,
    l4_umword_t asid ) [inline]
```

Set ASID of task.

On Cortex-R52 platforms, it might be necessary to control the VMID of a task or virtual machine explicitly. The IOMPU on such platforms will use it for further access control of device memory accesses. A privileged component can use this call to control the value.

The caller must have write permissions to the destination task.

## Parameters

<i>pfc</i>	Capability selector for the platform-control object.
<i>task</i>	Capability selector of destination task
<i>asid</i>	New ASID value

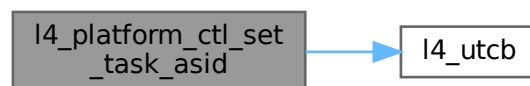
## Returns

Syscall return tag

Definition at line 62 of file [\\_\\_platform\\_control-arm.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.11.9.2.5 l4\_platform\_ctl\_system\_shutdown()

```
l4_msgtag_t l4_platform_ctl_system_shutdown (
    l4_cap_idx_t pfc,
    l4_umword_t reboot ) [inline]
```

Shutdown or reboot the system.

## Parameters

<i>pfc</i>	Capability selector for the platform-control object.
<i>reboot</i>	Shutdown when 0, or reboot when 1.

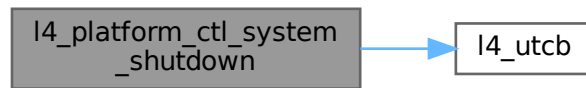
## Returns

Syscall return tag

Definition at line 232 of file [platform\\_control.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.11.9.2.6 l4\_platform\_ctl\_system\_suspend()

```

l4_msgtag_t l4_platform_ctl_system_suspend (
    l4_cap_idx_t pfc,
    l4_umword_t extras ) [inline]
  
```

Enter suspend to RAM.

##### Precondition

Must only be invoked on the boot CPU. Furthermore it must be ensured that the invoking thread is not migrated to a different CPU during the suspend.

##### Parameters

<i>pfc</i>	Capability selector for the platform-control object.
<i>extras</i>	Some extra platform-specific information needed to enter suspend to RAM. On x86 platforms and when using the Platform_control object provided by Fiasco, the value defines the sleep state. The sleep states are defined in the ACPI table. Other platforms as well as Io's Platform_control object don't make use of this value at the moment.

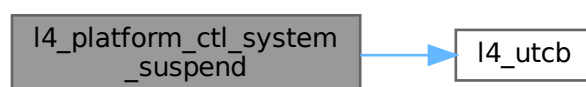
##### Returns

Syscall return tag

Definition at line 225 of file [platform\\_control.h](#).

References [l4\\_utcb\(\)](#).

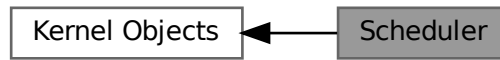
Here is the call graph for this function:



### 13.1.11.10 Scheduler

C interface of the Scheduler kernel object, see [L4::Scheduler](#) for the C++ interface.

Collaboration diagram for Scheduler:



#### Data Structures

- struct [l4\\_sched\\_cpu\\_set\\_t](#)  
*CPU sets.*
- struct [l4\\_sched\\_param\\_t](#)  
*Scheduler parameter set.*

#### Typedefs

- typedef struct [l4\\_sched\\_cpu\\_set\\_t](#) [l4\\_sched\\_cpu\\_set\\_t](#)  
*CPU sets.*
- typedef struct [l4\\_sched\\_param\\_t](#) [l4\\_sched\\_param\\_t](#)  
*Scheduler parameter set.*

#### Enumerations

- enum [L4\\_scheduler\\_classes](#) { [L4\\_SCHEDULER\\_CLASS\\_FIXED\\_PRIO](#) = 1UL << 1, [L4\\_SCHEDULER\\_CLASS\\_WFQ](#) = 1UL << 2 }  
*Supported scheduler classes.*
- enum [L4\\_scheduler\\_ops](#) { [L4\\_SCHEDULER\\_INFO\\_OP](#) = 0UL, [L4\\_SCHEDULER\\_RUN\\_THREAD\\_OP](#) = 1UL, [L4\\_SCHEDULER\\_IDLE\\_TIME\\_OP](#) = 2UL }  
*Operations on the Scheduler object.*

#### Functions

- [l4\\_sched\\_cpu\\_set\\_t l4\\_sched\\_cpu\\_set](#) ([l4\\_umword\\_t](#) offset, unsigned char granularity, [l4\\_umword\\_t](#) map=1) [L4\\_NOTHROW](#)
- [l4\\_msgtag\\_t l4\\_scheduler\\_info](#) ([l4\\_cap\\_idx\\_t](#) scheduler, [l4\\_umword\\_t](#) \*cpu\_max, [l4\\_sched\\_cpu\\_set\\_t](#) \*cpus) [L4\\_NOTHROW](#))  
*Get scheduler information.*
- [l4\\_msgtag\\_t l4\\_scheduler\\_info\\_with\\_classes](#) ([l4\\_cap\\_idx\\_t](#) scheduler, [l4\\_umword\\_t](#) \*cpu\_max, [l4\\_sched\\_cpu\\_set\\_t](#) \*cpus, [l4\\_umword\\_t](#) \*sched\_classes) [L4\\_NOTHROW](#))  
*Get scheduler information.*
- [l4\\_sched\\_param\\_t l4\\_sched\\_param](#) (unsigned prio, [l4\\_umword\\_t](#) quantum=0) [L4\\_NOTHROW](#)

*Construct scheduler parameter.*

- `l4_msgtag_t l4_scheduler_run_thread (l4_cap_idx_t scheduler, l4_cap_idx_t thread, l4_sched_param_t const *sp) L4_NOTHROW`

*Run a thread on a Scheduler.*

- `l4_msgtag_t l4_scheduler_idle_time (l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus, l4_kernel_clock_t *us) L4_NOTHROW`

*Query the idle time (in  $\mu$ s) of a CPU.*

- `int l4_scheduler_is_online (l4_cap_idx_t scheduler, l4_umword_t cpu) L4_NOTHROW`

*Query if a CPU is online.*

### 13.1.11.10.1 Detailed Description

C interface of the Scheduler kernel object, see [L4::Scheduler](#) for the C++ interface.

The Scheduler interface allows a client to manage CPU resources. The API provides functions to query scheduler information, check the online state of CPUs, query CPU idle time and to start threads on defined CPU sets.

The scheduler offers a virtual device IRQ which triggers when the number of online cores changes, e.g. due to hotplug events. In contrast to hardware IRQs, this IRQ implements a limited functionality:

- Only IRQ line 0 is supported, no MSI vectors.
- The IRQ is edge-triggered and the IRQ mode cannot be changed.
- As the IRQ is edge-triggered, it does not have to be explicitly unmasked.

It depends on the platform, which hotplug events actually trigger the IRQ. Many platforms only support triggering the IRQ when a CPU core different from the boot CPU goes online.

#### Include File

```
#include <l4/sys/scheduler.h>
```

### 13.1.11.10.2 Enumeration Type Documentation

#### 13.1.11.10.2.1 L4\_scheduler\_classes

```
enum L4_scheduler_classes
```

Supported scheduler classes.

#### Enumerator

<code>L4_SCHEDULER_CLASS_FIXED_PRIO</code>	Fixed-priority scheduler.
<code>L4_SCHEDULER_CLASS_WFQ</code>	Weighted fair queuing scheduler.

Definition at line 57 of file [scheduler.h](#).



### 13.1.11.10.2.2 L4\_scheduler\_ops

enum [L4\\_scheduler\\_ops](#)

Operations on the Scheduler object.

#### Enumerator

<code>L4_SCHEDULER_INFO_OP</code>	Query infos about the scheduler.
<code>L4_SCHEDULER_RUN_THREAD_OP</code>	Run a thread on this scheduler.
<code>L4_SCHEDULER_IDLE_TIME_OP</code>	Query idle time for the scheduler.

Definition at line [271](#) of file [scheduler.h](#).

### 13.1.11.10.3 Function Documentation

#### 13.1.11.10.3.1 l4\_sched\_cpu\_set()

```
l4_sched_cpu_set_t l4_sched_cpu_set (
    l4_umword_t offset,
    unsigned char granularity,
    l4_umword_t map = 1 ) [inline]
```

#### Parameters

<i>offset</i>	Offset. Must be a multiple of $2^{\text{granularity}}$ .
<i>granularity</i>	Granularity in log2 notation.
<i>map</i>	Bitmap of CPUs, defaults to 1 in C++.

#### Returns

CPU set.

#### Examples

[examples/sys/migrate/thread\\_migrate.cc](#).

Definition at line [281](#) of file [scheduler.h](#).

References [l4\\_sched\\_cpu\\_set\\_t::gran\\_offset](#), and [l4\\_sched\\_cpu\\_set\\_t::map](#).

Referenced by [l4\\_sched\\_param\(\)](#).

Here is the caller graph for this function:



### 13.1.11.10.3.2 l4\_sched\_param()

```
l4_sched_param_t l4_sched_param (
    unsigned prio,
    l4_umword_t quantum = 0 ) [inline]
```

Construct scheduler parameter.

The `l4_sched_param_t::affinity` of the returned value contains all CPUs.

#### Examples

`examples/sys/aliens/main.c`, `examples/sys/migrate/thread_migrate.cc`, `examples/sys/singlestep/main.c`, `examples/sys/start-with-exc/main.c`, and `examples/sys/utcb-ipc/main.c`.

Definition at line 291 of file `scheduler.h`.

References `l4_sched_param_t::affinity`, `l4_sched_cpu_set()`, `l4_sched_param_t::prio`, and `l4_sched_param_t::quantum`.

Here is the call graph for this function:



### 13.1.11.10.3.3 l4\_scheduler\_idle\_time()

```
l4_msgtag_t l4_scheduler_idle_time (
    l4_cap_idx_t scheduler,
    l4_sched_cpu_set_t const * cpus,
    l4_kernel_clock_t * us ) [inline]
```

Query the idle time (in  $\mu$ s) of a CPU.

#### Parameters

	<i>scheduler</i>	Scheduler object.
	<i>cpus</i>	Set of CPUs to query. Only the idle time of the first selected CPU in <code>cpus.map</code> is queried.
out	<i>us</i>	Idle time of queried CPU in $\mu$ s.

#### Return values

0	Success.
-L4_EINVAL	Invalid CPU requested in cpu set.

This function retrieves the idle time in  $\mu\text{s}$  of the first selected CPU in `cpus.map`. The idle time is the accumulated time a CPU has spent in the idle thread since its last reset. To calculate a load estimate  $l$  one has to retrieve the idle time at the beginning ( $i1$ ) and the end ( $i2$ ) of a known time interval  $t$ . The load is then calculated as  $l = 1 - (i2 - i1)/t$ .

The idle time is only defined for online CPUs. Reading the idle time from offline CPUs is undefined and may result in either getting `-L4_EINVAL` or calculating an estimated (incorrect) load of 1.

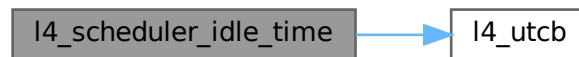
#### Note

The idle time statistics of remote CPUs is updated on context switch events only, hence may not be up-to-date when requested cross-CPU. To get up-to-date idle time you should use a thread running on the same CPU of which the idle time is requested.

Definition at line 405 of file `scheduler.h`.

References `l4_utcb()`.

Here is the call graph for this function:



#### 13.1.11.10.3.4 l4\_scheduler\_info()

```

l4_msgtag_t l4_scheduler_info (
    l4_cap_idx_t scheduler,
    l4_umword_t * cpu_max,
    l4_sched_cpu_set_t * cpus ) [inline]
  
```

Get scheduler information.

#### Parameters

	<i>scheduler</i>	Scheduler object.
out	<i>cpu_max</i>	Maximum number of CPUs ever available. Optional, can be NULL.
in, out	<i>cpus</i>	<i>cpus.offset</i> is first CPU of interest. <i>cpusgranularity</i> (see <code>l4_sched_cpu_set_t</code> ). <i>cpus.map</i> Bitmap of online CPUs. Must not be NULL.

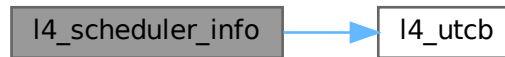
#### Return values

0	Success.
<code>-L4_ERANGE</code>	The given CPU offset is larger than the maximum number of CPUs.

Definition at line 383 of file [scheduler.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 13.1.11.10.3.5 l4\_scheduler\_info\_with\_classes()

```

l4_msgtag_t l4_scheduler_info_with_classes (
    l4_cap_idx_t scheduler,
    l4_umword_t * cpu_max,
    l4_sched_cpu_set_t * cpus,
    l4_umword_t * sched_classes ) [inline]
  
```

Get scheduler information.

#### Parameters

	<i>scheduler</i>	Scheduler object.
out	<i>cpu_max</i>	Maximum number of CPUs ever available. Optional, can be NULL.
in, out	<i>cpus</i>	<i>cpus.offset</i> is first CPU of interest. <i>cpus.granularity</i> (see <a href="#">l4_sched_cpu_set_t</a> ). <i>cpus.map</i> Bitmap of online CPUs. Must not be NULL.
out	<i>sched_classes</i>	A bitmap of available scheduling classes (see <a href="#">L4_scheduler_classes</a> ). Optional, can be NULL.

#### Return values

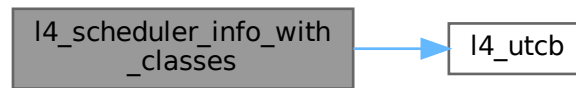
0	Success.
-L4_ERANGE	The given CPU offset is larger than the maximum number of CPUs.

This function delivers the same information as [l4\\_scheduler\\_info](#) plus the available scheduler classes (see [L4\\_scheduler\\_classes](#)).

Definition at line 390 of file [scheduler.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.11.10.3.6 l4\_scheduler\_is\_online()

```
int l4_scheduler_is_online (
    l4_cap_idx_t scheduler,
    l4_umword_t cpu ) [inline]
```

Query if a CPU is online.

##### Parameters

<i>scheduler</i>	Scheduler object.
<i>cpu</i>	CPU number whose online status should be queried.

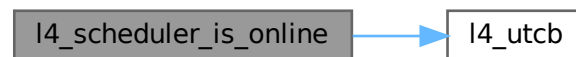
##### Return values

<i>true</i>	The CPU is online.
<i>false</i>	The CPU is offline

Definition at line 412 of file [scheduler.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.11.10.3.7 l4\_scheduler\_run\_thread()

```
l4_msgtag_t l4_scheduler_run_thread (
    l4_cap_idx_t scheduler,
```

```
l4_cap_idx_t thread,
l4_sched_param_t const * sp ) [inline]
```

Run a thread on a Scheduler.

#### Parameters

<i>scheduler</i>	Scheduler object.
<i>thread</i>	Capability of the thread to run.
<i>sp</i>	Scheduling parameters.

#### Return values

0	Success.
-L4_EINVAL	Invalid size of the scheduling parameter.

This function launches a thread on a CPU determined by the scheduling parameter `sp.affinity`. A thread can be intentionally stopped by migrating it on an offline or an invalid CPU. The thread is only guaranteed to run if the CPU it is migrated to is currently online.

#### Note

If the target CPU is currently not online, there is no guarantee that the thread will ever run, even if the CPU comes online later on.

A scheduler may impose a policy with regard to selecting CPUs. However the scheduler is required to ensure the following two properties:

- Two threads with disjoint CPU sets must be scheduled to different CPUs.
- Two threads with identical CPU sets selecting only a single CPU must be scheduled to the same CPU.

#### Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 398 of file [scheduler.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 13.1.11.11 Task

C interface of the Task kernel object, see [L4::Task](#) for the C++ interface.

Collaboration diagram for Task:



### Enumerations

- enum [l4\\_unmap\\_flags\\_t](#) { [L4\\_FP\\_ALL\\_SPACES](#) , [L4\\_FP\\_DELETE\\_OBJ](#) , [L4\\_FP\\_OTHER\\_SPACES](#) }  
*Flags for the unmap operation.*

### Functions

- [l4\\_msgtag\\_t l4\\_task\\_vgicc\\_map](#) ([l4\\_cap\\_idx\\_t](#) task, [l4\\_fpage\\_t](#) vgicc\_fpage) [L4\\_NOTHROW](#)  
*Map the GIC virtual CPU interface page to the task in case Fiasco detected a GIC version 2.*
- [l4\\_msgtag\\_t l4\\_task\\_map](#) ([l4\\_cap\\_idx\\_t](#) dst\_task, [l4\\_cap\\_idx\\_t](#) src\_task, [l4\\_fpage\\_t](#) snd\_fpage, [l4\\_umword\\_t](#) snd\_base) [L4\\_NOTHROW](#)  
*Map resources available in the source task to a destination task.*
- [l4\\_msgtag\\_t l4\\_task\\_unmap](#) ([l4\\_cap\\_idx\\_t](#) task, [l4\\_fpage\\_t](#) fpage, [l4\\_umword\\_t](#) map\_mask) [L4\\_NOTHROW](#)  
*Revoke rights from the task.*
- [l4\\_msgtag\\_t l4\\_task\\_unmap\\_batch](#) ([l4\\_cap\\_idx\\_t](#) task, [l4\\_fpage\\_t](#) const \*fpages, unsigned num\_fpages, [l4\\_umword\\_t](#) map\_mask) [L4\\_NOTHROW](#)  
*Revoke rights from a task.*
- [l4\\_msgtag\\_t l4\\_task\\_delete\\_obj](#) ([l4\\_cap\\_idx\\_t](#) task, [l4\\_cap\\_idx\\_t](#) obj) [L4\\_NOTHROW](#)  
*Release capability and delete object.*
- [l4\\_msgtag\\_t l4\\_task\\_release\\_cap](#) ([l4\\_cap\\_idx\\_t](#) task, [l4\\_cap\\_idx\\_t](#) cap) [L4\\_NOTHROW](#)  
*Release object capability.*
- [l4\\_msgtag\\_t l4\\_task\\_cap\\_valid](#) ([l4\\_cap\\_idx\\_t](#) task, [l4\\_cap\\_idx\\_t](#) cap) [L4\\_NOTHROW](#)  
*Check whether a capability is present (refers to an object).*
- [l4\\_msgtag\\_t l4\\_task\\_cap\\_equal](#) ([l4\\_cap\\_idx\\_t](#) task, [l4\\_cap\\_idx\\_t](#) cap\_a, [l4\\_cap\\_idx\\_t](#) cap\_b) [L4\\_NOTHROW](#)  
*Test whether two capabilities point to the same object with the same rights.*
- [l4\\_msgtag\\_t l4\\_task\\_add\\_ku\\_mem](#) ([l4\\_cap\\_idx\\_t](#) task, [l4\\_fpage\\_t](#) \*ku\_mem) [L4\\_NOTHROW](#)  
*Add kernel-user memory.*

#### 13.1.11.11.1 Detailed Description

C interface of the Task kernel object, see [L4::Task](#) for the C++ interface.

A task represents a combination of the address spaces provided by the [L4Re](#) micro kernel. A task consists of at least a memory address space and an object address space. On IA32 there is also an IO-port address space.

Task objects are created using the [Factory](#) interface.

### Include File

```
#include <l4/sys/task.h>
```

### 13.1.11.11.2 Enumeration Type Documentation

#### 13.1.11.11.2.1 l4\_unmap\_flags\_t

enum [l4\\_unmap\\_flags\\_t](#)

Flags for the unmap operation.

See also

[L4::Task::unmap\(\)](#) and [l4\\_task\\_unmap\(\)](#)

Enumerator

<a href="#">L4_FP_ALL_SPACES</a>	<p>Flag to tell the unmap operation to revoke permissions from all child mappings including the mapping in the invoked task.</p> <p><b>Note</b></p> <p>Object capabilities are not hierarchical – they have no children. The result of the map operation on an object capability is a copy of that capability in the object space of the destination task. An unmap operation on object capabilities is a no-op if this flag is not specified.</p> <p><b>See also</b></p> <p><a href="#">L4::Task::unmap()</a> <a href="#">l4_task_unmap()</a></p>
<a href="#">L4_FP_DELETE_OBJ</a>	<p>Flag that indicates that an unmap operation on object capabilities shall try to delete the corresponding objects immediately. This flag implies the <a href="#">L4_FP_ALL_SPACES</a> flag. The concept of deletion is only applicable to kernel objects. Therefore, for memory and I/O port capabilities, this flag has the same effect as <a href="#">L4_FP_ALL_SPACES</a> alone.</p> <p><b>See also</b></p> <p><a href="#">L4::Task::unmap()</a> <a href="#">l4_task_unmap()</a></p>
<a href="#">L4_FP_OTHER_SPACES</a>	<p>Counterpart to <a href="#">L4_FP_ALL_SPACES</a>; revoke permissions from child mappings only.</p> <p><b>See also</b></p> <p><a href="#">L4::Task::unmap()</a> <a href="#">l4_task_unmap()</a></p>

Definition at line [184](#) of file [consts.h](#).

### 13.1.11.11.3 Function Documentation

#### 13.1.11.11.3.1 l4\_task\_add\_ku\_mem()

```
l4\_msgtag\_t l4_task_add_ku_mem (
    l4\_cap\_idx\_t task,
    l4\_fpage\_t * ku_mem ) [inline]
```

Add kernel-user memory.



## Parameters

	<i>task</i>	Capability selector of the task to add the memory to.
<i>in, out</i>	<i>ku_mem</i>	Flexpage describing the virtual area the memory goes to. On systems without MMU, the flexpage is adjusted to reflect the actually allocated physical address.

## Returns

Syscall return tag

Kernel-user memory (ku\_mem) is memory that is shared between the kernel and user-space. It is needed for the UTCB area of threads (see [l4\\_thread\\_control\\_bind\(\)](#)) and for (extended) vCPU state. Note that existing kernel-user memory cannot be unmapped or mapped somewhere else.

## Note

The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page (`L4_PAGE_SIZE`). A portable implementation should not depend on allocations greater than 16KiB to succeed.

This function is only guaranteed to work on [L4::Task](#) objects. It might or might not work on [L4::Vm](#) objects or on [L4Re::Dma\\_space](#) objects but there is no practical use for adding kernel-user memory to [L4::Vm](#) objects or to [L4Re::Dma\\_space](#) objects.

Definition at line 468 of file [task.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



## 13.1.11.11.3.2 l4\_task\_cap\_equal()

```

l4_msgtag_t l4_task_cap_equal (
    l4_cap_idx_t task,
    l4_cap_idx_t cap_a,
    l4_cap_idx_t cap_b ) [inline]
  
```

Test whether two capabilities point to the same object with the same rights.

## Parameters

<i>task</i>	Capability selector of the destination task to do the lookup in
<i>cap<sub>a</sub></i>	Capability selector to compare
<i>cap<sub>b</sub></i>	Capability selector to compare

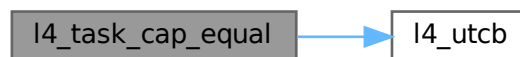
**Returns**

label contains 1 if equal, 0 if not equal

Definition at line 461 of file [task.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**13.1.11.11.3.3 l4\_task\_cap\_valid()**

```

l4_msgtag_t l4_task_cap_valid (
    l4_cap_idx_t task,
    l4_cap_idx_t cap ) [inline]
  
```

Check whether a capability is present (refers to an object).

**Parameters**

<i>task</i>	Task to check the capability in.
<i>cap</i>	Valid capability to check for presence.

**Return values**

<i>l4_msgtag_t::label()</i> > 0	Capability is present (refers to an object).
<i>l4_msgtag_t::label()</i> == 0	No capability present (void object).

A capability is considered present when it refers to an existing kernel object.

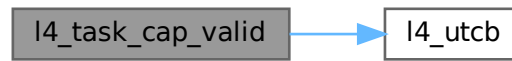
**Precondition**

*cap* must be a valid capability index (i.e. not L4\_INVALID\_CAP or the like).

Definition at line 455 of file [task.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.11.11.3.4 l4\_task\_delete\_obj()

```

l4_msgtag_t l4_task_delete_obj (
    l4_cap_idx_t task,
    l4_cap_idx_t obj ) [inline]
  
```

Release capability and delete object.

##### Parameters

<i>task</i>	Capability selector of destination task.
<i>obj</i>	Capability index of the object to delete.

##### Returns

Syscall return tag

If `obj` has the delete permission, initiates the deletion of the object. This implies that all capabilities for that object are gone afterwards. However, kernel-internally, objects are not destroyed until all other kernel objects holding a reference to it drop the reference. Hence, quota used by that object might not be freed immediately.

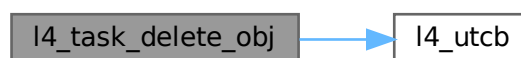
If `obj` does not have the delete permission, no error will be reported and only the capability `obj` is removed. (Note that, depending on the object's reference counter, this might still imply initiation of deletion.)

This operation is equivalent to [l4\\_task\\_unmap\(\)](#) with `L4_FP_DELETE_OBJ` flag.

Definition at line [434](#) of file [task.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 13.1.11.11.3.5 l4\_task\_map()

```
l4_msgtag_t l4_task_map (
    l4_cap_idx_t dst_task,
    l4_cap_idx_t src_task,
    l4_fpage_t snd_fpage,
    l4_umword_t snd_base ) [inline]
```

Map resources available in the source task to a destination task.

#### Parameters

<i>dst_task</i>	Capability selector of the destination task.
<i>src_task</i>	Capability selector of the source task.
<i>snd_fpage</i>	Send flexpage that describes an area in the address space or object space of the source task.
<i>snd_base</i>	Send base that describes an offset in the receive window of the destination task. The lower bits contain additional map control flags (see <a href="#">l4_fpage_cacheability_opt_t</a> for memory mappings, <a href="#">L4_obj_fpage_ctl</a> for object mappings, and <a href="#">L4_MAP_ITEM_GRANT</a> ; also see <a href="#">l4_map_control()</a> and <a href="#">l4_map_obj_control()</a> ).

#### Returns

Syscall return tag. The function [l4\\_error\(\)](#) shall be used to test if the map operation was successful.

#### Return values

<i>L4_EOK</i>	Operation successful (but see notes below).
<i>-L4_EPERM</i>	No <a href="#">L4_CAP_FPAGE_W</a> right on <i>dst_task</i> .
<i>-L4_EINVAL</i>	Invalid source task capability.
<i>-L4_IPC_SEMAPFAILED</i>	The map operation failed due to limited quota.

This method allows for asynchronous transfer of capabilities, memory mappings, and IO-port mappings (on IA32) from one task to another. The receive window is the whole address space of *dst\_task*. By specifying proper rights in *snd\_fpage* and *snd\_base*, it is possible to remove rights during transfer.

#### Note

If the send flex page is of type [L4\\_FPAGE\\_OBJ](#), the [L4\\_CAP\\_FPAGE\\_S](#) right is removed from the transferred capability unless both the source and destination task capabilities possess the [L4\\_CAP\\_FPAGE\\_S](#) right themselves.

Even with [l4\\_error\(\)](#) returning [L4\\_EOK](#) there might be cases where not all pages of the send flexpage were mapped respectively granted to the destination task, for instance, if the corresponding mapping in the destination task does already exist.

For more information on spaces and mappings, see [Spaces and Mappings](#). The flexpage API is described in more detail at [Flex pages](#).

**Note**

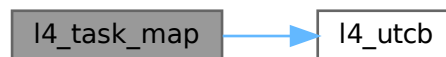
For peculiarities when using grant, see [L4\\_MAP\\_ITEM\\_GRANT](#).

Definition at line 404 of file [task.h](#).

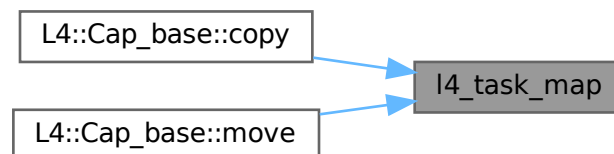
References [l4\\_utcb\(\)](#).

Referenced by [L4::Cap\\_base::copy\(\)](#), and [L4::Cap\\_base::move\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.11.11.3.6 l4\_task\_release\_cap()

```

l4_msgtag_t l4_task_release_cap (
    l4_cap_idx_t task,
    l4_cap_idx_t cap ) [inline]
  
```

Release object capability.

**Parameters**

<i>task</i>	Capability selector of destination task
<i>cap</i>	Capability selector of object to release

**Returns**

Syscall return tag

This operation unmaps the capability from the specified task.

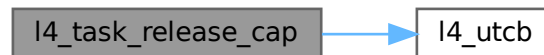
**Note**

If the reference counter of the kernel object referenced by `cap` goes down to zero, deletion of the object is initiated. Objects are not destroyed until all other kernel objects holding a reference to it drop the reference.

Definition at line 449 of file [task.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**13.1.11.11.3.7 l4\_task\_unmap()**

```

l4_msgtag_t l4_task_unmap (
    l4_cap_idx_t task,
    l4_fpage_t fpage,
    l4_umword_t map_mask ) [inline]
  
```

Revoke rights from the task.

**Parameters**

<i>task</i>	Capability selector of destination task
<i>fpage</i>	Flexpage that describes an area in one capability space of the destination task
<i>map_mask</i>	Unmap mask, see <a href="#">l4_unmap_flags_t</a>

**Returns**

Syscall return tag

This method allows to revoke rights from the destination task. For a flex page describing IO ports or memory, it also revokes rights from all the tasks that got the rights delegated from the destination task (i.e., this operation does a recursive rights revocation). If the set of rights is empty after the revocation, the capability is unmapped. It is guaranteed that the rights revocation is completed before this function returns.

**Note**

If the reference counter of a kernel object referenced in `fpage` goes down to zero (as a result of deleting capabilities), the deletion of the object is initiated. Objects are not destroyed until all other kernel objects holding a reference to it drop the reference.

**Examples**

[examples/sys/utcb-ipc/main.c](#).

Definition at line 411 of file [task.h](#).

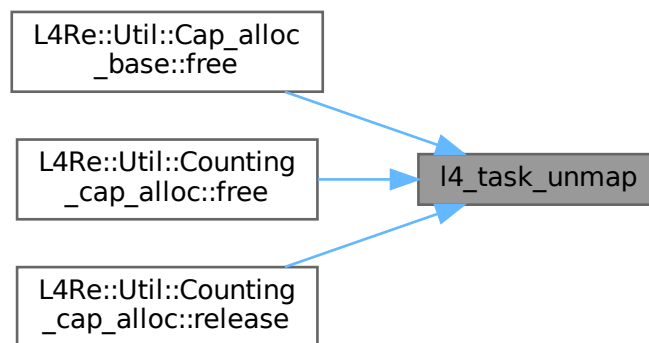
References [l4\\_utcb\(\)](#).

Referenced by [L4Re::Util::Cap\\_alloc\\_base::free\(\)](#), [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE, Dbg >::free\(\)](#), and [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE, Dbg >::release\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**13.1.11.11.3.8 l4\_task\_unmap\_batch()**

```

l4_msgtag_t l4_task_unmap_batch (
    l4_cap_idx_t task,
    l4_fpage_t const * fpages,
    unsigned num_fpages,
    l4_umword_t map_mask ) [inline]
  
```

Revoke rights from a task.

**Parameters**

<i>task</i>	Capability selector of destination task
<i>fpages</i>	An array of flexpages. Each item describes an area in one capability space of the destination task.
<i>num_fpages</i>	The size of the fpages array in elements (number of fpages sent).
<i>map_mask</i>	Unmap mask, see <a href="#">l4_unmap_flags_t</a>

**Returns**

Syscall return tag

Revoke rights specified in an array of flexpages, see [l4\\_task\\_unmap](#) for details.

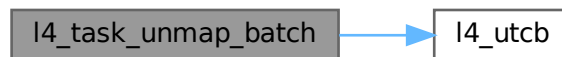
**Precondition**

The caller needs to take care that `num_fpages` is not bigger than `L4_UTCB_GENERIC_DATA_SIZE - 2`.

Definition at line [418](#) of file [task.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**13.1.11.11.3.9 l4\_task\_vgicc\_map()**

```

l4_msgtag_t l4_task_vgicc_map (
    l4_cap_idx_t task,
    l4_fpage_t vgicc_fpage ) [inline]
  
```

Map the GIC virtual CPU interface page to the task in case Fiasco detected a GIC version 2.

**Parameters**

<i>task</i>	Capability selector of destination task
<i>vgicc_fpage</i>	Flexpage that describes an area in the address space of the destination task to map the vGICC page to



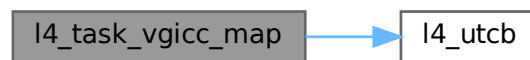
**Returns**

Syscall return tag

Definition at line 57 of file [\\_\\_task-arm.h](#).

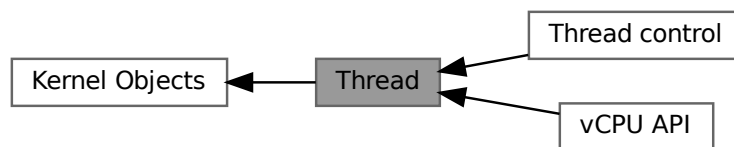
References [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**13.1.11.12 Thread**

C Thread object interface, see [L4::Thread](#) for the C++ interface.

Collaboration diagram for Thread:

**Modules**

- [Thread control](#)  
*API for Thread Control method.*
- [vCPU API](#)  
*vCPU API.*

## Enumerations

- enum `L4_thread_control_flags` {  
`L4_THREAD_CONTROL_SET_PAGER` = 0x0010000 , `L4_THREAD_CONTROL_BIND_TASK` = 0x0200000  
, `L4_THREAD_CONTROL_ALIEN` = 0x0400000 , `L4_THREAD_CONTROL_UX_NATIVE` = 0x0800000 ,  
`L4_THREAD_CONTROL_SET_EXC_HANDLER` = 0x1000000 }

*Flags for the thread control operation.*

- enum `L4_thread_control_mr_indices` {  
`L4_THREAD_CONTROL_MR_IDX_FLAGS` = 0 , `L4_THREAD_CONTROL_MR_IDX_PAGER` = 1 ,  
`L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER` = 2 , `L4_THREAD_CONTROL_MR_IDX_FLAG_VALS`  
= 4 ,  
`L4_THREAD_CONTROL_MR_IDX_BIND_UTCB` = 5 , `L4_THREAD_CONTROL_MR_IDX_BIND_TASK` = 6  
}

*Indices for the values in the message register for thread control.*

- enum `L4_thread_ex_regs_flags` { `L4_THREAD_EX_REGS_CANCEL` = 0x10000UL , `L4_THREAD_EX_REGS_TRIGGER_EXC`  
= 0x20000UL , `L4_THREAD_EX_REGS_ARCH_MASK` = 0xff000000UL }

*Flags for the thread ex-regs operation.*

- enum `L4_thread_ex_regs_flags_arm` { `L4_THREAD_EX_REGS_ARM_SET_EL_MASK` = 0x3 << 24 ,  
`L4_THREAD_EX_REGS_ARM_SET_EL_KEEP` = 0x0 << 24 , `L4_THREAD_EX_REGS_ARM_SET_EL_EL0`  
= 0x1 << 24 , `L4_THREAD_EX_REGS_ARM_SET_EL_EL1` = 0x2 << 24 }

*Arm specific `L4::Thread::ex_regs()` flags.*

## Functions

- `l4_msgtag_t l4_thread_ex_regs` (`l4_cap_idx_t` thread, `l4_addr_t` ip, `l4_addr_t` sp, `l4_umword_t` flags)  
`L4_NOTHROW`

*Exchange basic thread registers.*

- `l4_msgtag_t l4_thread_ex_regs_u` (`l4_cap_idx_t` thread, `l4_addr_t` ip, `l4_addr_t` sp, `l4_umword_t` flags,  
`l4_utcb_t` \*utcb) `L4_NOTHROW`

*Exchange basic thread registers.*

- `l4_msgtag_t l4_thread_ex_regs_ret` (`l4_cap_idx_t` thread, `l4_addr_t` \*ip, `l4_addr_t` \*sp, `l4_umword_t` \*flags)  
`L4_NOTHROW`

*Exchange basic thread registers and return previous values.*

- `l4_msgtag_t l4_thread_ex_regs_ret_u` (`l4_cap_idx_t` thread, `l4_addr_t` \*ip, `l4_addr_t` \*sp, `l4_umword_t`  
\*flags, `l4_utcb_t` \*utcb) `L4_NOTHROW`

*Exchange basic thread registers and return previous values.*

- `l4_msgtag_t l4_thread_yield` (void) `L4_NOTHROW`

*Yield current time slice.*

- `l4_msgtag_t l4_thread_switch` (`l4_cap_idx_t` to\_thread) `L4_NOTHROW`

*Switch to another thread (and donate the remaining time slice).*

- `l4_msgtag_t l4_thread_stats_time` (`l4_cap_idx_t` thread, `l4_kernel_clock_t` \*us) `L4_NOTHROW`

*Get consumed time of thread in  $\mu$ s.*

- `l4_msgtag_t l4_thread_vcpu_resume_start` (void) `L4_NOTHROW`

*vCPU return from event handler.*

- `l4_msgtag_t l4_thread_vcpu_resume_commit` (`l4_cap_idx_t` thread, `l4_msgtag_t` tag) `L4_NOTHROW`

*Commit vCPU resume.*

- `l4_msgtag_t l4_thread_vcpu_control` (`l4_cap_idx_t` thread, `l4_addr_t` vcpu\_state) `L4_NOTHROW`

*Enable the vCPU feature for the thread.*

- `l4_msgtag_t l4_thread_vcpu_control_u` (`l4_cap_idx_t` thread, `l4_addr_t` vcpu\_state, `l4_utcb_t` \*utcb)  
`L4_NOTHROW`

*Enable the vCPU feature for the thread.*

- `l4_msgtag_t l4_thread_vcpu_control_ext` (`l4_cap_idx_t` thread, `l4_addr_t` ext\_vcpu\_state) `L4_NOTHROW`

- Enable the extended vCPU feature for the thread.*

  - `l4_msgtag_t l4_thread_vcpu_control_ext_u (l4_cap_idx_t thread, l4_addr_t ext_vcpu_state, l4_utcb_t *utcb) L4_NOTHROW`
- Enable the extended vCPU feature for the thread.*

  - `l4_msgtag_t l4_thread_register_del_irq (l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW`
- Register an IRQ that will trigger upon deletion events.*

  - `l4_msgtag_t l4_thread_modify_sender_start (void) L4_NOTHROW`
- Start a thread sender modification sequence.*

  - `int l4_thread_modify_sender_add (l4_umword_t match_mask, l4_umword_t match, l4_umword_t del_bits, l4_umword_t add_bits, l4_msgtag_t *tag) L4_NOTHROW`
- Add a modification pattern to a sender modification sequence.*

  - `l4_msgtag_t l4_thread_modify_sender_commit (l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW`
- Apply (commit) a sender modification sequence.*

  - `l4_msgtag_t l4_thread_arm_set_tpidruro (l4_cap_idx_t thread, l4_addr_t tpidruro) L4_NOTHROW`
- Set the TPIDRURO thread specific register.*

### 13.1.11.12.1 Detailed Description

C Thread object interface, see [L4::Thread](#) for the C++ interface.

An [L4](#) thread is a thread of execution in the [L4](#) context. Usually user-level and kernel threads are mapped 1:1 to each other. Thread kernel objects are created using a factory, see [Factory](#) (`l4_factory_create_thread()`).

Amongst other things an [L4](#) thread encapsulates:

- CPU state
  - General-purpose registers
  - Program counter
  - Stack pointer
- FPU state
- Scheduling parameters, see the [Scheduler](#) API
- Execution state
  - Blocked, Runnable, Running

Thread objects provide an API for

- Thread configuration and manipulation
- Thread switching.

The thread control functions are used to control various aspects of a thread. See `l4_thread_control_start()` for more information.

#### Include File

```
#include <l4/sys/thread.h>
```

For the C++ interface refer to [L4::Thread](#).

### 13.1.11.12.2 Enumeration Type Documentation

#### 13.1.11.12.2.1 L4\_thread\_control\_flags

```
enum L4_thread_control_flags
```

Flags for the thread control operation.

## Enumerator

L4_THREAD_CONTROL_SET_PAGER	The pager will be given.
L4_THREAD_CONTROL_BIND_TASK	The task to bind the thread to will be given.
L4_THREAD_CONTROL_ALIEN	Alien state of the thread is set.
L4_THREAD_CONTROL_UX_NATIVE	Fiasco-UX only: pass-through of host system calls is set.
L4_THREAD_CONTROL_SET_EXC_HANDLER	The exception handler of the thread will be given.

Definition at line 721 of file [thread.h](#).

## 13.1.11.12.2.2 L4\_thread\_control\_mr\_indices

```
enum L4_thread_control_mr_indices
```

Indices for the values in the message register for thread control.

## Enumerator

L4_THREAD_CONTROL_MR_IDX_FLAGS	See also <a href="#">L4_thread_control_flags</a> .
L4_THREAD_CONTROL_MR_IDX_PAGER	Index for pager cap.
L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER	Index for exception handler.
L4_THREAD_CONTROL_MR_IDX_FLAG_VALS	Index for feature values.
L4_THREAD_CONTROL_MR_IDX_BIND_UTCB	Index for UTCB address for bind.
L4_THREAD_CONTROL_MR_IDX_BIND_TASK	Index for task flex-page for bind.

Definition at line 744 of file [thread.h](#).

## 13.1.11.12.2.3 L4\_thread\_ex\_regs\_flags

```
enum L4_thread_ex_regs_flags
```

Flags for the thread ex-regs operation.

## Enumerator

L4_THREAD_EX_REGS_CANCEL	Cancel ongoing IPC in the thread.
L4_THREAD_EX_REGS_TRIGGER_EXCEPTION	Trigger artificial exception in thread.
L4_THREAD_EX_REGS_ARCH_MASK	Arch specific flags.

Definition at line 759 of file [thread.h](#).

## 13.1.11.12.2.4 L4\_thread\_ex\_regs\_flags\_arm

```
enum L4_thread_ex_regs_flags_arm
```

Arm specific [L4::Thread::ex\\_regs\(\)](#) flags.

Only one option must be used in calls to [L4::Thread::ex\\_regs\(\)](#). Using more than one option results in undefined behaviour.

#### Enumerator

L4_THREAD_EX_REGS_ARM_SET_EL_MASK	Exception level set mask.
L4_THREAD_EX_REGS_ARM_SET_EL_KEEP	Keep current exception level of thread (default).
L4_THREAD_EX_REGS_ARM_SET_EL_EL0	Set exception level of thread to EL0 (usr mode).
L4_THREAD_EX_REGS_ARM_SET_EL_EL1	Set exception level of thread to EL1 (sys mode).

Definition at line 53 of file [thread.h](#).

### 13.1.11.12.3 Function Documentation

#### 13.1.11.12.3.1 l4\_thread\_arm\_set\_tpidruro()

```
l4_msgtag_t l4_thread_arm_set_tpidruro (
    l4_cap_idx_t thread,
    l4_addr_t tpidruro ) [inline]
```

Set the TPIDRURO thread specific register.

#### Parameters

<i>thread</i>	Thread to manipulate
<i>tpidruro</i>	The value to be set

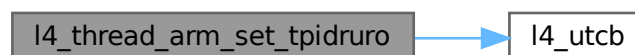
#### Returns

System call return tag

Definition at line 79 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 13.1.11.12.3.2 l4\_thread\_ex\_regs()

```
l4_msgtag_t l4_thread_ex_regs (
    l4_cap_idx_t thread,
    l4_addr_t ip,
    l4_addr_t sp,
    l4_umword_t flags ) [inline]
```

Exchange basic thread registers.

#### Parameters

<i>thread</i>	Capability selector of the thread to manipulate.
<i>ip</i>	New instruction pointer, use ~0UL to leave the instruction pointer unchanged.
<i>sp</i>	New stack pointer, use ~0UL to leave the stack pointer unchanged.
<i>flags</i>	Ex-regs flags, see <a href="#">L4_thread_ex_regs_flags</a> .

#### Returns

System call return tag

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if [L4\\_THREAD\\_EX\\_REGS\\_TRIGGER\\_EXCEPTION](#) forces an IPC then changes in IP and SP take effect directly after returning from this IPC.

The thread is started using [l4\\_scheduler\\_run\\_thread\(\)](#). However, if at the time [l4\\_scheduler\\_run\\_thread\(\)](#) is called, the instruction pointer of the thread is invalid, a later call to [l4\\_thread\\_ex\\_regs\(\)](#) with a valid instruction pointer might start the thread.

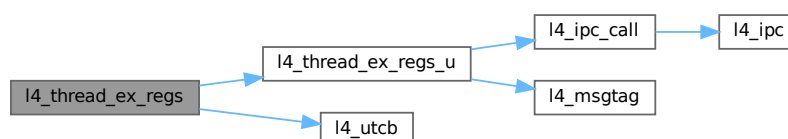
#### Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 912 of file [thread.h](#).

References [l4\\_thread\\_ex\\_regs\\_u\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 13.1.11.12.3.3 `l4_thread_ex_regs_ret()`

```
l4_msgtag_t l4_thread_ex_regs_ret (
    l4_cap_idx_t thread,
    l4_addr_t * ip,
    l4_addr_t * sp,
    l4_umword_t * flags ) [inline]
```

Exchange basic thread registers and return previous values.

#### Parameters

	<i>thread</i>	Capability selector of the thread to manipulate.
in, out	<i>ip</i>	New instruction pointer, use ~0UL to leave the instruction pointer unchanged, return previous instruction pointer.
in, out	<i>sp</i>	New stack pointer, use ~0UL to leave the stack pointer unchanged, returns previous stack pointer.
in, out	<i>flags</i>	Ex-regs flags, see <a href="#">L4_thread_ex_regs_flags</a> , return previous CPU flags of the thread.

#### Returns

System call return tag

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if [L4\\_THREAD\\_EX\\_REGS\\_TRIGGER\\_EXCEPTION](#) forces an IPC then changes in IP and SP take effect directly after returning from this IPC.

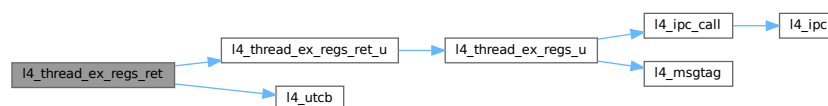
The thread is started using [l4\\_scheduler\\_run\\_thread\(\)](#). However, if at the time [l4\\_scheduler\\_run\\_thread\(\)](#) is called, the instruction pointer of the thread is invalid, a later call to [l4\\_thread\\_ex\\_regs\(\)](#) with a valid instruction pointer might start the thread.

Returned values are valid only if function returns successfully.

Definition at line 919 of file [thread.h](#).

References [l4\\_thread\\_ex\\_regs\\_ret\\_u\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 13.1.11.12.3.4 `l4_thread_ex_regs_ret_u()`

```
l4_msgtag_t l4_thread_ex_regs_ret_u (
    l4_cap_idx_t thread,
    l4_addr_t * ip,
    l4_addr_t * sp,
    l4_umword_t * flags,
    l4_utcb_t * utcb ) [inline]
```

Exchange basic thread registers and return previous values.

## Parameters

	<i>thread</i>	Capability selector of the thread to manipulate.
in, out	<i>ip</i>	New instruction pointer, use ~0UL to leave the instruction pointer unchanged, return previous instruction pointer.
in, out	<i>sp</i>	New stack pointer, use ~0UL to leave the stack pointer unchanged, returns previous stack pointer.
in, out	<i>flags</i>	Ex-regs flags, see <a href="#">L4_thread_ex_regs_flags</a> , return previous CPU flags of the thread.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

## Returns

System call return tag. [out] parameters are only valid if the function returns successfully. Use [l4\\_error\(\)](#) to check.

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see *flags*). If the thread is in an IPC operation or if [L4\\_THREAD\\_EX\\_REGS\\_TRIGGER\\_EXCEPTION](#) forces an IPC then changes in IP and SP take effect directly after returning from this IPC.

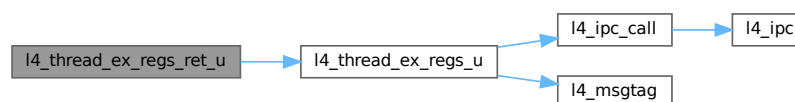
The thread is started using [L4::Scheduler::run\\_thread\(\)](#). However, if at the time [L4::Scheduler::run\\_thread\(\)](#) is called, the instruction pointer of the thread is invalid, a later call to *ex\_regs()* with a valid instruction pointer might start the thread.

Definition at line 785 of file [thread.h](#).

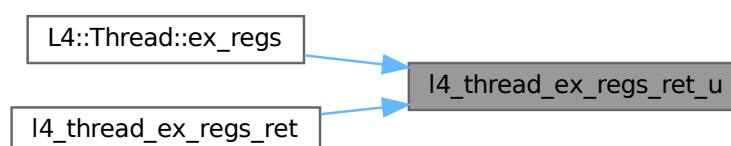
References [l4\\_thread\\_ex\\_regs\\_u\(\)](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [L4::Thread::ex\\_regs\(\)](#), and [l4\\_thread\\_ex\\_regs\\_ret\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:





13.1.11.12.3.5 `l4_thread_ex_regs_u()`

```
l4_msgtag_t l4_thread_ex_regs_u (
    l4_cap_idx_t thread,
    l4_addr_t ip,
    l4_addr_t sp,
    l4_umword_t flags,
    l4_utcb_t * utcb ) [inline]
```

Exchange basic thread registers.

## Parameters

<i>thread</i>	Capability selector of the thread to manipulate.
<i>ip</i>	New instruction pointer, use <code>~0UL</code> to leave the instruction pointer unchanged.
<i>sp</i>	New stack pointer, use <code>~0UL</code> to leave the stack pointer unchanged.
<i>flags</i>	Ex-regs flags, see <a href="#">L4_thread_ex_regs_flags</a> .
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

## Returns

System call return tag.

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if [L4\\_THREAD\\_EX\\_REGS\\_TRIGGER\\_EXCEPTION](#) forces an IPC then changes in IP and SP take effect directly after returning from this IPC.

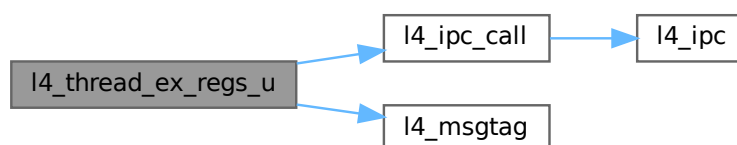
The thread is started using [L4::Scheduler::run\\_thread\(\)](#). However, if at the time [L4::Scheduler::run\\_thread\(\)](#) is called, the instruction pointer of the thread is invalid, a later call to `ex_regs()` with a valid instruction pointer might start the thread.

Definition at line 774 of file [thread.h](#).

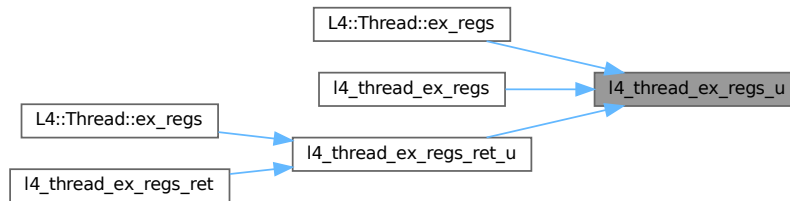
References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_THREAD](#), [L4\\_THREAD\\_EX\\_REGS\\_OP](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [L4::Thread::ex\\_regs\(\)](#), [l4\\_thread\\_ex\\_regs\(\)](#), and [l4\\_thread\\_ex\\_regs\\_ret\\_u\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 13.1.11.12.3.6 l4\_thread\_modify\_sender\_add()

```

int l4_thread_modify_sender_add (
    l4_umword_t match_mask,
    l4_umword_t match,
    l4_umword_t del_bits,
    l4_umword_t add_bits,
    l4_msgtag_t * tag ) [inline]

```

Add a modification pattern to a sender modification sequence.

#### Parameters

<i>tag</i>	Tag received from <a href="#">l4_thread_modify_sender_start()</a> or previous <a href="#">l4_thread_modify_sender_add()</a> calls from the same sequence.
<i>match_mask</i>	Bitmask of bits to match the label.
<i>match</i>	Bitmask that must be equal to the label after applying <i>match_mask</i> .
<i>del_bits</i>	Bits to be deleted from the label.
<i>add_bits</i>	Bits to be added to the label.

#### Returns

0 on success, <0 on error

In pseudo code: if ((sender\_label & match\_mask) == match) { sender\_label = (sender\_label & ~del\_bits) | add\_bits; }

Only the first match is applied.

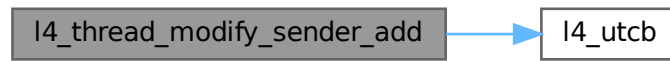
#### See also

[l4\\_thread\\_modify\\_sender\\_start](#)  
[l4\\_thread\\_modify\\_sender\\_commit](#)

Definition at line 1092 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.11.12.3.7 l4\_thread\_modify\_sender\_commit()

```
l4_msgtag_t l4_thread_modify_sender_commit (  
    l4_cap_idx_t thread,  
    l4_msgtag_t tag ) [inline]
```

Apply (commit) a sender modification sequence.

The modification rules are applied to all IPCs to the thread (whether directly or by IPC gate) that are already in flight, that is that the sender is already blocking on.

##### Note

Modifying the senders of a thread running on a different CPU core is not supported.

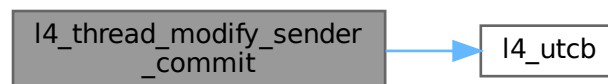
##### See also

[l4\\_thread\\_modify\\_sender\\_start](#)  
[l4\\_thread\\_modify\\_sender\\_add](#)

Definition at line 1103 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 13.1.11.12.3.8 l4\_thread\_modify\_sender\_start()

```
l4_msgtag_t l4_thread_modify_sender_start (
    void ) [inline]
```

Start a thread sender modification sequence.

Add modification rules with [l4\\_thread\\_modify\\_sender\\_add\(\)](#) and commit with [l4\\_thread\\_modify\\_sender\\_commit\(\)](#). Do not touch the UTCB between [l4\\_thread\\_modify\\_sender\\_start\(\)](#) and [l4\\_thread\\_modify\\_sender\\_commit\(\)](#).

This mechanism shall be used to change the source object labels of every pending IPC of an IPC gate or an IRQ if the labels in such pending IPC become invalid for the receiving thread, potentially because:

- a thread was unbound from an IPC gate / IRQ, or
- an IPC gate /IRQ was removed, or
- the label of an IPC gate /IRQ bound to a thread was changed.

It is not required to perform the modify\_sender mechanism after an IPC gate or an IRQ was bound to a thread for the first time.

See also

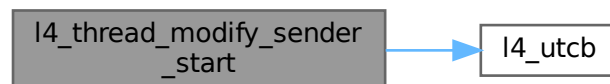
[l4\\_thread\\_modify\\_sender\\_add](#)

[l4\\_thread\\_modify\\_sender\\_commit](#)

Definition at line 1086 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 13.1.11.12.3.9 l4\_thread\_register\_del\_irq()

```
l4_msgtag_t l4_thread_register_del_irq (
    l4_cap_idx_t thread,
    l4_cap_idx_t irq ) [inline]
```

Register an IRQ that will trigger upon deletion events.

## Parameters

<i>thread</i>	Thread to register IRQ for.
<i>irq</i>	Capability selector for the IRQ object to be triggered.

## Returns

System call return tag containing the return code.

## Return values

<code>-L4_EPERM</code>	<code>L4_CAP_FPAGE_W</code> missing on <code>irq</code>
------------------------	---

In case the `irq` is already bound to an interrupt source, it is unbound first. When `irq` is deleted, it will be deregistered first. A registered deletion `Irq` can only be deregistered by deleting the `Irq` or the thread.

List of deletion events:

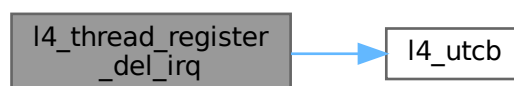
- deletion of one or several IPC gates bound to this thread.

When the deletion event is delivered, there is no indication about which IPC gate was deleted.

Definition at line 1013 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



## 13.1.11.12.3.10 l4\_thread\_stats\_time()

```

l4_msgtag_t l4_thread_stats_time (
    l4_cap_idx_t thread,
    l4_kernel_clock_t * us ) [inline]
  
```

Get consumed time of thread in  $\mu$ s.

## Parameters

	<i>thread</i>	Thread to get the consumed time from.
<i>out</i>	<i>us</i>	Consumed time in $\mu$ s.

**Returns**

system call return tag

Definition at line 981 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**13.1.11.12.3.11 l4\_thread\_switch()**

```
l4_msgtag_t l4_thread_switch (
    l4_cap_idx_t to_thread ) [inline]
```

Switch to another thread (and donate the remaining time slice).

**Parameters**

<i>to_thread</i>	The thread to switch to.
------------------	--------------------------

**Returns**

system call return tag

Definition at line 972 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 13.1.11.12.3.12 l4\_thread\_vcpu\_control()

```
l4_msgtag_t l4_thread_vcpu_control (
    l4_cap_idx_t thread,
    l4_addr_t vcpu_state ) [inline]
```

Enable the vCPU feature for the thread.

#### Parameters

<i>thread</i>	Capability selector of the thread for which the vCPU feature shall be enabled.
<i>vcpu_state</i>	The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see <a href="#">l4_task_add_ku_mem()</a> ).

#### Returns

Syscall return tag.

This function enables the vCPU feature of the `thread`.

The kernel-user memory area starting at `vcpu_state` must be at least 128-byte aligned and must cover the size of [l4\\_vcpu\\_state\\_t](#).

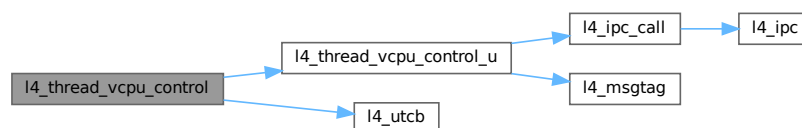
#### Note

Disabling of the vCPU feature is optional and currently not supported.

Definition at line 1030 of file [thread.h](#).

References [l4\\_thread\\_vcpu\\_control\\_u\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 13.1.11.12.3.13 l4\_thread\_vcpu\_control\_ext()

```
l4_msgtag_t l4_thread_vcpu_control_ext (
    l4_cap_idx_t thread,
    l4_addr_t ext_vcpu_state ) [inline]
```

Enable the extended vCPU feature for the thread.

## Parameters

<i>thread</i>	Capability selector of the thread for which the extended vCPU feature shall be enabled.
<i>ext_vcpu_state</i>	The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see <a href="#">l4_task_add_ku_mem()</a> ).

## Returns

Syscall return tag.

The extended vCPU feature allows the use of hardware-virtualization features such as Intel's VT or AMD's SVM.

This function enables the extended vCPU feature of the `thread`. Enabling the extended vCPU feature also enables the vCPU feature.

The kernel-user memory area starting at `ext_vcpu_state` must be at least 4 KiB aligned and must cover a size of `L4_PAGESIZE`. It includes the data of [l4\\_vcpu\\_state\\_t](#) at offset 0, the extended vCPU state at offset `L4_VCPU_OFFSET_EXT_STATE`, and, on some platforms, the extended vCPU information at offset `L4_VCPU_OFFSET_EXT_INFOS`.

## Note

Enabling the extended vCPU feature for a thread running on a different CPU core is currently not supported.

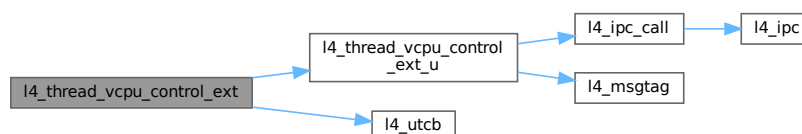
Disabling of the extended vCPU feature is currently not supported.

Upgrading from non-extended vCPU feature to extended vCPU feature is currently not supported.

Definition at line 1045 of file [thread.h](#).

References [l4\\_thread\\_vcpu\\_control\\_ext\\_u\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 13.1.11.12.3.14 l4\_thread\_vcpu\_control\_ext\_u()

```

l4_msgtag_t l4_thread_vcpu_control_ext_u (
    l4_cap_idx_t thread,
    l4_addr_t ext_vcpu_state,
    l4_utcb_t * utcb ) [inline]

```

Enable the extended vCPU feature for the thread.



## Parameters

<i>thread</i>	Capability selector of the thread for which the extended vCPU feature shall be enabled.
<i>ext_vcpu_state</i>	The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see <a href="#">L4::Task::add_ku_mem()</a> ).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

## Returns

Syscall return tag.

The extended vCPU feature allows the use of hardware-virtualization features such as Intel's VT or AMD's SVM.

This function enables the extended vCPU feature of `this` thread. Enabling the extended vCPU feature also enables the vCPU feature.

The kernel-user memory area starting at `ext_vcpu_state` must be at least 4 KiB aligned and must cover a size of `L4_PAGESIZE`. It includes the data of [l4\\_vcpu\\_state\\_t](#) at offset 0, the extended vCPU state at offset `L4_VCPU_OFFSET_EXT_STATE`, and, on some platforms, the extended vCPU information at offset `L4_VCPU_OFFSET_EXT_INFOS`.

## Note

Enabling the extended vCPU feature for a thread running on a different CPU core is currently not supported.

Disabling of the extended vCPU feature is currently not supported.

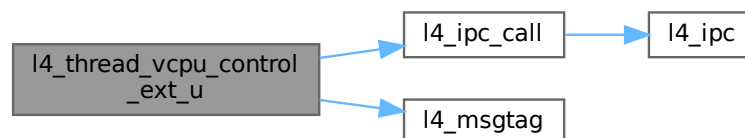
Upgrading from non-extended vCPU feature to extended vCPU feature is currently not supported.

Definition at line 1035 of file [thread.h](#).

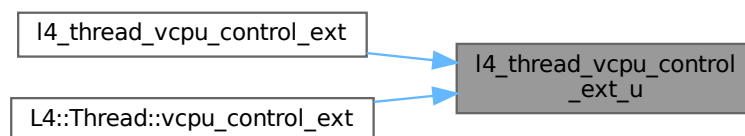
References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_THREAD](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [l4\\_thread\\_vcpu\\_control\\_ext\(\)](#), and [L4::Thread::vcpu\\_control\\_ext\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 13.1.11.12.3.15 l4\_thread\_vcpu\_control\_u()

```
l4_msgtag_t l4_thread_vcpu_control_u (
    l4_cap_idx_t thread,
    l4_addr_t vcpu_state,
    l4_utcb_t * utcb ) [inline]
```

Enable the vCPU feature for the thread.

#### Parameters

<i>thread</i>	Capability selector of the thread for which the vCPU feature shall be enabled.
<i>vcpu_state</i>	A virtual address pointing to a <a href="#">l4_vcpu_state_t</a> . It must be a valid kernel-user-memory address (see <a href="#">L4::Task::add_ku_mem()</a> ).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

#### Returns

Syscall return tag.

This function enables the vCPU feature of `this` thread

The kernel-user memory starting at `vcpu_state` must be at least 128-byte aligned and must cover the size of [l4\\_vcpu\\_state\\_t](#).

The asynchronous IPC handling is described at [vCPU API](#).

#### Note

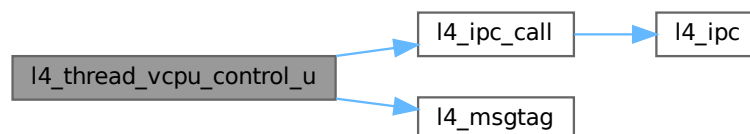
Disabling of the vCPU feature is optional and currently not supported.

Definition at line [1020](#) of file [thread.h](#).

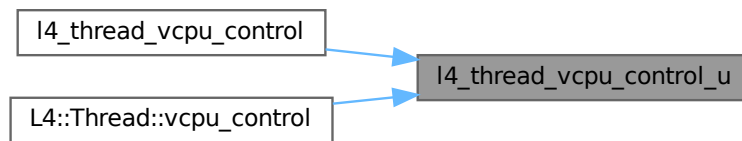
References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_THREAD](#), [L4\\_THREAD\\_VCPU\\_CONTROL\\_OP](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [l4\\_thread\\_vcpu\\_control\(\)](#), and [L4::Thread::vcpu\\_control\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 13.1.11.12.3.16 l4\_thread\_vcpu\_resume\_commit()

```
l4_msgtag_t l4_thread_vcpu_resume_commit (
    l4_cap_idx_t thread,
    l4_msgtag_t tag ) [inline]
```

Commit vCPU resume.

#### Parameters

<i>thread</i>	Thread to be resumed, the invalid cap can be used for the current thread.
<i>tag</i>	Tag to use, returned by <a href="#">l4_thread_vcpu_resume_start()</a>

#### Returns

Syscall return tag containing one of the following return codes.

#### Return values

0	Indicates a VM exit, provided that <i>thread</i> is in extended vCPU mode with virtual interrupts cleared.
1	Indicates an incoming IPC message, provided that the <i>thread</i> is in extended vCPU mode with virtual interrupts cleared.
-L4_EPERM	The user task capability set in the vCPU state is missing the <a href="#">L4_CAP_FPAGE_S</a> right.
-L4_ENOENT	The user task capability set in the vCPU state is invalid.
-L4_EINVAL	<i>thread</i> is not the current running thread, or does not have the vCPU feature enabled.
<0	A supplied mapping failed.

All flex pages in the UTCB (added with [l4\\_sndfpage\\_add\(\)](#) after [l4\\_thread\\_vcpu\\_resume\\_start\(\)](#)) are unconditionally mapped into the user task configured in the vCPU state.

To resume into another address space, the capability to the target [Task](#) (or [L4::Vm](#)) must be set in [l4\\_vcpu\\_state\\_t::user\\_task](#) together with [L4\\_VCPU\\_F\\_USER\\_MODE](#). The capability selector must have all lower bits clear (see [L4\\_CAP\\_MASK](#)). The kernel adds the [L4\\_SYSF\\_SEND](#) flag there to indicate that the capability has been referenced in the kernel. Consecutive resumes will not reference the task capability again until all lower bits are cleared again. To release a task use a different task capability or use an invalid capability with the [L4\\_SYSF\\_REPLY](#) flag set.

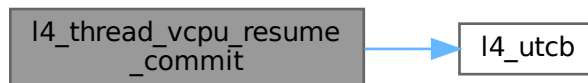
See also

[l4\\_vcpu\\_state\\_t](#)

Definition at line 993 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.11.12.3.17 l4\_thread\_vcpu\_resume\_start()

```
l4_msgtag_t l4_thread_vcpu_resume_start (  
    void ) [inline]
```

vCPU return from event handler.

##### Returns

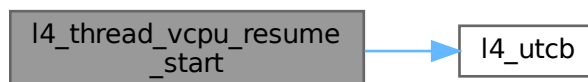
Message tag to be used for [l4\\_sndfpage\\_add\(\)](#) and [l4\\_thread\\_vcpu\\_resume\\_commit\(\)](#)

The vCPU resume functionality is split in multiple functions to allow the specification of additional send-flex-pages using [l4\\_sndfpage\\_add\(\)](#).

Definition at line 987 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.11.12.3.18 l4\_thread\_yield()

```
l4_msgtag_t l4_thread_yield (  
    void ) [inline]
```

Yield current time slice.

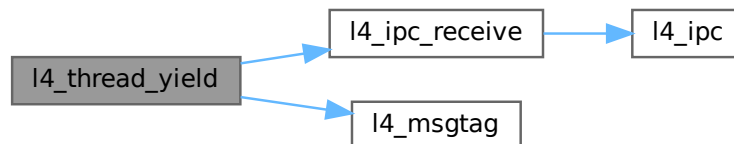
##### Returns

system call return tag

Definition at line 861 of file [thread.h](#).

References [L4\\_INVALID\\_CAP](#), [L4\\_IPC\\_BOTH\\_TIMEOUT\\_0](#), [l4\\_ipc\\_receive\(\)](#), and [l4\\_msgtag\(\)](#).

Here is the call graph for this function:



#### 13.1.11.12.4 Thread control

API for Thread Control method.

Collaboration diagram for Thread control:



## Functions

- void [l4\\_thread\\_control\\_start](#) (void) [L4\\_NOTHROW](#)  
*Start a thread control API sequence.*
- void [l4\\_thread\\_control\\_pager](#) ([l4\\_cap\\_idx\\_t](#) pager) [L4\\_NOTHROW](#)  
*Set the pager.*
- void [l4\\_thread\\_control\\_exc\\_handler](#) ([l4\\_cap\\_idx\\_t](#) exc\_handler) [L4\\_NOTHROW](#)  
*Set the exception handler.*
- void [l4\\_thread\\_control\\_bind](#) ([l4\\_utcb\\_t](#) \*thread\_utcb, [l4\\_cap\\_idx\\_t](#) task) [L4\\_NOTHROW](#)  
*Bind the thread to a task.*
- void [l4\\_thread\\_control\\_alien](#) (int on) [L4\\_NOTHROW](#)  
*Enable alien mode.*
- void [l4\\_thread\\_control\\_ux\\_host\\_syscall](#) (int on) [L4\\_NOTHROW](#)  
*Enable pass through of native host (Linux) system calls.*
- [l4\\_msgtag\\_t](#) [l4\\_thread\\_control\\_commit](#) ([l4\\_cap\\_idx\\_t](#) thread) [L4\\_NOTHROW](#)  
*Commit the thread control parameters.*

### 13.1.11.12.4.1 Detailed Description

API for Thread Control method.

The thread control API provides access to almost any parameter of a thread object. The API is based on a single invocation of the thread object. However, because of the huge amount of parameters, the API provides a set of functions to set specific parameters of a thread and a commit function to commit the thread control call (see [l4\\_thread\\_control\\_commit\(\)](#)).

A thread control operation must always start with [l4\\_thread\\_control\\_start\(\)](#) and be committed with [l4\\_thread\\_control\\_commit\(\)](#). All other thread control parameter setter functions must be called between these two functions.

An example for a sequence of thread control API calls can be found below.

```
l4\_thread\_control\_start();
l4\_thread\_control\_pager(pager\_cap);
l4\_thread\_control\_bind (thread\_utcb, task);
l4\_thread\_control\_commit(thread\_cap);
```

### 13.1.11.12.4.2 Function Documentation

#### [l4\\_thread\\_control\\_alien\(\)](#)

```
void l4_thread_control_alien (
    int on ) [inline]
```

Enable alien mode.

#### Parameters

<i>on</i>	Boolean value defining the state of the feature.
-----------	--

For a thread in alien mode the kernel produces just an exception IPC for each IPC and exception caused by the

alien thread instead of handling these events regularly. (Page faults of alien threads and interrupts occurring while the alien thread is running are always handled regularly.) While the alien thread is blocking, the exception handler can inspect and modify the state of the alien thread and potentially also the syscall arguments. If the exception handler replies with [L4\\_PROTO\\_ALLOW\\_SYSCALL](#) as message tag, the kernel handles the next IPC or exception of the alien thread in a regular way. If the exception handler leaves certain thread state unchanged (in particular the instruction pointer), this will be the IPC or exception that caused the call of the exception handler. For a regularly processed IPC or exception of the alien thread the kernel also performs an exception IPC on kernel exit.

This feature can be used to attach a debugger to a thread and trace all object invocations and their results. It could also be used to handle other systems that use the same syscall instruction as [L4Re](#).

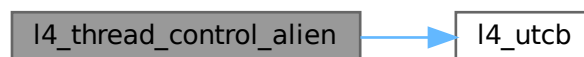
#### Examples

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line 951 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### **`l4_thread_control_bind()`**

```
void l4_thread_control_bind (
    l4_utcb_t * thread_utcb,
    l4_cap_idx_t task ) [inline]
```

Bind the thread to a task.

#### Parameters

<i>thread_utcb</i>	The thread's UTCB address within the task it shall be bound to. The address must be aligned (architecture dependent; at least word aligned) and it must point to at least L4_UTCB_OFFSET bytes of kernel-user memory.
<i>task</i>	The task the thread shall be bound to.

#### Precondition

The thread must not be bound to a task yet.

A thread may execute code in the context of a task if and only if the thread is bound to the task. To actually start execution, [l4\\_thread\\_ex\\_regs\(\)](#) needs to be used. Execution in the context of the task means that the code has access to all the task's resources (and only those). The executed code itself must be one of those resources. A thread can be bound at most once to a task.

**Note**

The UTCBs of different threads in the same task should not overlap in order to prevent data corruption.

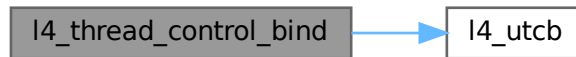
**Examples**

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 945 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**`l4_thread_control_commit()`**

```
l4_msgtag_t l4_thread_control_commit (
    l4_cap_idx_t thread ) [inline]
```

Commit the thread control parameters.

**Parameters**

<i>thread</i>	Capability selector of target thread to commit to.
---------------	--

**Returns**

Syscall return tag containing one of the following return codes.

**Return values**

<i>L4_EOK</i>	Operation successful.
<i>-L4_EPERM</i>	No <a href="#">L4_CAP_FPAGE_S</a> right on <i>thread</i> or the task capability set in <a href="#">l4_thread_control_bind()</a> .
<i>-L4_EINVAL</i>	Malformed thread control parameters.

**Examples**

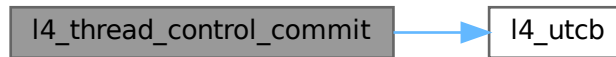
[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).



Definition at line 963 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### **`l4_thread_control_exc_handler()`**

```
void l4_thread_control_exc_handler (
    l4_cap_idx_t exc_handler ) [inline]
```

Set the exception handler.

#### **Parameters**

<i>exc_handler</i>	Capability selector invoked to send an exception IPC.
--------------------	---

#### **Note**

The exception-handler capability selector is interpreted in the task the thread is bound to (executes in).

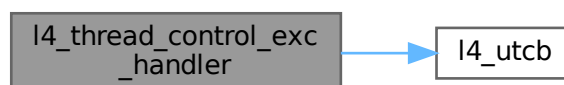
#### **Examples**

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 938 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



**l4\_thread\_control\_pager()**

```
void l4_thread_control_pager (
    l4_cap_idx_t pager ) [inline]
```

Set the pager.

**Parameters**

<i>pager</i>	Capability selector invoked to send a page-fault IPC.
--------------	---

**Note**

The pager capability selector is interpreted in the task the thread is bound to (executes in).

**Examples**

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 932 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**l4\_thread\_control\_start()**

```
void l4_thread_control_start (
    void ) [inline]
```

Start a thread control API sequence.

This function starts a sequence of thread control API functions. After this functions any of following functions may be called in any order.

- [l4\\_thread\\_control\\_pager\(\)](#)
- [l4\\_thread\\_control\\_exc\\_handler\(\)](#)
- [l4\\_thread\\_control\\_bind\(\)](#)
- [l4\\_thread\\_control\\_alien\(\)](#)
- [l4\\_thread\\_control\\_ux\\_host\\_syscall\(\)](#) (Fiasco-UX only)

To commit the changes to the thread [l4\\_thread\\_control\\_commit\(\)](#) must be called in the end.

**Note**

The thread control API calls store the parameters for the thread in the UTCB of the caller (see [l4\\_utcb\(\)](#)), this means between [l4\\_thread\\_control\\_start\(\)](#) and [l4\\_thread\\_control\\_commit\(\)](#) no functions that modify the UTCB contents must be called.

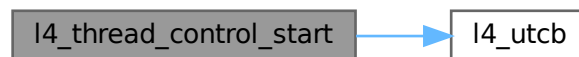
**Examples**

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line [926](#) of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**`l4_thread_control_ux_host_syscall()`**

```
void l4_thread_control_ux_host_syscall (
    int on ) [inline]
```

Enable pass through of native host (Linux) system calls.

**Parameters**

<i>on</i>	Boolean value defining the state of the feature.
-----------	--

**Precondition**

Running on Fiasco-UX

This enables the thread to do host system calls. This feature is only available in Fiasco-UX and ignored in other environments.

Definition at line [957](#) of file [thread.h](#).

References [l4\\_utcb\(\)](#).

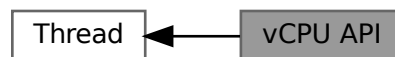
Here is the call graph for this function:



### 13.1.11.12.5 vCPU API

vCPU API.

Collaboration diagram for vCPU API:



### Data Structures

- struct [l4\\_vcpu\\_state\\_t](#)  
*State of a vCPU.*
- struct [l4\\_vcpu\\_regs\\_t](#)  
*vCPU registers.*
- struct [l4\\_vcpu\\_ipc\\_regs\\_t](#)  
*vCPU message registers.*

### Typedefs

- typedef struct [l4\\_vcpu\\_state\\_t](#) [l4\\_vcpu\\_state\\_t](#)  
*State of a vCPU.*
- typedef struct [l4\\_vcpu\\_regs\\_t](#) [l4\\_vcpu\\_regs\\_t](#)  
*vCPU registers.*
- typedef struct [l4\\_vcpu\\_ipc\\_regs\\_t](#) [l4\\_vcpu\\_ipc\\_regs\\_t](#)  
*vCPU message registers.*
- typedef struct [l4\\_vcpu\\_regs\\_t](#) [l4\\_vcpu\\_regs\\_t](#)  
*vCPU registers.*
- typedef struct [l4\\_vcpu\\_ipc\\_regs\\_t](#) [l4\\_vcpu\\_ipc\\_regs\\_t](#)  
*vCPU message registers.*
- typedef struct [l4\\_vcpu\\_regs\\_t](#) [l4\\_vcpu\\_regs\\_t](#)  
*vCPU registers.*
- typedef struct [l4\\_vcpu\\_ipc\\_regs\\_t](#) [l4\\_vcpu\\_ipc\\_regs\\_t](#)  
*vCPU message registers.*

## Enumerations

- enum `L4_vcpu_state_flags` {  
`L4_VCPU_F_IRQ` = 0x01 , `L4_VCPU_F_PAGE_FAULTS` = 0x02 , `L4_VCPU_F_EXCEPTIONS` = 0x04 ,  
`L4_VCPU_F_USER_MODE` = 0x20 ,  
`L4_VCPU_F_FPU_ENABLED` = 0x80 }
- State flags of a vCPU.*
- enum `L4_vcpu_sticky_flags` { `L4_VCPU_SF_IRQ_PENDING` = 0x01 }
- Sticky flags of a vCPU.*
- enum `L4_vcpu_state_offset` { `L4_VCPU_OFFSET_EXT_STATE` = 0x180 , `L4_VCPU_OFFSET_EXT_INFOS` = 0x100 }
- Offsets for vCPU state layouts.*
- enum `L4_vcpu_state_offset` { `L4_VCPU_OFFSET_EXT_STATE` = 0x400 , `L4_VCPU_OFFSET_EXT_INFOS` = 0x200 }
- Offsets for vCPU state layouts.*
- enum `L4_vcpu_state_offset` { `L4_VCPU_OFFSET_EXT_STATE` = 0x400 , `L4_VCPU_OFFSET_EXT_INFOS` = 0x200 }
- Offsets for vCPU state layouts.*

### 13.1.11.12.5.1 Detailed Description

vCPU API.

The vCPU API in [L4Re](#) implements virtual processors (vCPUs) on top of [L4::Thread](#). This API can be used for user level threading, operating system rehosting (see [L4Linux](#)) and virtualization.

You switch a thread into vCPU operation with [L4::Thread::vcpu\\_control](#).

In vCPU mode, incoming IPC can be redirected to a handler function. If an IPC is sent to the vCPU, the thread's normal execution is interrupted and the handler called. Which kind of IPC is redirected is specified by the [L4\\_vcpu\\_state\\_flags](#) set in the [l4\\_vcpu\\_state\\_t::state](#) field of [vcpu\\_state](#). All events enabled in the [vcpu\\_state](#) field are redirected to the handler. The handler is set via [l4\\_vcpu\\_state\\_t::entry\\_ip](#) and [l4\\_vcpu\\_state\\_t::entry\\_sp](#). IPC redirection works independent of "kernel" and "user" mode, but see [l4\\_vcpu\\_state\\_t::entry\\_sp](#). When the entry handler is called, the UTCB contains the result of the IPC and content normally found in CPU register is in [l4\\_vcpu\\_state\\_t::i](#).

Furthermore, the thread can execute in the context of different tasks, called the "kernel" and the "user" mode. The kernel task is the one to which the thread was originally bound via [L4::Thread::control\(\)](#). Execution starts in the kernel task and it is always switched to when the asynchronous IPC handler is invoked. When returning from the handler via [l4\\_thread\\_vcpu\\_resume\\_start\(\)](#) and [l4\\_thread\\_vcpu\\_resume\\_commit\(\)](#), a different user task can be specified by setting [l4\\_vcpu\\_state\\_t::user\\_task](#) and enabling the [L4\\_VCPU\\_F\\_USER\\_MODE](#) flag in [l4\\_vcpu\\_state\\_t::state](#). Note that the kernel may cache the user task internally, see [l4\\_thread\\_vcpu\\_resume\\_commit\(\)](#).

If the [L4\\_VCPU\\_F\\_USER\\_MODE](#) flag is enabled, the following flags will be automatically enabled in [l4\\_vcpu\\_state\\_t::state](#) on [L4::Thread::vcpu\\_resume\\_commit\(\)](#):

- [L4\\_VCPU\\_F\\_IRQ](#)
- [L4\\_VCPU\\_F\\_PAGE\\_FAULTS](#)
- [L4\\_VCPU\\_F\\_EXCEPTIONS](#)

When the kernel mode is entered, the following flags will be automatically disabled in [l4\\_vcpu\\_state\\_t::state](#):

- [L4\\_VCPU\\_F\\_IRQ](#)
- [L4\\_VCPU\\_F\\_PAGE\\_FAULTS](#)
- [L4\\_VCPU\\_F\\_USER\\_MODE](#)

Extended vCPU operation is used for hardware CPU virtualization. It can be enabled with [L4::Thread::vcpu\\_control\\_ext\(\)](#).

[vCPU Support Library](#) defines a convenience API for working with vCPUs.

See also

[vCPU Support Library](#)

### 13.1.11.12.5.2 Enumeration Type Documentation

#### L4\_vcpu\_state\_flags

enum [L4\\_vcpu\\_state\\_flags](#)

State flags of a vCPU.

Enumerator

<a href="#">L4_VCPU_F_IRQ</a>	<p>Receiving of IRQs and IPC enabled. While this flag is not set, the corresponding vCPU thread will not receive any IPC and threads attempting to send an IPC to this thread will block (according to the selected send timeout).</p> <p><b>Note</b></p> <p>On <a href="#">L4::Thread::vcpu_resume_commit()</a> this flag is automatically enabled in <a href="#">l4_vcpu_state_t::state</a> if <a href="#">L4_VCPU_F_USER_MODE</a> is enabled.</p> <p>When the kernel mode is entered, this flags is automatically disabled in <a href="#">l4_vcpu_state_t::state</a>.</p>
<a href="#">L4_VCPU_F_PAGE_FAULTS</a>	<p>Page faults enabled. If this flag is set, a page fault switches to kernel mode (potentially causing a VM exit) and calls the entry handler. If this flag is not set, a page fault generates a page fault IPC to the pager of the vCPU thread.</p> <p><b>Note</b></p> <p>IPC redirection for page faults controlled by this flag works independent of "kernel" and "user" mode.</p> <p>On <a href="#">L4::Thread::vcpu_resume_commit()</a> this flag is automatically enabled in <a href="#">l4_vcpu_state_t::state</a> if <a href="#">L4_VCPU_F_USER_MODE</a> is enabled.</p> <p>When the kernel mode is entered, this flags is automatically disabled in <a href="#">l4_vcpu_state_t::state</a>.</p>

## Enumerator

L4_VCPU_F_EXCEPTIONS	<p>Exceptions enabled. If this flag is set, then, on the event of an exception, the vCPU switches to kernel mode (potentially causing a VM exit) and calls the entry handler. If this flag is not set, an exception generates an exception IPC to the exception handler of the vCPU thread.</p> <p><b>Note</b></p> <p>IPC redirection for exceptions controlled by this flag works independent of "kernel" and "user" mode.</p> <p>On <a href="#">L4::Thread::vcpu_resume_commit()</a> this flag is automatically enabled in <a href="#">l4_vcpu_state_t::state</a> if <a href="#">L4_VCPU_F_USER_MODE</a> is enabled.</p>
L4_VCPU_F_USER_MODE	<p>User task will be used. If set, the vCPU switches to user mode on next <a href="#">L4::Thread::vcpu_resume_commit()</a>. If clear, the vCPU stays in "kernel" mode.</p> <p><b>Note</b></p> <p>When the kernel mode is entered, this flags is automatically disabled in <a href="#">l4_vcpu_state_t::state</a>.</p>
L4_VCPU_F_FPU_ENABLED	<p>FPU enabled. This flag is only relevant if <a href="#">L4_VCPU_F_USER_MODE</a> is set. Setting this flag allows code in vCPU mode to use the FPU. IF this flag is not set, any FPU operation will trigger a corresponding exception (FPU fault).</p>

Definition at line 112 of file [vcpu.h](#).

**L4\_vcpu\_state\_offset** [1/3]

```
enum L4_vcpu_state_offset
```

Offsets for vCPU state layouts.

## Enumerator

L4_VCPU_OFFSET_EXT_STATE	Offset where extended state begins.
L4_VCPU_OFFSET_EXT_INFOS	Offset where extended infos begin.

Definition at line 45 of file [\\_\\_vcpu-arch.h](#).

**L4\_vcpu\_state\_offset** [2/3]

```
enum L4_vcpu_state_offset
```

Offsets for vCPU state layouts.

## Enumerator

L4_VCPU_OFFSET_EXT_STATE	Offset where extended state begins.
L4_VCPU_OFFSET_EXT_INFOS	Offset where extended infos begin.

Definition at line 46 of file [\\_\\_vcpu-arch.h](#).

#### **L4\_vcpu\_state\_offset** [3/3]

enum [L4\\_vcpu\\_state\\_offset](#)

Offsets for vCPU state layouts.

##### Enumerator

L4_VCPU_OFFSET_EXT_STATE	Offset where extended state begins.
L4_VCPU_OFFSET_EXT_INFOS	Offset where extended infos begin.

Definition at line 45 of file [\\_\\_vcpu-arch.h](#).

#### **L4\_vcpu\_sticky\_flags**

enum [L4\\_vcpu\\_sticky\\_flags](#)

Sticky flags of a vCPU.

##### Enumerator

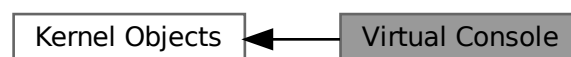
L4_VCPU_SF_IRQ_PENDING	An event is pending: Either an IRQ or another thread attempts to send an IPC to this vCPU thread.
------------------------	---

Definition at line 178 of file [vcpu.h](#).

#### **13.1.11.13 Virtual Console**

C Virtual console interface for simple character based input and output, see [L4::Vcon](#) for the C++ interface.

Collaboration diagram for Virtual Console:



#### **Data Structures**

- struct [l4\\_vcon\\_attr\\_t](#)  
*Vcon attribute structure.*



## Typedefs

- typedef struct [l4\\_vcon\\_attr\\_t](#) [l4\\_vcon\\_attr\\_t](#)  
*Vcon attribute structure.*

## Enumerations

- enum [L4\\_vcon\\_size\\_consts](#) { [L4\\_VCON\\_WRITE\\_SIZE](#) = (L4\_UTCB\_GENERIC\_DATA\_SIZE - 2) \* sizeof([l4\\_umword\\_t](#)) , [L4\\_VCON\\_READ\\_SIZE](#) = (L4\_UTCB\_GENERIC\_DATA\_SIZE - 1) \* sizeof([l4\\_umword\\_t](#)) }  
*Size constants.*
- enum [L4\\_vcon\\_i\\_flags](#) { [L4\\_VCON\\_INLCR](#) = 000100 , [L4\\_VCON\\_IGNCR](#) = 000200 , [L4\\_VCON\\_ICRNL](#) = 000400 }  
*Input flags.*
- enum [L4\\_vcon\\_o\\_flags](#) { [L4\\_VCON\\_ONLCR](#) = 000004 , [L4\\_VCON\\_OCRNL](#) = 000010 , [L4\\_VCON\\_ONLRET](#) = 000040 }  
*Output flags.*
- enum [L4\\_vcon\\_l\\_flags](#) { [L4\\_VCON\\_ICANON](#) = 000002 , [L4\\_VCON\\_ECHO](#) = 000010 }  
*Local flags.*

## Functions

- [l4\\_msgtag\\_t](#) [l4\\_vcon\\_send](#) ([l4\\_cap\\_idx\\_t](#) vcon, char const \*buf, unsigned size) [L4\\_NOTHROW](#)  
*Send data to virtual console.*
- [l4\\_msgtag\\_t](#) [l4\\_vcon\\_send\\_u](#) ([l4\\_cap\\_idx\\_t](#) vcon, char const \*buf, unsigned size, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Send data to *this* virtual console.*
- long [l4\\_vcon\\_write](#) ([l4\\_cap\\_idx\\_t](#) vcon, char const \*buf, unsigned size) [L4\\_NOTHROW](#)  
*Write data to virtual console.*
- long [l4\\_vcon\\_write\\_u](#) ([l4\\_cap\\_idx\\_t](#) vcon, char const \*buf, unsigned size, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Write data to *this* virtual console.*
- int [l4\\_vcon\\_read](#) ([l4\\_cap\\_idx\\_t](#) vcon, char \*buf, unsigned size) [L4\\_NOTHROW](#)  
*Read data from virtual console.*
- int [l4\\_vcon\\_read\\_u](#) ([l4\\_cap\\_idx\\_t](#) vcon, char \*buf, unsigned size, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Read data from *this* virtual console.*
- int [l4\\_vcon\\_read\\_with\\_flags](#) ([l4\\_cap\\_idx\\_t](#) vcon, char \*buf, unsigned size) [L4\\_NOTHROW](#)  
*Read data from virtual console, extended version including flags.*
- [l4\\_msgtag\\_t](#) [l4\\_vcon\\_set\\_attr](#) ([l4\\_cap\\_idx\\_t](#) vcon, [l4\\_vcon\\_attr\\_t](#) const \*attr) [L4\\_NOTHROW](#)  
*Set attributes of a Vcon.*
- [l4\\_msgtag\\_t](#) [l4\\_vcon\\_set\\_attr\\_u](#) ([l4\\_cap\\_idx\\_t](#) vcon, [l4\\_vcon\\_attr\\_t](#) const \*attr, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Set the attributes of *this* virtual console.*
- [l4\\_msgtag\\_t](#) [l4\\_vcon\\_get\\_attr](#) ([l4\\_cap\\_idx\\_t](#) vcon, [l4\\_vcon\\_attr\\_t](#) \*attr) [L4\\_NOTHROW](#)  
*Get attributes of a Vcon.*
- [l4\\_msgtag\\_t](#) [l4\\_vcon\\_get\\_attr\\_u](#) ([l4\\_cap\\_idx\\_t](#) vcon, [l4\\_vcon\\_attr\\_t](#) \*attr, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Get attributes of *this* virtual console.*
- void [l4\\_vcon\\_set\\_attr\\_raw](#) ([l4\\_vcon\\_attr\\_t](#) \*attr) [L4\\_NOTHROW](#)  
*Set terminal attributes to disable all special processing.*

### 13.1.11.13.1 Detailed Description

C Virtual console interface for simple character based input and output, see [L4::Vcon](#) for the C++ interface.

The interrupt for read events is provided by the virtual key interrupt which, in contrast to hardware IRQs, implements a limited functionality:

- Only IRQ line 0 is supported, no MSI vectors.
- The IRQ is edge-triggered and the IRQ mode cannot be changed.
- As the IRQ is edge-triggered, it does not have to be explicitly unmasked.

A server implementing the virtual console protocol has a queue for input events. When the first input event is added to the empty queue, the virtual key interrupt is triggered. Further events are added to the queue without generating further interrupts. The queue is emptied when a client reads all queued input events.

#### Include File

```
#include <l4/sys/vcon.h>
```

See [L4::Vcon](#) for the C++ interface.

### 13.1.11.13.2 Typedef Documentation

#### 13.1.11.13.2.1 l4\_vcon\_attr\_t

```
typedef struct l4_vcon_attr_t l4_vcon_attr_t
```

Vcon attribute structure.

The flags members can be a combination of their respective enums.

See also

[L4\\_vcon\\_i\\_flags](#)

[L4\\_vcon\\_o\\_flags](#)

[L4\\_vcon\\_l\\_flags](#)

### 13.1.11.13.3 Enumeration Type Documentation

#### 13.1.11.13.3.1 L4\_vcon\_i\_flags

```
enum L4_vcon_i_flags
```

Input flags.

#### Enumerator

L4_VCON_INLCR	Translate NL to CR.
L4_VCON_IGNCR	Ignore CR.
L4_VCON_ICRNL	Translate CR to NL if L4_VCON_IGNCR is not set.

Definition at line 217 of file [vcon.h](#).

#### 13.1.11.13.3.2 L4\_vcon\_l\_flags

enum [L4\\_vcon\\_l\\_flags](#)

Local flags.

Enumerator

L4_VCON_ICANON	Canonical mode.
L4_VCON_ECHO	Echo input.

Definition at line 239 of file [vcon.h](#).

#### 13.1.11.13.3.3 L4\_vcon\_o\_flags

enum [L4\\_vcon\\_o\\_flags](#)

Output flags.

Enumerator

L4_VCON_ONLCR	Translate NL to CR-NL.
L4_VCON_OCRNL	Translate CR to NL.
L4_VCON_ONLRET	Do not output CR.

Definition at line 228 of file [vcon.h](#).

#### 13.1.11.13.3.4 L4\_vcon\_size\_consts

enum [L4\\_vcon\\_size\\_consts](#)

Size constants.

Enumerator

L4_VCON_WRITE_SIZE	Maximum size that can be written with one <code>l4_vcon_write</code> call.
L4_VCON_READ_SIZE	Maximum size that can be read with one <code>l4_vcon_read*</code> call.

Definition at line 106 of file [vcon.h](#).

### 13.1.11.13.4 Function Documentation

#### 13.1.11.13.4.1 l4\_vcon\_get\_attr()

```
l4\_msgtag\_t l4_vcon_get_attr (
```

```
l4_cap_idx_t vcon,
l4_vcon_attr_t * attr ) [inline]
```

Get attributes of a Vcon.

#### Parameters

	<i>vcon</i>	Vcon object.
out	<i>attr</i>	Attribute structure.

#### Returns

Syscall return tag

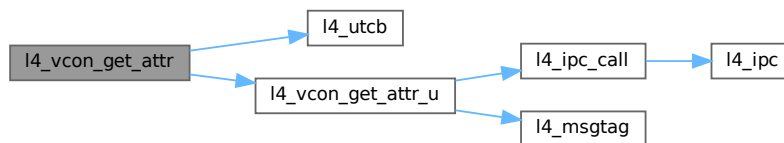
#### Examples

[examples/sys/isr/main.c](#).

Definition at line 444 of file [vcon.h](#).

References [l4\\_utcb\(\)](#), and [l4\\_vcon\\_get\\_attr\\_u\(\)](#).

Here is the call graph for this function:



#### 13.1.11.13.4.2 l4\_vcon\_get\_attr\_u()

```
l4_msgtag_t l4_vcon_get_attr_u (
    l4_cap_idx_t vcon,
    l4_vcon_attr_t * attr,
    l4_utcb_t * utcb ) [inline]
```

Get attributes of this virtual console.

#### Parameters

	<i>vcon</i>	Capability index of the vcon object.
out	<i>attr</i>	Attribute structure. Contains the attributes after a successful call of this function.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

**Returns**

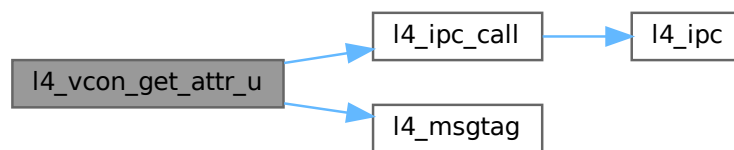
Syscall return tag.

Definition at line 426 of file [vcon.h](#).

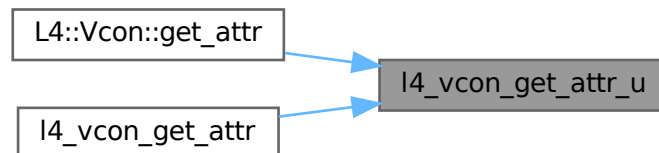
References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_LOG](#), [L4\\_VCON\\_GET\\_ATTR\\_OP](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [L4::Vcon::get\\_attr\(\)](#), and [l4\\_vcon\\_get\\_attr\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**13.1.11.13.4.3 l4\_vcon\_read()**

```

int l4_vcon_read (
    l4_cap_idx_t vcon,
    char * buf,
    unsigned size ) [inline]
  
```

Read data from virtual console.

**Parameters**

	<i>vcon</i>	Vcon object.
out	<i>buf</i>	Pointer to data buffer.
	<i>size</i>	Size of buffer in bytes.

## Return values

<code>-L4_EPERM</code>	The Vcon instance requires the <a href="#">L4_CAP_FPAGE_W</a> right on the <code>vcon</code> capability and this right is not present.
<code>&gt;size</code>	More bytes to read, <code>size</code> bytes are in the buffer <code>buf</code> .
<code>&lt;=size</code>	Number of bytes read.

## Note

Size must not exceed [L4\\_VCON\\_READ\\_SIZE](#).

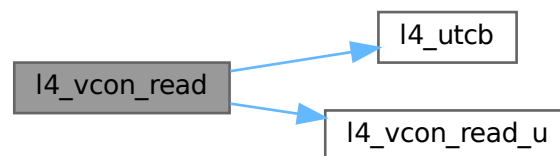
## Examples

[examples/sys/isr/main.c](#).

Definition at line [400](#) of file [vcon.h](#).

References [l4\\_utcb\(\)](#), and [l4\\_vcon\\_read\\_u\(\)](#).

Here is the call graph for this function:



## 13.1.11.13.4.4 l4\_vcon\_read\_u()

```

int l4_vcon_read_u (
    l4_cap_idx_t vcon,
    char * buf,
    unsigned size,
    l4_utcb_t * utcb ) [inline]
  
```

Read data from `this` virtual console.

## Parameters

	<i>vcon</i>	Capability index of the vcon object.
out	<i>buf</i>	Pointer to data buffer.
	<i>size</i>	Size of the data buffer in bytes.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

## Return values

<code>-L4_EPERM</code>	The Vcon instance requires the <a href="#">L4_CAP_FPAGE_W</a> right on the capability used to invoke this operation and this right is not present.
<code>&gt;size</code>	More bytes to read, <code>size</code> bytes are in the buffer <code>buf</code> .
<code>&lt;=size</code>	Number of bytes read.

## Note

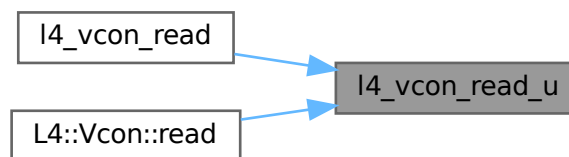
Size must not exceed [L4\\_VCON\\_READ\\_SIZE](#).

Definition at line 390 of file [vcon.h](#).

References [L4\\_VCON\\_READ\\_SIZE\\_MASK](#).

Referenced by [l4\\_vcon\\_read\(\)](#), and [L4::Vcon::read\(\)](#).

Here is the caller graph for this function:



## 13.1.11.13.4.5 l4\_vcon\_read\_with\_flags()

```

int l4_vcon_read_with_flags (
    l4_cap_idx_t vcon,
    char * buf,
    unsigned size ) [inline]
  
```

Read data from virtual console, extended version including flags.

## Parameters

	<i>vcon</i>	Vcon object.
out	<i>buf</i>	Pointer to data buffer.
	<i>size</i>	Size of buffer in bytes.

If this function returns a positive value the caller can check the [L4\\_VCON\\_READ\\_STAT\\_BREAK](#) flag bit for a break condition. The bytes read can be obtained by masking the return value with [L4\\_VCON\\_READ\\_SIZE\\_MASK](#).

If a break condition is signaled, it is always the first event in the transmitted content, i.e. all characters supplied by this read call follow the break condition.

`buf` might be a `NULL`, in this case the input data will be dropped.

#### Note

Size must not exceed [L4\\_VCON\\_READ\\_SIZE](#).

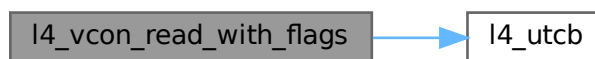
#### Return values

<code>-L4_EPERM</code>	The Vcon instance requires the <a href="#">L4_CAP_FPAGE_W</a> right on the <code>vcon</code> capability and this right is not present.
<code>&gt;size</code>	More bytes to read, <code>size</code> bytes are in the buffer <code>buf</code> .
<code>&lt;=size</code>	Number of bytes read.

Definition at line [384](#) of file [vcon.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.11.13.4.6 l4\_vcon\_send()

```

l4_msgtag_t l4_vcon_send (
    l4_cap_idx_t vcon,
    char const * buf,
    unsigned size ) [inline]
  
```

Send data to virtual console.

#### Parameters

<code>vcon</code>	Vcon object.
<code>buf</code>	Pointer to data buffer.
<code>size</code>	Size of buffer in bytes.

#### Returns

Syscall return tag



**Note**

Size must not exceed [L4\\_VCON\\_WRITE\\_SIZE](#), a proper value of the `size` parameter is NOT checked. Also, this function is a send only operation, this means there is no return value except for a failed send operation. Use [l4\\_ipc\\_error\(\)](#) to check for send errors, and **do not** use [l4\\_error\(\)](#).

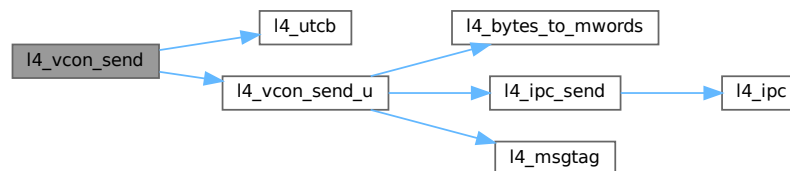
**Examples**

[examples/sys/utcb-ipc/main.c](#).

Definition at line 324 of file [vcon.h](#).

References [l4\\_utcb\(\)](#), and [l4\\_vcon\\_send\\_u\(\)](#).

Here is the call graph for this function:

**13.1.11.13.4.7 l4\_vcon\_send\_u()**

```

l4_msgtag_t l4_vcon_send_u (
    l4_cap_idx_t vcon,
    char const * buf,
    unsigned size,
    l4_utcb_t * utcb ) [inline]

```

Send data to this virtual console.

**Parameters**

<i>vcon</i>	Capability index of the Vcon object.
<i>buf</i>	Pointer to the data buffer.
<i>size</i>	Size of the data buffer in bytes.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

**Returns**

Syscall return tag

## Note

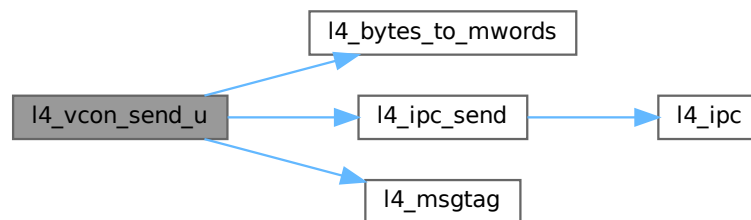
Size must not exceed [L4\\_VCON\\_WRITE\\_SIZE](#), a proper value of the `size` parameter is NOT checked. Also, this function is a send only operation, this means there is no return value except for a failed send operation. Use [l4\\_ipc\\_error\(\)](#) to check for send errors, do not use [l4\\_error\(\)](#), as [l4\\_error\(\)](#) will always return an error.

Definition at line 311 of file [vcon.h](#).

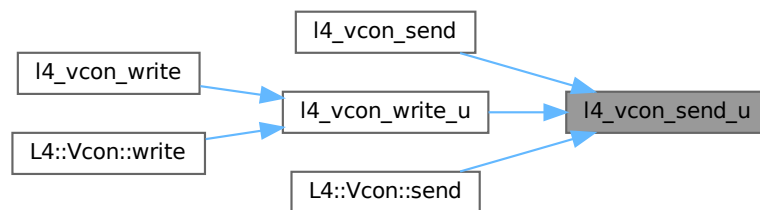
References [l4\\_bytes\\_to\\_mwords\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_ipc\\_send\(\)](#), [l4\\_msgtag\(\)](#), [L4\\_MSGTAG\\_SCHEDULE](#), [L4\\_PROTO\\_LOG](#), [L4\\_VCON\\_WRITE\\_OP](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [l4\\_vcon\\_send\(\)](#), [l4\\_vcon\\_write\\_u\(\)](#), and [L4::Vcon::send\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.11.13.4.8 l4\_vcon\_set\_attr()

```

l4_msgtag_t l4_vcon_set_attr (
    l4_cap_idx_t vcon,
    l4_vcon_attr_t const * attr ) [inline]
  
```

Set attributes of a Vcon.

## Parameters

<i>vcon</i>	Vcon object.
<i>attr</i>	Attribute structure.

## Returns

Syscall return tag

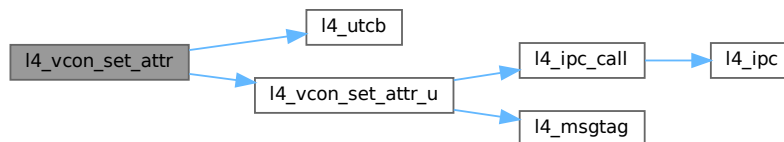
## Examples

[examples/sys/isr/main.c](#).

Definition at line 420 of file [vcon.h](#).

References [l4\\_utcb\(\)](#), and [l4\\_vcon\\_set\\_attr\\_u\(\)](#).

Here is the call graph for this function:



#### 13.1.11.13.4.9 l4\_vcon\_set\_attr\_raw()

```
void l4_vcon_set_attr_raw (
    l4_vcon_attr_t * attr ) [inline]
```

Set terminal attributes to disable all special processing.

Removes all flags that would mangle the read or written characters. Also disables echoing and any special processing of characters.

## Parameters

<i>in, out</i>	<i>attr</i>	Attribute structure to update.
----------------	-------------	--------------------------------

Definition at line 450 of file [vcon.h](#).

Referenced by [l4\\_vcon\\_attr\\_t::set\\_raw\(\)](#).

Here is the caller graph for this function:



#### 13.1.11.13.4.10 l4\_vcon\_set\_attr\_u()

```

l4_msgtag_t l4_vcon_set_attr_u (
    l4_cap_idx_t vcon,
    l4_vcon_attr_t const * attr,
    l4_utcb_t * utcb ) [inline]
  
```

Set the attributes of `this` virtual console.

##### Parameters

<i>vcon</i>	Capability index of the vcon object.
<i>attr</i>	Attribute structure with the attributes for the virtual console.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

##### Returns

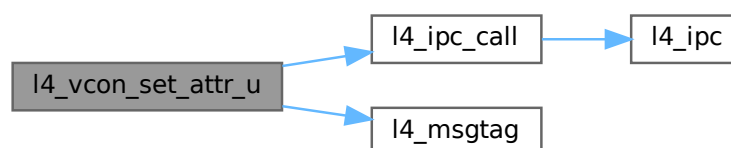
Syscall return tag.

Definition at line 406 of file [vcon.h](#).

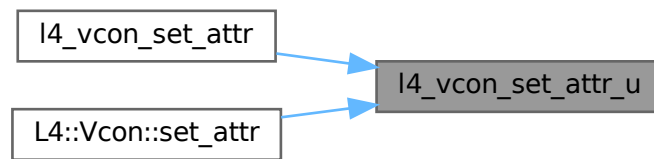
References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_LOG](#), [L4\\_VCON\\_SET\\_ATTR\\_OP](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [l4\\_vcon\\_set\\_attr\(\)](#), and [L4::Vcon::set\\_attr\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.11.13.4.11 l4\_vcon\_write()

```

long l4_vcon_write (
    l4_cap_idx_t vcon,
    char const * buf,
    unsigned size ) [inline]
  
```

Write data to virtual console.

##### Parameters

<i>vcon</i>	Vcon object.
<i>buf</i>	Pointer to data buffer.
<i>size</i>	Size of buffer in bytes.

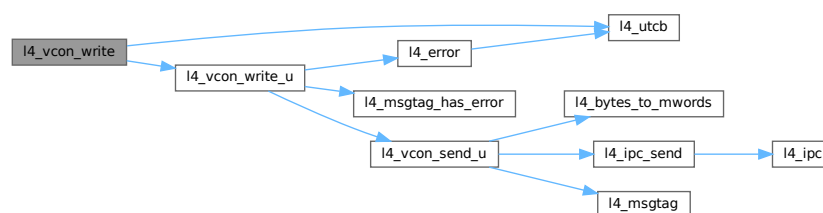
##### Return values

< 0	Error.
>= 0	Number of bytes written to the virtual console

Definition at line 345 of file [vcon.h](#).

References [l4\\_utcb\(\)](#), and [l4\\_vcon\\_write\\_u\(\)](#).

Here is the call graph for this function:



### 13.1.11.13.4.12 l4\_vcon\_write\_u()

```
long l4_vcon_write_u (
    l4_cap_idx_t vcon,
    char const * buf,
    unsigned size,
    l4_utcb_t * utcb ) [inline]
```

Write data to this virtual console.

#### Parameters

<i>vcon</i>	Capability index of the vcon object.
<i>buf</i>	Pointer to the data buffer.
<i>size</i>	Size of the data buffer in bytes.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

#### Return values

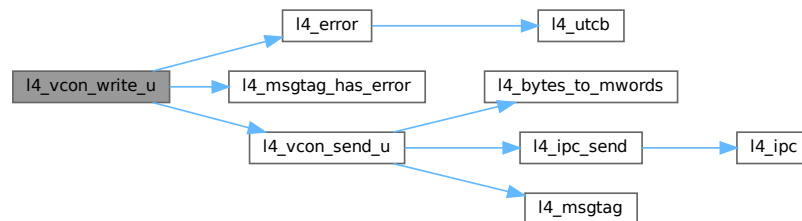
<0	Error.
>=0	Number of bytes written to the virtual console.

Definition at line 330 of file [vcon.h](#).

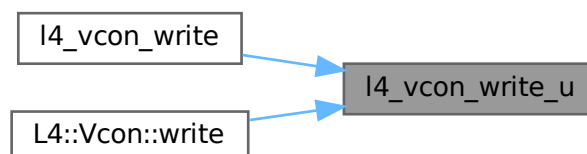
References [l4\\_error\(\)](#), [l4\\_msgtag\\_has\\_error\(\)](#), [l4\\_vcon\\_send\\_u\(\)](#), and [L4\\_VCON\\_WRITE\\_SIZE](#).

Referenced by [l4\\_vcon\\_write\(\)](#), and [L4::Vcon::write\(\)](#).

Here is the call graph for this function:



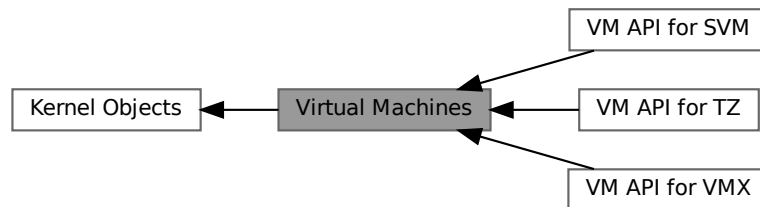
Here is the caller graph for this function:



#### 13.1.11.14 Virtual Machines

Virtual Machine API.

Collaboration diagram for Virtual Machines:



#### Modules

- [VM API for SVM](#)  
*Virtual machine API for SVM.*
- [VM API for TZ](#)  
*Virtual Machine API for ARM TrustZone.*
- [VM API for VMX](#)  
*Virtual machine API for VMX.*

#### 13.1.11.14.1 Detailed Description

Virtual Machine API.

#### 13.1.11.14.2 VM API for SVM

Virtual machine API for SVM.

Collaboration diagram for VM API for SVM:



## Data Structures

- struct [l4\\_vm\\_svm\\_vmcb\\_control\\_area](#)  
*VMCB structure for SVM VMs.*
- struct [l4\\_vm\\_svm\\_vmcb\\_state\\_save\\_area\\_seg](#)  
*State save area segment selector struct.*
- struct [l4\\_vm\\_svm\\_vmcb\\_state\\_save\\_area](#)  
*State save area structure for SVM VMs.*
- struct [l4\\_vm\\_svm\\_vmcb\\_t](#)  
*Control structure for SVM VMs.*

## Typedefs

- typedef struct [l4\\_vm\\_svm\\_vmcb\\_control\\_area](#) [l4\\_vm\\_svm\\_vmcb\\_control\\_area\\_t](#)  
*VMCB structure for SVM VMs.*
- typedef struct [l4\\_vm\\_svm\\_vmcb\\_state\\_save\\_area\\_seg](#) [l4\\_vm\\_svm\\_vmcb\\_state\\_save\\_area\\_seg\\_t](#)  
*State save area segment selector struct.*
- typedef struct [l4\\_vm\\_svm\\_vmcb\\_state\\_save\\_area](#) [l4\\_vm\\_svm\\_vmcb\\_state\\_save\\_area\\_t](#)  
*State save area structure for SVM VMs.*
- typedef struct [l4\\_vm\\_svm\\_vmcb\\_t](#) [l4\\_vm\\_svm\\_vmcb\\_t](#)  
*Control structure for SVM VMs.*

### 13.1.11.14.2.1 Detailed Description

Virtual machine API for SVM.

### 13.1.11.14.3 VM API for TZ

Virtual Machine API for ARM TrustZone.

Collaboration diagram for VM API for TZ:



## Data Structures

- struct [l4\\_vm\\_tz\\_state](#)  
*state structure for TrustZone VMs*



### 13.1.11.14.3.1 Detailed Description

Virtual Machine API for ARM TrustZone.

### 13.1.11.14.4 VM API for VMX

Virtual machine API for VMX.

Collaboration diagram for VM API for VMX:



### Data Structures

- struct [l4\\_vmx\\_offset\\_table\\_t](#)  
*Software VMCS field offset table.*
- struct [l4\\_ext\\_vcpu\\_state\\_vmx\\_t](#)  
*VMX extended vCPU state.*

### Typedefs

- typedef struct [l4\\_vmx\\_offset\\_table\\_t](#) [l4\\_vmx\\_offset\\_table\\_t](#)  
*Software VMCS field offset table.*
- typedef struct [l4\\_ext\\_vcpu\\_state\\_vmx\\_t](#) [l4\\_ext\\_vcpu\\_state\\_vmx\\_t](#)  
*VMX extended vCPU state.*

### Enumerations

- enum [L4\\_vm\\_vmx\\_caps\\_regs](#) {  
[L4\\_VM\\_VMX\\_BASIC\\_REG](#) = 0 , [L4\\_VM\\_VMX\\_TRUE\\_PINBASED\\_CTLS\\_REG](#) = 1 , [L4\\_VM\\_VMX\\_TRUE\\_PROCBASED\\_CTL](#)  
 = 2 , [L4\\_VM\\_VMX\\_TRUE\\_EXIT\\_CTLS\\_REG](#) = 3 ,  
[L4\\_VM\\_VMX\\_TRUE\\_ENTRY\\_CTLS\\_REG](#) = 4 , [L4\\_VM\\_VMX\\_MISC\\_REG](#) = 5 , [L4\\_VM\\_VMX\\_CR0\\_FIXED0\\_REG](#)  
 = 6 , [L4\\_VM\\_VMX\\_CR0\\_FIXED1\\_REG](#) = 7 ,  
[L4\\_VM\\_VMX\\_CR4\\_FIXED0\\_REG](#) = 8 , [L4\\_VM\\_VMX\\_CR4\\_FIXED1\\_REG](#) = 9 , [L4\\_VM\\_VMX\\_VMCS\\_ENUM\\_REG](#)  
 = 0xa , [L4\\_VM\\_VMX\\_PROCBASED\\_CTLS2\\_REG](#) = 0xb ,  
[L4\\_VM\\_VMX\\_EPT\\_VPID\\_CAP\\_REG](#) = 0xc , [L4\\_VM\\_VMX\\_NESTED\\_REVISION](#) = 0xd , [L4\\_VM\\_VMX\\_NUM\\_CAPS\\_REGS](#)  
 }  
*Exported VMX capability registers.*
- enum [L4\\_vm\\_vmx\\_dfl1\\_regs](#) {  
[L4\\_VM\\_VMX\\_PINBASED\\_CTLS\\_DFL1\\_REG](#) = 0x1 , [L4\\_VM\\_VMX\\_PROCBASED\\_CTLS\\_DFL1\\_REG](#) =  
 0x2 , [L4\\_VM\\_VMX\\_EXIT\\_CTLS\\_DFL1\\_REG](#) = 0x3 , [L4\\_VM\\_VMX\\_ENTRY\\_CTLS\\_DFL1\\_REG](#) = 0x4 ,  
[L4\\_VM\\_VMX\\_NUM\\_DFL1\\_REGS](#) }  
*Exported VMX capability registers (default to 1 bits).*

- enum [L4\\_vm\\_vmx\\_sw\\_fields](#) {  
[L4\\_VM\\_VMX\\_VMCS\\_CR2](#) = 0x683e , [L4\\_VM\\_VMX\\_VMCS\\_NAT\\_ARG0](#) = 0x6840 , [L4\\_VM\\_VMX\\_VMCS\\_NAT\\_ARG1](#) = 0x6842 , [L4\\_VM\\_VMX\\_VMCS\\_NAT\\_ARG2](#) = 0x6844 ,  
[L4\\_VM\\_VMX\\_VMCS\\_NAT\\_ARG3](#) = 0x6846 , [L4\\_VM\\_VMX\\_VMCS\\_XCR0](#) = 0x2840 , [L4\\_VM\\_VMX\\_VMCS\\_MSR\\_SYSCALL\\_I](#)  
= 0x2842 , [L4\\_VM\\_VMX\\_VMCS\\_MSR\\_LSTAR](#) = 0x2844 ,  
[L4\\_VM\\_VMX\\_VMCS\\_MSR\\_CSTAR](#) = 0x2846 , [L4\\_VM\\_VMX\\_VMCS\\_MSR\\_TSC\\_AUX](#) = 0x2848 ,  
[L4\\_VM\\_VMX\\_VMCS\\_MSR\\_STAR](#) = 0x284a , [L4\\_VM\\_VMX\\_VMCS\\_MSR\\_KERNEL\\_GS\\_BASE](#) = 0x284c }  
Additional (virtual) VMCS fields.

## Functions

- [l4\\_uint64\\_t l4\\_vm\\_vmx\\_get\\_caps](#) (void const \*vcpu\_state, unsigned cap\_msr) [L4\\_NOTHROW](#)  
Get a capability register for VMX.
- [l4\\_uint32\\_t l4\\_vm\\_vmx\\_get\\_caps\\_default1](#) (void const \*vcpu\_state, unsigned cap\_msr) [L4\\_NOTHROW](#)  
Get a default to one capability register for VMX.
- unsigned [l4\\_vm\\_vmx\\_field\\_len](#) (unsigned field) [L4\\_NOTHROW](#)  
Return length in bytes of a VMCS field.
- unsigned [l4\\_vm\\_vmx\\_field\\_order](#) (unsigned field) [L4\\_NOTHROW](#)  
Return length in power of two (bytes) of a VMCS field.
- void [l4\\_vm\\_vmx\\_clear](#) (void \*vmcs, void \*user\_vmcs) [L4\\_NOTHROW](#)  
Saves cached state from the kernel software VMCS to the user software VMCS.
- void [l4\\_vm\\_vmx\\_ptr\\_load](#) (void \*vmcs, void \*user\_vmcs) [L4\\_NOTHROW](#)  
Loads the user\_vmcs as the current software VMCS.
- [l4\\_uint32\\_t l4\\_vm\\_vmx\\_get\\_cr2\\_index](#) (void const \*vmcs) [L4\\_NOTHROW](#)  
Get the software VMCS field index of the virtual CR2 register.
- [l4\\_umword\\_t l4\\_vm\\_vmx\\_read\\_nat](#) (void \*vmcs, unsigned field) [L4\\_NOTHROW](#)  
Read a natural-width software VMCS field.
- [l4\\_uint16\\_t l4\\_vm\\_vmx\\_read\\_16](#) (void \*vmcs, unsigned field) [L4\\_NOTHROW](#)  
Read a 16-bit software VMCS field.
- [l4\\_uint32\\_t l4\\_vm\\_vmx\\_read\\_32](#) (void \*vmcs, unsigned field) [L4\\_NOTHROW](#)  
Read a 32-bit software VMCS field.
- [l4\\_uint64\\_t l4\\_vm\\_vmx\\_read\\_64](#) (void \*vmcs, unsigned field) [L4\\_NOTHROW](#)  
Read a 64-bit software VMCS field.
- [l4\\_uint64\\_t l4\\_vm\\_vmx\\_read](#) (void \*vmcs, unsigned field) [L4\\_NOTHROW](#)  
Read any software VMCS field.
- void [l4\\_vm\\_vmx\\_write\\_nat](#) (void \*vmcs, unsigned field, [l4\\_umword\\_t](#) val) [L4\\_NOTHROW](#)  
Write to a natural-width software VMCS field.
- void [l4\\_vm\\_vmx\\_write\\_16](#) (void \*vmcs, unsigned field, [l4\\_uint16\\_t](#) val) [L4\\_NOTHROW](#)  
Write to a 16-bit software VMCS field.
- void [l4\\_vm\\_vmx\\_write\\_32](#) (void \*vmcs, unsigned field, [l4\\_uint32\\_t](#) val) [L4\\_NOTHROW](#)  
Write to a 32-bit software VMCS field.
- void [l4\\_vm\\_vmx\\_write\\_64](#) (void \*vmcs, unsigned field, [l4\\_uint64\\_t](#) val) [L4\\_NOTHROW](#)  
Write to a 64-bit software VMCS field.
- void [l4\\_vm\\_vmx\\_write](#) (void \*vmcs, unsigned field, [l4\\_uint64\\_t](#) val) [L4\\_NOTHROW](#)  
Write to an arbitrary software VMCS field.
- void [l4\\_vm\\_vmx\\_set\\_hw\\_vmcs](#) (void \*vmcs, [l4\\_cap\\_idx\\_t](#) vmcs\_cap) [L4\\_NOTHROW](#)  
Associate the software VMCS with a hardware VMCS object capability.
- [l4\\_cap\\_idx\\_t l4\\_vm\\_vmx\\_get\\_hw\\_vmcs](#) (void \*vmcs) [L4\\_NOTHROW](#)  
Get the hardware VMCS object capability associated with the software VMCS.

#### 13.1.11.14.4.1 Detailed Description

Virtual machine API for VMX.

#### 13.1.11.14.4.2 Typedef Documentation

##### **l4\_ext\_vcpu\_state\_vmx\_t**

```
typedef struct l4_ext_vcpu_state_vmx_t l4_ext_vcpu_state_vmx_t
```

VMX extended vCPU state.

For completeness, this is the overall memory layout of the vCPU:

0x000 - 0x1ff: Standard vCPU state [l4\\_vcpu\\_state\\_t](#) (with padding). 0x200 - 0x3ff: VMX capabilities (with padding). 0x400 - 0xffff: VMX extended vCPU state.

The memory layout of the VMX extended vCPU state is as follows:

0x000 - 0x007: Reserved (ignored by the kernel). In the hardware VMCS, the revision identifier and the abort indicator are stored in this area. Hereby we simply ignore these two entries. 0x008 - 0x00f: User space data (ignored by the kernel). This currently stores the pointer to a different VMX extended vCPU state that has been loaded into the given state. 0x010 - 0x013: VMCS field index of the software-defined CR2 field in the software VMCS. 0x014 - 0x017: Reserved. 0x018 - 0x01f: Capability of the hardware VMCS object (with padding). 0x020 - 0x047: Software VMCS field offset table [l4\\_vmx\\_offset\\_table\\_t](#). 0x048 - 0x0bf: Reserved. 0x0c0 - 0xabf: Software VMCS fields (with padding). 0xac0 - 0xbff: Software VMCS fields dirty bitmap (with padding).

##### **l4\_vmx\_offset\_table\_t**

```
typedef struct l4_vmx_offset_table_t l4_vmx_offset_table_t
```

Software VMCS field offset table.

The memory layout is as follows:

0x00 - 0x02: 3 offsets for 16-bit fields. 0x03: Reserved. 0x04 - 0x06: 3 offsets for 64-bit fields. 0x07: Reserved. 0x08 - 0x0a: 3 offsets for 32-bit fields. 0x0b: Reserved. 0x0c - 0x0e: 3 offsets for natural-width fields. 0x0f: Reserved. 0x10 - 0x12: 3 limits for 16-bit fields. 0x13: Reserved. 0x14 - 0x16: 3 limits for 64-bit fields. 0x17: Reserved. 0x18 - 0x1a: 3 limits for 32-bit fields. 0x1b: Reserved. 0x1c - 0x1e: 3 limits for natural-width fields. 0x1f: Reserved. 0x20 - 0x23: 4 index shifts. 0x24: Offset of the first software VMCS field. 0x25: Size of the software VMCS fields. 0x26 - 0x27: Reserved.

The offsets/limits in each size category are in the following order:

- Control fields.
- Read-only fields.
- Guest fields.

The index shifts are in the following order:

- 16-bit.
- 64-bit.
- 32-bit.
- Natural-width.

All offsets/limits/sizes are represented in a 64-byte granule.

The offsets (after being multiplied by 64) are indexes in the values array in [l4\\_ext\\_vcpu\\_state\\_vmx\\_t](#) and bit indexes in the dirty\_bitmap array in [l4\\_ext\\_vcpu\\_state\\_vmx\\_t](#).

The limits (after being multiplied by 64) represent the range of the available indexes.

### 13.1.11.14.4.3 Enumeration Type Documentation

#### L4\_vm\_vmx\_caps\_regs

enum [L4\\_vm\\_vmx\\_caps\\_regs](#)

Exported VMX capability registers.

##### Enumerator

L4_VM_VMX_BASIC_REG	Basic VMX capabilities.
L4_VM_VMX_TRUE_PINBASED_CTLN_REG	True pin-based control caps.
L4_VM_VMX_TRUE_PROCBASED_CTLN_REG	True processor based control caps.
L4_VM_VMX_TRUE_EXIT_CTLN_REG	True exit control caps.
L4_VM_VMX_TRUE_ENTRY_CTLN_REG	True entry control caps.
L4_VM_VMX_MISC_REG	Misc caps.
L4_VM_VMX_CR0_FIXED0_REG	Fixed to 0 bits of CR0.
L4_VM_VMX_CR0_FIXED1_REG	Fixed to 1 bits of CR0.
L4_VM_VMX_CR4_FIXED0_REG	Fixed to 0 bits of CR4.
L4_VM_VMX_CR4_FIXED1_REG	Fixed to 1 bits of CR4.
L4_VM_VMX_VMCS_ENUM_REG	VMCS enumeration info.
L4_VM_VMX_PROCBASED_CTLN2_REG	Processor based control 2 caps.
L4_VM_VMX_EPT_VPID_CAP_REG	EPT and VPID caps.
L4_VM_VMX_NESTED_REVISION	Nested VMCS revision.
L4_VM_VMX_NUM_CAPS_REGS	Total number of VMX capability registers.

Definition at line 39 of file [\\_\\_vm-vmx.h](#).

#### L4\_vm\_vmx\_dfl1\_regs

enum [L4\\_vm\\_vmx\\_dfl1\\_regs](#)

Exported VMX capability registers (default to 1 bits).

##### Enumerator

L4_VM_VMX_PINBASED_CTLN_DFL1_REG	Default 1 bits in pin-based controls.
L4_VM_VMX_PROCBASED_CTLN_DFL1_REG	Default 1 bits in processor-based controls.
L4_VM_VMX_EXIT_CTLN_DFL1_REG	Default 1 bits in exit controls.
L4_VM_VMX_ENTRY_CTLN_DFL1_REG	Default 1 bits in entry controls.
L4_VM_VMX_NUM_DFL1_REGS	Total number of default on registers.

Definition at line 63 of file [\\_\\_vm-vmx.h](#).

#### L4\_vm\_vmx\_sw\_fields

enum [L4\\_vm\\_vmx\\_sw\\_fields](#)

Additional (virtual) VMCS fields.

The VMCS offsets defined here are actually not in the hardware VMCS. However our VMMs run in user mode and need to have access to certain registers available in kernel mode only. So we put them into our version of the VMCS.

#### Enumerator

L4_VM_VMX_VMCS_CR2	VMCS offset for CR2.  <b>Note</b>  You usually need to check this value against the value you get from <a href="#">l4_vm_vmx_get_cr2_index()</a> to make sure you are running on a compatible kernel.
L4_VM_VMX_VMCS_XCR0	VMCS offset of extended control register XCR0.
L4_VM_VMX_VMCS_MSR_SYSCALL_MASK	VMCS offset of system call flag mask MSR.
L4_VM_VMX_VMCS_MSR_LSTAR	VMCS offset of IA32e mode system call target address MSR.
L4_VM_VMX_VMCS_MSR_CSTAR	VMCS offset of IA32 mode system call target address MSR.
L4_VM_VMX_VMCS_MSR_TSC_AUX	VMCS offset of auxiliary TSC signature MSR.
L4_VM_VMX_VMCS_MSR_STAR	VMCS offset of system call target address MSR.
L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE	VMCS offset of GS base address swap target MSR.

Definition at line 106 of file [\\_\\_vm-vmx.h](#).

#### 13.1.11.14.4.4 Function Documentation

##### **l4\_vm\_vmx\_clear()**

```
void l4_vm_vmx_clear (
    void * vmcs,
    void * user_vmcs ) [inline]
```

Saves cached state from the kernel software VMCS to the user software VMCS.

#### Parameters

<i>vmcs</i>	Pointer to the kernel software VMCS.
<i>user_vmcs</i>	Pointer to the user software VMCS.

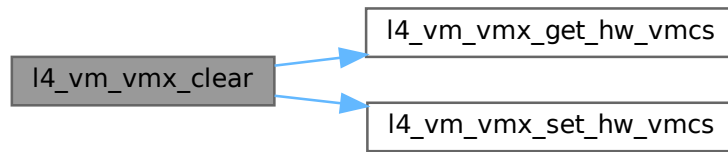
This function is comparable to VMX vmclear.

Definition at line 623 of file [\\_\\_vm-vmx.h](#).

References [l4\\_vm\\_vmx\\_get\\_hw\\_vmcs\(\)](#), and [l4\\_vm\\_vmx\\_set\\_hw\\_vmcs\(\)](#).

Referenced by [l4\\_vm\\_vmx\\_ptr\\_load\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### `l4_vm_vmx_field_len()`

```
unsigned l4_vm_vmx_field_len (  
    unsigned field ) [inline]
```

Return length in bytes of a VMCS field.

#### Parameters

<i>field</i>	Field number.
--------------	---------------

#### Returns

Width of field in bytes.

Definition at line 535 of file `__vm-vmx.h`.

References `l4_vm_vmx_field_order()`.

Here is the call graph for this function:



### **`l4_vm_vmx_field_order()`**

```
unsigned l4_vm_vmx_field_order (
    unsigned field ) [inline]
```

Return length in power of two (bytes) of a VMCS field.

#### **Parameters**

<i>field</i>	Field number.
--------------	---------------

#### **Returns**

Width of field in power of two (bytes).

Definition at line 518 of file `__vm-vmx.h`.

Referenced by `l4_vm_vmx_field_len()`.

Here is the caller graph for this function:



### **`l4_vm_vmx_get_caps()`**

```
l4_uint64_t l4_vm_vmx_get_caps (
    void const * vcpu_state,
    unsigned cap_msr ) [inline]
```

Get a capability register for VMX.

**Parameters**

<i>vcpu_state</i>	Pointer to the VCPU state of the VCPU.
<i>cap_msr</i>	Caps register index (see <a href="#">L4_vm_vmx_caps_regs</a> ).

**Returns**

The value of the capability register.

Definition at line 805 of file [\\_\\_vm-vmx.h](#).

References [L4\\_VCPU\\_OFFSET\\_EXT\\_INFOS](#).

**`l4_vm_vmx_get_caps_default1()`**

```
l4_uint32_t l4_vm_vmx_get_caps_default1 (  
    void const * vcpu_state,  
    unsigned cap_msr ) [inline]
```

Get a default to one capability register for VMX.

**Parameters**

<i>vcpu_state</i>	Pointer to the VCPU state of the VCPU.
<i>cap_msr</i>	Default 1 caps register index (see <a href="#">L4_vm_vmx_dfl1_regs</a> ).

**Returns**

The value of the capability register.

Definition at line 813 of file [\\_\\_vm-vmx.h](#).

References [L4\\_VCPU\\_OFFSET\\_EXT\\_INFOS](#), [L4\\_VM\\_VMX\\_NUM\\_CAPS\\_REGS](#), and [L4\\_VM\\_VMX\\_PINBASED\\_CTL5\\_DFL1\\_REGS](#).

**`l4_vm_vmx_get_cr2_index()`**

```
l4_uint32_t l4_vm_vmx_get_cr2_index (  
    void const * vmcs ) [inline]
```

Get the software VMCS field index of the virtual CR2 register.

**Parameters**

<i>vmcs</i>	Pointer to the software VMCS.
-------------	-------------------------------

**Returns**

The field index used for the virtual CR2 register as used by the current Fiasco.OC interface.



The CR2 register is actually not in the hardware VMCS, however our VMMs run in user mode and need to have access to this register so we put it into our software version of the VMCS.

See also

[L4\\_VM\\_VMX\\_VMCS\\_CR2](#)

Definition at line 821 of file [\\_\\_vm-vmx.h](#).

### **l4\_vm\_vmx\_get\_hw\_vmcs()**

```
l4_cap_idx_t l4_vm_vmx_get_hw_vmcs (
    void * vmcs ) [inline]
```

Get the hardware VMCS object capability associated with the software VMCS.

#### **Parameters**

<i>vmcs</i>	Pointer to the software VMCS.
-------------	-------------------------------

#### **Returns**

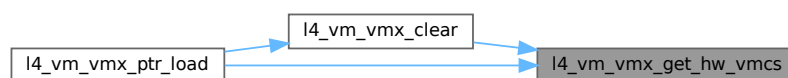
Hardware VMCS object capability.

Definition at line 837 of file [\\_\\_vm-vmx.h](#).

References [L4\\_CAP\\_MASK](#).

Referenced by [l4\\_vm\\_vmx\\_clear\(\)](#), and [l4\\_vm\\_vmx\\_ptr\\_load\(\)](#).

Here is the caller graph for this function:



### **l4\_vm\_vmx\_ptr\_load()**

```
void l4_vm_vmx_ptr_load (
    void * vmcs,
    void * user_vmcs ) [inline]
```

Loads the `user_vmcs` as the current software VMCS.

## Parameters

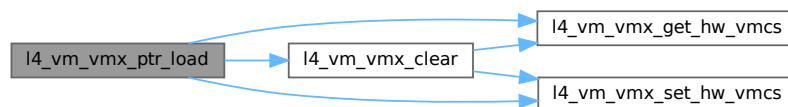
<i>vmcs</i>	Pointer to the kernel software VMCS.
<i>user_vmcs</i>	Pointer to the user software VMCS.

This function is comparable to VMX `vmprld`.

Definition at line 644 of file `__vm-vmx.h`.

References `l4_vm_vmx_clear()`, `l4_vm_vmx_get_hw_vmcs()`, and `l4_vm_vmx_set_hw_vmcs()`.

Here is the call graph for this function:

**`l4_vm_vmx_read()`**

```

l4_uint64_t l4_vm_vmx_read (
    void * vmcs,
    unsigned field ) [inline]
  
```

Read any software VMCS field.

## Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.

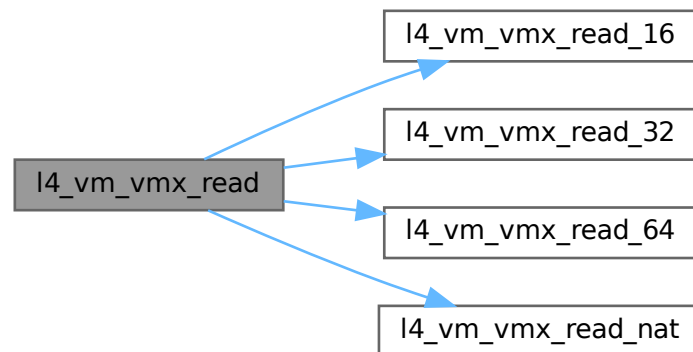
## Returns

The value of the software VMCS field with the given index.

Definition at line 713 of file `__vm-vmx.h`.

References `l4_vm_vmx_read_16()`, `l4_vm_vmx_read_32()`, `l4_vm_vmx_read_64()`, and `l4_vm_vmx_read_nat()`.

Here is the call graph for this function:



### **l4\_vm\_vmx\_read\_16()**

```

l4_uint16_t l4_vm_vmx_read_16 (
    void * vmcs,
    unsigned field ) [inline]
  
```

Read a 16-bit software VMCS field.

#### Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.

#### Returns

The value of the software VMCS field with the given index.

Definition at line 680 of file [\\_\\_vm-vmx.h](#).

Referenced by [l4\\_vm\\_vmx\\_read\(\)](#).

Here is the caller graph for this function:



### **l4\_vm\_vmx\_read\_32()**

```
l4_uint32_t l4_vm_vmx_read_32 (
    void * vmcs,
    unsigned field ) [inline]
```

Read a 32-bit software VMCS field.

#### **Parameters**

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.

#### **Returns**

The value of the software VMCS field with the given index.

Definition at line 691 of file [\\_\\_vm-vmx.h](#).

Referenced by [l4\\_vm\\_vmx\\_read\(\)](#).

Here is the caller graph for this function:



### **l4\_vm\_vmx\_read\_64()**

```
l4_uint64_t l4_vm_vmx_read_64 (
    void * vmcs,
    unsigned field ) [inline]
```

Read a 64-bit software VMCS field.

#### **Parameters**

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.

#### **Returns**

The value of the software VMCS field with the given index.

Definition at line 702 of file [\\_\\_vm-vmx.h](#).

Referenced by [l4\\_vm\\_vmx\\_read\(\)](#).

Here is the caller graph for this function:



### **`l4_vm_vmx_read_nat()`**

```
l4_umword_t l4_vm_vmx_read_nat (
    void * vmcs,
    unsigned field ) [inline]
```

Read a natural-width software VMCS field.

#### **Parameters**

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.

#### **Returns**

The value of the software VMCS field with the given index.

Definition at line [669](#) of file [\\_\\_vm-vmx.h](#).

Referenced by [l4\\_vm\\_vmx\\_read\(\)](#).

Here is the caller graph for this function:



## l4\_vm\_vmx\_set\_hw\_vmcs()

```
void l4_vm_vmx_set_hw_vmcs (
    void * vmcs,
    l4_cap_idx_t vmcs_cap ) [inline]
```

Associate the software VMCS with a hardware VMCS object capability.

The VMX extended vCPU is unable to run unless it is associated with a hardware VMCS object (i.e. a Vcpu\_context object).

### Note

When replacing the hardware VMCS object, the dirty bitmap of the software VMCS fields is not touched. This is on purpose, to enable efficient switching between separate VMs. The user is responsible for explicitly setting those software VMCS bitmap fields that need to be synchronized to the hardware VMCS.

The kernel might cache the VMCS object internally (i.e. the capability is not looked up on every vCPU resume). To remove the association of the current hardware VMCS object, store an invalid capability with the bit 3 set.

If the hardware limitations of the usage of the hardware VMCS are not observed (i.e. no hardware VMCS being active on more than one physical CPU), the vCPU will fail to resume.

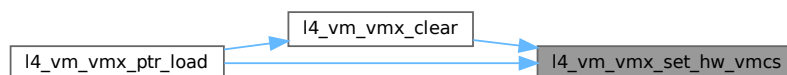
### Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>vmcs_cap</i>	Hardware VMCS object capability.

Definition at line 829 of file [\\_\\_vm-vmx.h](#).

Referenced by [l4\\_vm\\_vmx\\_clear\(\)](#), and [l4\\_vm\\_vmx\\_ptr\\_load\(\)](#).

Here is the caller graph for this function:



## l4\_vm\_vmx\_write()

```
void l4_vm_vmx_write (
    void * vmcs,
    unsigned field,
    l4_uint64_t val ) [inline]
```

Write to an arbitrary software VMCS field.

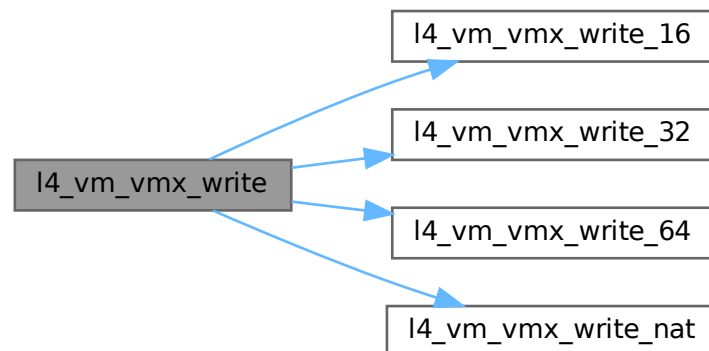
## Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.
<i>val</i>	The value that shall be written to the given field.

Definition at line 790 of file [\\_\\_vm-vmx.h](#).

References [l4\\_vm\\_vmx\\_write\\_16\(\)](#), [l4\\_vm\\_vmx\\_write\\_32\(\)](#), [l4\\_vm\\_vmx\\_write\\_64\(\)](#), and [l4\\_vm\\_vmx\\_write\\_nat\(\)](#).

Here is the call graph for this function:

**`l4_vm_vmx_write_16()`**

```
void l4_vm_vmx_write_16 (
    void * vmcs,
    unsigned field,
    l4_uint16_t val ) [inline]
```

Write to a 16-bit software VMCS field.

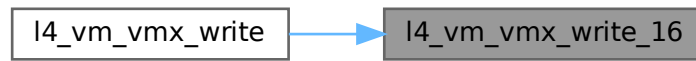
## Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.
<i>val</i>	The value that shall be written to the given field.

Definition at line 745 of file [\\_\\_vm-vmx.h](#).

Referenced by [l4\\_vm\\_vmx\\_write\(\)](#).

Here is the caller graph for this function:



### **`l4_vm_vmx_write_32()`**

```
void l4_vm_vmx_write_32 (
    void * vmcs,
    unsigned field,
    l4_uint32_t val ) [inline]
```

Write to a 32-bit software VMCS field.

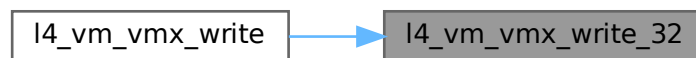
#### **Parameters**

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.
<i>val</i>	The value that shall be written to the given field.

Definition at line 760 of file [\\_\\_vm-vmx.h](#).

Referenced by [l4\\_vm\\_vmx\\_write\(\)](#).

Here is the caller graph for this function:



### **`l4_vm_vmx_write_64()`**

```
void l4_vm_vmx_write_64 (
    void * vmcs,
    unsigned field,
    l4_uint64_t val ) [inline]
```

Write to a 64-bit software VMCS field.



## Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.
<i>val</i>	The value that shall be written to the given field.

Definition at line 775 of file [\\_\\_vm-vmx.h](#).

Referenced by [l4\\_vm\\_vmx\\_write\(\)](#).

Here is the caller graph for this function:

**`l4_vm_vmx_write_nat()`**

```
void l4_vm_vmx_write_nat (
    void * vmcs,
    unsigned field,
    l4_umword_t val ) [inline]
```

Write to a natural-width software VMCS field.

## Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.
<i>val</i>	The value that shall be written to the given field.

Definition at line 730 of file [\\_\\_vm-vmx.h](#).

Referenced by [l4\\_vm\\_vmx\\_write\(\)](#).

Here is the caller graph for this function:



### 13.1.12 Memory operations.

Operations for memory access.

Collaboration diagram for Memory operations.:



#### Enumerations

- enum `L4_mem_op_widths` { `L4_MEM_WIDTH_1BYTE` = 0 , `L4_MEM_WIDTH_2BYTE` = 1 , `L4_MEM_WIDTH_4BYTE` = 2 }

*Memory access width definitions.*

#### Functions

- unsigned long `l4_mem_read` (unsigned long virtaddress, unsigned width)  
*Read user task memory from kernel privilege level.*
- void `l4_mem_write` (unsigned long virtaddress, unsigned width, unsigned long value)  
*Write user task memory from kernel privilege level.*

#### 13.1.12.1 Detailed Description

Operations for memory access.

This module provides functionality to access user task memory from the kernel. This is needed for some devices that are only accessible from privileged processor mode. Only use this when absolutely required. This functionality is only available on the ARM architecture.

```
#include <l4/sys/mem_op.h>
```

#### 13.1.12.2 Enumeration Type Documentation

##### 13.1.12.2.1 L4\_mem\_op\_widths

```
enum L4_mem_op_widths
```

Memory access width definitions.

#### Enumerator

<code>L4_MEM_WIDTH_1BYTE</code>	Access one byte (8-bit width)
<code>L4_MEM_WIDTH_2BYTE</code>	Access two bytes (16-bit width)
<code>L4_MEM_WIDTH_4BYTE</code>	Access four bytes (32-bit width)

Definition at line 51 of file [mem\\_op.h](#).

### 13.1.12.3 Function Documentation

#### 13.1.12.3.1 l4\_mem\_read()

```
unsigned long l4_mem_read (
    unsigned long virtaddress,
    unsigned width ) [inline]
```

Read user task memory from kernel privilege level.

##### Parameters

<i>virtaddress</i>	Virtual address in the calling task.
<i>width</i>	Width of access in bytes in log2,

##### See also

[L4\\_mem\\_op\\_widths](#)

##### Returns

Read value.

Upon an given invalid address or invalid width value the function does nothing.

Definition at line 141 of file [mem\\_op.h](#).

References [l4\\_mem\\_arm\\_op\\_call\(\)](#).

Here is the call graph for this function:



#### 13.1.12.3.2 l4\_mem\_write()

```
void l4_mem_write (
    unsigned long virtaddress,
    unsigned width,
    unsigned long value ) [inline]
```

Write user task memory from kernel privilege level.

## Parameters

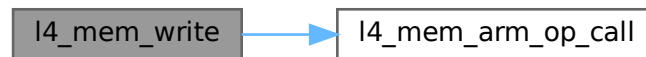
<i>virtaddress</i>	Virtual address in the calling task.
<i>width</i>	Width of access in bytes in log2 (i.e. allowed values: 0, 1, 2)
<i>value</i>	Value to write.

Upon an given invalid address or invalid width value the function does nothing.

Definition at line 147 of file [mem\\_op.h](#).

References [l4\\_mem\\_arm\\_op\\_call\(\)](#).

Here is the call graph for this function:



### 13.1.13 Memory related

Memory related constants, data types and functions.

Collaboration diagram for Memory related:



## Macros

- `#define L4_PAGESIZE`  
*Minimal page size (in bytes).*
- `#define L4_PAGEMASK`  
*Mask for the page number.*
- `#define L4_LOG2_PAGESIZE`  
*Number of bits used for page offset.*
- `#define L4_SUPERPAGESIZE`  
*Size of a large page.*
- `#define L4_SUPERPAGEMASK`  
*Mask for the number of a large page.*

- `#define L4_LOG2_SUPERPAGESIZE`  
*Number of bits used as offset for a large page.*
- `#define L4_INVALID_PTR ((void *)L4_INVALID_ADDR)`  
*Invalid address as pointer type.*
- `#define L4_PAGESHIFT 12`  
*Size of a page, log2-based.*
- `#define L4_SUPERPAGESHIFT 21`  
*Size of a large page, log2-based.*
- `#define L4_PAGESHIFT 12`  
*Size of a page, log2-based.*
- `#define L4_SUPERPAGESHIFT 21`  
*Size of a large page, log2-based.*
- `#define L4_PAGESHIFT 12`  
*Size of a page log2-based.*
- `#define L4_SUPERPAGESHIFT 22`  
*Size of a large page log2-based.*

### Enumerations

- `enum l4_addr_consts_t { L4_INVALID_ADDR = ~0UL }`  
*Address related constants.*

### Functions

- `l4_addr_t l4_trunc_page (l4_addr_t address) L4_NOTHROW`  
*Round an address down to the next lower page boundary.*
- `l4_addr_t l4_trunc_size (l4_addr_t address, unsigned char bits) L4_NOTHROW`  
*Round an address down to the next lower flex page with size bits.*
- `l4_addr_t l4_round_page (l4_addr_t address) L4_NOTHROW`  
*Round address up to the next page.*
- `l4_addr_t l4_round_size (l4_addr_t value, unsigned char bits) L4_NOTHROW`  
*Round value up to the next alignment with bits size.*
- `unsigned l4_bytes_to_mwords (unsigned size) L4_NOTHROW`  
*Determine how many machine words (l4\_umword\_t) are required to store a buffer of 'size' bytes.*

#### 13.1.13.1 Detailed Description

Memory related constants, data types and functions.

#### 13.1.13.2 Macro Definition Documentation

##### 13.1.13.2.1 L4\_LOG2\_PAGESIZE

```
#define L4_LOG2_PAGESIZE
```

Number of bits used for page offset.

Size of page in log2.

Definition at line 398 of file [consts.h](#).

#### 13.1.13.2.2 L4\_LOG2\_SUPERPAGESIZE

```
#define L4_LOG2_SUPERPAGESIZE
```

Number of bits used as offset for a large page.

Size of large page in log2

Definition at line 424 of file [consts.h](#).

#### 13.1.13.2.3 L4\_PAGEMASK

```
#define L4_PAGEMASK
```

Mask for the page number.

##### Note

The most significant bits are set.

Definition at line 389 of file [consts.h](#).

#### 13.1.13.2.4 L4\_PAGESHIFT [1/2]

```
#define L4_PAGESHIFT 12
```

Size of a page, log2-based.

Size of a page log2-based.

Definition at line 35 of file [consts.h](#).

#### 13.1.13.2.5 L4\_PAGESHIFT [2/2]

```
#define L4_PAGESHIFT 12
```

Size of a page, log2-based.

Size of a page log2-based.

##### Examples

[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), [examples/libs/l4re/c/ma+rm.c](#), [examples/libs/l4re/streammap/client.cc](#),  
and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 37 of file [consts.h](#).

#### 13.1.13.2.6 L4\_SUPERPAGEMASK

```
#define L4_SUPERPAGEMASK
```

Mask for the number of a large page.

##### Note

The most significant bits are set.

Definition at line 416 of file [consts.h](#).

#### 13.1.13.2.7 L4\_SUPERPAGESHIFT [1/2]

```
#define L4_SUPERPAGESHIFT 21
```

Size of a large page, log2-based.

Size of a large page log2-based.

Definition at line 41 of file [consts.h](#).

#### 13.1.13.2.8 L4\_SUPERPAGESHIFT [2/2]

```
#define L4_SUPERPAGESHIFT 21
```

Size of a large page, log2-based.

Size of a large page log2-based.

##### Examples

[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), and [examples/libs/l4re/c/ma+rm.c](#).

Definition at line 42 of file [consts.h](#).

#### 13.1.13.2.9 L4\_SUPERPAGESIZE

```
#define L4_SUPERPAGESIZE
```

Size of a large page.

A large page is a *super page* on IA32 or a *section* on ARM.

Definition at line 407 of file [consts.h](#).

### 13.1.13.3 Enumeration Type Documentation

#### 13.1.13.3.1 l4\_addr\_consts\_t

```
enum l4_addr_consts_t
```

Address related constants.

## Enumerator

L4_INVALID_ADDR	Invalid address.
-----------------	------------------

Definition at line 492 of file [consts.h](#).

## 13.1.13.4 Function Documentation

## 13.1.13.4.1 l4\_bytes\_to\_mwords()

```
unsigned l4_bytes_to_mwords (
    unsigned size ) [inline]
```

Determine how many machine words (l4\_umword\_t) are required to store a buffer of 'size' bytes.

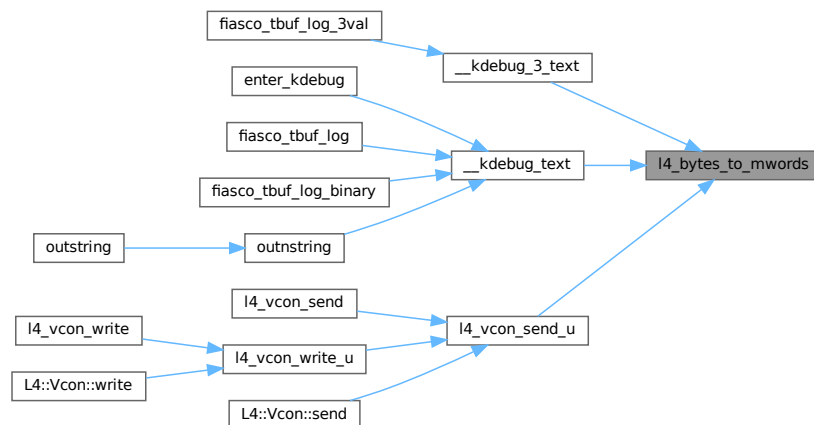
## Parameters

size	The number of bytes to be translated into machine words.
------	--

Definition at line 485 of file [consts.h](#).

Referenced by [\\_\\_kdebug\\_3\\_text\(\)](#), [\\_\\_kdebug\\_text\(\)](#), and [l4\\_vcon\\_send\\_u\(\)](#).

Here is the caller graph for this function:



## 13.1.13.4.2 l4\_round\_page()

```
l4_addr_t l4_round_page (
    l4_addr_t address ) [inline]
```

Round address up to the next page.

The address is rounded up to the next minimal page boundary. On most architectures this is a 4k page. Check [L4\\_PAGESIZE](#) for the minimal page size.



## Parameters

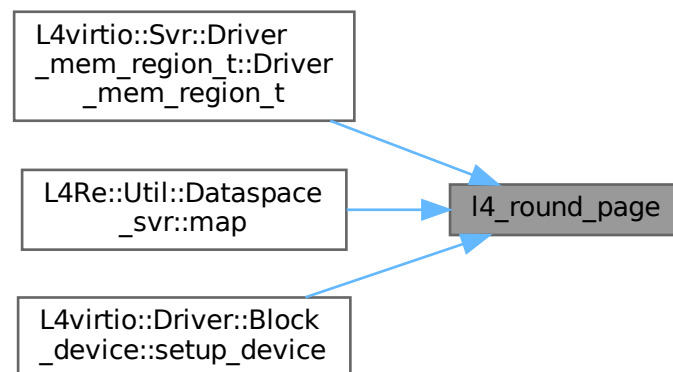
<i>address</i>	The address to round up.
----------------	--------------------------

Definition at line 462 of file [consts.h](#).

References [L4\\_PAGEMASK](#), and [L4\\_PAGESIZE](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_region\\_t< DATA >::Driver\\_mem\\_region\\_t\(\)](#), [L4Re::Util::Dataspace\\_svr::map\(\)](#), and [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#).

Here is the caller graph for this function:



#### 13.1.13.4.3 l4\_round\_size()

```
l4_addr_t l4_round_size (
    l4_addr_t value,
    unsigned char bits ) [inline]
```

Round value up to the next alignment with *bits* size.

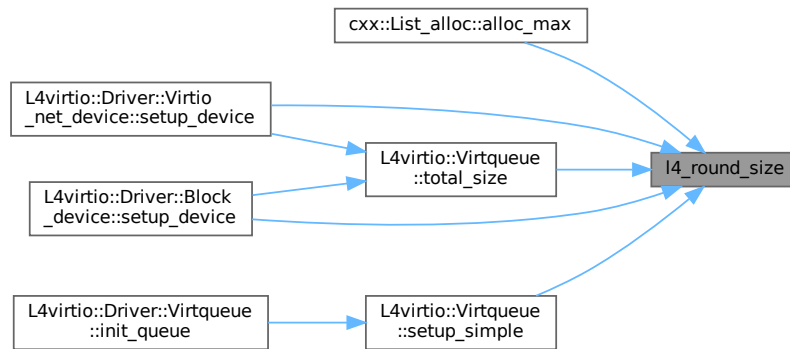
## Parameters

<i>value</i>	The value to round up to the next size-alignment.
<i>bits</i>	The size of the alignment (log2).

Definition at line 473 of file [consts.h](#).

Referenced by [cxx::List\\_alloc::alloc\\_max\(\)](#), [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#), [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#), [L4virtio::Virtqueue::setup\\_simple\(\)](#), and [L4virtio::Virtqueue::total\\_size\(\)](#).

Here is the caller graph for this function:



#### 13.1.13.4.4 l4\_trunc\_page()

```
l4_addr_t l4_trunc_page (
    l4_addr_t address ) [inline]
```

Round an address down to the next lower page boundary.

The address is rounded down to the next lower minimal page boundary. On most architectures this is a 4k page. Check [L4\\_PAGESIZE](#) for the minimal page size.

##### Parameters

<i>address</i>	The address to round.
----------------	-----------------------

##### Examples

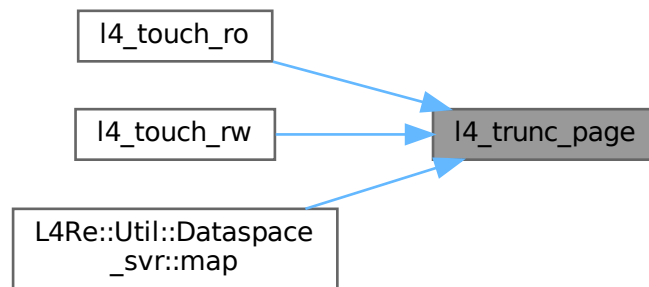
[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), and [examples/libs/l4re/c/ma+rm.c](#).

Definition at line [437](#) of file [consts.h](#).

References [L4\\_PAGEMASK](#).

Referenced by [l4\\_touch\\_ro\(\)](#), [l4\\_touch\\_rw\(\)](#), and [L4Re::Util::Dataspace\\_svr::map\(\)](#).

Here is the caller graph for this function:



#### 13.1.13.4.5 l4\_trunc\_size()

```
l4_addr_t l4_trunc_size (
    l4_addr_t address,
    unsigned char bits ) [inline]
```

Round an address down to the next lower flex page with size *bits*.

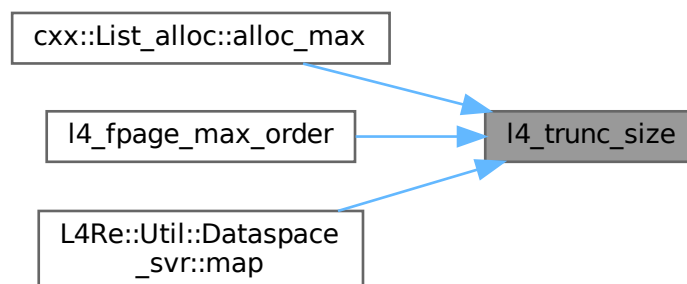
##### Parameters

<i>address</i>	The address to round.
<i>bits</i>	The size of the flex page (log2).

Definition at line 448 of file [consts.h](#).

Referenced by [cxx::List\\_alloc::alloc\\_max\(\)](#), [l4\\_fpage\\_max\\_order\(\)](#), and [L4Re::Util::Dataspace\\_svr::map\(\)](#).

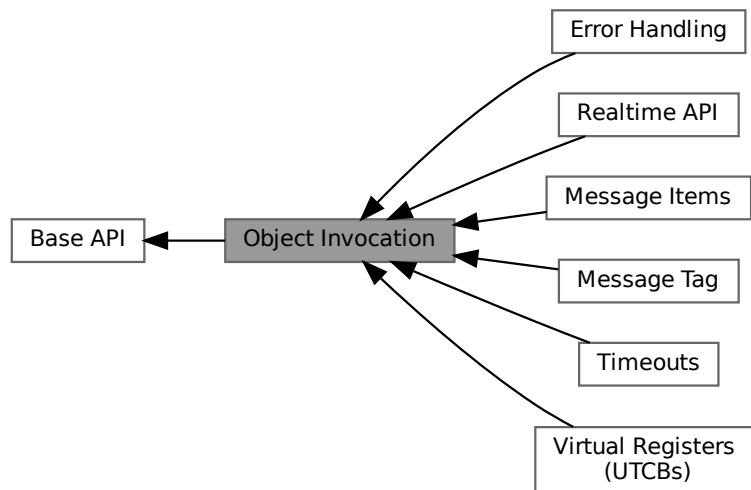
Here is the caller graph for this function:



### 13.1.14 Object Invocation

API for [L4](#) object invocation.

Collaboration diagram for Object Invocation:



#### Modules

- [Error Handling](#)  
*Error handling for [L4](#) object invocation.*
- [Message Items](#)  
*Message item related functions.*
- [Message Tag](#)  
*API related to the message tag data type.*
- [Realtime API](#)
- [Timeouts](#)  
*All kinds of timeouts and time related functions.*
- [Virtual Registers \(UTCBs\)](#)  
*[L4](#) Virtual Registers (UTCB).*

#### Files

- file [utcb.h](#)  
*UTCB definitions.*

#### Enumerations

- enum [l4\\_syscall\\_flags\\_t](#) {  
[L4\\_SYSF\\_NONE](#) , [L4\\_SYSF\\_SEND](#) , [L4\\_SYSF\\_RECV](#) , [L4\\_SYSF\\_OPEN\\_WAIT](#) ,  
[L4\\_SYSF\\_REPLY](#) , [L4\\_SYSF\\_CALL](#) , [L4\\_SYSF\\_WAIT](#) , [L4\\_SYSF\\_SEND\\_AND\\_WAIT](#) ,  
[L4\\_SYSF\\_REPLY\\_AND\\_WAIT](#) }  
*Capability selector flags.*

## Functions

- `l4_msgtag_t l4_ipc_send (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag, l4_timeout_t timeout) L4_NOTHROW`  
*Send a message to an object (do **not** wait for a reply).*
- `l4_msgtag_t l4_ipc_wait (l4_utcb_t *utcb, l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW`  
*Wait for an incoming message from any possible sender.*
- `l4_msgtag_t l4_ipc_receive (l4_cap_idx_t object, l4_utcb_t *utcb, l4_timeout_t timeout) L4_NOTHROW`  
*Wait for a message from a specific source.*
- `l4_msgtag_t l4_ipc_call (l4_cap_idx_t object, l4_utcb_t *utcb, l4_msgtag_t tag, l4_timeout_t timeout) L4_NOTHROW`  
*Object call (usual invocation).*
- `l4_msgtag_t l4_ipc_reply_and_wait (l4_utcb_t *utcb, l4_msgtag_t tag, l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW`  
*Reply and wait operation (uses the reply capability).*
- `l4_msgtag_t l4_ipc_send_and_wait (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag, l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW`  
*Send a message and do an open wait.*
- `l4_msgtag_t l4_ipc (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_umword_t flags, l4_umword_t slabel, l4_msgtag_t tag, l4_umword_t *rlabel, l4_timeout_t timeout) L4_NOTHROW`  
*Generic L4 object invocation.*
- `l4_msgtag_t l4_ipc_sleep (l4_timeout_t timeout) L4_NOTHROW`  
*Sleep for an amount of time.*
- `l4_msgtag_t l4_ipc_sleep_ms (l4_uint32_t ms) L4_NOTHROW`  
*Sleep for a certain amount of milliseconds.*
- `l4_msgtag_t l4_ipc_sleep_us (l4_uint64_t us) L4_NOTHROW`  
*Sleep for a certain amount of microseconds.*
- `int l4_sndfpage_add (l4_fpage_t const snd_fpage, unsigned long snd_base, l4_msgtag_t *tag) L4_NOTHROW`  
*Add a flex-page to be sent to the UTCB.*

### 13.1.14.1 Detailed Description

API for L4 object invocation.

#### Include File

```
#include <l4/sys/ipc.h>
```

General abstractions for L4 object invocation. The basic principle is that all objects are denoted by a capability that is accessed via a capability selector (see [Capabilities](#)).

This set of functions is common to all kinds of objects provided by the L4 micro kernel. The concrete semantics of an invocation depends on the object that shall be invoked.

Objects may be invoked in various ways, the most common way is to use a *call* operation (`l4_ipc_call()`). However, there are a lot more flavours available that have a semantics depending on the object.

#### See also

[IPC-Gate API](#)

[L4 Inter-Process Communication \(IPC\)](#)

### 13.1.14.2 Timeouts during IPC

IPC operation between two communication partners may consist of up to two phases (send phase and receive phase). For both phases, a timeout may be specified (send timeout and receive timeout).

#### Note

When IPC communication happens across CPU cores and a timeout is specified, then the counting of the timeout only begins after the target thread has been scheduled at least once. In particular, this means that an IPC timeout, including a timeout of zero, may be delayed depending on the scheduling on the target CPU core. If a higher priority thread on the target core is executing a busy loop, that delay may even be indefinitely.

#### See also

[Timeouts](#)

### 13.1.14.3 Enumeration Type Documentation

#### 13.1.14.3.1 l4\_syscall\_flags\_t

```
enum l4_syscall_flags_t
```

Capability selector flags.

These flags determine the concrete operation when a kernel object is invoked.

The following combinations of flags are supported when invoking IPC (see [l4\\_ipc\(\)](#)); with other combinations, behavior is undefined:

- [L4\\_SYSF\\_SEND](#): send to specified partner
- [L4\\_SYSF\\_RECV](#): receive from specified partner
- [L4\\_SYSF\\_RECV](#) | [L4\\_SYSF\\_OPEN\\_WAIT](#): receive from any sending partner; see [L4\\_SYSF\\_WAIT](#)
- [L4\\_SYSF\\_SEND](#) | [L4\\_SYSF\\_RECV](#): call specified partner; see [L4\\_SYSF\\_CALL](#)
- [L4\\_SYSF\\_SEND](#) | [L4\\_SYSF\\_RECV](#) | [L4\\_SYSF\\_OPEN\\_WAIT](#): send to specified partner and receive from any sending partner; see [L4\\_SYSF\\_SEND\\_AND\\_WAIT](#)
- [L4\\_SYSF\\_REPLY](#) | [L4\\_SYSF\\_SEND](#): reply to caller
- [L4\\_SYSF\\_REPLY](#) | [L4\\_SYSF\\_SEND](#) | [L4\\_SYSF\\_RECV](#): call the caller
- [L4\\_SYSF\\_REPLY](#) | [L4\\_SYSF\\_SEND](#) | [L4\\_SYSF\\_RECV](#) | [L4\\_SYSF\\_OPEN\\_WAIT](#): reply to caller and receive from any sending partner; see [L4\\_SYSF\\_REPLY\\_AND\\_WAIT](#)

#### Enumerator

L4_SYSF_NONE	<p>Empty set of flags.</p> <p><b>Deprecated</b> Default flags (call to a kernel object).</p> <p>Using this value as flags in the capability selector for an invocation indicates a call (send and wait for a reply).</p>
--------------	--

## Enumerator

L4_SYSF_SEND	Send-phase flag. Setting this flag in a capability selector induces a send phase, this means a message is sent to the object denoted by the capability. For receive phase see <a href="#">L4_SYSF_RECV</a> . In <a href="#">l4_vcpu_state_t::user_task</a> this flag means that the kernel has cached the user task capability internally, see <a href="#">l4_thread_vcpu_resume_commit()</a> .
L4_SYSF_RECV	Receive-phase flag. Setting this flag in a capability selector induces a receive phase, this means the invoking thread waits for a message from the object denoted by the capability. For a send phase see <a href="#">L4_SYSF_SEND</a> .
L4_SYSF_OPEN_WAIT	Open-wait flag. This flag indicates that the receive operation (see <a href="#">L4_SYSF_RECV</a> ) shall be an <i>open wait</i> . <i>Open wait</i> means that the invoking thread shall wait for a message from any possible sender and <i>not</i> from the sender denoted by the capability.
L4_SYSF_REPLY	Reply flag. This flag indicates that the send phase shall use the in-kernel reply capability instead of the capability denoted by the selector index.
L4_SYSF_CALL	Call flags (combines send and receive). Combines <a href="#">L4_SYSF_SEND</a> and <a href="#">L4_SYSF_RECV</a> .
L4_SYSF_WAIT	Wait flags (combines receive and open wait). Combines <a href="#">L4_SYSF_RECV</a> and <a href="#">L4_SYSF_OPEN_WAIT</a> .
L4_SYSF_SEND_AND_WAIT	Send-and-wait flags. Combines <a href="#">L4_SYSF_SEND</a> and <a href="#">L4_SYSF_WAIT</a> .
L4_SYSF_REPLY_AND_WAIT	Reply-and-wait flags. Combines <a href="#">L4_SYSF_SEND</a> , <a href="#">L4_SYSF_REPLY</a> , and <a href="#">L4_SYSF_WAIT</a> .

Definition at line 61 of file [consts.h](#).

## 13.1.14.4 Function Documentation

## 13.1.14.4.1 l4\_ipc()

```
l4_msgtag_t l4_ipc (
    l4_cap_idx_t dest,
    l4_utcb_t * utcb,
    l4_umword_t flags,
    l4_umword_t slabel,
    l4_msgtag_t tag,
    l4_umword_t * rlabel,
    l4_timeout_t timeout ) [inline]
```

Generic [L4](#) object invocation.

## Parameters

	<i>dest</i>	Destination object. <a href="#">L4_INVALID_CAP</a> denotes the current thread. An IPC to the current thread will always abort after the specified timeout and can be used for sleeping without busy waiting.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
	<i>flags</i>	Invocation flags (see <a href="#">l4_syscall_flags_t</a> ).
	<i>slabel</i>	Send label if applicable (may be seen by the receiver).
	<i>tag</i>	Sending message tag.
out	<i>rlabel</i>	Receiving label.
	<i>timeout</i>	Timeout pair (see <a href="#">l4_timeout_t</a> ).

**Returns**

return tag

**See also**

[L4 Inter-Process Communication \(IPC\)](#)

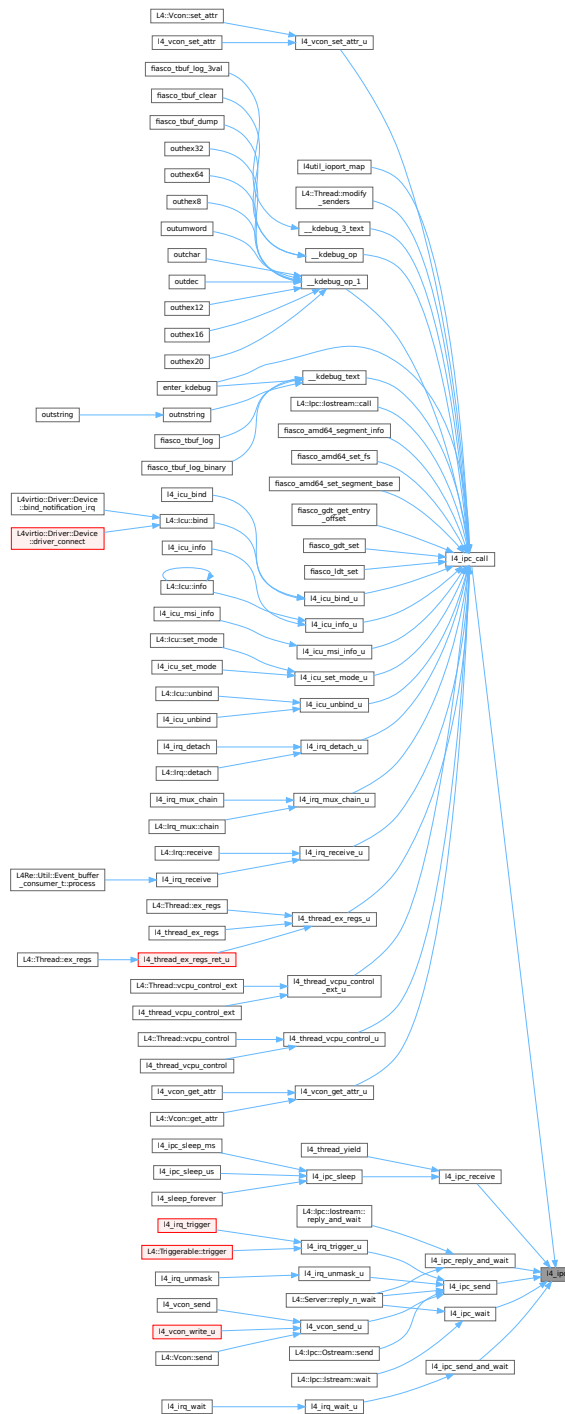
Definition at line 34 of file [ipc.h](#).

References [l4\\_msgtag\\_t::raw](#).

Referenced by [l4\\_ipc\\_call\(\)](#), [l4\\_ipc\\_receive\(\)](#), [l4\\_ipc\\_reply\\_and\\_wait\(\)](#), [l4\\_ipc\\_send\(\)](#), [l4\\_ipc\\_send\\_and\\_wait\(\)](#), and [l4\\_ipc\\_wait\(\)](#).



Here is the caller graph for this function:



#### 13.1.14.4.2 l4\_ipc\_call()

```
l4_msgtag_t l4_ipc_call (
    l4_cap_idx_t object,
    l4_utcb_t * utcb,
```

```
l4_msgtag_t tag,
l4_timeout_t timeout ) [inline]
```

Object call (usual invocation).

#### Parameters

<i>object</i>	Capability selector for the object to call. A value of <a href="#">L4_INVALID_CAP</a> denotes the current thread and will abort the IPC after the time specified in the <code>snd</code> part of the <code>timeout</code> parameter has expired.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
<i>tag</i>	Message tag to describe the message to be sent.
<i>timeout</i>	Timeout pair for send an receive phase (see <a href="#">l4_timeout_t</a> ).

#### Returns

result tag

A message is sent to the object and the invoker waits for a reply from the object. Messages from other sources are not accepted.

#### Note

The send-to-receive transition needs no time, the object can reply with a send timeout of zero.

If a finite receive timeout is specified, the IPC receive operation could abort before the partner was able to send the reply message. Under certain circumstances the partner may still have the temporary reply capability to the calling thread and may use this capability to reply to the caller at a later, unexpected time specifying an arbitrary IPC label. This case is relevant for servers which call another, possibly untrusted, server while serving a client request.

#### See also

[L4 Inter-Process Communication \(IPC\)](#)

#### Examples

[examples/sys/aliens/main.c](#), [examples/sys/ipc/ipc\\_example.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line 576 of file [ipc.h](#).

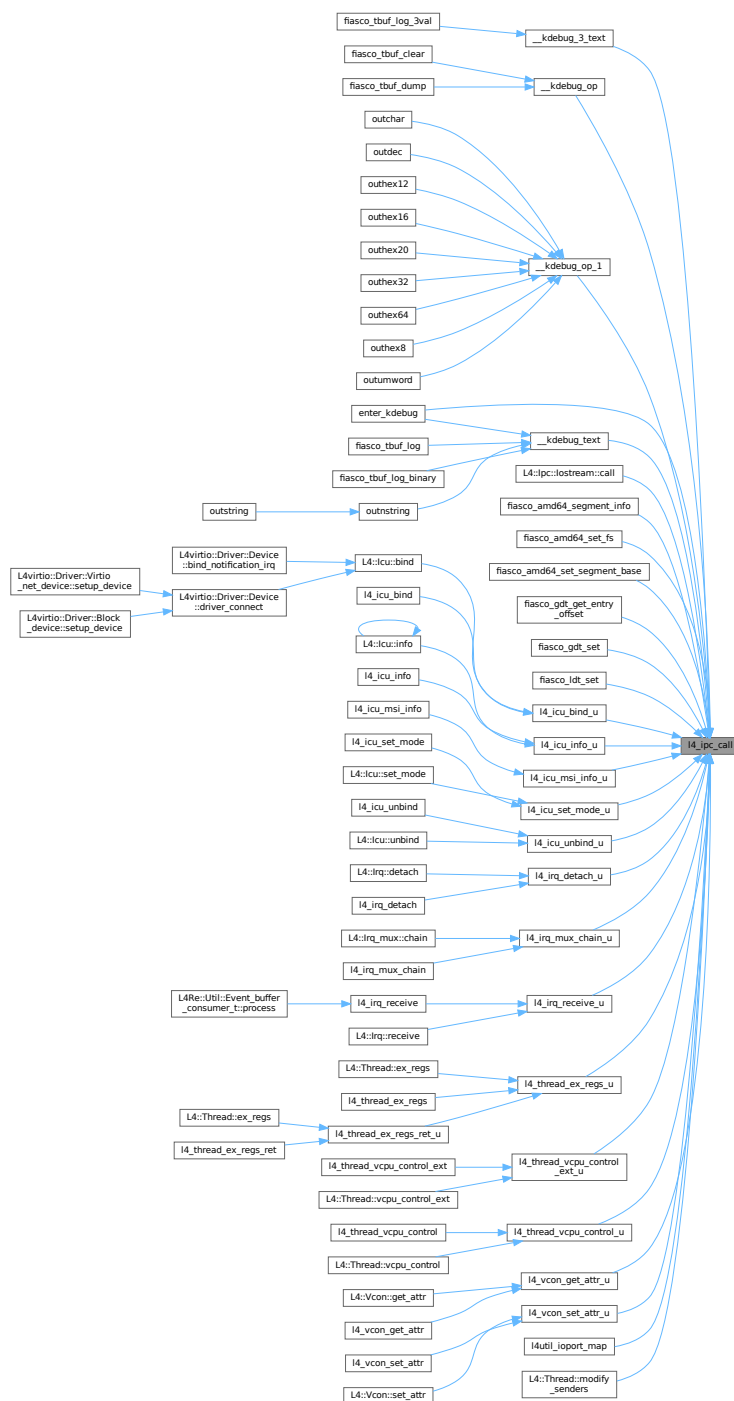
References [l4\\_ipc\(\)](#), and [L4\\_SYSF\\_CALL](#).

Referenced by [\\_\\_kdebug\\_3\\_text\(\)](#), [\\_\\_kdebug\\_op\(\)](#), [\\_\\_kdebug\\_op\\_1\(\)](#), [\\_\\_kdebug\\_text\(\)](#), [L4::ipc::loststream::call\(\)](#), [enter\\_kdebug\(\)](#), [fiasco\\_amd64\\_segment\\_info\(\)](#), [fiasco\\_amd64\\_set\\_fs\(\)](#), [fiasco\\_amd64\\_set\\_segment\\_base\(\)](#), [fiasco\\_gdt\\_get\\_entry\\_offset\(\)](#), [fiasco\\_gdt\\_set\(\)](#), [fiasco\\_ldt\\_set\(\)](#), [l4\\_icu\\_bind\\_u\(\)](#), [l4\\_icu\\_info\\_u\(\)](#), [l4\\_icu\\_msi\\_info\\_u\(\)](#), [l4\\_icu\\_set\\_mode\\_u\(\)](#), [l4\\_icu\\_unbind\\_u\(\)](#), [l4\\_irq\\_detach\\_u\(\)](#), [l4\\_irq\\_mux\\_chain\\_u\(\)](#), [l4\\_irq\\_receive\\_u\(\)](#), [l4\\_thread\\_ex\\_regs\\_u\(\)](#), [l4\\_thread\\_vcpu\\_control\\_ext\\_u\(\)](#), [l4\\_thread\\_vcpu\\_control\\_u\(\)](#), [l4\\_vcon\\_get\\_attr\\_u\(\)](#), [l4\\_vcon\\_set\\_attr\\_u\(\)](#), [l4util\\_ioport\\_map\(\)](#), and [L4::Thread::modify\\_senders\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.14.4.3 l4\_ipc\_receive()

```
14_msgtag_t 14_ipc_receive (
    14_cap_idx_t object,
    14_utcb_t * utcb,
    14_timeout_t timeout ) [inline]
```

Wait for a message from a specific source.

## Parameters

<i>object</i>	Object to receive a message from. A value of <a href="#">L4_INVALID_CAP</a> denotes the current thread. It could be used for sleeping without busy waiting for the time specified in the <code>rcv</code> part of the <code>timeout</code> parameter.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
<i>timeout</i>	Timeout pair (see <a href="#">l4_timeout_t</a> , only the receive part matters).

## Returns

result tag.

This operation waits for a message from the specified object. Messages from other sources are not accepted by this operation. The operation does not include a send phase, this means no message is sent to the object.

## Note

This operation is usually used to receive messages from a specific IRQ or thread. However, it is not common to use this operation for normal applications.

## See also

[L4 Inter-Process Communication \(IPC\)](#)

## Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 613 of file [ipc.h](#).

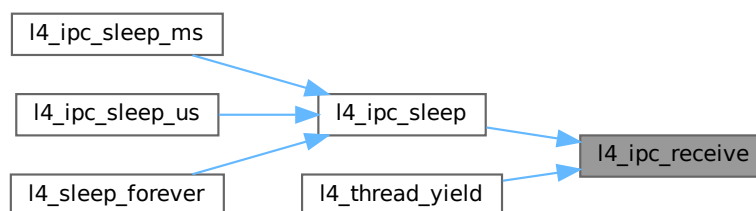
References [l4\\_ipc\(\)](#), [L4\\_SYSF\\_RECV](#), and [l4\\_msgtag\\_t::raw](#).

Referenced by [l4\\_ipc\\_sleep\(\)](#), and [l4\\_thread\\_yield\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.14.4.4 l4\_ipc\_reply\_and\_wait()

```
l4_msgtag_t l4_ipc_reply_and_wait (
    l4_utcb_t * utcb,
    l4_msgtag_t tag,
    l4_umword_t * label,
    l4_timeout_t timeout ) [inline]
```

Reply and wait operation (uses the *reply* capability).

##### Parameters

	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
	<i>tag</i>	Describes the message to be sent as reply.
out	<i>label</i>	Label assigned to the source object of the received message.
	<i>timeout</i>	Timeout pair (see <a href="#">l4_timeout_t</a> ).

##### Returns

result tag

A message is sent to the previous caller using the implicit reply capability. Afterwards the invoking thread waits for a message from any source.

##### Note

This is the standard server operation: it sends a reply to the actual client and waits for the next incoming request, which may come from any other client.

In case of multiple senders trying to send to the thread performing this system call, the thread receives from a sender with the highest priority. In this respect, IRQ sources have the highest priority 255.

##### See also

[L4 Inter-Process Communication \(IPC\)](#)

##### Examples

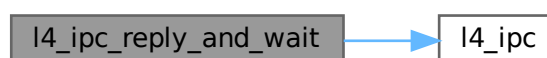
[examples/sys/ipc/ipc\\_example.c](#).

Definition at line 583 of file [ipc.h](#).

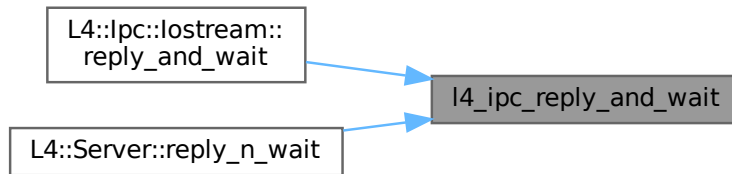
References [L4\\_INVALID\\_CAP](#), [l4\\_ipc\(\)](#), and [L4\\_SYSF\\_REPLY\\_AND\\_WAIT](#).

Referenced by [L4::ipc::loststream::reply\\_and\\_wait\(\)](#), and [L4::Server< LOOP\\_HOOKS >::reply\\_n\\_wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.14.4.5 l4\_ipc\_send()

```

l4_msgtag_t l4_ipc_send (
    l4_cap_idx_t dest,
    l4_utcb_t * utcb,
    l4_msgtag_t tag,
    l4_timeout_t timeout ) [inline]
  
```

Send a message to an object (do **not** wait for a reply).

##### Parameters

<i>dest</i>	Capability selector for the destination object. A value of <a href="#">L4_INVALID_CAP</a> denotes the current thread and could be used for sleeping without busy waiting for the time specified in the <code>snd</code> part of the <code>timeout</code> parameter.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
<i>tag</i>	Descriptor for the message to be sent.
<i>timeout</i>	Timeout pair (see <a href="#">l4_timeout_t</a> ) only send part is relevant.

##### Returns

Syscall return tag for the send-only operation, this means there is no return value except [L4\\_MSGTAG\\_ERROR](#) indicating success or failure of the send operation. Use [l4\\_ipc\\_error\(\)](#) to check for errors and **do not** use [l4\\_error\(\)](#).

A message is sent to the destination object. There is no receive phase included. The invoker continues working after sending the message.

##### Note

This is a special-purpose message transfer. Objects usually support only invocation via [l4\\_ipc\\_call\(\)](#) consisting of a send phase and a receive phase for returning the result of the object invocation. For example, [l4\\_icu\\_unmask\(\)](#), [l4\\_icu\\_mask\(\)](#) and [l4\\_irq\\_trigger\(\)](#) use send-only IPC operations for object invocation.

See also

[L4 Inter-Process Communication \(IPC\)](#)

Examples

[examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 597 of file [ipc.h](#).

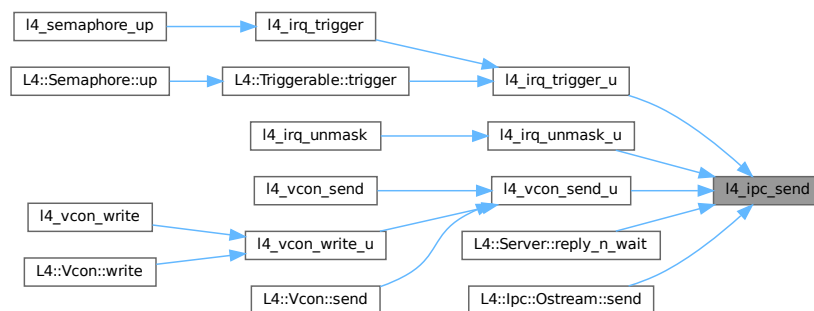
References [l4\\_ipc\(\)](#), and [L4\\_SYSF\\_SEND](#).

Referenced by [l4\\_irq\\_trigger\\_u\(\)](#), [l4\\_irq\\_unmask\\_u\(\)](#), [l4\\_vcon\\_send\\_u\(\)](#), [L4::Server< LOOP\\_HOOKS >::reply\\_n\\_wait\(\)](#), and [L4::Ipc::Ostream::send\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.14.4.6 l4\_ipc\_send\_and\_wait()

```

l4_msgtag_t l4_ipc_send_and_wait (
    l4_cap_idx_t dest,
    l4_utcb_t * utcb,
    l4_msgtag_t tag,
    l4_umword_t * label,
    l4_timeout_t timeout ) [inline]
  
```

Send a message and do an open wait.



## Parameters

	<i>dest</i>	Object to send a message to. A value of <a href="#">L4_INVALID_CAP</a> denotes the current thread and will abort the IPC after the time specified in the <code>snd</code> part of the <code>timeout</code> parameter has expired.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
	<i>tag</i>	Describes the message that shall be sent.
out	<i>label</i>	Label assigned to the source object of the receive phase.
	<i>timeout</i>	Timeout pair (see <a href="#">l4_timeout_t</a> ).

## Returns

result tag

A message is sent to the destination object and the invoking thread waits for a reply from any source.

## Note

This is a special-purpose operation and shall not be used in general applications.

In case of multiple senders trying to send to the thread performing this system call, the thread receives from a sender with the highest priority. In this respect, IRQ sources have the highest priority 255.

## See also

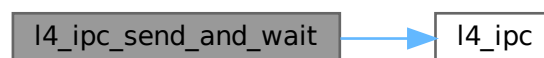
[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 590 of file [ipc.h](#).

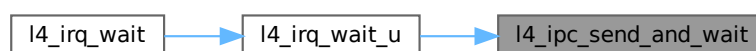
References [l4\\_ipc\(\)](#), and [L4\\_SYSF\\_SEND\\_AND\\_WAIT](#).

Referenced by [l4\\_irq\\_wait\\_u\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.14.4.7 l4\_ipc\_sleep()

```
l4_msgtag_t l4_ipc_sleep (
    l4_timeout_t timeout ) [inline]
```

Sleep for an amount of time.

##### Parameters

<i>timeout</i>	Timeout pair (see <a href="#">l4_timeout_t</a> , the receive part matters).
----------------	---

##### Returns

error code:

- [L4\\_IPC\\_RETIMEOUT](#): success
- [L4\\_IPC\\_RECANCELED](#) woken up by a different thread ([l4\\_thread\\_ex\\_regs\(\)](#)).

The invoking thread waits until the timeout is expired or the wait was aborted by another thread by [l4\\_thread\\_ex\\_regs\(\)](#).

##### See also

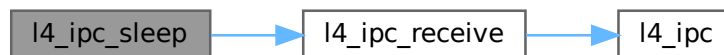
[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 622 of file [ipc.h](#).

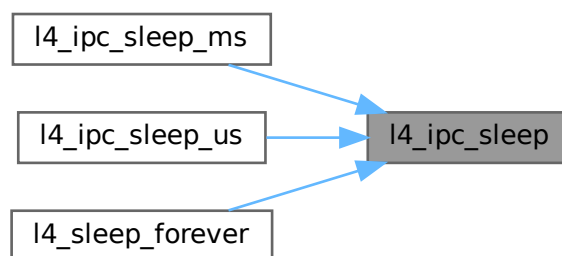
References [L4\\_INVALID\\_CAP](#), and [l4\\_ipc\\_receive\(\)](#).

Referenced by [l4\\_ipc\\_sleep\\_ms\(\)](#), [l4\\_ipc\\_sleep\\_us\(\)](#), and [l4\\_sleep\\_forever\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.14.4.8 l4\_ipc\_sleep\_ms()

```
l4_msgtag_t l4_ipc_sleep_ms (
    l4_uint32_t ms ) [inline]
```

Sleep for a certain amount of milliseconds.

##### Parameters

<i>ms</i>	Number of milliseconds to wait.
-----------	---------------------------------

##### Returns

error code:

- [L4\\_IPC\\_RETIMEOUT](#): success
- [L4\\_IPC\\_RECANCELED](#) woken up by a different thread ([l4\\_thread\\_ex\\_regs\(\)](#)).

The invoking thread waits until the timeout is expired or the wait was aborted by another thread by [l4\\_thread\\_ex\\_regs\(\)](#).

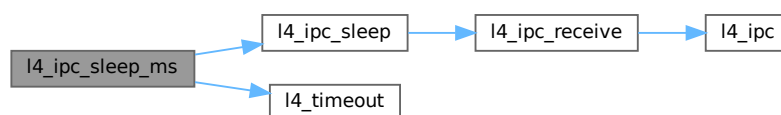
##### See also

[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 626 of file [ipc.h](#).

References [l4\\_ipc\\_sleep\(\)](#), [L4\\_IPC\\_TIMEOUT\\_NEVER](#), and [l4\\_timeout\(\)](#).

Here is the call graph for this function:



#### 13.1.14.4.9 l4\_ipc\_sleep\_us()

```
l4_msgtag_t l4_ipc_sleep_us (
    l4_uint64_t us ) [inline]
```

Sleep for a certain amount of microseconds.

##### Parameters

<i>us</i>	Number of microseconds to wait.
-----------	---------------------------------

**Returns**

error code:

- [L4\\_IPC\\_RETIMEOUT](#): success
- [L4\\_IPC\\_RECANCELED](#) woken up by a different thread ([l4\\_thread\\_ex\\_regs\(\)](#)).

The invoking thread waits until the timeout is expired or the wait was aborted by another thread by [l4\\_thread\\_ex\\_regs\(\)](#).

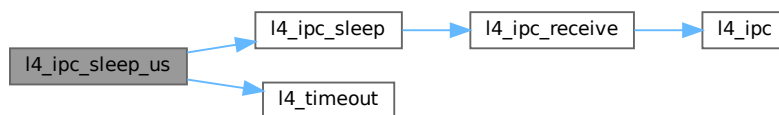
**See also**

[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 633 of file [ipc.h](#).

References [l4\\_ipc\\_sleep\(\)](#), [L4\\_IPC\\_TIMEOUT\\_NEVER](#), and [l4\\_timeout\(\)](#).

Here is the call graph for this function:

**13.1.14.4.10 l4\_ipc\_wait()**

```

l4_msgtag_t l4_ipc_wait (
    l4_utcb_t * utcb,
    l4_umword_t * label,
    l4_timeout_t timeout ) [inline]
  
```

Wait for an incoming message from any possible sender.

**Parameters**

	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
out	<i>label</i>	Label assigned to the source object (IPC gate or IRQ).
	<i>timeout</i>	Timeout pair (see <a href="#">l4_timeout_t</a> , only the receive part is used).

**Returns**

return tag

This operation does an open wait, and therefore needs no capability to denote the possible source of a message. This means the calling thread waits for an incoming message from any possible source. There is no send phase included in this operation.

The usual usage of this function is to call that function when entering a server loop in a user-level server that implements user-level objects, see also [l4\\_ipc\\_reply\\_and\\_wait\(\)](#).

#### Note

In case of multiple senders trying to send to the thread performing this system call, the thread receives from a sender with the highest priority. In this respect, IRQ sources have the highest priority 255.

#### See also

[L4 Inter-Process Communication \(IPC\)](#)

#### Examples

[examples/sys/ipc/ipc\\_example.c](#).

Definition at line 604 of file [ipc.h](#).

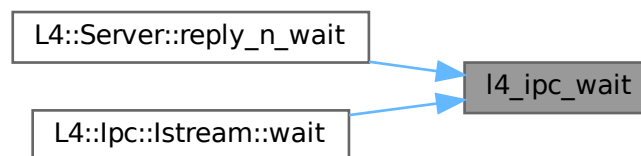
References [L4\\_INVALID\\_CAP](#), [l4\\_ipc\(\)](#), [L4\\_SYSF\\_WAIT](#), and [l4\\_msgtag\\_t::raw](#).

Referenced by [L4::Server< LOOP\\_HOOKS >::reply\\_n\\_wait\(\)](#), and [L4::lpc::lstream::wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.14.4.11 l4\_sndfpage\_add()

```

int l4_sndfpage_add (
    l4_fpage_t const snd_fpage,
    unsigned long snd_base,
    l4_msgtag_t * tag ) [inline]
  
```

Add a flex-page to be sent to the UTCB.

## Parameters

	<i>snd_fpage</i>	Flex-page.
	<i>snd_base</i>	Send base.
<i>in, out</i>	<i>tag</i>	Tag to be updated. Only the number of items is incremented in the updated tag, all other members remain unmodified.

## Returns

0 on success, negative error code otherwise

## See also

[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 696 of file [ipc.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 13.1.14.5 Error Handling

Error handling for [L4](#) object invocation.

Collaboration diagram for Error Handling:



## Enumerations

- enum `l4_ipc_tcr_error_t` {  
`L4_IPC_ERROR_MASK` = 0x1F , `L4_IPC_SND_ERR_MASK` = 0x01 , `L4_IPC_ENOT_EXISTENT` = 0x04 ,  
`L4_IPC_RETIMEOUT` = 0x03 ,  
`L4_IPC_SETIMEOUT` = 0x02 , `L4_IPC_RECANCELED` = 0x07 , `L4_IPC_SECANCELED` = 0x06 ,  
`L4_IPC_REMAPFAILED` = 0x11 ,  
`L4_IPC_SEMAPFAILED` = 0x10 , `L4_IPC_RESNDPFTO` = 0x0b , `L4_IPC_SESNDPFTO` = 0x0a ,  
`L4_IPC_RERCVPFTO` = 0x0d ,  
`L4_IPC_SERCVPFTO` = 0x0c , `L4_IPC_REABORTED` = 0x0f , `L4_IPC_SEABORTED` = 0x0e ,  
`L4_IPC_REMSGCUT` = 0x09 ,  
`L4_IPC_SEMSGCUT` = 0x08 }

Error codes in the error TCR.

## Functions

- `l4_umword_t l4_ipc_error (l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW`  
Get the IPC error code for an IPC operation.
- long `l4_error (l4_msgtag_t tag) L4_NOTHROW`  
Get IPC error code if any or message tag label otherwise for an IPC call.
- int `l4_ipc_is_snd_error (l4_utcb_t *utcb) L4_NOTHROW`  
Returns whether an error occurred in send phase of an invocation.
- int `l4_ipc_is_rcv_error (l4_utcb_t *utcb) L4_NOTHROW`  
Returns whether an error occurred in receive phase of an invocation.
- int `l4_ipc_error_code (l4_utcb_t *utcb) L4_NOTHROW`  
Get the error condition of the last invocation from the TCR.

### 13.1.14.5.1 Detailed Description

Error handling for L4 object invocation.

#### Include File

```
#include <l4/sys/ipc.h>
```

### 13.1.14.5.2 Enumeration Type Documentation

#### 13.1.14.5.2.1 l4\_ipc\_tcr\_error\_t

```
enum l4_ipc_tcr_error_t
```

Error codes in the *error* TCR.

The error codes are accessible via the *error* TCR, see `l4_thread_regs_t.error`.

#### Enumerator

<code>L4_IPC_ERROR_MASK</code>	Mask for error bits.
<code>L4_IPC_SND_ERR_MASK</code>	Send error mask.
<code>L4_IPC_ENOT_EXISTENT</code>	Non-existing destination or source.
<code>L4_IPC_RETIMEOUT</code>	Timeout during receive operation.

## Enumerator

L4_IPC_SETIMEOUT	Timeout during send operation.
L4_IPC_RECANCELED	Receive operation canceled.
L4_IPC_SECANCELED	Send operation canceled.
L4_IPC_REMAPFAILED	Map flexpage failed in receive operation.
L4_IPC_SEMAPFAILED	Map flexpage failed in send operation.
L4_IPC_RESNDPFTO	Send-pagefault timeout in receive operation.
L4_IPC_SESNDPFTO	Send-pagefault timeout in send operation.
L4_IPC_RERCVPFTO	Receive-pagefault timeout in receive operation.
L4_IPC_SERCVPFTO	Receive-pagefault timeout in send operation.
L4_IPC_REABORTED	Receive operation aborted.
L4_IPC_SEABORTED	Send operation aborted.
L4_IPC_REMSGCUT	Received message truncated. Usually returned when the typed items to be sent by the IPC partner exceed the buffer registers of the respective types.
L4_IPC_SEMSGCUT	Sent message truncated. Usually returned when the typed items to be sent exceed the IPC partner's buffer registers of the respective types.

Definition at line 92 of file [ipc.h](#).

### 13.1.14.5.3 Function Documentation

#### 13.1.14.5.3.1 l4\_error()

```
long l4_error (
    l4_msgtag_t tag ) [inline]
```

Get IPC error code if any or message tag label otherwise for an IPC call.

This function shall only be used if the IPC operation includes a receive phase (usually a call operation), otherwise no tag label is received and the return value of this function is undefined.

## Parameters

<i>tag</i>	Message tag returned by the IPC call.
------------	---------------------------------------

## Returns

In case of an IPC error, a negative error code in the range of [L4\\_EIPC\\_LO](#) to [L4\\_EIPC\\_HI](#) (see [l4\\_ipc\\_to\\_errno\(\)](#) and [l4\\_ipc\\_tcr\\_error\\_t](#)), otherwise the tag label. By convention, the callee can signal errors via a negative tag label (negated value from [l4\\_error\\_code\\_t](#)) and success via a non-negative value.

## Examples

[examples/libs/l4re/streammap/client.cc](#), [examples/sys/aliens/main.c](#), [examples/sys/isr/main.c](#), [examples/sys/migrate/thread\\_migration.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 657 of file [ipc.h](#).

References [l4\\_utcb\(\)](#).

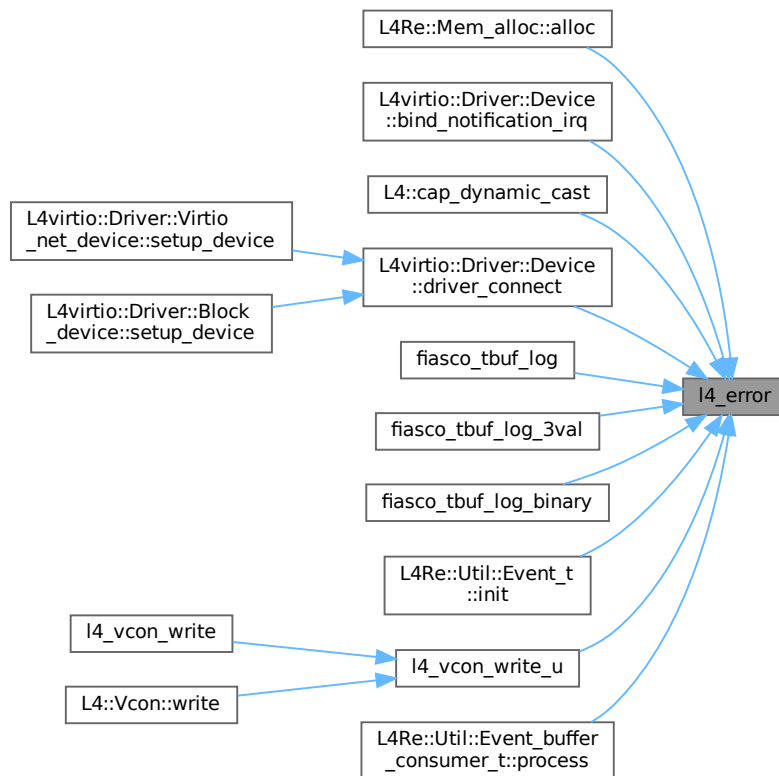


Referenced by [L4Re::Mem\\_alloc::alloc\(\)](#), [L4virtio::Driver::Device::bind\\_notification\\_irq\(\)](#), [L4::cap\\_dynamic\\_cast\(\)](#), [L4virtio::Driver::Device::driver\\_connect\(\)](#), [fiasco\\_tbuf\\_log\(\)](#), [fiasco\\_tbuf\\_log\\_3val\(\)](#), [fiasco\\_tbuf\\_log\\_binary\(\)](#), [L4Re::Util::Event\\_t< PAYLOAD >::init\(\)](#), [l4\\_vcon\\_write\\_u\(\)](#), and [L4Re::Util::Event\\_buffer\\_consumer\\_t< PAYLOAD >::process\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 13.1.14.5.3.2 l4\_ipc\_error()

```

l4_umword_t l4_ipc_error (
    l4_msgtag_t tag,
    l4_utcb_t * utcb ) [inline]
  
```

Get the IPC error code for an IPC operation.

## Parameters

<i>tag</i>	Message tag returned by the IPC operation.
<i>utcb</i>	UTCB that was used for the IPC operation.

## Returns

0 if no error condition is set, error code otherwise (see [l4\\_ipc\\_tcr\\_error\\_t](#)).

## Examples

[examples/sys/ipc/ipc\\_example.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 640 of file [ipc.h](#).

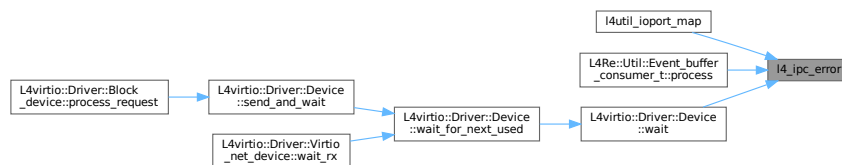
References [l4\\_thread\\_regs\\_t::error](#), [L4\\_IPC\\_ERROR\\_MASK](#), [L4\\_LIKELY](#), and [l4\\_msgtag\\_has\\_error\(\)](#).

Referenced by [l4util\\_ioport\\_map\(\)](#), [L4Re::Util::Event\\_buffer\\_consumer\\_t< PAYLOAD >::process\(\)](#), and [L4virtio::Driver::Device::wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 13.1.14.5.3.3 l4\_ipc\_error\_code()

```
int l4_ipc_error_code (
    l4_utcb_t * utcb ) [inline]
```

Get the error condition of the last invocation from the TCR.

## Precondition

`l4_msgtag_has_error(tag) == true`

**Parameters**

<i>utcb</i>	UTCB to check.
-------------	----------------

**Returns**

Error condition of type `l4_ipc_tcr_error_t`.

Definition at line 669 of file `ipc.h`.

References `l4_thread_regs_t::error`, and `L4_IPC_ERROR_MASK`.

**13.1.14.5.3.4 l4\_ipc\_is\_rcv\_error()**

```
int l4_ipc_is_rcv_error (  
    l4_utcb_t * utcb ) [inline]
```

Returns whether an error occurred in receive phase of an invocation.

**Precondition**

`l4_msgtag_has_error(tag) == true`

**Parameters**

<i>utcb</i>	UTCB to check.
-------------	----------------

**Returns**

Boolean value.

Definition at line 666 of file `ipc.h`.

References `l4_thread_regs_t::error`.

**13.1.14.5.3.5 l4\_ipc\_is\_snd\_error()**

```
int l4_ipc_is_snd_error (  
    l4_utcb_t * utcb ) [inline]
```

Returns whether an error occurred in send phase of an invocation.

**Precondition**

`l4_msgtag_has_error(tag) == true`

## Parameters

<i>utcb</i>	UTCB to check.
-------------	----------------

## Returns

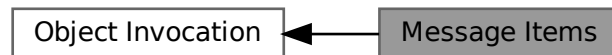
Boolean value.

Definition at line 663 of file [ipc.h](#).

### 13.1.14.6 Message Items

Message item related functions.

Collaboration diagram for Message Items:



## Enumerations

- enum [l4\\_msg\\_item\\_consts\\_t](#) {  
[L4\\_ITEM\\_MAP](#) = 8 , [L4\\_ITEM\\_CONT](#) = 1 , [L4\\_MAP\\_ITEM\\_GRANT](#) = 2 , [L4\\_MAP\\_ITEM\\_MAP](#) = 0 ,  
[L4\\_RCV\\_ITEM\\_SINGLE\\_CAP](#) = [L4\\_ITEM\\_MAP](#) | 2 , [L4\\_RCV\\_ITEM\\_LOCAL\\_ID](#) = 4 }

*Constants for message items.*

## Functions

- [l4\\_umword\\_t l4\\_map\\_control](#) ([l4\\_umword\\_t](#) spot, unsigned char cache, unsigned grant) [L4\\_NOTHROW](#)  
*Create the first word for a map item for the memory space.*
- [l4\\_umword\\_t l4\\_map\\_obj\\_control](#) ([l4\\_umword\\_t](#) spot, unsigned grant) [L4\\_NOTHROW](#)  
*Create the first word for a map item for the object space.*

#### 13.1.14.6.1 Detailed Description

Message item related functions.

Message items are typed items that can be transferred via IPC operations. Message items are also used to specify receive windows for typed items to be received. Message items are placed in the message registers (MRs) of the UTCB of the sending thread. Receive items are placed in the buffer registers (BRs) of the UTCB of the receiving thread.

Message items are usually two-word data structures. The first word denotes the type of the message item (for example a memory flex-page, io flex-page or object flex-page) and the second word contains information depending on the type. There is actually one exception that is a small (one word) receive buffer item for a single capability.

### 13.1.14.6.2 Enumeration Type Documentation

#### 13.1.14.6.2.1 l4\_msg\_item\_consts\_t

enum `l4_msg_item_consts_t`

Constants for message items.

Enumerator

L4_ITEM_MAP	Identify a message item as <i>map item</i> .
L4_ITEM_CONT	Denote that the following item shall be put into the same receive item as this one.
L4_MAP_ITEM_GRANT	<p>Flag as <i>grant</i> instead of <i>map</i> operation. This means, the sender delegates access to the receiver and the kernel removes the rights from the sender (basically a move operation). The mapping in the receiver gets the new parent of any child mappings of the mapping of the sender. Rights revocation via send item/flexpage is <i>not</i> guaranteed to be applied to descendant mappings in case of grant. See <a href="#">Spaces and Mappings</a> for more details on map/grant.</p> <p><b>Note</b></p> <p>The grant operation is not performed if the resulting rights of the receiver mapping would not contain the <a href="#">L4_CAP_FPAGE_R</a> bit (for object capabilities) or none of the <a href="#">L4_FPAGE_RWX</a> bits (memory and IO ports). In that case, the mapping is not created in the receiver space and not removed from the sender space.</p> <p>If the removal of the whole mapping from the sender is not possible because the size of the mapped frame at the sender exceeds the size defined by the send or receive flexpage, the grant operation is turned into a regular map operation and the mapping is <i>not</i> removed from the sender. This would happen if, for example, a smaller part of an <a href="#">L4</a> superpage mapping shall be granted.</p>
L4_MAP_ITEM_MAP	Flag as usual <i>map</i> operation.
L4_RCV_ITEM_SINGLE_CAP	Mark the receive buffer to be a small receive item that describes a buffer for a single object capability.
L4_RCV_ITEM_LOCAL_ID	<p>The receiver requests to receive a local ID instead of a mapping whenever possible. This flag may only be used for small buffers, see <a href="#">L4_RCV_ITEM_SINGLE_CAP</a>.</p> <p>When this flag is set, then,</p> <ul style="list-style-type: none"> <li>• when sender and receiver are bound to the same task, then no mapping is done for this item and just the capability index is transferred,</li> <li>• otherwise, when the sender specified an IPC gate for transfer that is bound to a thread that is bound to the same task as the receiving thread, then no mapping is done for this item and just the label bitwise disjoint with the <a href="#">L4_CAP_FPAGE_W</a> and <a href="#">L4_CAP_FPAGE_S</a> permissions that would have been mapped is transferred,</li> <li>• otherwise a regular mapping is done for this item.</li> </ul>

Definition at line 224 of file [consts.h](#).



## Parameters

<i>spot</i>	Hot spot address, used to determine what is actually mapped when send and receive flex pages have different size.
<i>grant</i>	Indicates if it is a map item or a grant item. Allowed values: <a href="#">L4_MAP_ITEM_MAP</a> , <a href="#">L4_MAP_ITEM_GRANT</a> .

## Returns

The value to be used as first word in a map item for kernel objects or IO-ports.

Definition at line 714 of file [\\_\\_l4\\_fpage.h](#).

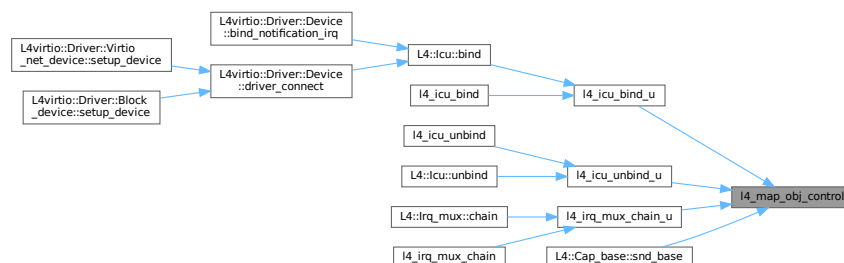
References [l4\\_map\\_control\(\)](#).

Referenced by [l4\\_icu\\_bind\\_u\(\)](#), [l4\\_icu\\_unbind\\_u\(\)](#), [l4\\_irq\\_mux\\_chain\\_u\(\)](#), and [L4::Cap\\_base::snd\\_base\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 13.1.14.7 Message Tag

API related to the message tag data type.

Collaboration diagram for Message Tag:



## Data Structures

- struct [l4\\_msgtag\\_t](#)  
*Message tag data structure.*

## Typedefs

- typedef struct [l4\\_msgtag\\_t](#) [l4\\_msgtag\\_t](#)  
*Message tag data structure.*

## Enumerations

- enum [L4\\_platform\\_ctl\\_proto](#) { [L4\\_PROTO\\_PLATFORM\\_CTL](#) = 0 }  
*Predefined protocol type for messages to platform-control objects.*
- enum [L4\\_msgtag\\_protocol](#) {  
[L4\\_PROTO\\_NONE](#) = 0 , [L4\\_PROTO\\_ALLOW\\_SYSCALL](#) = 1 , [L4\\_PROTO\\_PF\\_EXCEPTION](#) = 1 ,  
[L4\\_PROTO\\_IRQ](#) = -1L ,  
[L4\\_PROTO\\_PAGE\\_FAULT](#) = -2L , [L4\\_PROTO\\_PREEMPTION](#) = -3L , [L4\\_PROTO\\_SYS\\_EXCEPTION](#) = -4L ,  
[L4\\_PROTO\\_EXCEPTION](#) = -5L ,  
[L4\\_PROTO\\_SIGMA0](#) = -6L , [L4\\_PROTO\\_IO\\_PAGE\\_FAULT](#) = -8L , [L4\\_PROTO\\_KOBJECT](#) = -10L ,  
[L4\\_PROTO\\_TASK](#) = -11L ,  
[L4\\_PROTO\\_THREAD](#) = -12L , [L4\\_PROTO\\_LOG](#) = -13L , [L4\\_PROTO\\_SCHEDULER](#) = -14L ,  
[L4\\_PROTO\\_FACTORY](#) = -15L ,  
[L4\\_PROTO\\_VM](#) = -16L , [L4\\_PROTO\\_DMA\\_SPACE](#) = -17L , [L4\\_PROTO\\_IRQ\\_SENDER](#) = -18L ,  
[L4\\_PROTO\\_IRQ\\_MUX](#) = -19L ,  
[L4\\_PROTO\\_SEMAPHORE](#) = -20L , [L4\\_PROTO\\_META](#) = -21L , [L4\\_PROTO\\_IOMMU](#) = -22L ,  
[L4\\_PROTO\\_DEBUGGER](#) = -23L ,  
[L4\\_PROTO\\_SMCCC](#) = -24L , [L4\\_PROTO\\_VCPU\\_CONTEXT](#) = -25L }  
*Message tag for IPC operations.*
- enum [L4\\_msgtag\\_flags](#) { [L4\\_MSGTAG\\_ERROR](#) , [L4\\_MSGTAG\\_TRANSFER\\_FPU](#) , [L4\\_MSGTAG\\_SCHEDULE](#) ,  
[L4\\_MSGTAG\\_PROPAGATE](#) = 0x4000 , [L4\\_MSGTAG\\_FLAGS](#) }  
*Flags for message tags.*

## Functions

- [l4\\_msgtag\\_t](#) [l4\\_msgtag](#) (long label, unsigned words, unsigned items, unsigned flags) [L4\\_NOTHROW](#)  
*Create a message tag from the specified values.*
- long [l4\\_msgtag\\_label](#) ([l4\\_msgtag\\_t](#) t) [L4\\_NOTHROW](#)  
*Get the protocol of tag.*
- unsigned [l4\\_msgtag\\_words](#) ([l4\\_msgtag\\_t](#) t) [L4\\_NOTHROW](#)  
*Get the number of untyped words.*
- unsigned [l4\\_msgtag\\_items](#) ([l4\\_msgtag\\_t](#) t) [L4\\_NOTHROW](#)  
*Get the number of typed items.*
- unsigned [l4\\_msgtag\\_flags](#) ([l4\\_msgtag\\_t](#) t) [L4\\_NOTHROW](#)  
*Get the flags.*
- unsigned [l4\\_msgtag\\_has\\_error](#) ([l4\\_msgtag\\_t](#) t) [L4\\_NOTHROW](#)  
*Test for error indicator flag.*
- unsigned [l4\\_msgtag\\_is\\_page\\_fault](#) ([l4\\_msgtag\\_t](#) t) [L4\\_NOTHROW](#)  
*Test for page-fault protocol.*
- unsigned [l4\\_msgtag\\_is\\_preemption](#) ([l4\\_msgtag\\_t](#) t) [L4\\_NOTHROW](#)  
*Test for preemption protocol.*



- unsigned `l4_msgtag_is_sys_exception` (`l4_msgtag_t` `t`) `L4_NOTHROW`  
*Test for system-exception protocol.*
- unsigned `l4_msgtag_is_exception` (`l4_msgtag_t` `t`) `L4_NOTHROW`  
*Test for exception protocol.*
- unsigned `l4_msgtag_is_sigma0` (`l4_msgtag_t` `t`) `L4_NOTHROW`  
*Test for sigma0 protocol.*
- unsigned `l4_msgtag_is_io_page_fault` (`l4_msgtag_t` `t`) `L4_NOTHROW`  
*Test for IO-page-fault protocol.*

#### 13.1.14.7.1 Detailed Description

API related to the message tag data type.

##### Include File

```
#include <l4/sys/types.h>
```

#### 13.1.14.7.2 Typedef Documentation

##### 13.1.14.7.2.1 `l4_msgtag_t`

```
typedef struct l4_msgtag_t l4_msgtag_t
```

Message tag data structure.

##### Include File

```
#include <l4/sys/types.h>
```

Describes the details of an IPC operation, in particular which parts of the UTCB have to be transmitted, and also flags to enable real-time and FPU extensions.

The message tag also contains a user-defined label that could be used to specify a protocol ID. Some negative values are reserved for kernel protocols such as page faults and exceptions.

The type must be treated completely opaque.

#### 13.1.14.7.3 Enumeration Type Documentation

##### 13.1.14.7.3.1 `L4_msgtag_flags`

```
enum L4_msgtag_flags
```

Flags for message tags.

##### Enumerator

<code>L4_MSGTAG_ERROR</code>	Error indicator flag.
<code>L4_MSGTAG_TRANSFER_FPU</code>	Enable FPU transfer flag for IPC. By enabling this flag when sending IPC, the sender indicates that the contents of the FPU shall be transferred to the receiving thread. However, the receiver has to indicate its willingness to receive FPU context in its buffer descriptor register (BDR).
Generated for L4Re by Doxygen	
<code>L4_MSGTAG_SCHEDULE</code>	Enable schedule in IPC flag. Usually IPC operations donate the remaining time slice of a thread to the called thread. Enabling this flag when sending IPC does a real scheduling decision. However, this flag decreases IPC

Definition at line 96 of file [types.h](#).

### 13.1.14.7.3.2 L4\_msgtag\_protocol

```
enum L4_msgtag_protocol
```

Message tag for IPC operations.

All predefined protocols used by the kernel.

#### Enumerator

L4_PROTO_NONE	Default protocol tag to reply to kernel.
L4_PROTO_ALLOW_SYSCALL	Allow an alien the system call.
L4_PROTO_PF_EXCEPTION	Make an exception out of a page fault.
L4_PROTO_IRQ	IRQ message.
L4_PROTO_PAGE_FAULT	Page fault message.
L4_PROTO_PREEMPTION	Preemption message.
L4_PROTO_SYS_EXCEPTION	System exception.
L4_PROTO_EXCEPTION	Exception.
L4_PROTO_SIGMA0	Sigma0 protocol.
L4_PROTO_IO_PAGE_FAULT	I/O page fault message.
L4_PROTO_KOBJECT	Protocol for messages to a generic kobject.
L4_PROTO_TASK	Protocol for messages to a task object.
L4_PROTO_THREAD	Protocol for messages to a thread object.
L4_PROTO_LOG	Protocol for messages to a log object.
L4_PROTO_SCHEDULER	Protocol for messages to a scheduler object.
L4_PROTO_FACTORY	Protocol for messages to a factory object.
L4_PROTO_VM	Protocol for messages to a virtual machine object.
L4_PROTO_DMA_SPACE	Protocol for (creating) kernel DMA space objects.
L4_PROTO_IRQ_SENDER	Protocol for IRQ senders (IRQ -> IPC)
L4_PROTO_IRQ_MUX	Protocol for IRQ mux (IRQ -> n x IRQ)
L4_PROTO_SEMAPHORE	Protocol for semaphore objects.
L4_PROTO_META	Meta information protocol.
L4_PROTO_IOMMU	Protocol ID for IO-MMUs.
L4_PROTO_DEBUGGER	Protocol ID for the debugger.
L4_PROTO_SMCCC	Protocol ID for ARM SMCCC calls.
L4_PROTO_VCPU_CONTEXT	Protocol for hardware vCPU contexts.

Definition at line 49 of file [types.h](#).

### 13.1.14.7.3.3 L4\_platform\_ctl\_proto

```
enum L4_platform_ctl_proto
```

Predefined protocol type for messages to platform-control objects.

## Enumerator

L4_PROTO_PLATFORM_CTL	Protocol messages to a platform control object. See <a href="#">L4_platform_ctl_ops</a> for allowed operations.
-----------------------	---

Definition at line 185 of file [platform\\_control.h](#).

## 13.1.14.7.4 Function Documentation

## 13.1.14.7.4.1 l4\_msgtag()

```
l4_msgtag_t l4_msgtag (
    long label,
    unsigned words,
    unsigned items,
    unsigned flags ) [inline]
```

Create a message tag from the specified values.

Message tag functions.

## Parameters

<i>label</i>	The user-defined label
<i>words</i>	The number of untyped words within the UTCB
<i>items</i>	The number of typed items (e.g., flex pages) within the UTCB
<i>flags</i>	The IPC flags for realtime and FPU extensions

## Returns

Message tag

## Examples

[examples/sys/aliens/main.c](#), [examples/sys/ipc/ipc\\_example.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 428 of file [types.h](#).

Referenced by [\\_\\_kdebug\\_3\\_text\(\)](#), [\\_\\_kdebug\\_op\(\)](#), [\\_\\_kdebug\\_op\\_1\(\)](#), [\\_\\_kdebug\\_text\(\)](#), [L4::Irqep\\_t< Derived, BASE, bool >::dispatch\(\)](#), [enter\\_kdebug\(\)](#), [fiasco\\_amd64\\_segment\\_info\(\)](#), [fiasco\\_amd64\\_set\\_fs\(\)](#), [fiasco\\_amd64\\_set\\_segment\\_base\(\)](#), [fiasco\\_gdt\\_get\\_entry\\_offset\(\)](#), [fiasco\\_gdt\\_set\(\)](#), [fiasco\\_ldt\\_set\(\)](#), [L4::Server< LOOP\\_HOOKS >::internal\\_loop\(\)](#), [l4\\_icu\\_bind\\_u\(\)](#), [l4\\_icu\\_info\\_u\(\)](#), [l4\\_icu\\_msi\\_info\\_u\(\)](#), [l4\\_icu\\_set\\_mode\\_u\(\)](#), [l4\\_icu\\_unbind\\_u\(\)](#), [l4\\_irq\\_detach\\_u\(\)](#), [l4\\_irq\\_mux\\_chain\\_u\(\)](#), [l4\\_irq\\_receive\\_u\(\)](#), [l4\\_irq\\_trigger\\_u\(\)](#), [l4\\_irq\\_unmask\\_u\(\)](#), [l4\\_irq\\_wait\\_u\(\)](#), [l4\\_thread\\_ex\\_regs\\_u\(\)](#), [l4\\_thread\\_vcpu\\_control\\_ext\\_u\(\)](#), [l4\\_thread\\_vcpu\\_control\\_u\(\)](#), [l4\\_thread\\_yield\(\)](#), [l4\\_vcon\\_get\\_attr\\_u\(\)](#), [l4\\_vcon\\_send\\_u\(\)](#), [l4\\_vcon\\_set\\_attr\\_u\(\)](#), [l4util\\_ioport\\_map\(\)](#), [L4::Thread::modify\\_senders\(\)](#), [L4::lpc\\_svr::Exc\\_dispatch< R, Exc >::operator\(\)\(\)](#), [L4::Cap\\_base::validate\(\)](#), and [L4::Cap\\_base::validate\(\)](#).



## Parameters

<i>t</i>	The tag
----------	---------

## Returns

Flags

Definition at line 452 of file [types.h](#).**13.1.14.7.4.3 l4\_msgtag\_has\_error()**

```
unsigned l4_msgtag_has_error (
    l4_msgtag_t t ) [inline]
```

Test for error indicator flag.

## Parameters

<i>t</i>	The tag
----------	---------

## Returns

&gt;0 for yes, 0 for no

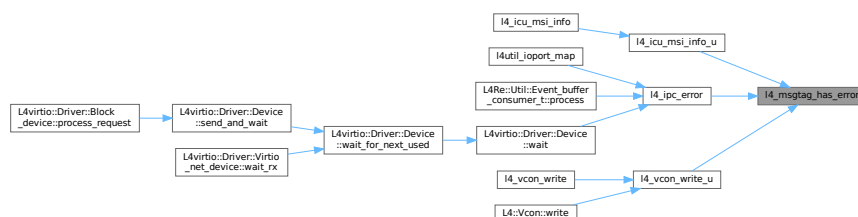
Return whether the kernel operation caused a communication error, e.g. with IPC. if true: utcb->error is valid, otherwise utcb->error is not valid

## Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 457 of file [types.h](#).References [L4\\_MSGTAG\\_ERROR](#).Referenced by [l4\\_icu\\_msi\\_info\\_u\(\)](#), [l4\\_ipc\\_error\(\)](#), and [l4\\_vcon\\_write\\_u\(\)](#).

Here is the caller graph for this function:



#### 13.1.14.7.4.4 l4\_msgtag\_is\_exception()

```
unsigned l4_msgtag_is_exception (
    l4_msgtag_t t ) [inline]
```

Test for exception protocol.

**Parameters**

<i>t</i>	The tag
----------	---------

**Returns**

Boolean value

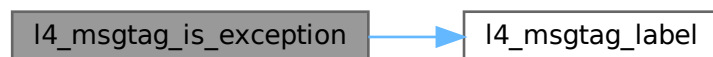
**Examples**

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 471 of file [types.h](#).

References [l4\\_msgtag\\_label\(\)](#), and [L4\\_PROTO\\_EXCEPTION](#).

Here is the call graph for this function:

**13.1.14.7.4.5 l4\_msgtag\_is\_io\_page\_fault()**

```
unsigned l4_msgtag_is_io_page_fault (  
    l4_msgtag_t t ) [inline]
```

Test for IO-page-fault protocol.

**Parameters**

<i>t</i>	The tag
----------	---------

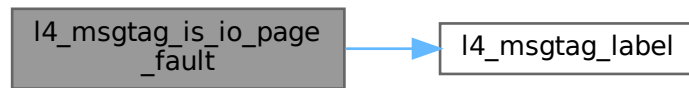
**Returns**

Boolean value

Definition at line 477 of file [types.h](#).

References [l4\\_msgtag\\_label\(\)](#), and [L4\\_PROTO\\_IO\\_PAGE\\_FAULT](#).

Here is the call graph for this function:



#### 13.1.14.7.4.6 `l4_msgtag_is_page_fault()`

```
unsigned l4_msgtag_is_page_fault (
    l4_msgtag_t t ) [inline]
```

Test for page-fault protocol.

##### Parameters

<code>t</code>	The tag
----------------	---------

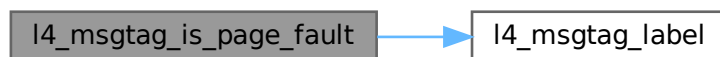
##### Returns

Boolean value

Definition at line 462 of file [types.h](#).

References [l4\\_msgtag\\_label\(\)](#), and [L4\\_PROTO\\_PAGE\\_FAULT](#).

Here is the call graph for this function:



#### 13.1.14.7.4.7 `l4_msgtag_is_preemption()`

```
unsigned l4_msgtag_is_preemption (
    l4_msgtag_t t ) [inline]
```

Test for preemption protocol.



**Parameters**

<i>t</i>	The tag
----------	---------

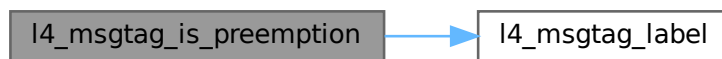
**Returns**

Boolean value

Definition at line 465 of file [types.h](#).

References [l4\\_msgtag\\_label\(\)](#), and [L4\\_PROTO\\_PREEMPTION](#).

Here is the call graph for this function:

**13.1.14.7.4.8 l4\_msgtag\_is\_sigma0()**

```
unsigned l4_msgtag_is_sigma0 (  
    l4_msgtag_t t ) [inline]
```

Test for sigma0 protocol.

**Parameters**

<i>t</i>	The tag
----------	---------

**Returns**

Boolean value

Definition at line 474 of file [types.h](#).

References [l4\\_msgtag\\_label\(\)](#), and [L4\\_PROTO\\_SIGMA0](#).

Here is the call graph for this function:



#### 13.1.14.7.4.9 l4\_msgtag\_is\_sys\_exception()

```
unsigned l4_msgtag_is_sys_exception (
    l4_msgtag_t t ) [inline]
```

Test for system-exception protocol.

##### Parameters

<i>t</i>	The tag
----------	---------

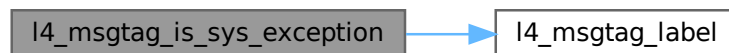
##### Returns

Boolean value

Definition at line 468 of file [types.h](#).

References [l4\\_msgtag\\_label\(\)](#), and [L4\\_PROTO\\_SYS\\_EXCEPTION](#).

Here is the call graph for this function:



#### 13.1.14.7.4.10 l4\_msgtag\_items()

```
unsigned l4_msgtag_items (
    l4_msgtag_t t ) [inline]
```

Get the number of typed items.

##### Parameters

<i>t</i>	The tag
----------	---------

##### Returns

Number of items.

Definition at line 448 of file [types.h](#).

Referenced by [l4util\\_ioport\\_map\(\)](#).

Here is the caller graph for this function:



#### 13.1.14.7.4.11 l4\_msgtag\_label()

```
long l4_msgtag_label (  
    l4_msgtag_t t )    [inline]
```

Get the protocol of tag.

##### Parameters

<i>t</i>	The tag
----------	---------

##### Returns

Label

##### Examples

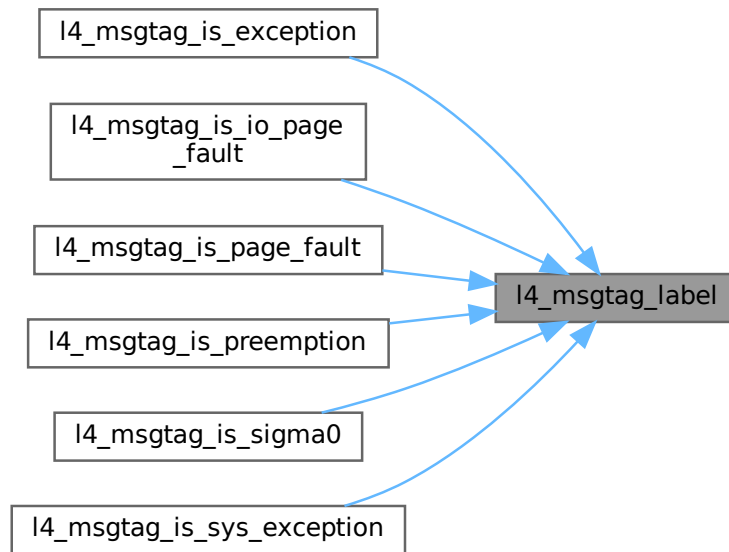
[examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line [440](#) of file [types.h](#).

References [l4\\_msgtag\\_t::raw](#).

Referenced by [l4\\_msgtag\\_is\\_exception\(\)](#), [l4\\_msgtag\\_is\\_io\\_page\\_fault\(\)](#), [l4\\_msgtag\\_is\\_page\\_fault\(\)](#), [l4\\_msgtag\\_is\\_preemption\(\)](#), [l4\\_msgtag\\_is\\_sigma0\(\)](#), and [l4\\_msgtag\\_is\\_sys\\_exception\(\)](#).

Here is the caller graph for this function:



#### 13.1.14.7.4.12 l4\_msgtag\_words()

```
unsigned l4_msgtag_words (
    l4_msgtag_t t ) [inline]
```

Get the number of untyped words.

##### Parameters

<i>t</i>	The tag
----------	---------

##### Returns

Number of words

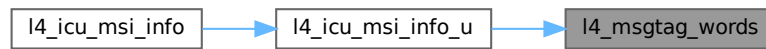
##### Examples

[examples/sys/utcb-ipc/main.c](#).

Definition at line 444 of file [types.h](#).

Referenced by [l4\\_icu\\_msi\\_info\\_u\(\)](#).

Here is the caller graph for this function:



#### 13.1.14.8 Realtime API

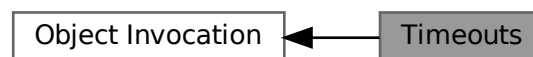
Collaboration diagram for Realtime API:



#### 13.1.14.9 Timeouts

All kinds of timeouts and time related functions.

Collaboration diagram for Timeouts:



#### Data Structures

- struct [l4\\_timeout\\_s](#)  
*Basic timeout specification.*
- union [l4\\_timeout\\_t](#)  
*Timeout pair.*

## Macros

- `#define L4_IPC_TIMEOUT_0 ((l4_timeout_s){0x0400})`  
*Timeout constants.*
- `#define L4_IPC_TIMEOUT_NEVER ((l4_timeout_s){0})`  
*never timeout*
- `#define L4_IPC_NEVER_INITIALIZER {0}`  
*never timeout, initializer*
- `#define L4_IPC_NEVER ((l4_timeout_t){0})`  
*never timeout*
- `#define L4_IPC_RECV_TIMEOUT_0 ((l4_timeout_t){0x00000400})`  
*0 receive timeout*
- `#define L4_IPC_SEND_TIMEOUT_0 ((l4_timeout_t){0x04000000})`  
*0 send timeout*
- `#define L4_IPC_BOTH_TIMEOUT_0 ((l4_timeout_t){0x04000400})`  
*0 receive and send timeout*
- `#define L4_TIMEOUT_US_NEVER (~0ULL)`  
*The waiting period in microseconds which is interpreted as "never" by l4\_timeout\_from\_us().*
- `#define L4_TIMEOUT_US_MAX ((1ULL << 41) - 1)`  
*The longest waiting period in microseconds accepted by l4\_timeout\_from\_us().*

## Typedefs

- `typedef struct l4_timeout_s l4_timeout_s`  
*Basic timeout specification.*
- `typedef union l4_timeout_t l4_timeout_t`  
*Timeout pair.*

## Functions

- `L4_CONSTEXPR l4_timeout_s l4_timeout_rel (unsigned man, unsigned exp) L4_NOTHROW`  
*Get relative timeout consisting of mantissa and exponent.*
- `L4_CONSTEXPR l4_timeout_t l4_ipc_timeout (unsigned snd_man, unsigned snd_exp, unsigned rcv_man, unsigned rcv_exp) L4_NOTHROW`  
*Convert explicit timeout values to l4\_timeout\_t type.*
- `L4_CONSTEXPR l4_timeout_t l4_timeout (l4_timeout_s snd, l4_timeout_s rcv) L4_NOTHROW`  
*Combine send and receive timeout in a timeout.*
- `L4_CONSTEXPR void l4_snd_timeout (l4_timeout_s snd, l4_timeout_t *to) L4_NOTHROW`  
*Set send timeout in given to timeout.*
- `L4_CONSTEXPR void l4_rcv_timeout (l4_timeout_s rcv, l4_timeout_t *to) L4_NOTHROW`  
*Set receive timeout in given to timeout.*
- `L4_CONSTEXPR l4_kernel_clock_t l4_timeout_rel_get (l4_timeout_s to) L4_NOTHROW`  
*Get clock value of out timeout.*
- `L4_CONSTEXPR unsigned l4_timeout_is_absolute (l4_timeout_s to) L4_NOTHROW`  
*Return whether the given timeout is absolute or not.*
- `L4_CONSTEXPR l4_kernel_clock_t l4_timeout_get (l4_kernel_clock_t cur, l4_timeout_s to) L4_NOTHROW`  
*Get clock value for a clock + a timeout.*
- `l4_timeout_s l4_timeout_abs (l4_kernel_clock_t pint, int br) L4_NOTHROW`  
*Set an absolute timeout.*
- `unsigned l4_utcb_mr64_idx (unsigned idx) L4_NOTHROW`  
*Get index into 64bit message registers alias from native-sized index.*

### 13.1.14.9.1 Detailed Description

All kinds of timeouts and time related functions.

### 13.1.14.9.2 Macro Definition Documentation

#### 13.1.14.9.2.1 L4\_IPC\_TIMEOUT\_0

```
#define L4_IPC_TIMEOUT_0 ((l4_timeout_s){0x0400})
```

Timeout constants.

0 timeout

Definition at line 81 of file `__timeout.h`.

#### 13.1.14.9.2.2 L4\_TIMEOUT\_US\_MAX

```
#define L4_TIMEOUT_US_MAX ((1ULL << 41) - 1)
```

The longest waiting period in microseconds accepted by `l4_timeout_from_us()`.

See `l4_timeout_from_us()` for an explanation.

Definition at line 99 of file `__timeout.h`.

### 13.1.14.9.3 Typedef Documentation

#### 13.1.14.9.3.1 l4\_timeout\_s

```
typedef struct l4_timeout_s l4_timeout_s
```

Basic timeout specification.

Basically a floating point number with 10 bits mantissa and 5 bits exponent ( $t = m \cdot 2^e$ ).

If bit 15 == 1 the timeout is absolute and the lower 6 bits encode the index of the UTCB buffer register(s) holding the absolute 64-bit timeout value. On 32-bit systems, two consecutive UTCB buffer registers are used.

#### 13.1.14.9.3.2 l4\_timeout\_t

```
typedef union l4_timeout_t l4_timeout_t
```

Timeout pair.

For IPC there are usually a send and a receive timeout. So this structure contains a pair of timeouts.

### 13.1.14.9.4 Function Documentation

#### 13.1.14.9.4.1 l4\_ipc\_timeout()

```
L4_CONSTEXPR l4_timeout_t l4_ipc_timeout (
    unsigned snd_man,
    unsigned snd_exp,
    unsigned rcv_man,
    unsigned rcv_exp ) [inline]
```

Convert explicit timeout values to `l4_timeout_t` type.

## Parameters

<i>snd_man</i>	Mantissa of send timeout.
<i>snd_exp</i>	Exponent of send timeout.
<i>rcv_man</i>	Mantissa of receive timeout.
<i>rcv_exp</i>	Exponent of receive timeout.

Definition at line 211 of file [\\_\\_timeout.h](#).

References [l4\\_timeout\(\)](#).

Here is the call graph for this function:



#### 13.1.14.9.4.2 l4\_rcv\_timeout()

```

L4_CONSTEXPR void l4_rcv_timeout (
    l4_timeout_s rcv,
    l4_timeout_t * to ) [inline]
  
```

Set receive timeout in given to timeout.

## Parameters

	<i>rcv</i>	Receive timeout
out	<i>to</i>	<a href="#">L4</a> timeout

Definition at line 235 of file [\\_\\_timeout.h](#).

#### 13.1.14.9.4.3 l4\_snd\_timeout()

```

L4_CONSTEXPR void l4_snd_timeout (
    l4_timeout_s snd,
    l4_timeout_t * to ) [inline]
  
```

Set send timeout in given to timeout.

## Parameters

	<i>snd</i>	Send timeout
out	<i>to</i>	<a href="#">L4</a> timeout



Definition at line 228 of file [\\_\\_timeout.h](#).

References [l4\\_timeout\\_t::p](#), and [l4\\_timeout\\_t::snd](#).

#### 13.1.14.9.4.4 l4\_timeout()

```
L4_CONSTEXPR l4_timeout_t l4_timeout (
    l4_timeout_s snd,
    l4_timeout_s rcv ) [inline]
```

Combine send and receive timeout in a timeout.

##### Parameters

<i>snd</i>	Send timeout
<i>rcv</i>	Receive timeout

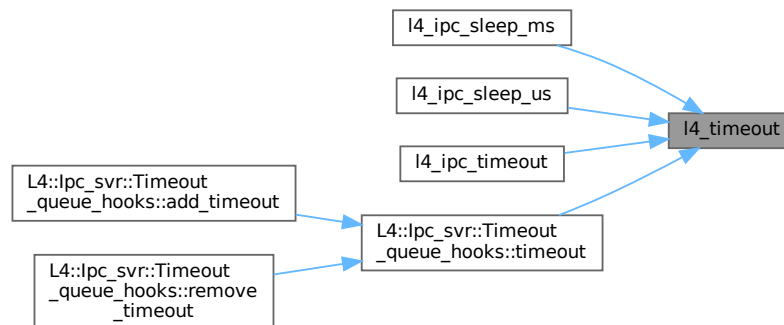
##### Returns

[L4](#) timeout

Definition at line 221 of file [\\_\\_timeout.h](#).

Referenced by [l4\\_ipc\\_sleep\\_ms\(\)](#), [l4\\_ipc\\_sleep\\_us\(\)](#), [l4\\_ipc\\_timeout\(\)](#), and [L4::lpc\\_svr::Timeout\\_queue\\_hooks< HOOKS, BR\\_MAN](#)

Here is the caller graph for this function:



#### 13.1.14.9.4.5 l4\_timeout\_abs()

```
l4_timeout_s l4_timeout_abs (
    l4_kernel_clock_t pint,
    int br ) [inline]
```

Set an absolute timeout.

## Parameters

<i>pint</i>	Point in time in clocks
<i>br</i>	The buffer register the timeout shall be placed in. (

## Note

On 32bit architectures the timeout needs two consecutive buffers.)

The absolute timeout value will be placed into the buffer register *br* of the current thread.

## Returns

timeout value

Definition at line 383 of file [utcb.h](#).

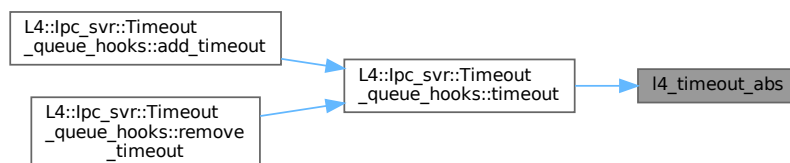
References [l4\\_utcb\(\)](#).

Referenced by [L4::lpc\\_svr::Timeout\\_queue\\_hooks<HOOKS, BR\\_MAN>::timeout\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.14.9.4.6 l4\_timeout\_get()

```

L4_CONSTEXPR l4_kernel_clock_t l4_timeout_get (
    l4_kernel_clock_t cur,
    l4_timeout_s to ) [inline]
  
```

Get clock value for a clock + a timeout.

## Parameters

<i>cur</i>	Clock value
<i>to</i>	<a href="#">L4</a> timeout

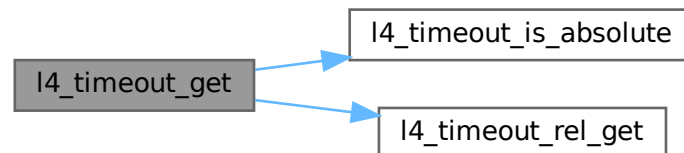
## Returns

Clock sum

Definition at line [265](#) of file [\\_\\_timeout.h](#).

References [l4\\_timeout\\_is\\_absolute\(\)](#), and [l4\\_timeout\\_rel\\_get\(\)](#).

Here is the call graph for this function:



#### 13.1.14.9.4.7 l4\_timeout\_is\_absolute()

```
L4\_CONSTEXPR unsigned l4_timeout_is_absolute (  
    l4\_timeout\_s to ) [inline]
```

Return whether the given timeout is absolute or not.

## Parameters

<i>to</i>	<a href="#">L4</a> timeout
-----------	----------------------------

## Returns

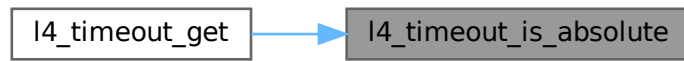
!= 0 if absolute, 0 if relative

Definition at line [258](#) of file [\\_\\_timeout.h](#).

References [l4\\_timeout\\_s::t](#).

Referenced by [l4\\_timeout\\_get\(\)](#).

Here is the caller graph for this function:



#### 13.1.14.9.4.8 l4\_timeout\_rel()

```
L4_CONSTEXPR l4_timeout_s l4_timeout_rel (  
    unsigned man,  
    unsigned exp ) [inline]
```

Get relative timeout consisting of mantissa and exponent.

##### Parameters

<i>man</i>	Mantissa of timeout
<i>exp</i>	Exponent of timeout

##### Returns

timeout value

Definition at line 242 of file [\\_\\_timeout.h](#).

#### 13.1.14.9.4.9 l4\_timeout\_rel\_get()

```
L4_CONSTEXPR l4_kernel_clock_t l4_timeout_rel_get (  
    l4_timeout_s to ) [inline]
```

Get clock value of out timeout.

##### Parameters

<i>to</i>	<a href="#">L4</a> timeout
-----------	----------------------------

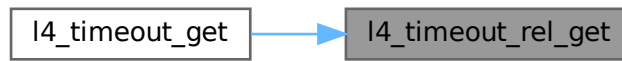
##### Returns

Clock value

Definition at line 249 of file [\\_\\_timeout.h](#).

Referenced by [l4\\_timeout\\_get\(\)](#).

Here is the caller graph for this function:



#### 13.1.14.9.4.10 l4\_utcb\_mr64\_idx()

```
unsigned l4_utcb_mr64_idx (  
    unsigned idx ) [inline]
```

Get index into 64bit message registers alias from native-sized index.

##### Parameters

<i>idx</i>	Index to native-sized message register
------------	--

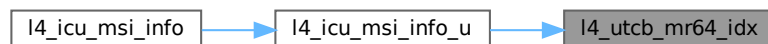
##### Returns

Index to 64bit message register alias

Definition at line [386](#) of file [utcb.h](#).

Referenced by [l4\\_icu\\_msi\\_info\\_u\(\)](#).

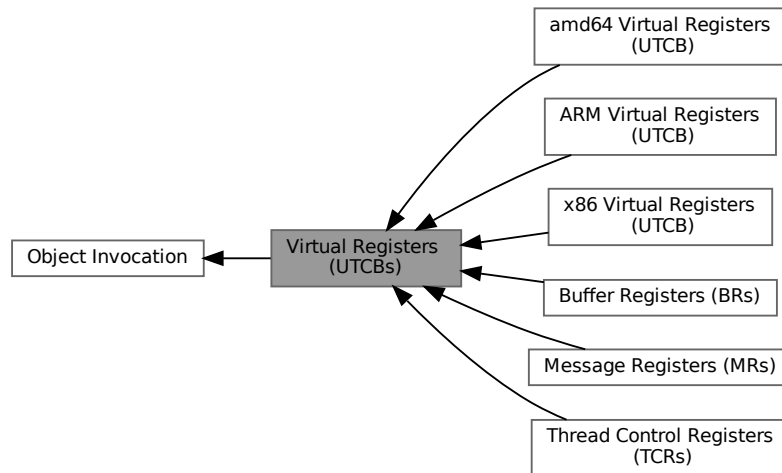
Here is the caller graph for this function:



#### 13.1.14.10 Virtual Registers (UTCBS)

[L4](#) Virtual Registers (UTCBS).

Collaboration diagram for Virtual Registers (UTCBs):



## Modules

- [ARM Virtual Registers \(UTCB\)](#)
- [Buffer Registers \(BRs\)](#)
- [Message Registers \(MRs\)](#)
- [Thread Control Registers \(TCRs\)](#)
- [amd64 Virtual Registers \(UTCB\)](#)
- [x86 Virtual Registers \(UTCB\)](#)

## Files

- file [utcb.h](#)  
*UTCB definitions for ARM.*
- file [utcb.h](#)  
*UTCB definitions for amd64.*
- file [utcb.h](#)  
*UTCB definitions for X86.*

## Typedefs

- typedef struct [l4\\_utcb\\_t](#) [l4\\_utcb\\_t](#)  
*Opaque type for the UTCB.*

## Functions

- [l4\\_utcb\\_t](#) \* [l4\\_utcb](#) (void) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
*Get the UTCB address.*
- [l4\\_msg\\_regs\\_t](#) \* [l4\\_utcb\\_mr](#) (void) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
*Get the message-register block of a UTCB.*
- [l4\\_buf\\_regs\\_t](#) \* [l4\\_utcb\\_br](#) (void) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
*Get the buffer-register block of a UTCB.*
- [l4\\_thread\\_regs\\_t](#) \* [l4\\_utcb\\_tcr](#) (void) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
*Get the thread-control-register block of a UTCB.*

### 13.1.14.10.1 Detailed Description

[L4](#) Virtual Registers (UTCB).

#### Include File

```
#include <l4/sys/utcb.h>
```

The virtual registers are part of the micro-kernel API and are located in the user-level thread control block (UTCB). The UTCB is a data structure defined by the micro kernel and located on kernel-provided memory. Each [L4](#) thread gets a unique UTCB assigned when it is bound to a task (see [Thread Control](#) , [l4\\_thread\\_control\\_bind\(\)](#) for more information).

The UTCB is arranged in three blocks of virtual registers.

- [Thread Control Registers \(TCRs\)](#)
- [Message Registers \(MRs\)](#)
- [Buffer Registers \(BRs\)](#)

To access the contents of the virtual registers the [l4\\_utcb\\_mr\(\)](#), [l4\\_utcb\\_tcr\(\)](#), and [l4\\_utcb\\_br\(\)](#) functions must be used.

### 13.1.14.10.2 Typedef Documentation

#### 13.1.14.10.2.1 l4\_utcb\_t

```
typedef struct l4_utcb_t l4_utcb_t
```

Opaque type for the UTCB.

To access the contents of the virtual registers the [l4\\_utcb\\_mr\(\)](#), [l4\\_utcb\\_tcr\(\)](#), and [l4\\_utcb\\_br\(\)](#) functions must be used.

Definition at line [67](#) of file [utcb.h](#).

### 13.1.14.10.3 Function Documentation

#### 13.1.14.10.3.1 l4\_utcb\_br()

```
l4_buf_regs_t * l4_utcb_br (  
    void ) [inline]
```

Get the buffer-register block of a UTCB.

**Returns**

A pointer to the buffer-register block of `u`.

Definition at line 355 of file `utcb.h`.

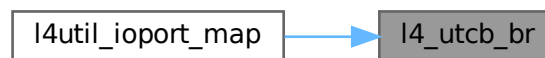
References `l4_utcb()`.

Referenced by `l4util_ioport_map()`.

Here is the call graph for this function:



Here is the caller graph for this function:

**13.1.14.10.3.2 l4\_utcb\_mr()**

```
l4_msg_regs_t * l4_utcb_mr (  
    void ) [inline]
```

Get the message-register block of a UTCB.

**Returns**

A pointer to the message-register block of `u`.

**Examples**

`examples/sys/aliens/main.c`, `examples/sys/ipc/ipc_example.c`, `examples/sys/singlestep/main.c`, and `examples/sys/utcb-ipc/main.c`.



Definition at line 352 of file [utcb.h](#).

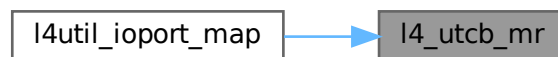
References [l4\\_utcb\(\)](#).

Referenced by [l4util\\_ioport\\_map\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.1.14.10.3.3 l4\_utcb\_tcr()

```
l4_thread_regs_t * l4_utcb_tcr (  
    void ) [inline]
```

Get the thread-control-register block of a UTCB.

##### Returns

A pointer to the thread-control-register block of `u`.

Definition at line 358 of file [utcb.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 13.1.14.10.4 ARM Virtual Registers (UTCB)

Collaboration diagram for ARM Virtual Registers (UTCB):



#### Data Structures

- struct [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

#### Typedefs

- typedef struct [l4\\_exc\\_regs\\_t](#) [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

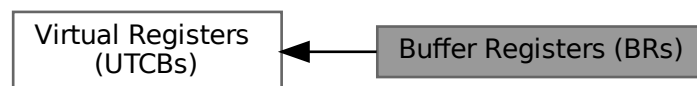
#### Enumerations

- enum [L4\\_utcb\\_consts\\_arm](#)  
*UTCB constants for ARM.*

#### 13.1.14.10.4.1 Detailed Description

#### 13.1.14.10.5 Buffer Registers (BRs)

Collaboration diagram for Buffer Registers (BRs):



#### Data Structures

- struct [l4\\_buf\\_regs\\_t](#)  
*Encapsulation of the buffer-registers block in the UTCB.*

## Typedefs

- typedef struct [l4\\_buf\\_regs\\_t](#) [l4\\_buf\\_regs\\_t](#)  
*Encapsulation of the buffer-registers block in the UTCB.*

## Enumerations

- enum [l4\\_buffer\\_desc\\_consts\\_t](#) { [L4\\_BDR\\_MEM\\_SHIFT](#) = 0 , [L4\\_BDR\\_IO\\_SHIFT](#) = 5 , [L4\\_BDR\\_OBJ\\_SHIFT](#) = 10 , [L4\\_BDR\\_OFFSET\\_MASK](#) = (1UL << 20) - 1 }  
*Constants for buffer descriptors.*

## Functions

- void [l4\\_utcb\\_inherit\\_fpu](#) (int switch\_on) [L4\\_NOTHROW](#)  
*Enable or disable inheritance of FPU state to receiver.*

### 13.1.14.10.5.1 Detailed Description

### 13.1.14.10.5.2 Enumeration Type Documentation

#### [l4\\_buffer\\_desc\\_consts\\_t](#)

enum [l4\\_buffer\\_desc\\_consts\\_t](#)

Constants for buffer descriptors.

#### Enumerator

<a href="#">L4_BDR_MEM_SHIFT</a>	Bit offset for the memory-buffer index.
<a href="#">L4_BDR_IO_SHIFT</a>	Bit offset for the IO-buffer index.
<a href="#">L4_BDR_OBJ_SHIFT</a>	Bit offset for the capability-buffer index.

Definition at line [292](#) of file [consts.h](#).

### 13.1.14.10.6 Message Registers (MRs)

Collaboration diagram for Message Registers (MRs):



## Modules

- [Exception registers](#)

*Overly definition of the MRs for exception messages.*

## Data Structures

- union [l4\\_msg\\_regs\\_t](#)

*Encapsulation of the message-register block in the UTCB.*

## Typedefs

- typedef union [l4\\_msg\\_regs\\_t](#) [l4\\_msg\\_regs\\_t](#)

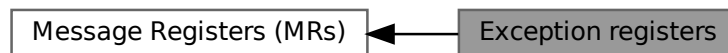
*Encapsulation of the message-register block in the UTCB.*

### 13.1.14.10.6.1 Detailed Description

### 13.1.14.10.6.2 Exception registers

Overly definition of the MRs for exception messages.

Collaboration diagram for Exception registers:



## Functions

- [l4\\_exc\\_regs\\_t](#) \* [l4\\_utcb\\_exc](#) (void) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
*Get the message-register block of a UTCB (for an exception IPC).*
- [l4\\_umword\\_t](#) [l4\\_utcb\\_exc\\_pc](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
*Access function to get the program counter of the exception state.*
- void [l4\\_utcb\\_exc\\_pc\\_set](#) ([l4\\_exc\\_regs\\_t](#) \*u, [l4\\_addr\\_t](#) pc) [L4\\_NOTHROW](#)  
*Set the program counter register in the exception state.*
- unsigned long [l4\\_utcb\\_exc\\_typeval](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
*Get the value out of an exception UTCB that describes the type of exception.*
- int [l4\\_utcb\\_exc\\_is\\_pf](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
*Check whether an exception IPC is a page fault.*
- [l4\\_addr\\_t](#) [l4\\_utcb\\_exc\\_pfa](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
*Function to get the L4 style page fault address out of an exception.*
- int [l4\\_utcb\\_exc\\_is\\_ex\\_regs\\_exception](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
*Check whether an exception IPC was triggered via [l4\\_thread\\_ex\\_regs\(\)](#).*

## Detailed Description

Overly definition of the MRs for exception messages.

## Function Documentation

### **l4\_utcb\_exc()**

```
l4_exc_regs_t * l4_utcb_exc (  
    void ) [inline]
```

Get the message-register block of a UTCB (for an exception IPC).

#### Returns

A pointer to the exception message in `u`.

#### Examples

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line 361 of file [utcb.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### **l4\_utcb\_exc\_is\_ex\_regs\_exception()**

```
int l4_utcb_exc_is_ex_regs_exception (  
    l4_exc_regs_t const * u ) [inline]
```

Check whether an exception IPC was triggered via [l4\\_thread\\_ex\\_regs\(\)](#).

#### Return values

0	Exception was not triggered through <code>ex_regs</code> .
<code>!=0</code>	Exception was triggered through <code>ex_regs</code> .

This function checks if the exception was emitted by using the `L4_THREAD_EX_REGS_TRIGGER_EXCEPTION` flag in an `l4_thread_ex_regs()` call.

Definition at line 121 of file `utcb.h`.

References `l4_utcb_exc_typeval()`.

Here is the call graph for this function:



### **l4\_utcb\_exc\_is\_pf()**

```
int l4_utcb_exc_is_pf (
    l4_exc_regs_t const * u ) [inline]
```

Check whether an exception IPC is a page fault.

#### **Returns**

0 if not, != 0 if yes

Function to check whether an exception IPC is a page fault, also applies to I/O pagefaults.

Definition at line 111 of file `utcb.h`.

### **l4\_utcb\_exc\_pc()**

```
l4_umword_t l4_utcb_exc_pc (
    l4_exc_regs_t const * u ) [inline]
```

Access function to get the program counter of the exception state.

#### **Parameters**

<i>u</i>	UTCB
----------	------

#### **Returns**

The program counter register out of the exception state.

#### **Examples**

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line 96 of file [utcb.h](#).

### **l4\_utcb\_exc\_pc\_set()**

```
void l4_utcb_exc_pc_set (
    l4_exc_regs_t * u,
    l4_addr_t pc ) [inline]
```

Set the program counter register in the exception state.

#### **Parameters**

<i>u</i>	UTCB
<i>pc</i>	The program counter to set.

Definition at line 101 of file [utcb.h](#).

#### **13.1.14.10.7 Thread Control Registers (TCRs)**

Collaboration diagram for Thread Control Registers (TCRs):



#### **Data Structures**

- struct [l4\\_thread\\_regs\\_t](#)  
*Encapsulation of the thread-control-register block of the UTCB.*

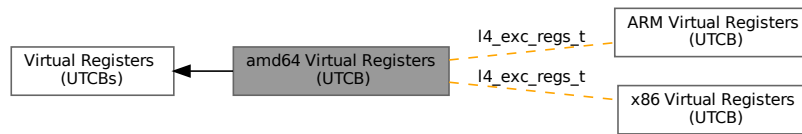
#### **Typedefs**

- typedef struct [l4\\_thread\\_regs\\_t](#) **l4\_thread\_regs\_t**  
*Encapsulation of the thread-control-register block of the UTCB.*

### 13.1.14.10.7.1 Detailed Description

### 13.1.14.10.8 amd64 Virtual Registers (UTCB)

Collaboration diagram for amd64 Virtual Registers (UTCB):



### Data Structures

- struct [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

### Typedefs

- typedef struct [l4\\_exc\\_regs\\_t](#) [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

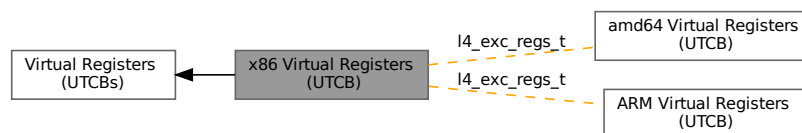
### Enumerations

- enum [L4\\_utcb\\_consts\\_amd64](#)  
*UTCB constants for AMD64.*

### 13.1.14.10.8.1 Detailed Description

### 13.1.14.10.9 x86 Virtual Registers (UTCB)

Collaboration diagram for x86 Virtual Registers (UTCB):



### Data Structures

- struct [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*



## Typedefs

- typedef struct [l4\\_exc\\_regs\\_t](#) [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

## Enumerations

- enum [L4\\_utcb\\_consts\\_x86](#) {  
[L4\\_UTCB\\_EXCEPTION\\_REGS\\_SIZE](#) = 19 , [L4\\_UTCB\\_GENERIC\\_DATA\\_SIZE](#) = 63 , [L4\\_UTCB\\_GENERIC\\_BUFFERS\\_SIZE](#) = 58 , [L4\\_UTCB\\_MSG\\_REGS\\_OFFSET](#) = 0 ,  
[L4\\_UTCB\\_BUF\\_REGS\\_OFFSET](#) = 64 \* sizeof(l4\_umword\_t) , [L4\\_UTCB\\_THREAD\\_REGS\\_OFFSET](#) = 123 \* sizeof(l4\_umword\_t) , [L4\\_UTCB\\_INHERIT\\_FPU](#) = 1UL << 24 , [L4\\_UTCB\\_OFFSET](#) = 512 }  
*UTCB constants for x86.*

### 13.1.14.10.9.1 Detailed Description

### 13.1.14.10.9.2 Enumeration Type Documentation

#### [L4\\_utcb\\_consts\\_x86](#)

enum [L4\\_utcb\\_consts\\_x86](#)

UTCB constants for x86.

#### Enumerator

<a href="#">L4_UTCB_EXCEPTION_REGS_SIZE</a>	Number if message registers used for exception IPC.
<a href="#">L4_UTCB_GENERIC_DATA_SIZE</a>	Total number of message register (MRs) available.
<a href="#">L4_UTCB_GENERIC_BUFFERS_SIZE</a>	Total number of buffer registers (BRs) available.
<a href="#">L4_UTCB_MSG_REGS_OFFSET</a>	Offset of MR[0] relative to the UTCB pointer.
<a href="#">L4_UTCB_BUF_REGS_OFFSET</a>	Offset of BR[0] relative to the UTCB pointer.
<a href="#">L4_UTCB_THREAD_REGS_OFFSET</a>	Offset of TCR[0] relative to the UTCB pointer.
<a href="#">L4_UTCB_INHERIT_FPU</a>	BDR flag to accept reception of FPU state.
<a href="#">L4_UTCB_OFFSET</a>	Offset of two consecutive UTCBs.

Definition at line [41](#) of file [utcb.h](#).

## 13.2 EDID parsing functionality

## Enumerations

- enum [Libedid\\_consts](#) { [Libedid\\_block\\_size](#) = 128 }  
*EDID constants.*

## Functions

- int [libedid\\_check\\_header](#) (const unsigned char \*edid)  
*Check for valid EDID header.*
- int [libedid\\_checksum](#) (const unsigned char \*edid)  
*Calculates the EDID checksum.*
- unsigned [libedid\\_version](#) (const unsigned char \*edid)  
*Returns the EDID version number.*
- unsigned [libedid\\_revision](#) (const unsigned char \*edid)  
*Returns the EDID revision number.*
- void [libedid\\_pnp\\_id](#) (const unsigned char \*edid, unsigned char \*id)  
*Extracts the display's PnP ID.*
- void [libedid\\_prefered\\_resolution](#) (const unsigned char \*edid, unsigned \*w, unsigned \*h)  
*Extract the display's prefered mode.*
- unsigned [libedid\\_num\\_ext\\_blocks](#) (const unsigned char \*edid)  
*Get the number of EDID extension blocks.*
- unsigned [libedid\\_dump\\_standard\\_timings](#) (const unsigned char \*edid)  
*Dump the standard timings to stdout.*
- void [libedid\\_dump](#) (const unsigned char \*edid)  
*Dump raw EDID data to stdout.*

### 13.2.1 Detailed Description

### 13.2.2 Enumeration Type Documentation

#### 13.2.2.1 Libedid\_consts

enum [Libedid\\_consts](#)

EDID constants.

Enumerator

<a href="#">Libedid_block_size</a>	Size of one EDID block in bytes.
------------------------------------	----------------------------------

Definition at line 23 of file [edid.h](#).

### 13.2.3 Function Documentation

#### 13.2.3.1 libedid\_check\_header()

```
int libedid_check_header (
    const unsigned char * edid )
```

Check for valid EDID header.

**Parameters**

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

**Returns**

0 if the header is correct, -EINVAL otherwise

**13.2.3.2 libedid\_checksum()**

```
int libedid_checksum (
    const unsigned char * edid )
```

Calculates the EDID checksum.

**Parameters**

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

**Returns**

0 if checksum is correct, -EINVAL otherwise

**13.2.3.3 libedid\_dump()**

```
void libedid_dump (
    const unsigned char * edid )
```

Dump raw EDID data to stdout.

**Parameters**

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

**13.2.3.4 libedid\_dump\_standard\_timings()**

```
unsigned libedid_dump_standard_timings (
    const unsigned char * edid )
```

Dump the standard timings to stdout.

**Parameters**

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

**Returns**

Number of standard timings stored in EDID

**13.2.3.5 libedid\_num\_ext\_blocks()**

```
unsigned libedid_num_ext_blocks (
    const unsigned char * edid )
```

Get the number of EDID extension blocks.

**Parameters**

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

**Returns**

Number of EDID extension blocks

**13.2.3.6 libedid\_pnp\_id()**

```
void libedid_pnp_id (
    const unsigned char * edid,
    unsigned char * id )
```

Extracts the display's PnP ID.

**Parameters**

	<i>edid</i>	Pointer to a 128byte EDID block
out	<i>id</i>	Return the PnP id. Must point to 4 bytes.

**13.2.3.7 libedid\_prefered\_resolution()**

```
void libedid_prefered_resolution (
    const unsigned char * edid,
    unsigned * w,
    unsigned * h )
```

Extract the display's preferred mode.

**Parameters**

	<i>edid</i>	Pointer to a 128byte EDID block
out	<i>w</i>	X resolution of preferred video mode in pixels.
out	<i>h</i>	Y resolution of preferred video mode in pixels.

**13.2.3.8 libedid\_revision()**

```
unsigned libedid_revision (
    const unsigned char * edid )
```

Returns the EDID revision number.

**Parameters**

<i>edid</i>	Pointer to a 128 EDID block
-------------	-----------------------------

**Returns**

Revision number

**13.2.3.9 libedid\_version()**

```
unsigned libedid_version (
    const unsigned char * edid )
```

Returns the EDID version number.

**Parameters**

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

**Returns**

Version number

**13.3 IO interface****Typedefs**

- typedef [l4vbus\\_resource\\_t](#) [l4io\\_resource\\_t](#)  
*Resource descriptor.*
- typedef [l4vbus\\_device\\_t](#) [l4io\\_device\\_t](#)  
*Device descriptor.*

**Enumerations**

- enum [l4io\\_iomem\\_flags\\_t](#) {  
[L4IO\\_MEM\\_NONCACHED](#) = 0 , [L4IO\\_MEM\\_CACHED](#) = 1 , [L4IO\\_MEM\\_USE\\_MTRR](#) = 2 , [L4IO\\_MEM\\_ATTR\\_MASK](#) = 0xf ,  
[L4IO\\_MEM\\_WRITE\\_COMBINED](#) = [L4IO\\_MEM\\_USE\\_MTRR](#) | [L4IO\\_MEM\\_CACHED](#) , [L4IO\\_MEM\\_USE\\_RESERVED\\_AREA](#)  
= 0x40 << 8 , [L4IO\\_MEM\\_EAGER\\_MAP](#) = 0x80 << 8 }  
*Flags for IO memory.*

- enum `l4io_device_types_t` {  
`L4IO_DEVICE_INVALID` = 0 , `L4IO_DEVICE_PCI` , `L4IO_DEVICE_USB` , `L4IO_DEVICE_OTHER` ,  
`L4IO_DEVICE_ANY` = ~0 }  
*Device types.*
- enum `l4io_resource_types_t` {  
`L4IO_RESOURCE_INVALID` = `L4VBUS_RESOURCE_INVALID` , `L4IO_RESOURCE_IRQ` = `L4VBUS_RESOURCE_IRQ` , `L4IO_RESOURCE_MEM` = `L4VBUS_RESOURCE_MEM` , `L4IO_RESOURCE_PORT` = `L4VBUS_RESOURCE_PORT` ,  
`L4IO_RESOURCE_ANY` = ~0 }  
*Resource types.*

## Functions

- long `l4io_request_iomem` (`l4_addr_t` phys, unsigned long size, int flags, `l4_addr_t` \*virt)  
*Request an IO memory region.*
- long `l4io_request_iomem_region` (`l4_addr_t` phys, `l4_addr_t` virt, unsigned long size, int flags)  
*Request an IO memory region and map it to a specified region.*
- long `l4io_release_iomem` (`l4_addr_t` virt, unsigned long size)  
*Release an IO memory region.*
- long `l4io_request_ioport` (unsigned portnum, unsigned len)  
*Request an IO port region.*
- long `l4io_release_ioport` (unsigned portnum, unsigned len)  
*Release an IO port region.*
- int `l4io_lookup_device` (const char \*devname, `l4io_device_handle_t` \*dev\_handle, `l4io_device_t` \*dev, `l4io_resource_handle_t` \*res\_handle)  
*Find a device by name.*
- int `l4io_lookup_resource` (`l4io_device_handle_t` devhandle, enum `l4io_resource_types_t` type, `l4io_resource_handle_t` \*reshandle, `l4io_resource_t` \*res)  
*Request a specific resource from a device description.*
- `l4_addr_t` `l4io_request_resource_iomem` (`l4io_device_handle_t` devhandle, `l4io_resource_handle_t` \*reshandle)  
*Request IO memory.*
- int `l4io_has_resource` (enum `l4io_resource_types_t` type, `l4vbus_paddr_t` start, `l4vbus_paddr_t` end)  
*Check if a resource is available.*

### 13.3.1 Detailed Description

### 13.3.2 Typedef Documentation

#### 13.3.2.1 `l4io_resource_t`

```
typedef l4vbus_resource_t l4io_resource_t
```

Resource descriptor.

For IRQ types, the end field is not used, i.e. only a single interrupt can be described with a `l4io_resource_t`

Definition at line 69 of file [types.h](#).

### 13.3.3 Enumeration Type Documentation

#### 13.3.3.1 `l4io_device_types_t`

```
enum l4io_device_types_t
```

Device types.

## Enumerator

L4IO_DEVICE_INVALID	Invalid type.
L4IO_DEVICE_PCI	PCI device.
L4IO_DEVICE_USB	USB device.
L4IO_DEVICE_OTHER	Any other device without unique IDs.
L4IO_DEVICE_ANY	any type

Definition at line 38 of file [types.h](#).

**13.3.3.2 l4io\_iomem\_flags\_t**

```
enum l4io_iomem_flags_t
```

Flags for IO memory.

## Enumerator

L4IO_MEM_NONCACHED	Non-cache memory.
L4IO_MEM_CACHED	Cache memory.
L4IO_MEM_USE_MTRR	Use MTRR.
L4IO_MEM_USE_RESERVED_AREA	Use reserved area for mapping I/O memory. Flag only valid for <a href="#">l4io_request_iomem_region()</a>
L4IO_MEM_EAGER_MAP	Eagerly map the I/O memory. Passthrough to the l4re-rm.

Definition at line 16 of file [types.h](#).

**13.3.3.3 l4io\_resource\_types\_t**

```
enum l4io_resource_types_t
```

Resource types.

## Enumerator

L4IO_RESOURCE_INVALID	Invalid type.
L4IO_RESOURCE_IRQ	Interrupt resource.
L4IO_RESOURCE_MEM	I/O memory resource.
L4IO_RESOURCE_PORT	I/O port resource (x86 only)
L4IO_RESOURCE_ANY	any type

Definition at line 50 of file [types.h](#).

**13.3.4 Function Documentation****13.3.4.1 l4io\_has\_resource()**

```
int l4io_has_resource (
```

```
enum l4io_resource_types_t type,
l4vbus_paddr_t start,
l4vbus_paddr_t end )
```

Check if a resource is available.

#### Parameters

<i>type</i>	Type of resource
<i>start</i>	Minimal value.
<i>end</i>	Maximum value.

### 13.3.4.2 l4io\_lookup\_device()

```
int l4io_lookup_device (
    const char * devname,
    l4io_device_handle_t * dev_handle,
    l4io_device_t * dev,
    l4io_resource_handle_t * res_handle )
```

Find a device by name.

#### Parameters

	<i>devname</i>	Name of device.
out	<i>dev_handle</i>	Device handle for found device, can be NULL.
out	<i>dev</i>	Device information, filled by the function, can be NULL.
out	<i>res_handle</i>	Resource handle, can be NULL.

#### Returns

0 on success, error code otherwise

### 13.3.4.3 l4io\_lookup\_resource()

```
int l4io_lookup_resource (
    l4io_device_handle_t devhandle,
    enum l4io_resource_types_t type,
    l4io_resource_handle_t * reshandle,
    l4io_resource_t * res )
```

Request a specific resource from a device description.

#### Parameters

	<i>devhandle</i>	Device handle.
	<i>type</i>	Type of resource to request (see <a href="#">l4io_resource_types_t</a> ).
in, out	<i>reshandle</i>	Resource handle, start with handle returned by device functions. The next resource handle is returned here.
out	<i>res</i>	Device descriptor.



**Returns**

0 on success, error code otherwise, esp. -L4\_ENOENT if no more resources found

**13.3.4.4 l4io\_release\_iomem()**

```
long l4io_release_iomem (
    l4_addr_t virt,
    unsigned long size )
```

Release an IO memory region.

**Parameters**

<i>virt</i>	Virtual address of region to free, see <a href="#">l4io_request_iomem</a>
<i>size</i>	Size of the region to release.

**Returns**

0 on success, <0 on error

**13.3.4.5 l4io\_release\_ioport()**

```
long l4io_release_ioport (
    unsigned portnum,
    unsigned len )
```

Release an IO port region.

**Parameters**

<i>portnum</i>	Start of port range to release
<i>len</i>	Length of range to request

**Returns**

0 on success, <0 on error

**Note**

X86 architecture only

**13.3.4.6 l4io\_request\_iomem()**

```
long l4io_request_iomem (
    l4_addr_t phys,
    unsigned long size,
    int flags,
    l4_addr_t * virt )
```

Request an IO memory region.

## Parameters

	<i>phys</i>	Physical address of the I/O memory region
	<i>size</i>	Size of the region in Bytes, granularity pages.
	<i>flags</i>	See <a href="#">l4io_iomem_flags_t</a>
<i>in, out</i>	<i>virt</i>	Virtual address where the IO memory region should be mapped to. If the caller passes '0' a region in the caller's address space is searched and the virtual address is returned.

## Return values

0	Success.
-L4_ENOENT	No area in the caller's address space could be found to map the IO memory region.
-L4_EPERM	Operation not allowed.
-L4_EINVAL	Invalid value.
-L4_EADDRNOTAVAIL	The requested virtual address is not available.
-L4_ENOMEM	The requested IO memory region could not be allocated.
<0	IPC errors.

## Note

This function uses [L4Re](#) functionality to reserve a part of the virtual address space of the caller.

## 13.3.4.7 l4io\_request\_iomem\_region()

```
long l4io_request_iomem_region (
    l4_addr_t phys,
    l4_addr_t virt,
    unsigned long size,
    int flags )
```

Request an IO memory region and map it to a specified region.

## Parameters

<i>phys</i>	Physical address of the I/O memory region
<i>virt</i>	Virtual address.
<i>size</i>	Size of the region in Bytes, granularity pages.
<i>flags</i>	See <a href="#">l4io_iomem_flags_t</a>

## Return values

0	Success.
-L4_ENOENT	No area could be found to map the IO memory region.
-L4_EPERM	Operation not allowed.
-L4_EINVAL	Invalid value.
-L4_EADDRNOTAVAIL	The requested virtual address is not available.
-L4_ENOMEM	The requested IO memory region could not be allocated.
<0	IPC errors.

**Note**

This function uses [L4Re](#) functionality to reserve a part of the virtual address space of the caller.

**13.3.4.8 l4io\_request\_ioport()**

```
long l4io_request_ioport (
    unsigned portnum,
    unsigned len )
```

Request an IO port region.

**Parameters**

<i>portnum</i>	Start of port range to request
<i>len</i>	Length of range to request

**Returns**

0 on success, <0 on error

**Note**

X86 architecture only

**13.3.4.9 l4io\_request\_resource\_iomem()**

```
l4_addr_t l4io_request_resource_iomem (
    l4io_device_handle_t devhandle,
    l4io_resource_handle_t * reshandle )
```

Request IO memory.

**Parameters**

	<i>devhandle</i>	Device handle.
<i>in, out</i>	<i>reshandle</i>	Resource handle from which IO memory should be requested. Upon successful completion 'reshandle' points to the device's next resource.

**Return values**

0	An error occurred. The value of 'reshandle' is undefined.
>0	The virtual address of the IO memory mapping.

## 13.4 IPC Helpers

### Functions

- void [L4::throw\\_ipc\\_exception](#) ([L4::Cap](#)< void > const &o, [l4\\_msgtag\\_t](#) const &err, [l4\\_utcb\\_t](#) \*utcb)  
*Throw an [L4](#) IPC error as exception.*
- void [L4::throw\\_ipc\\_exception](#) (void const \*o, [l4\\_msgtag\\_t](#) const &err, [l4\\_utcb\\_t](#) \*utcb)  
*Throw an [L4](#) IPC error as exception.*

### 13.4.1 Detailed Description

### 13.4.2 Function Documentation

#### 13.4.2.1 [throw\\_ipc\\_exception\(\)](#) [1/2]

```
void L4::throw_ipc_exception (
    L4::Cap< void > const & o,
    l4\_msgtag\_t const & err,
    l4\_utcb\_t * utcb ) [inline]
```

Throw an [L4](#) IPC error as exception.

#### Parameters

<i>o</i>	The client side object, for which the IPC was invoked.
<i>err</i>	The IPC result code (error code).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

Definition at line [45](#) of file [ipc\\_helper](#).

References [l4\\_msgtag\\_t::has\\_error\(\)](#).

Referenced by [L4::throw\\_ipc\\_exception\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 13.4.2.2 `throw_ipc_exception()` [2/2]

```

void L4::throw_ipc_exception (
    void const * o,
    l4_msgtag_t const & err,
    l4_utcb_t * utcb ) [inline]
  
```

Throw an [L4](#) IPC error as exception.

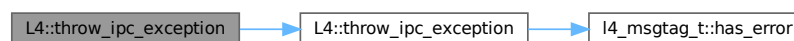
#### Parameters

<i>o</i>	The client side object, for which the IPC was invoked.
<i>err</i>	The IPC result code (error code).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

Definition at line [61](#) of file [ipc\\_helper](#).

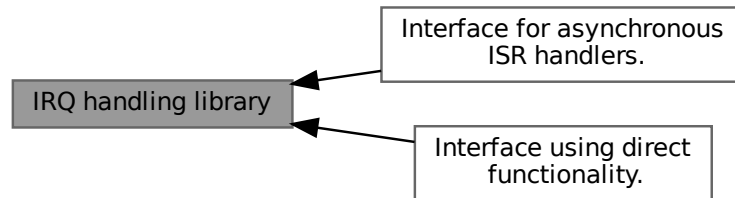
References [L4::throw\\_ipc\\_exception\(\)](#).

Here is the call graph for this function:



## 13.5 IRQ handling library

Collaboration diagram for IRQ handling library:



### Modules

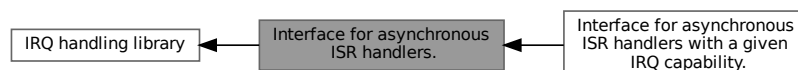
- [Interface for asynchronous ISR handlers.](#)  
*This interface has just two (main) functions.*
- [Interface using direct functionality.](#)

### 13.5.1 Detailed Description

### 13.5.2 Interface for asynchronous ISR handlers.

This interface has just two (main) functions.

Collaboration diagram for Interface for asynchronous ISR handlers.:



### Modules

- [Interface for asynchronous ISR handlers with a given IRQ capability.](#)  
*This group is just an enhanced version to [l4irq\\_request\(\)](#) which takes a capability object instead of a plain number.*

### Functions

- `l4irq_t * l4irq\_request (int irqnum, void(*isr_handler)(void *), void *isr_data, int irq_thread_prio, unsigned mode)`  
*Attach asynchronous ISR handler to IRQ.*
- `long l4irq\_release (l4irq_t *irq)`  
*Release asynchronous ISR handler and free resources.*

### 13.5.2.1 Detailed Description

This interface has just two (main) functions.

`l4irq_request` to install a handler for an interrupt and `l4irq_release` to uninstall the handler again and release all resources associated with it.

### 13.5.2.2 Function Documentation

#### 13.5.2.2.1 `l4irq_release()`

```
long l4irq_release (
    l4irq_t * irq )
```

Release asynchronous ISR handler and free resources.

##### Parameters

<i>irq</i>	IRQ data structure
------------	--------------------

##### Returns

0 success, != 0 failure

##### Examples

[examples/libs/libirq/async\\_isr.c](#).

#### 13.5.2.2.2 `l4irq_request()`

```
l4irq_t * l4irq_request (
    int irqnum,
    void(*) (void *) isr_handler,
    void * isr_data,
    int irq_thread_prio,
    unsigned mode )
```

Attach asynchronous ISR handler to IRQ.

##### Parameters

<i>irqnum</i>	IRQ number to request
<i>isr_handler</i>	Handler routine that is called when an interrupt triggers
<i>isr_data</i>	Pointer given as argument to <code>isr_handler</code>
<i>irq_thread_prio</i>	<a href="#">L4</a> thread priority of the ISR handler. Give -1 for same priority as creator.
<i>mode</i>	Interrupt type,

See also

[L4\\_irq\\_mode](#)

Returns

Pointer to `l4irq_t` structure, 0 on error

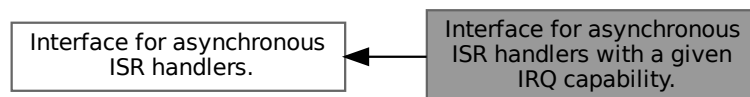
Examples

[examples/libs/libirq/async\\_isr.c](#).

### 13.5.2.3 Interface for asynchronous ISR handlers with a given IRQ capability.

This group is just an enhanced version to [l4irq\\_request\(\)](#) which takes a capability object instead of a plain number.

Collaboration diagram for Interface for asynchronous ISR handlers with a given IRQ capability.:



## Functions

- `l4irq_t * l4irq_request_cap (l4_cap_idx_t irqcap, void(*isr_handler)(void *), void *isr_data, int irq_thread_prio, unsigned mode)`

*Attach asynchronous ISR handler to IRQ.*

### 13.5.2.3.1 Detailed Description

This group is just an enhanced version to [l4irq\\_request\(\)](#) which takes a capability object instead of a plain number.

### 13.5.2.3.2 Function Documentation

#### 13.5.2.3.2.1 l4irq\_request\_cap()

```

l4irq_t * l4irq_request_cap (
    l4_cap_idx_t irqcap,
    void(*) (void *) isr_handler,
    void * isr_data,
    int irq_thread_prio,
    unsigned mode )
  
```

Attach asynchronous ISR handler to IRQ.



## Parameters

<i>irqcap</i>	IRQ capability
<i>isr_handler</i>	Handler routine that is called when an interrupt triggers
<i>isr_data</i>	Pointer given as argument to <i>isr_handler</i>
<i>irq_thread_prio</i>	<a href="#">L4</a> thread priority of the ISR handler. Give -1 for same priority as creator.
<i>mode</i>	Interrupt type,

## See also

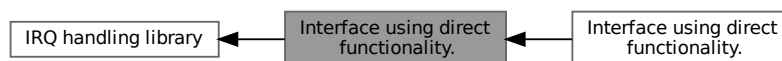
[L4\\_irq\\_mode](#)

## Returns

Pointer to `l4irq_t` structure, 0 on error

### 13.5.3 Interface using direct functionality.

Collaboration diagram for Interface using direct functionality.:



## Modules

- [Interface using direct functionality.](#)

## Functions

- `l4irq_t * l4irq\_attach (int irqnum)`  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq\_attach\_ft (int irqnum, unsigned mode)`  
*Attach/connect to IRQ using given type.*
- `l4irq_t * l4irq\_attach\_thread (int irqnum, l4\_cap\_idx\_t to_thread)`  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq\_attach\_thread\_ft (int irqnum, l4\_cap\_idx\_t to_thread, unsigned mode)`  
*Attach/connect to IRQ using given type.*
- `long l4irq\_wait (l4irq_t *irq)`  
*Wait for specified IRQ.*
- `long l4irq\_unmask\_and\_wait\_any (l4irq_t *unmask_irq, l4irq_t **ret_irq)`  
*Unmask a specific IRQ and wait for any attached IRQ.*
- `long l4irq\_wait\_any (l4irq_t **irq)`  
*Wait for any attached IRQ.*
- `long l4irq\_unmask (l4irq_t *irq)`  
*Unmask a specific IRQ.*
- `long l4irq\_detach (l4irq_t *irq)`  
*Detach from IRQ.*

### 13.5.3.1 Detailed Description

### 13.5.3.2 Function Documentation

#### 13.5.3.2.1 `l4irq_attach()`

```
l4irq_t * l4irq_attach (
    int irqnum )
```

Attach/connect to IRQ.

#### Parameters

<i>irqnum</i>	IRQ number to request
---------------	-----------------------

#### Returns

Pointer to `l4irq_t` structure, 0 on error

This `l4irq_attach` has to be called in the same thread as `l4irq_wait` and caller has to be a pthread thread.

#### Examples

[examples/libs/libirq/loop.c](#).

#### 13.5.3.2.2 `l4irq_attach_ft()`

```
l4irq_t * l4irq_attach_ft (
    int irqnum,
    unsigned mode )
```

Attach/connect to IRQ using given type.

#### Parameters

<i>irqnum</i>	IRQ number to request
<i>mode</i>	Interrupt type,

#### See also

[L4\\_irq\\_mode](#)

#### Returns

Pointer to `l4irq_t` structure, 0 on error

This `l4irq_attach` has to be called in the same thread as `l4irq_wait` and caller has to be a pthread thread.

### 13.5.3.2.3 l4irq\_attach\_thread()

```
l4irq_t * l4irq_attach_thread (
    int irqnum,
    l4_cap_idx_t to_thread )
```

Attach/connect to IRQ.

#### Parameters

<i>irqnum</i>	IRQ number to request
<i>to_thread</i>	Attach IRQ to this specified thread.

#### Returns

Pointer to l4irq\_t structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

### 13.5.3.2.4 l4irq\_attach\_thread\_ft()

```
l4irq_t * l4irq_attach_thread_ft (
    int irqnum,
    l4_cap_idx_t to_thread,
    unsigned mode )
```

Attach/connect to IRQ using given type.

#### Parameters

<i>irqnum</i>	IRQ number to request
<i>to_thread</i>	Attach IRQ to this specified thread.
<i>mode</i>	Interrupt type,

#### See also

[L4\\_irq\\_mode](#)

#### Returns

Pointer to l4irq\_t structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

### 13.5.3.2.5 l4irq\_detach()

```
long l4irq_detach (
    l4irq_t * irq )
```

Detach from IRQ.

**Parameters**

<i>irq</i>	IRQ data structure
------------	--------------------

**Returns**

0 on success, != 0 on error

**13.5.3.2.6 l4irq\_unmask()**

```
long l4irq_unmask (
    l4irq_t * irq )
```

Unmask a specific IRQ.

**Parameters**

<i>irq</i>	IRQ data structure
------------	--------------------

**Returns**

0 on success, != 0 on error

This function is useful if a thread wants to wait for multiple IRQs using l4\_ipc\_wait.

**13.5.3.2.7 l4irq\_unmask\_and\_wait\_any()**

```
long l4irq_unmask_and_wait_any (
    l4irq_t * unmask_irq,
    l4irq_t ** ret_irq )
```

Unmask a specific IRQ and wait for any attached IRQ.

**Parameters**

	<i>unmask_irq</i>	IRQ data structure for unmask.
out	<i>ret_irq</i>	Received interrupt.

**Returns**

0 on success, != 0 on error

**13.5.3.2.8 l4irq\_wait()**

```
long l4irq_wait (
    l4irq_t * irq )
```

Wait for specified IRQ.

## Parameters

<i>irq</i>	IRQ data structure
------------	--------------------

## Returns

0 on success, != 0 on error

## Examples

[examples/libs/libirq/loop.c](#).

## 13.5.3.2.9 l4irq\_wait\_any()

```
long l4irq_wait_any (
    l4irq_t ** irq )
```

Wait for any attached IRQ.

## Return values

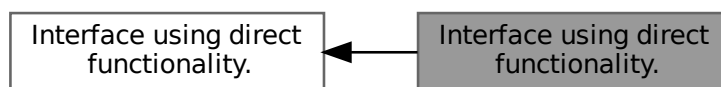
<i>irq</i>	Received interrupt.
------------	---------------------

## Returns

0 on success, != 0 on error

## 13.5.3.3 Interface using direct functionality.

Collaboration diagram for Interface using direct functionality.:



## Functions

- `l4irq_t * l4irq_attach_cap (l4_cap_idx_t irqcap)`  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq_attach_cap_ft (l4_cap_idx_t irqcap, unsigned mode)`  
*Attach/connect to IRQ using given type.*
- `l4irq_t * l4irq_attach_thread_cap (l4_cap_idx_t irqcap, l4_cap_idx_t to_thread)`  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq_attach_thread_cap_ft (l4_cap_idx_t irqcap, l4_cap_idx_t to_thread, unsigned mode)`  
*Attach/connect to IRQ using given type.*

### 13.5.3.3.1 Detailed Description

### 13.5.3.3.2 Function Documentation

#### 13.5.3.3.2.1 l4irq\_attach\_cap()

```
l4irq_t * l4irq_attach_cap (
    l4_cap_idx_t irqcap )
```

Attach/connect to IRQ.

#### Parameters

<i>irqcap</i>	IRQ capability
---------------	----------------

#### Returns

Pointer to l4irq\_t structure, 0 on error

This l4irq\_attach has to be called in the same thread as l4irq\_wait and caller has to be a pthread thread.

#### 13.5.3.3.2.2 l4irq\_attach\_cap\_ft()

```
l4irq_t * l4irq_attach_cap_ft (
    l4_cap_idx_t irqcap,
    unsigned mode )
```

Attach/connect to IRQ using given type.

#### Parameters

<i>irqcap</i>	IRQ capability
<i>mode</i>	Interrupt type,

#### See also

[L4\\_irq\\_mode](#)

#### Returns

Pointer to l4irq\_t structure, 0 on error

This l4irq\_attach has to be called in the same thread as l4irq\_wait and caller has to be a pthread thread.

#### 13.5.3.3.2.3 l4irq\_attach\_thread\_cap()

```
l4irq_t * l4irq_attach_thread_cap (
    l4_cap_idx_t irqcap,
    l4_cap_idx_t to_thread )
```

Attach/connect to IRQ.

**Parameters**

<i>irqcap</i>	IRQ capability
<i>to_thread</i>	Attach IRQ to this thread.

**Returns**

Pointer to `l4irq_t` structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

**13.5.3.3.2.4 l4irq\_attach\_thread\_cap\_ft()**

```
l4irq_t * l4irq_attach_thread_cap_ft (
    l4_cap_idx_t irqcap,
    l4_cap_idx_t to_thread,
    unsigned mode )
```

Attach/connect to IRQ using given type.

**Parameters**

<i>irqcap</i>	IRQ capability
<i>to_thread</i>	Attach IRQ to this thread.
<i>mode</i>	Interrupt type,

**See also**

[L4\\_irq\\_mode](#)

**Returns**

Pointer to `l4irq_t` structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

## 13.6 L4 IPC Opcodes

List of protocol specific opcodes used for communication with [L4Re](#) and Kernel objects.

**Enumerations**

- enum [L4\\_icu\\_opcode](#) {  
[L4\\_ICU\\_OP\\_BIND](#) , [L4\\_ICU\\_OP\\_UNBIND](#) , [L4\\_ICU\\_OP\\_INFO](#) , [L4\\_ICU\\_OP\\_MSI\\_INFO](#) ,  
[L4\\_ICU\\_OP\\_UNMASK](#) , [L4\\_ICU\\_OP\\_MASK](#) , [L4\\_ICU\\_OP\\_SET\\_MODE](#) }  
*Opcodes to the ICU interface.*
- enum [L4\\_ipc\\_gate\\_ops](#) { [L4\\_IPC\\_GATE\\_BIND\\_OP](#) = 0x10 , [L4\\_IPC\\_GATE\\_GET\\_INFO\\_OP](#) = 0x11 }

*Operations on the IPC-gate.*

- enum [L4\\_platform\\_ctl\\_ops](#) {  
[L4\\_PLATFORM\\_CTL\\_SYS\\_SUSPEND\\_OP](#) = 0UL , [L4\\_PLATFORM\\_CTL\\_SYS\\_SHUTDOWN\\_OP](#) = 1UL ,  
[L4\\_PLATFORM\\_CTL\\_CPU\\_ALLOW\\_SHUTDOWN\\_OP](#) = 2UL , [L4\\_PLATFORM\\_CTL\\_CPU\\_ENABLE\\_OP](#) =  
3UL ,  
[L4\\_PLATFORM\\_CTL\\_CPU\\_DISABLE\\_OP](#) = 4UL , [L4\\_PLATFORM\\_CTL\\_SET\\_TASK\\_ASID\\_OP](#) = 0x10UL }

*Operations on platform-control objects.*

- enum [L4\\_task\\_ops](#) {  
[L4\\_TASK\\_MAP\\_OP](#) = 0UL , [L4\\_TASK\\_UNMAP\\_OP](#) = 1UL , [L4\\_TASK\\_CAP\\_INFO\\_OP](#) = 2UL ,  
[L4\\_TASK\\_ADD\\_KU\\_MEM\\_OP](#) = 3UL ,  
[L4\\_TASK\\_LDT\\_SET\\_X86\\_OP](#) = 0x11UL , [L4\\_TASK\\_MAP\\_VGICC\\_ARM\\_OP](#) = 0x12UL }

*Operations on task objects.*

- enum [L4\\_thread\\_ops](#) {  
[L4\\_THREAD\\_CONTROL\\_OP](#) = 0UL , [L4\\_THREAD\\_EX\\_REGS\\_OP](#) = 1UL , [L4\\_THREAD\\_SWITCH\\_OP](#) =  
2UL , [L4\\_THREAD\\_STATS\\_OP](#) = 3UL ,  
[L4\\_THREAD\\_VCPU\\_RESUME\\_OP](#) = 4UL , [L4\\_THREAD\\_REGISTER\\_DELETE\\_IRQ\\_OP](#) = 5UL ,  
[L4\\_THREAD\\_MODIFY\\_SENDER\\_OP](#) = 6UL , [L4\\_THREAD\\_VCPU\\_CONTROL\\_OP](#) = 7UL ,  
[L4\\_THREAD\\_VCPU\\_CONTROL\\_EXT\\_OP](#) = [L4\\_THREAD\\_VCPU\\_CONTROL\\_OP](#) | 0x10000 , [L4\\_THREAD\\_X86\\_GDT\\_OP](#)  
= 0x10UL , [L4\\_THREAD\\_ARM\\_TPIDRURO\\_OP](#) = 0x10UL , [L4\\_THREAD\\_AMD64\\_SET\\_SEGMENT\\_BASE\\_OP](#)  
= 0x12UL ,  
[L4\\_THREAD\\_AMD64\\_GET\\_SEGMENT\\_INFO\\_OP](#) = 0x13UL , [L4\\_THREAD\\_OPCODE\\_MASK](#) = 0xffff }

*Operations on thread objects.*

- enum [L4\\_vcon\\_ops](#) { [L4\\_VCON\\_WRITE\\_OP](#) = 0UL , [L4\\_VCON\\_READ\\_OP](#) = 1UL , [L4\\_VCON\\_SET\\_ATTR\\_OP](#)  
= 2UL , [L4\\_VCON\\_GET\\_ATTR\\_OP](#) = 3UL }

*Operations on vcon objects.*

### 13.6.1 Detailed Description

List of protocol specific opcodes used for communication with [L4Re](#) and Kernel objects.

### 13.6.2 Enumeration Type Documentation

#### 13.6.2.1 L4\_icu\_opcode

enum [L4\\_icu\\_opcode](#)

Opcodes to the ICU interface.

Enumerator

<a href="#">L4_ICU_OP_BIND</a>	Bind opcode.  See also  <a href="#">l4_icu_bind()</a>
<a href="#">L4_ICU_OP_UNBIND</a>	Unbind opcode.  See also  <a href="#">l4_icu_unbind()</a>



## Enumerator

L4_ICU_OP_INFO	Info opcode.  See also <a href="#">l4_icu_info()</a>
L4_ICU_OP_MSI_INFO	Msi-info opcode.  See also <a href="#">l4_icu_msi_info()</a>
L4_ICU_OP_UNMASK	Unmask opcode.  See also <a href="#">l4_icu_unmask()</a>
L4_ICU_OP_MASK	Mask opcode.  See also <a href="#">l4_icu_mask()</a>
L4_ICU_OP_SET_MODE	Set-mode opcode.  See also <a href="#">l4_icu_set_mode()</a>

Definition at line 106 of file [icu.h](#).

### 13.6.2.2 L4\_ipc\_gate\_ops

```
enum L4_ipc_gate_ops
```

Operations on the IPC-gate.

## Enumerator

L4_IPC_GATE_BIND_OP	Bind operation.
L4_IPC_GATE_GET_INFO_OP	Info operation.

Definition at line 116 of file [ipc\\_gate.h](#).

### 13.6.2.3 L4\_platform\_ctl\_ops

```
enum L4_platform_ctl_ops
```

Operations on platform-control objects.

See [L4\\_PROTO\\_PLATFORM\\_CTL](#) for the protocol type to use for messages to platform-control objects.

## Enumerator

L4_PLATFORM_CTL_SYS_SUSPEND_OP	Suspend.
L4_PLATFORM_CTL_SYS_SHUTDOWN_OP	shutdown/reboot
L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP	allow CPU shutdown
L4_PLATFORM_CTL_CPU_ENABLE_OP	enable an offline CPU
L4_PLATFORM_CTL_CPU_DISABLE_OP	disable an online CPU
L4_PLATFORM_CTL_SET_TASK_ASID_OP	Arm: set task ASID.

Definition at line 170 of file [platform\\_control.h](#).

### 13.6.2.4 L4\_task\_ops

enum [L4\\_task\\_ops](#)

Operations on task objects.

## Enumerator

L4_TASK_MAP_OP	Map.
L4_TASK_UNMAP_OP	Unmap.
L4_TASK_CAP_INFO_OP	Cap info.
L4_TASK_ADD_KU_MEM_OP	Add kernel-user memory.
L4_TASK_LDT_SET_X86_OP	x86: LDT set
L4_TASK_MAP_VGICC_ARM_OP	Arm: Map virtual GICC area.

Definition at line 312 of file [task.h](#).

### 13.6.2.5 L4\_thread\_ops

enum [L4\\_thread\\_ops](#)

Operations on thread objects.

## Enumerator

L4_THREAD_CONTROL_OP	Control operation.
L4_THREAD_EX_REGS_OP	Exchange registers operation.
L4_THREAD_SWITCH_OP	Do a thread switch.
L4_THREAD_STATS_OP	Thread statistics.
L4_THREAD_VCPU_RESUME_OP	VCPU resume.
L4_THREAD_REGISTER_DELETE_IRQ_OP	Register an IPC-gate deletion IRQ.
L4_THREAD_MODIFY_SENDER_OP	Modify all senders IDs that match the given pattern.
L4_THREAD_VCPU_CONTROL_OP	Enable / disable VCPU feature.
L4_THREAD_X86_GDT_OP	Gdt.
L4_THREAD_ARM_TPIDRURO_OP	Set TPIDRURO register.
L4_THREAD_AMD64_SET_SEGMENT_BASE_OP	Set segment base.
L4_THREAD_AMD64_GET_SEGMENT_INFO_OP	Get segment information.
L4_THREAD_OPCODE_MASK	Mask for opcodes.

Definition at line 693 of file [thread.h](#).

### 13.6.2.6 L4\_vcon\_ops

enum [L4\\_vcon\\_ops](#)

Operations on vcon objects.

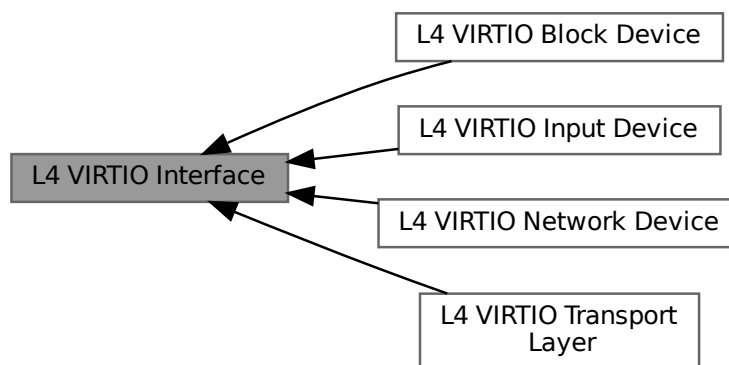
Enumerator

L4_VCON_WRITE_OP	Write.
L4_VCON_READ_OP	Read.
L4_VCON_SET_ATTR_OP	Get console attributes.
L4_VCON_GET_ATTR_OP	Set console attributes.

Definition at line 300 of file [vcon.h](#).

## 13.7 L4 VIRTIO Interface

Collaboration diagram for L4 VIRTIO Interface:



### Modules

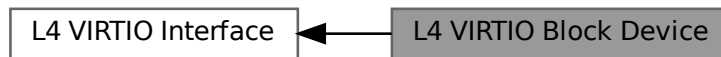
- [L4 VIRTIO Block Device](#)
- [L4 VIRTIO Input Device](#)
- [L4 VIRTIO Network Device](#)
- [L4 VIRTIO Transport Layer](#)

*[L4](#) specific VIRTIO Transport layer.*

### 13.7.1 Detailed Description

### 13.7.2 L4 VIRTIO Block Device

Collaboration diagram for L4 VIRTIO Block Device:



#### Data Structures

- struct [l4virtio\\_block\\_header\\_t](#)  
*Header structure of a request for a block device.*
- struct [l4virtio\\_block\\_discard\\_t](#)  
*Structure used for the write zeroes and discard commands.*
- struct [l4virtio\\_block\\_config\\_t](#)  
*Device configuration for block devices.*

#### Typedefs

- typedef struct [l4virtio\\_block\\_header\\_t](#) [l4virtio\\_block\\_header\\_t](#)  
*Header structure of a request for a block device.*
- typedef struct [l4virtio\\_block\\_discard\\_t](#) [l4virtio\\_block\\_discard\\_t](#)  
*Structure used for the write zeroes and discard commands.*
- typedef struct [l4virtio\\_block\\_config\\_t](#) [l4virtio\\_block\\_config\\_t](#)  
*Device configuration for block devices.*

#### Enumerations

- enum [L4virtio\\_block\\_operations](#) {  
[L4VIRTIO\\_BLOCK\\_T\\_IN](#) = 0 , [L4VIRTIO\\_BLOCK\\_T\\_OUT](#) = 1 , [L4VIRTIO\\_BLOCK\\_T\\_FLUSH](#) = 4 ,  
[L4VIRTIO\\_BLOCK\\_T\\_GET\\_ID](#) = 8 ,  
[L4VIRTIO\\_BLOCK\\_T\\_DISCARD](#) = 11 , [L4VIRTIO\\_BLOCK\\_T\\_WRITE\\_ZEROES](#) = 13 }  
*Kinds of operation over a block device.*
- enum [L4virtio\\_block\\_status](#) { [L4VIRTIO\\_BLOCK\\_S\\_OK](#) = 0 , [L4VIRTIO\\_BLOCK\\_S\\_IOERR](#) = 1 ,  
[L4VIRTIO\\_BLOCK\\_S\\_UNSUPP](#) = 2 }  
*Status of a finished block request.*

#### 13.7.2.1 Detailed Description

#### 13.7.2.2 Enumeration Type Documentation

##### 13.7.2.2.1 L4virtio\_block\_operations

enum [L4virtio\\_block\\_operations](#)

Kinds of operation over a block device.

## Enumerator

L4VIRTIO_BLOCK_T_IN	Read from device.
L4VIRTIO_BLOCK_T_OUT	Write to device.
L4VIRTIO_BLOCK_T_FLUSH	Flush data to disk.
L4VIRTIO_BLOCK_T_GET_ID	Get device ID.
L4VIRTIO_BLOCK_T_DISCARD	Discard a range of sectors.
L4VIRTIO_BLOCK_T_WRITE_ZEROES	Write zeroes to a range of sectors.

Definition at line 19 of file [virtio\\_block.h](#).

## 13.7.2.2.2 L4virtio\_block\_status

enum [L4virtio\\_block\\_status](#)

Status of a finished block request.

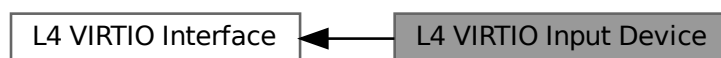
## Enumerator

L4VIRTIO_BLOCK_S_OK	Request finished successfully.
L4VIRTIO_BLOCK_S_IOERR	IO error on device.
L4VIRTIO_BLOCK_S_UNSUPP	Operation is not supported.

Definition at line 32 of file [virtio\\_block.h](#).

## 13.7.3 L4 VIRTIO Input Device

Collaboration diagram for L4 VIRTIO Input Device:



## Data Structures

- struct [l4virtio\\_input\\_absinfo\\_t](#)  
*Information about the absolute axis in the underlying evdev implementation.*
- struct [l4virtio\\_input\\_devids\\_t](#)  
*Device ID information for the device.*
- struct [l4virtio\\_input\\_config\\_t](#)  
*Device configuration for input devices.*
- struct [l4virtio\\_input\\_event\\_t](#)  
*Single event in event or status queue.*

## Typedefs

- typedef struct [l4virtio\\_input\\_absinfo\\_t](#) [l4virtio\\_absinfo\\_t](#)  
*Information about the absolute axis in the underlying evdev implementation.*
- typedef struct [l4virtio\\_input\\_devids\\_t](#) [l4virtio\\_input\\_devids\\_t](#)  
*Device ID information for the device.*
- typedef struct [l4virtio\\_input\\_config\\_t](#) [l4virtio\\_input\\_config\\_t](#)  
*Device configuration for input devices.*
- typedef struct [l4virtio\\_input\\_event\\_t](#) [l4virtio\\_input\\_event\\_t](#)  
*Single event in event or status queue.*

## Enumerations

- enum [L4virtio\\_input\\_config\\_select](#)  
*Device information selectors.*

### 13.7.3.1 Detailed Description

## 13.7.4 L4 VIRTIO Network Device

Collaboration diagram for L4 VIRTIO Network Device:



## Data Structures

- struct [l4virtio\\_net\\_header\\_t](#)  
*Header structure of a request for a network device.*
- struct [l4virtio\\_net\\_config\\_t](#)  
*Device configuration for network devices.*

## Typedefs

- typedef struct [l4virtio\\_net\\_header\\_t](#) [l4virtio\\_net\\_header\\_t](#)  
*Header structure of a request for a network device.*
- typedef struct [l4virtio\\_net\\_config\\_t](#) [l4virtio\\_net\\_config\\_t](#)  
*Device configuration for network devices.*

## Enumerations

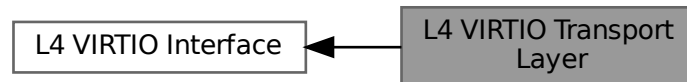
- enum [L4virtio\\_net\\_feature\\_bits](#)  
*Network device feature bits.*

### 13.7.4.1 Detailed Description

## 13.7.5 L4 VIRTIO Transport Layer

L4 specific VIRTIO Transport layer.

Collaboration diagram for L4 VIRTIO Transport Layer:



### Namespaces

- namespace [L4virtio](#)  
*L4-VIRTIO Transport C++ API.*

### Data Structures

- struct [l4virtio\\_config\\_hdr\\_t](#)  
*L4-VIRTIO config header, provided in shared data space.*
- struct [l4virtio\\_config\\_queue\\_t](#)  
*Queue configuration entry.*

### Typedefs

- typedef struct [l4virtio\\_config\\_hdr\\_t](#) [l4virtio\\_config\\_hdr\\_t](#)  
*L4-VIRTIO config header, provided in shared data space.*
- typedef struct [l4virtio\\_config\\_queue\\_t](#) [l4virtio\\_config\\_queue\\_t](#)  
*Queue configuration entry.*

### Enumerations

- enum [L4\\_virtio\\_protocol](#)  
*L4-VIRTIO protocol number.*
- enum [L4\\_virtio\\_opcodes](#) {  
[L4VIRTIO\\_OP\\_SET\\_STATUS](#) = 0 , [L4VIRTIO\\_OP\\_CONFIG\\_QUEUE](#) = 1 , [L4VIRTIO\\_OP\\_REGISTER\\_DS](#)  
= 3 , [L4VIRTIO\\_OP\\_DEVICE\\_CONFIG](#) = 4 ,  
[L4VIRTIO\\_OP\\_GET\\_DEVICE\\_IRQ](#) = 5 }  
*Opcodes to setup and configure a device.*

- enum `L4virtio_device_ids` {  
`L4VIRTIO_ID_NET` = 1 , `L4VIRTIO_ID_BLOCK` = 2 , `L4VIRTIO_ID_CONSOLE` = 3 , `L4VIRTIO_ID_RNG` = 4  
, `L4VIRTIO_ID_BALLOON` = 5 , `L4VIRTIO_ID_RPMSG` = 7 , `L4VIRTIO_ID_SCSI` = 8 , `L4VIRTIO_ID_9P` = 9 ,  
`L4VIRTIO_ID_RPROC_SERIAL` = 11 , `L4VIRTIO_ID_CAIF` = 12 , `L4VIRTIO_ID_GPU` = 16 , `L4VIRTIO_ID_INPUT`  
= 18 ,  
`L4VIRTIO_ID_VSOCK` = 19 , `L4VIRTIO_ID_CRYPT` = 20 , `L4VIRTIO_ID_FS` = 26 , `L4VIRTIO_ID_SCMI` =  
32 ,  
`L4VIRTIO_ID_SOCK` = 0x9999 }  
*Virtio device IDs as reported in the driver's config space.*
- enum `L4virtio_device_status` {  
`L4VIRTIO_STATUS_ACKNOWLEDGE` = 1 , `L4VIRTIO_STATUS_DRIVER` = 2 , `L4VIRTIO_STATUS_DRIVER_OK`  
= 4 , `L4VIRTIO_STATUS_FEATURES_OK` = 8 ,  
`L4VIRTIO_STATUS_DEVICE_NEEDS_RESET` = 0x40 , `L4VIRTIO_STATUS_FAILED` = 0x80 }  
*Virtio device status bits.*
- enum `L4virtio_feature_bits` { `L4VIRTIO_FEATURE_VERSION_1` = 32 , `L4VIRTIO_FEATURE_CMD_CONFIG`  
= 160 }  
*L4virtio-specific feature bits.*
- enum `L4_virtio_irq_status` { `L4VIRTIO_IRQ_STATUS_VRING` = 1 , `L4VIRTIO_IRQ_STATUS_CONFIG` = 2 }  
*VIRTIO IRQ status codes (l4virtio\_config\_hdr\_t::irq\_status).*
- enum `L4_virtio_cmd` {  
`L4VIRTIO_CMD_NONE` = 0x00000000 , `L4VIRTIO_CMD_SET_STATUS` = 0x01000000 , `L4VIRTIO_CMD_CFG_QUEUE`  
= 0x02000000 , `L4VIRTIO_CMD_CFG_CHANGED` = 0x04000000 ,  
`L4VIRTIO_CMD_NOTIFY_QUEUE` = 0x08000000 , `L4VIRTIO_CMD_MASK` = 0xff000000 }  
*Virtio commands for device configuration.*

## Functions

- `l4virtio_config_queue_t * l4virtio_config_queues` (`l4virtio_config_hdr_t` const \*cfg)  
*Get the pointer to the first queue config.*
- void \* `l4virtio_device_config` (`l4virtio_config_hdr_t` const \*cfg)  
*Get the pointer to the device configuration.*
- void `l4virtio_set_feature` (`l4_uint32_t` \*feature\_map, unsigned feat)  
*Set the given feature bit in a feature map.*
- void `l4virtio_clear_feature` (`l4_uint32_t` \*feature\_map, unsigned feat)  
*Clear the given feature bit in a feature map.*
- unsigned `l4virtio_get_feature` (`l4_uint32_t` \*feature\_map, unsigned feat)  
*Check if the given bit in a feature map is set.*
- int `l4virtio_set_status` (`l4_cap_idx_t` cap, unsigned status) `L4_NOTHROW`
- int `l4virtio_config_queue` (`l4_cap_idx_t` cap, unsigned queue) `L4_NOTHROW`
- int `l4virtio_register_ds` (`l4_cap_idx_t` cap, `l4_cap_idx_t` ds\_cap, `l4_uint64_t` base, `l4_umword_t` offset,  
`l4_umword_t` size) `L4_NOTHROW`
- int `l4virtio_device_config_ds` (`l4_cap_idx_t` cap, `l4_cap_idx_t` config\_ds, `l4_addr_t` \*ds\_offset) `L4_NOTHROW`
- int `l4virtio_device_notification_irq` (`l4_cap_idx_t` cap, unsigned index, `l4_cap_idx_t` irq) `L4_NOTHROW`

### 13.7.5.1 Detailed Description

`L4` specific VIRTIO Transport layer.

The `L4` specific VIRTIO Transport layer is based on `L4Re::Dataspace` as shared memory and `L4::Irq` for signaling. The VIRTIO configuration space is mostly based on a shared memory implementation too and accompanied by two IPC functions to synchronize the configuration between device and driver.



### 13.7.5.2 Typedef Documentation

#### 13.7.5.2.1 l4virtio\_config\_queue\_t

```
typedef struct l4virtio_config_queue_t l4virtio_config_queue_t
```

Queue configuration entry.

An array of such entries is available at the `l4virtio_config_hdr_t::queues_offset` in the config data space.

Consistency rules for the queue config are:

- A driver might read `num_max` at any time.
- A driver must write to `num`, `desc_addr`, `avail_addr`, and `used_addr` only when `ready` is zero (0). Values in these fields are validated and used by the device only after successfully setting `ready` to one (1), either by the IPC or by `L4VIRTIO_CMD_CFG_QUEUE`.
- The value of `device_notify_index` is valid only when `ready` is one.
- The driver might write to `device_notify_index` at any time, however the change is guaranteed to take effect after a successful `L4VIRTIO_CMD_CFG_QUEUE` or after a `config_queue` IPC. Note, the change might also have immediate effect, depending on the device implementation.

### 13.7.5.3 Enumeration Type Documentation

#### 13.7.5.3.1 L4\_virtio\_cmd

```
enum L4_virtio_cmd
```

Virtio commands for device configuration.

Enumerator

<code>L4VIRTIO_CMD_NONE</code>	No command pending.
<code>L4VIRTIO_CMD_SET_STATUS</code>	Set the status register.
<code>L4VIRTIO_CMD_CFG_QUEUE</code>	Configure a queue.
<code>L4VIRTIO_CMD_CFG_CHANGED</code>	Device config changed.
<code>L4VIRTIO_CMD_NOTIFY_QUEUE</code>	Configure a queue.
<code>L4VIRTIO_CMD_MASK</code>	Mask to get command bits.

Definition at line 116 of file [virtio.h](#).

#### 13.7.5.3.2 L4\_virtio\_irq\_status

```
enum L4_virtio_irq_status
```

VIRTIO IRQ status codes (`l4virtio_config_hdr_t::irq_status`).

Note

`l4virtio_config_hdr_t::irq_status` is currently unused.

## Enumerator

L4VIRTIO_IRQ_STATUS_VRING	VRING IRQ pending flag.
L4VIRTIO_IRQ_STATUS_CONFIG	CONFIG IRQ pending flag.

Definition at line 107 of file [virtio.h](#).

**13.7.5.3.3 L4\_virtio\_opcodes**

```
enum L4_virtio_opcodes
```

Opcodes to setup and configure a device.

## Enumerator

L4VIRTIO_OP_SET_STATUS	Write device status register.
L4VIRTIO_OP_CONFIG_QUEUE	Configure queue.
L4VIRTIO_OP_REGISTER_DS	Register shared memory with device.
L4VIRTIO_OP_DEVICE_CONFIG	Get device config page.
L4VIRTIO_OP_GET_DEVICE_IRQ	Retrieve device notification IRQ.

Definition at line 51 of file [virtio.h](#).

**13.7.5.3.4 L4virtio\_device\_ids**

```
enum L4virtio_device_ids
```

Virtio device IDs as reported in the driver's config space.

## Enumerator

L4VIRTIO_ID_NET	Virtual ethernet card.
L4VIRTIO_ID_BLOCK	General block device.
L4VIRTIO_ID_CONSOLE	Simple device for data IO via ports.
L4VIRTIO_ID_RNG	Entropy source.
L4VIRTIO_ID_BALLOON	Memory ballooning device.
L4VIRTIO_ID_RPMMSG	Device using rpmsg protocol.
L4VIRTIO_ID_SCSI	SCSI host device.
L4VIRTIO_ID_9P	Device using 9P transport protocol.
L4VIRTIO_ID_RPROC_SERIAL	Rproc serial device.
L4VIRTIO_ID_CAIF	Device using CAIF network protocol.
L4VIRTIO_ID_GPU	GPU.
L4VIRTIO_ID_INPUT	Input.
L4VIRTIO_ID_VSOCK	Vsock transport.
L4VIRTIO_ID_CRYPT	Crypto.
L4VIRTIO_ID_FS	FS.
L4VIRTIO_ID_SCSI	Scmi device.
L4VIRTIO_ID_SOCKET	Unofficial socket device.

Definition at line 61 of file [virtio.h](#).

### 13.7.5.3.5 L4virtio\_device\_status

enum [L4virtio\\_device\\_status](#)

Virtio device status bits.

#### Enumerator

L4VIRTIO_STATUS_ACKNOWLEDGE	Guest OS has found device.
L4VIRTIO_STATUS_DRIVER	Guest OS knows how to drive device.
L4VIRTIO_STATUS_DRIVER_OK	Driver is set up.
L4VIRTIO_STATUS_FEATURES_OK	Driver has acknowledged feature set.
L4VIRTIO_STATUS_DEVICE_NEEDS_RESET	Device detected fatal error.
L4VIRTIO_STATUS_FAILED	Driver detected fatal error.

Definition at line 84 of file [virtio.h](#).

### 13.7.5.3.6 L4virtio\_feature\_bits

enum [L4virtio\\_feature\\_bits](#)

L4virtio-specific feature bits.

#### Enumerator

L4VIRTIO_FEATURE_VERSION_1	Virtio protocol version 1 supported. Must be 1 for <a href="#">L4virtio</a> .
L4VIRTIO_FEATURE_CMD_CONFIG	Status and queue config are set via cmd field instead of via IPC.

Definition at line 95 of file [virtio.h](#).

## 13.7.5.4 Function Documentation

### 13.7.5.4.1 l4virtio\_config\_queue()

```
int l4virtio_config_queue (
    l4_cap_idx_t cap,
    unsigned queue )
```

#### Parameters

<i>cap</i>	Capability to the VIRTIO host.
------------	--------------------------------

Trigger queue configuration of the given queue.

Usually all queues are configured when the status is written to running. However, in some cases queues shall

be disabled or enabled dynamically, in this case this function triggers a reconfiguration from the shared memory register of the queue config.

#### Parameters

<i>queue</i>	Queue index for the queue to be configured.
--------------	---

#### Return values

0	on success.
-L4_EIO	The queue's status is invalid.
-L4_ERANGE	The queue index exceeds the number of queues.
-L4_EINVAL	Otherwise.

#### 13.7.5.4.2 l4virtio\_config\_queues()

```
l4virtio_config_queue_t * l4virtio_config_queues (
    l4virtio_config_hdr_t const * cfg ) [inline]
```

Get the pointer to the first queue config.

#### Parameters

<i>cfg</i>	Pointer to the config header.
------------	-------------------------------

#### Returns

pointer to queue config of queue 0.

Definition at line 249 of file [virtio.h](#).

References [l4virtio\\_config\\_hdr\\_t::queues\\_offset](#).

#### 13.7.5.4.3 l4virtio\_device\_config()

```
void * l4virtio_device_config (
    l4virtio_config_hdr_t const * cfg ) [inline]
```

Get the pointer to the device configuration.

#### Parameters

<i>cfg</i>	Pointer to the config header.
------------	-------------------------------

#### Returns

pointer to device configuration structure.

Definition at line 260 of file [virtio.h](#).

#### 13.7.5.4.4 l4virtio\_device\_config\_ds()

```
int l4virtio_device_config_ds (
    l4_cap_idx_t cap,
    l4_cap_idx_t config_ds,
    l4_addr_t * ds_offset )
```

##### Parameters

<i>cap</i>	Capability to the L4-VIRTIO host
------------	----------------------------------

Get the dataspace with the [L4virtio](#) configuration page.

##### Parameters

<i>config_ds</i>	Capability for receiving the dataspace capability for the shared L4-VIRTIO config data space.
<i>ds_offset</i>	Offset into the dataspace where the device configuration structure starts.

#### 13.7.5.4.5 l4virtio\_device\_notification\_irq()

```
int l4virtio_device_notification_irq (
    l4_cap_idx_t cap,
    unsigned index,
    l4_cap_idx_t irq )
```

##### Parameters

<i>cap</i>	Capability to the L4-VIRTIO host
------------	----------------------------------

Get the notification interrupt corresponding to the given index.

##### Parameters

	<i>index</i>	Index of the interrupt.
out	<i>irq</i>	Triggerable for the given index.

##### Return values

<i>L4_EOK</i>	Success.
<i>L4_ENOSYS</i>	IRQ notification not supported by device.
<i>&lt;0</i>	Other error.

An index is only guaranteed to return an IRQ object when the index is set in one of the device notify index fields. The device must return the same interrupt for a given index as long as the index is in use. If an index disappears as a result of a configuration change and then is reused later, the interrupt is not guaranteed to be the same.

Interrupts must always be rerequested after a device reset.

#### 13.7.5.4.6 l4virtio\_register\_ds()

```
int l4virtio_register_ds (
    l4_cap_idx_t cap,
    l4_cap_idx_t ds_cap,
    l4_uint64_t base,
    l4_umword_t offset,
    l4_umword_t size )
```

##### Parameters

<i>cap</i>	Capability to the VIRTIO host
------------	-------------------------------

Register a shared data space with VIRTIO host.

##### Parameters

<i>ds_cap</i>	Dataspace capability to register. The lower 8 bits determine the rights mask with which the guest's rights are masked during the registration of the dataspace at the VIRTIO host.
<i>base</i>	VIRTIO guest physical start address of shared memory region
<i>offset</i>	Offset within the data space that is attached to the given <i>base</i> in the guest physical memory.
<i>size</i>	Size of the memory region in the guest

##### Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EINVAL</i>	The <i>ds_cap</i> capability is invalid, does not refer to a valid dataspace, is not a trusted dataspace if trusted dataspace validation is enabled, or <i>size</i> and <i>offset</i> specify an invalid region.
<i>-L4_ENOMEM</i>	The limit of dataspaces that can be registered has been reached or no capability slot could be allocated.
<i>-L4_ERANGE</i>	<i>offset</i> is larger than the size of the dataspace.
<i>&lt;0</i>	Any error returned by the dataspace when queried for information during setup or any error returned by the region manager from attaching the dataspace.

#### 13.7.5.4.7 l4virtio\_set\_status()

```
int l4virtio_set_status (
    l4_cap_idx_t cap,
    unsigned status )
```

##### Parameters

<i>cap</i>	Capability to the VIRTIO host
------------	-------------------------------

Write the VIRTIO status register.

##### Parameters

<i>status</i>	Status word to write to the VIRTIO status.
---------------	--

## Return values

0	on success.
---	-------------

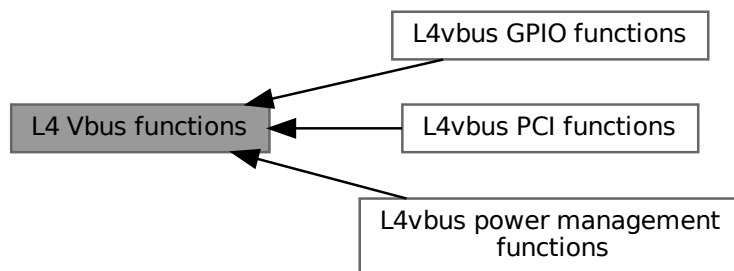
## Note

All other registers are accessed via shared memory.

## 13.8 L4 Vbus functions

C interface of the Vbus API.

Collaboration diagram for L4 Vbus functions:



## Modules

- [L4vbus GPIO functions](#)
- [L4vbus PCI functions](#)
- [L4vbus power management functions](#)

## Enumerations

- enum [L4vbus\\_dma\\_domain\\_assign\\_flags](#) { [L4VBUS\\_DMAD\\_UNBIND](#) = 0 , [L4VBUS\\_DMAD\\_BIND](#) = 1 , [L4VBUS\\_DMAD\\_L4RE\\_DMA\\_SPACE](#) = 0 , [L4VBUS\\_DMAD\\_KERNEL\\_DMA\\_SPACE](#) = 2 }
- Flags for [l4vbus\\_assign\\_dma\\_domain\(\)](#).*

## Functions

- `int l4vbus_get_device_by_hid (l4_cap_idx_t vbus, l4vbus_device_handle_t parent, l4vbus_device_handle_t *child, char const *hid, int depth, l4vbus_device_t *devinfo)`  
*Find a device by the hardware interface identifier (HID).*
- `int l4vbus_get_next_device (l4_cap_idx_t vbus, l4vbus_device_handle_t parent, l4vbus_device_handle_t *child, int depth, l4vbus_device_t *devinfo)`  
*Find next child following `child`.*
- `int l4vbus_get_device (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, l4vbus_device_t *devinfo)`  
*Obtain detailed information about a Vbus device.*
- `int l4vbus_get_resource (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, unsigned res_idx, l4vbus_resource_t *res)`  
*Obtain the resource description of an individual device resource.*
- `int l4vbus_is_compatible (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, char const *cid)`  
*Check if the given device has a compatibility ID (CID) or HID that matches `cid`.*
- `int l4vbus_get_hid (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, char *hid, unsigned long max_len)`  
*Get the HID (hardware identifier) of a device.*
- `int l4vbus_get_adr (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, l4_uint32_t *adr)`  
*Get the bus-specific address of a device.*
- `int l4vbus_request_ioport (l4_cap_idx_t vbus, l4vbus_resource_t const *res)`  
*Request an IO port resource.*
- `int l4vbus_assign_dma_domain (l4_cap_idx_t vbus, unsigned domain_id, unsigned flags, l4_cap_idx_t dma_space)`  
*Bind or unbind a kernel [DMA space](#) or a `L4Re::Dma_space` to a DMA domain.*
- `int l4vbus_release_ioport (l4_cap_idx_t vbus, l4vbus_resource_t const *res)`  
*Release a previously requested IO port resource.*
- `int l4vbus_vicu_get_cap (l4_cap_idx_t vbus, l4vbus_device_handle_t icu, l4_cap_idx_t cap)`  
*Get capability of ICU.*

### 13.8.1 Detailed Description

C interface of the Vbus API.

The virtual bus (Vbus) is a hierarchical (tree) structure of device nodes where each device has a set of resources attached to it. Each virtual bus provides an `lcu` ([Interrupt controller](#)) for interrupt handling.

The Vbus interface allows a client to find and query devices present on his virtual bus. After obtaining a device handle for a specific device the client can enumerate its resources.

#### Include File

```
#include <l4/vbus/vbus.h>
```

Refer to [L4vbus](#) for the C++ API.

### 13.8.2 Enumeration Type Documentation

#### 13.8.2.1 L4vbus\_dma\_domain\_assign\_flags

```
enum L4vbus_dma_domain_assign_flags
```

Flags for `l4vbus_assign_dma_domain()`.



## Enumerator

L4VBUS_DMAD_UNBIND	Unbind the given DMA space from the DMA domain.
L4VBUS_DMAD_BIND	Bind the given DMA space to the DMA domain.
L4VBUS_DMAD_L4RE_DMA_SPACE	The given DMA space is an <a href="#">L4Re::Dma_space</a> .
L4VBUS_DMAD_KERNEL_DMA_SPACE	The given DMA space is a kernel DMA space ( <a href="#">L4::Task</a> )

Definition at line 176 of file [vbus.h](#).

### 13.8.3 Function Documentation

#### 13.8.3.1 l4vbus\_assign\_dma\_domain()

```
int l4vbus_assign_dma_domain (
    l4_cap_idx_t vbus,
    unsigned domain_id,
    unsigned flags,
    l4_cap_idx_t dma_space )
```

Bind or unbind a kernel [DMA space](#) or a [L4Re::Dma\\_space](#) to a DMA domain.

## Parameters

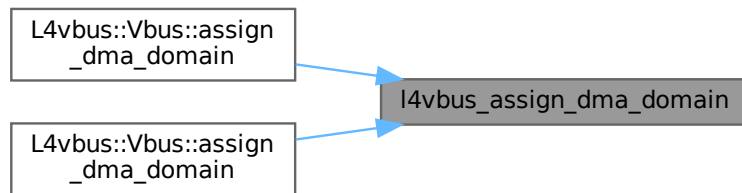
<i>vbus</i>	Capability of the system bus
<i>domain_id</i>	DMA domain ID (resource address of DMA domain found on the vBUS). If the value is ~0U the DMA space of the whole vBUS is used.
<i>flags</i>	A combination of <a href="#">L4vbus_dma_domain_assign_flags</a> .
<i>dma_space</i>	The DMA space capability to bind or unbind, this must either be an <a href="#">L4Re::Dma_space</a> or a kernel <a href="#">DMA space</a> ( <a href="#">L4::Task</a> created with L4_PROTO_DMA_SPACE) and the type must be reflected in the <i>flags</i> .

## Return values

0	Operation completed successfully.
-L4_ENOENT	The vbus does not support a global DMA domain or no DMA domain could be found.
-L4_EINVAL	Invalid argument used.
-L4_EBUSY	DMA domain is already active, this means another DMA space is already assigned.

Referenced by [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#), and [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#).

Here is the caller graph for this function:



### 13.8.3.2 l4vbus\_get\_adr()

```
int l4vbus_get_adr (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    l4_uint32_t * adr )
```

Get the bus-specific address of a device.

#### Parameters

	<i>vbus</i>	Capability of the system bus
	<i>dev</i>	Handle of the device
out	<i>adr</i>	Address

#### Return values

<i>L4_EOK</i>	Success.
<i>-L4_ENOSYS</i>	Device has no valid address.

### 13.8.3.3 l4vbus\_get\_device()

```
int l4vbus_get_device (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    l4vbus_device_t * devinfo )
```

Obtain detailed information about a Vbus device.

#### Parameters

	<i>vbus</i>	Capability of the vbus to which the device is connected.
	<i>dev</i>	Device handle of the device from which to retrieve the details.
out	<i>devinfo</i>	Information structure which contains details about the device. The pointer might be NULL.

## Return values

<i>0</i>	Success.
<i>-L4_ENODEV</i>	No device with the given device handle <i>dev</i> could be found.

Referenced by [L4vbus::Device::device\(\)](#).

Here is the caller graph for this function:



## 13.8.3.4 l4vbus\_get\_device\_by\_hid()

```

int l4vbus_get_device_by_hid (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t parent,
    l4vbus_device_handle_t * child,
    char const * hid,
    int depth,
    l4vbus_device_t * devinfo )

```

Find a device by the hardware interface identifier (HID).

## Parameters

<i>vbus</i>	Capability of the system bus
<i>parent</i>	Handle to the parent to start the search

This function searches the vbus for a device with the given HID and returns a handle to the first matching device. The HID usually conforms to an ACPI HID or a Linux device tree compatible identifier.

It is possible to have multiple devices with the same HID on a vbus. In order to find all matching devices this function has to be called repeatedly with *child* pointing to the device found in the previous iteration. The iteration starts at *child* that might be any device node in the tree.

## Parameters

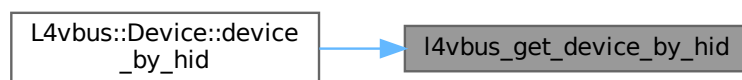
<i>in, out</i>	<i>child</i>	Handle of the device from where in the device tree the search should start. To start searching from the beginning <i>child</i> must be initialized using the default ( <a href="#">L4VBUS_NULL</a> ). If a matching device is found, its handle is returned through this parameter.
	<i>hid</i>	HID of the device
	<i>depth</i>	Maximum depth for the recursive lookup
<i>out</i>	<i>devinfo</i>	Device information structure (might be NULL)

## Return values

<code>&gt;=0</code>	A device with the given HID was found.
<code>-L4_ENOENT</code>	No device with the given HID could be found on the vbus.
<code>-L4_EINVAL</code>	Invalid or no HID provided.
<code>-L4_ENODEV</code>	Function called on a non-existing device.

Referenced by [L4vbus::Device::device\\_by\\_hid\(\)](#).

Here is the caller graph for this function:



### 13.8.3.5 l4vbus\_get\_hid()

```

int l4vbus_get_hid (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    char * hid,
    unsigned long max_len )
  
```

Get the HID (hardware identifier) of a device.

## Parameters

<i>vbus</i>	Capability of the system bus
<i>dev</i>	Handle of the device
<i>hid</i>	Pointer to a buffer for the HID string
<i>max_len</i>	The size of the buffer ( <i>hid</i> )

## Returns

the length of the HID string on success, else failure

### 13.8.3.6 l4vbus\_get\_next\_device()

```

int l4vbus_get_next_device (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t parent,
    l4vbus_device_handle_t * child,
  
```

```
int depth,  
l4vbus_device_t * devinfo )
```

Find next child following `child`.

## Parameters

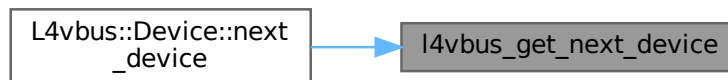
	<i>vbus</i>	Capability of the system bus
	<i>parent</i>	Handle to the parent device (use <a href="#">L4VBUS_ROOT_BUS</a> for the system bus)
in, out	<i>child</i>	Handle of the device that precedes the device that shall be returned. To start from the beginning, <i>child</i> must be initialized with <a href="#">L4VBUS_NULL</a> . If a device is found, its handle is returned through this parameter.
	<i>depth</i>	Depth to look for
out	<i>devinfo</i>	Device information (might be NULL)

## Returns

0 on success, else failure

Referenced by [L4vbus::Device::next\\_device\(\)](#).

Here is the caller graph for this function:



## 13.8.3.7 l4vbus\_get\_resource()

```

int l4vbus_get_resource (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    unsigned res_idx,
    l4vbus_resource_t * res )
  
```

Obtain the resource description of an individual device resource.

## Parameters

	<i>vbus</i>	Capability of the vbus to which the device is connected.
	<i>dev</i>	Device handle of the device on the vbus. The device handle can be obtained by using the <a href="#">l4vbus_get_device_by_hid()</a> and <a href="#">l4vbus_get_next_device()</a> functions.
	<i>res_idx</i>	Index of the resource for which the resource description should be returned. The total number of resources for a device is available in the <a href="#">l4vbus_device_t</a> structure that is returned by <a href="#">L4vbus::Device::device_by_hid()</a> and <a href="#">L4vbus::Device::next_device()</a> .
out	<i>res</i>	Descriptor of the resource.

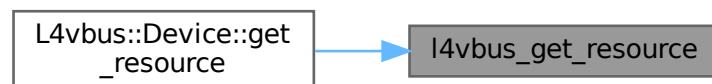
This function returns the resource descriptor of an individual device resource selected by the *res\_idx* parameter.

## Return values

0	Success.
-L4_ENOENT	Invalid resource index <code>res_idx</code> .

Referenced by [L4vbus::Device::get\\_resource\(\)](#).

Here is the caller graph for this function:



### 13.8.3.8 l4vbus\_is\_compatible()

```
int l4vbus_is_compatible (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    char const * cid )
```

Check if the given device has a compatibility ID (CID) or HID that matches *cid*.

## Parameters

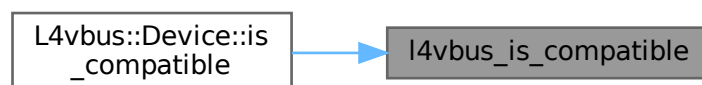
<i>vbus</i>	Capability of the system bus
<i>dev</i>	device handle for which the CID shall be tested
<i>cid</i>	the compatibility ID to test

## Returns

1 when the given ID (*cid*) matches this device, 0 when the given ID does not match, <0 on error.

Referenced by [L4vbus::Device::is\\_compatible\(\)](#).

Here is the caller graph for this function:



### 13.8.3.9 l4vbus\_release\_ioport()

```
int l4vbus_release_ioport (
    l4_cap_idx_t vbus,
    l4vbus_resource_t const * res )
```

Release a previously requested IO port resource.

#### Parameters

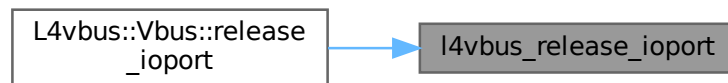
	<i>vbus</i>	Capability of the system bus.
in	<i>res</i>	The IO port resource to be released from the bus.

#### Returns

>=0 on success, <0 on error.

Referenced by [L4vbus::Vbus::release\\_ioport\(\)](#).

Here is the caller graph for this function:



### 13.8.3.10 l4vbus\_request\_ioport()

```
int l4vbus_request_ioport (
    l4_cap_idx_t vbus,
    l4vbus_resource_t const * res )
```

Request an IO port resource.

#### Parameters

	<i>vbus</i>	Capability of the system bus.
in	<i>res</i>	The IO port resource to be requested from the bus.

#### Return values

0	Success.
-L4_EINVAL	Resource is not an IO port resource.
-L4_ENOENT	No matching IO port resource found.



If any IO port resource is found that contains the requested IO port range the IO ports are obtained.

Referenced by [L4vbus::Vbus::request\\_ioport\(\)](#).

Here is the caller graph for this function:



#### 13.8.3.11 l4vbus\_vicu\_get\_cap()

```
int l4vbus_vicu_get_cap (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t icu,
    l4_cap_idx_t cap )
```

Get capability of ICU.

##### Parameters

<i>vbus</i>	Capability of the system bus.
<i>icu</i>	ICU device handle.
<i>cap</i>	Capability slot for the capability.

##### Returns

0 on success, else failure

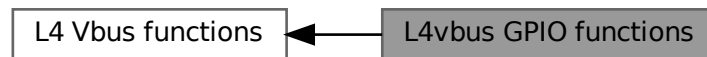
Referenced by [L4vbus::Icu::vicu\(\)](#).

Here is the caller graph for this function:



### 13.8.4 L4vbus GPIO functions

Collaboration diagram for L4vbus GPIO functions:



#### Enumerations

- enum `L4vbus_gpio_generic_func` { `L4VBUS_GPIO_SETUP_INPUT` = 0x100 , `L4VBUS_GPIO_SETUP_OUTPUT` = 0x200 , `L4VBUS_GPIO_SETUP_IRQ` = 0x300 }  
*Constants for generic GPIO functions.*
- enum `L4vbus_gpio_pull_modes` { `L4VBUS_GPIO_PIN_PULL_NONE` = 0x100 , `L4VBUS_GPIO_PIN_PULL_UP` = 0x200 , `L4VBUS_GPIO_PIN_PULL_DOWN` = 0x300 }  
*Constants for generic GPIO pull up/down resistor configuration.*

#### Functions

- int `l4vbus_gpio_setup` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, unsigned mode, int value)  
*Configure the function of a GPIO pin.*
- int `l4vbus_gpio_config_pull` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, unsigned mode)  
*Generic function to set pull up/down mode.*
- int `l4vbus_gpio_config_pad` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, unsigned func, unsigned value)  
*Hardware specific configuration function.*
- int `l4vbus_gpio_config_get` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, unsigned func, unsigned \*value)  
*Read hardware specific configuration.*
- int `l4vbus_gpio_get` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin)  
*Read value of GPIO input pin.*
- int `l4vbus_gpio_set` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, int value)  
*Set GPIO output pin.*
- int `l4vbus_gpio_multi_setup` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned offset, unsigned mask, unsigned mode, unsigned value)  
*Configure function of multiple GPIO pins at once.*
- int `l4vbus_gpio_multi_config_pad` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned offset, unsigned mask, unsigned func, unsigned value)  
*Hardware specific configuration function for multiple GPIO pins.*
- int `l4vbus_gpio_multi_get` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned offset, unsigned \*data)  
*Read values of multiple GPIO pins at once.*
- int `l4vbus_gpio_multi_set` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned offset, unsigned mask, unsigned data)  
*Set multiple GPIO output pins at once.*
- int `l4vbus_gpio_to_irq` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin)  
*Create IRQ for GPIO pin.*

### 13.8.4.1 Detailed Description

### 13.8.4.2 Enumeration Type Documentation

#### 13.8.4.2.1 L4vbus\_gpio\_generic\_func

enum [L4vbus\\_gpio\\_generic\\_func](#)

Constants for generic GPIO functions.

##### Enumerator

L4VBUS_GPIO_SETUP_INPUT	Set GPIO pin to input.
L4VBUS_GPIO_SETUP_OUTPUT	Set GPIO pin to output.
L4VBUS_GPIO_SETUP_IRQ	Set GPIO pin to IRQ.

Definition at line 26 of file [vbus\\_gpio.h](#).

#### 13.8.4.2.2 L4vbus\_gpio\_pull\_modes

enum [L4vbus\\_gpio\\_pull\\_modes](#)

Constants for generic GPIO pull up/down resistor configuration.

##### Enumerator

L4VBUS_GPIO_PIN_PULL_NONE	No pull up or pull down resistors.
L4VBUS_GPIO_PIN_PULL_UP	enable pull up resistor
L4VBUS_GPIO_PIN_PULL_DOWN	enable pull down resistor

Definition at line 36 of file [vbus\\_gpio.h](#).

### 13.8.4.3 Function Documentation

#### 13.8.4.3.1 l4vbus\_gpio\_config\_get()

```
int l4vbus_gpio_config_get (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin,
    unsigned func,
    unsigned * value )
```

Read hardware specific configuration.

##### Parameters

	<i>vbus</i>	V-BUS capability
--	-------------	------------------

## Parameters

	<i>handle</i>	Device handle for the GPIO chip
	<i>pin</i>	GPIO pin number
	<i>func</i>	Hardware specific configuration register to read from. Usually this is an offset to the GPIO chip's base address.
out	<i>value</i>	The configuration value.

## Returns

0 if OK, error code otherwise

Referenced by [L4vbus::Gpio\\_pin::config\\_get\(\)](#).

Here is the caller graph for this function:



## 13.8.4.3.2 l4vbus\_gpio\_config\_pad()

```

int l4vbus_gpio_config_pad (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin,
    unsigned func,
    unsigned value )
  
```

Hardware specific configuration function.

## Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin number
<i>func</i>	Hardware specific configuration register, usually offset to the GPIO chip's base address
<i>value</i>	Value which is written into the hardware specific configuration register for the specified pin

## Returns

0 if OK, error code otherwise

Referenced by [L4vbus::Gpio\\_pin::config\\_pad\(\)](#).

Here is the caller graph for this function:



#### 13.8.4.3.3 l4vbus\_gpio\_config\_pull()

```

int l4vbus_gpio_config_pull (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin,
    unsigned mode )
  
```

Generic function to set pull up/down mode.

##### Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin number
<i>mode</i>	mode for pull up/down resistors, see <a href="#">L4vbus_gpio_pull_modes</a>

##### Returns

0 if OK, error code otherwise

Referenced by [L4vbus::Gpio\\_pin::config\\_pull\(\)](#).

Here is the caller graph for this function:



#### 13.8.4.3.4 l4vbus\_gpio\_get()

```

int l4vbus_gpio_get (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin )
  
```

Read value of GPIO input pin.

## Parameters

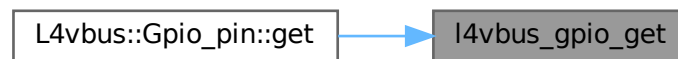
<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin number to read from

## Returns

Value of GPIO pin (usually 0 or 1), negative error code otherwise.

Referenced by [L4vbus::Gpio\\_pin::get\(\)](#).

Here is the caller graph for this function:



## 13.8.4.3.5 l4vbus\_gpio\_multi\_config\_pad()

```

int l4vbus_gpio_multi_config_pad (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned offset,
    unsigned mask,
    unsigned func,
    unsigned value )
  
```

Hardware specific configuration function for multiple GPIO pins.

## Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>offset</i>	Pin corresponding to the LSB in <i>mask</i> . Note: allowed may be hardware specific.
<i>mask</i>	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation.
<i>func</i>	Hardware specific configuration register, usually offset to the GPIO chip's base address.
<i>value</i>	Value which is written into the hardware specific configuration register for the specified pins

## Returns

0 if OK, error code otherwise

Referenced by [L4vbus::Gpio\\_module::config\\_pad\(\)](#).

Here is the caller graph for this function:



#### 13.8.4.3.6 l4vbus\_gpio\_multi\_get()

```

int l4vbus_gpio_multi_get (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned offset,
    unsigned * data )
  
```

Read values of multiple GPIO pins at once.

##### Parameters

	<i>vbus</i>	V-BUS capability
	<i>handle</i>	Device handle for the GPIO chip
	<i>offset</i>	Pin corresponding to the LSB in <i>data</i> . Note: allowed may be hardware specific.
out	<i>data</i>	Each bit returns the value (0 or 1) for the corresponding GPIO pin. The value of pins that are not accessible is undefined.

##### Returns

0 if OK, error code otherwise

Referenced by [L4vbus::Gpio\\_module::get\(\)](#).

Here is the caller graph for this function:



#### 13.8.4.3.7 l4vbus\_gpio\_multi\_set()

```
int l4vbus_gpio_multi_set (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned offset,
    unsigned mask,
    unsigned data )
```

Set multiple GPIO output pins at once.

##### Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>offset</i>	Pin corresponding to the LSB in <i>data</i> . Note: allowed may be hardware specific.
<i>mask</i>	Mask of GPIO pins to set. A bit set to 1 selects this pin. A maximum of 32 pins can be set at once. The real number depends on the hardware and the driver implementation.
<i>data</i>	Each bit corresponds to the GPIO pin in <i>mask</i> . The value of each bit is written to the GPIO pin if its bit in <i>mask</i> is set.

##### Returns

0 if OK, error code otherwise

Referenced by [L4vbus::Gpio\\_module::set\(\)](#).

Here is the caller graph for this function:



#### 13.8.4.3.8 l4vbus\_gpio\_multi\_setup()

```
int l4vbus_gpio_multi_setup (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned offset,
    unsigned mask,
    unsigned mode,
    unsigned value )
```

Configure function of multiple GPIO pins at once.



## Parameters

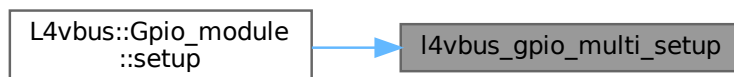
<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>offset</i>	Pin corresponding to the LSB in <i>mask</i> . Note: allowed may be hardware specific.
<i>mask</i>	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation.
<i>mode</i>	GPIO function, see <a href="#">L4vbus_gpio_generic_func</a> for generic functions. Hardware specific functions must be provided in the lower 8 bits.
<i>value</i>	Optional value to set the GPIO pins to if they are configured as output pins

## Returns

0 if OK, error code otherwise

Referenced by [L4vbus::Gpio\\_module::setup\(\)](#).

Here is the caller graph for this function:



## 13.8.4.3.9 l4vbus\_gpio\_set()

```

int l4vbus_gpio_set (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin,
    int value )

```

Set GPIO output pin.

## Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin number to write to
<i>value</i>	Value to write to the GPIO pin (usually 0 or 1)

## Returns

0 if OK, error code otherwise

Referenced by [L4vbus::Gpio\\_pin::set\(\)](#).

Here is the caller graph for this function:



#### 13.8.4.3.10 l4vbus\_gpio\_setup()

```

int l4vbus_gpio_setup (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin,
    unsigned mode,
    int value )
  
```

Configure the function of a GPIO pin.

##### Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin number
<i>mode</i>	GPIO function, see <a href="#">L4vbus_gpio_generic_func</a> for generic functions. Hardware specific functions must be provided in the lower 8 bits.
<i>value</i>	Optional value to set the GPIO pin to if it is configured as an output pin

##### Returns

0 if OK, error code otherwise

Referenced by [L4vbus::Gpio\\_pin::setup\(\)](#).

Here is the caller graph for this function:



#### 13.8.4.3.11 l4vbus\_gpio\_to\_irq()

```
int l4vbus_gpio_to_irq (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin )
```

Create IRQ for GPIO pin.

##### Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin to create an IRQ for.

##### Returns

IRQ number if OK, negative error code otherwise

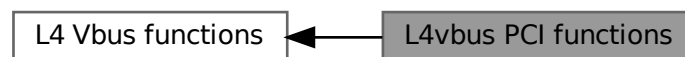
Referenced by [L4vbus::Gpio\\_pin::to\\_irq\(\)](#).

Here is the caller graph for this function:



### 13.8.5 L4vbus PCI functions

Collaboration diagram for L4vbus PCI functions:



## Functions

- `int l4vbus_pci_cfg_read (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width)`  
*Read from the vPCI configuration space using the PCI root bridge.*
- `int l4vbus_pci_cfg_write (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width)`  
*Write to the vPCI configuration space using the PCI root bridge.*
- `int l4vbus_pci_irq_enable (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t bus, l4_uint32_t devfn, int pin, unsigned char *trigger, unsigned char *polarity)`  
*Enable PCI interrupt for a specific device using the PCI root bridge.*
- `int l4vbus_pcidev_cfg_read (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width)`  
*Read from the device's vPCI configuration space.*
- `int l4vbus_pcidev_cfg_write (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width)`  
*Write to the device's vPCI configuration space.*
- `int l4vbus_pcidev_irq_enable (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, unsigned char *trigger, unsigned char *polarity)`  
*Enable the device's PCI interrupt.*

### 13.8.5.1 Detailed Description

### 13.8.5.2 Function Documentation

#### 13.8.5.2.1 l4vbus\_pci\_cfg\_read()

```
int l4vbus_pci_cfg_read (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    l4_uint32_t bus,
    l4_uint32_t devfn,
    l4_uint32_t reg,
    l4_uint32_t * value,
    l4_uint32_t width )
```

Read from the vPCI configuration space using the PCI root bridge.

#### Parameters

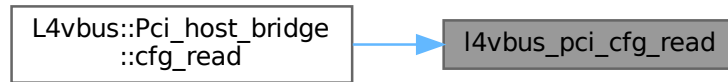
	<i>vbus</i>	Capability of the system bus
	<i>handle</i>	Device handle of the PCI root bridge
	<i>bus</i>	Bus number
	<i>devfn</i>	Device id (upper 16bit) and function (lower 16bit)
	<i>reg</i>	Register in configuration space to read
out	<i>value</i>	Value that has been read
	<i>width</i>	Width to read in bits (e.g. 8, 16, 32)

#### Returns

0 on success, else failure

Referenced by [L4vbus::Pci\\_host\\_bridge::cfg\\_read\(\)](#).

Here is the caller graph for this function:



#### 13.8.5.2.2 l4vbus\_pci\_cfg\_write()

```
int l4vbus_pci_cfg_write (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    l4_uint32_t bus,
    l4_uint32_t devfn,
    l4_uint32_t reg,
    l4_uint32_t value,
    l4_uint32_t width )
```

Write to the vPCI configuration space using the PCI root bridge.

##### Parameters

<i>vbus</i>	Capability of the system bus
<i>handle</i>	Device handle of the PCI root bridge
<i>bus</i>	Bus number
<i>devfn</i>	Device id (upper 16bit) and function (lower 16bit)
<i>reg</i>	Register in configuration space to write
<i>value</i>	Value to write
<i>width</i>	Width to write in bits (e.g. 8, 16, 32)

**Returns**

0 on success, else failure

Referenced by [L4vbus::Pci\\_host\\_bridge::cfg\\_write\(\)](#).

Here is the caller graph for this function:

**13.8.5.2.3 l4vbus\_pci\_irq\_enable()**

```

int l4vbus_pci_irq_enable (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    l4_uint32_t bus,
    l4_uint32_t devfn,
    int pin,
    unsigned char * trigger,
    unsigned char * polarity )
  
```

Enable PCI interrupt for a specific device using the PCI root bridge.

**Parameters**

	<i>vbus</i>	Capability of the system bus
	<i>handle</i>	Device handle of the PCI root bridge
	<i>bus</i>	Bus number
	<i>devfn</i>	Device id (upper 16bit) and function (lower 16bit)
	<i>pin</i>	Interrupt pin (normally as reported in configuration register INTR)
out	<i>trigger</i>	False if interrupt is level-triggered
out	<i>polarity</i>	True if interrupt is of low polarity

**Returns**

On success: Interrupt line to be used, else failure

Referenced by [L4vbus::Pci\\_host\\_bridge::irq\\_enable\(\)](#).

Here is the caller graph for this function:



#### 13.8.5.2.4 l4vbus\_pciddev\_cfg\_read()

```

int l4vbus_pciddev_cfg_read (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    l4_uint32_t reg,
    l4_uint32_t * value,
    l4_uint32_t width )
  
```

Read from the device's vPCI configuration space.

##### Parameters

	<i>vbus</i>	Capability of the system bus
	<i>handle</i>	Device handle of the PCI device
	<i>reg</i>	Register in configuration space to read
out	<i>value</i>	Value that has been read
	<i>width</i>	Width to read in bits (e.g. 8, 16, 32)

##### Returns

0 on success, else failure

Referenced by [L4vbus::Pci\\_dev::cfg\\_read\(\)](#).

Here is the caller graph for this function:



### 13.8.5.2.5 l4vbus\_pciddev\_cfg\_write()

```
int l4vbus_pciddev_cfg_write (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    l4_uint32_t reg,
    l4_uint32_t value,
    l4_uint32_t width )
```

Write to the device's vPCI configuration space.

#### Parameters

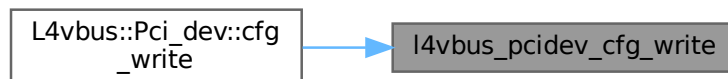
<i>vbus</i>	Capability of the system bus
<i>handle</i>	Device handle of the PCI device
<i>reg</i>	Register in configuration space to write
<i>value</i>	Value to write
<i>width</i>	Width to write in bits (e.g. 8, 16, 32)

#### Returns

0 on success, else failure

Referenced by [L4vbus::Pci\\_dev::cfg\\_write\(\)](#).

Here is the caller graph for this function:



### 13.8.5.2.6 l4vbus\_pciddev\_irq\_enable()

```
int l4vbus_pciddev_irq_enable (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned char * trigger,
    unsigned char * polarity )
```

Enable the device's PCI interrupt.

#### Parameters

	<i>vbus</i>	Capability of the system bus
	<i>handle</i>	Device handle of the PCI device
out	<i>trigger</i>	False if interrupt is level-triggered
out	<i>polarity</i>	True if interrupt is of low polarity

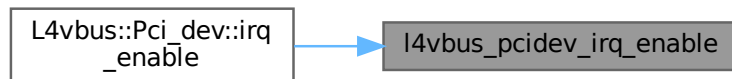


**Returns**

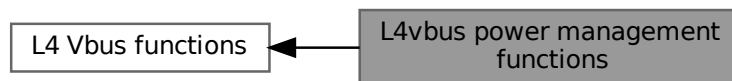
On success: Interrupt line to be used, else failure

Referenced by [L4vbus::Pci\\_dev::irq\\_enable\(\)](#).

Here is the caller graph for this function:

**13.8.6 L4vbus power management functions**

Collaboration diagram for L4vbus power management functions:

**Functions**

- `int l4vbus_pm_suspend (l4_cap_idx_t vbus, l4vbus_device_handle_t handle)`  
*Suspend the device.*
- `int l4vbus_pm_resume (l4_cap_idx_t vbus, l4vbus_device_handle_t handle)`  
*Resume the device.*

**13.8.6.1 Detailed Description****13.8.6.2 Function Documentation****13.8.6.2.1 l4vbus\_pm\_resume()**

```

int l4vbus_pm_resume (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle )
  
```

Resume the device.

**Parameters**

<i>vbus</i>	V-BUS capability
<i>handle</i>	Handle for the device to be resumed

Switches the device from low-power mode to normal operation and restores the saved state.

**Return values**

0	Success.
---	----------

Referenced by [L4vbus::Pm< DEC >::pm\\_resume\(\)](#).

Here is the caller graph for this function:

**13.8.6.2.2 l4vbus\_pm\_suspend()**

```
int l4vbus_pm_suspend (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle )
```

Suspend the device.

**Parameters**

<i>vbus</i>	V-BUS capability
<i>handle</i>	Handle for the device to be suspended

Saves the state of the device and puts it into a low-power mode.

**Return values**

0	Success.
---	----------

Referenced by [L4vbus::Pm< DEC >::pm\\_suspend\(\)](#).

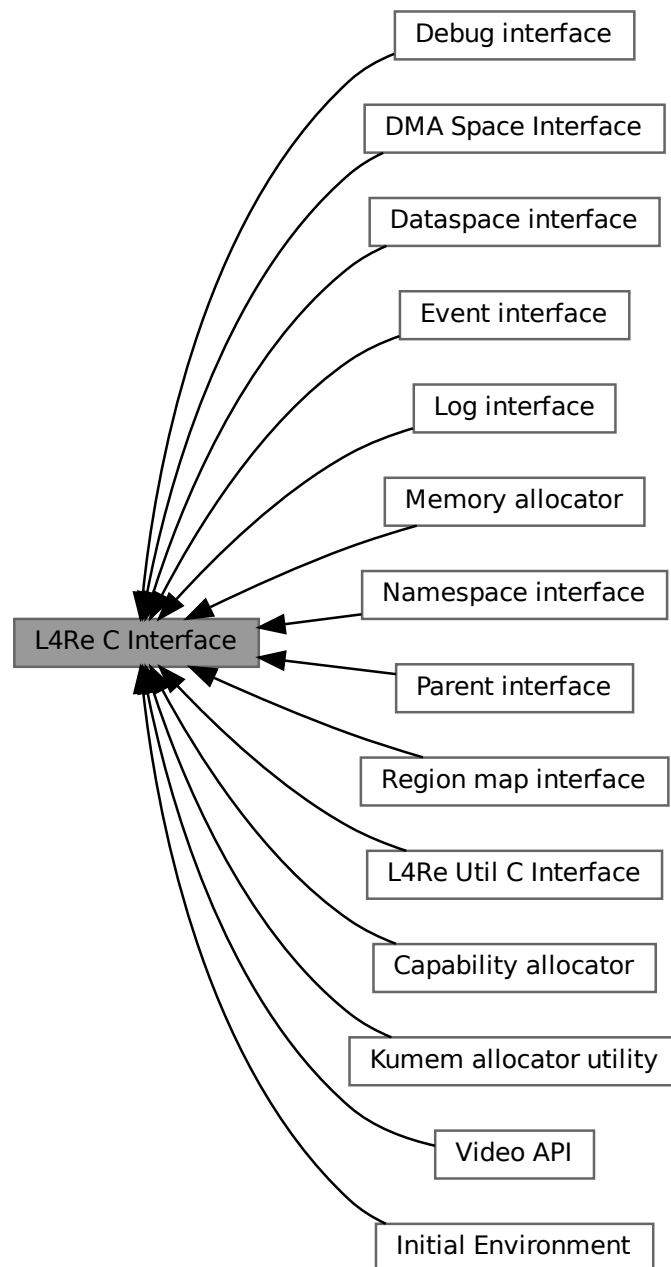
Here is the caller graph for this function:



## 13.9 L4Re C Interface

Documentation for the [L4Re](#) C Interface.

Collaboration diagram for L4Re C Interface:



## Modules

- [Capability allocator](#)  
*Capability allocator C interface.*
- [DMA Space Interface](#)  
*DMA Space C interface.*
- [Dataspace interface](#)

*Dataspace C interface.*

- [Debug interface](#)
- [Event interface](#)

*Event C interface.*

- [Initial Environment](#)

*C interface of the initial environment that is provided to an [L4](#) task.*

- [Kumem allocator utility](#)

*Kumem allocator utility C interface.*

- [L4Re Util C Interface](#)

*Documentation of the [L4](#) Runtime Environment utility functionality in C.*

- [Log interface](#)

*Log C interface.*

- [Memory allocator](#)

*Memory allocator C interface.*

- [Namespace interface](#)

*Namespace C interface.*

- [Parent interface](#)

- [Region map interface](#)

*Region map C interface.*

- [Video API](#)

## Files

- file [inhibitor.h](#)

*Inhibitor C interface.*

### 13.9.1 Detailed Description

Documentation for the [L4Re](#) C Interface.

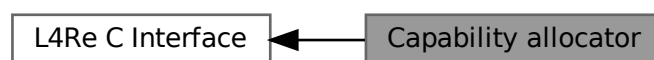
The interface functions closely align with the C++ functions and add no further functionalities.

For new programs it is advised to use the C++ interface.

### 13.9.2 Capability allocator

Capability allocator C interface.

Collaboration diagram for Capability allocator:



## Functions

- [l4\\_cap\\_idx\\_t](#) **l4re\_util\_cap\_alloc** (void) [L4\\_NOTHROW](#)  
*Get free capability index at capability allocator.*
- void **l4re\_util\_cap\_free** ([l4\\_cap\\_idx\\_t](#) cap) [L4\\_NOTHROW](#)  
*Return capability index to capability allocator.*
- void **l4re\_util\_cap\_free\_um** ([l4\\_cap\\_idx\\_t](#) cap) [L4\\_NOTHROW](#)  
*Return capability index to capability allocator, and unmaps the object.*
- long **l4re\_util\_cap\_last** (void) [L4\\_NOTHROW](#)  
*Return last capability index the allocator can return.*

### 13.9.2.1 Detailed Description

Capability allocator C interface.

### 13.9.2.2 Function Documentation

#### 13.9.2.2.1 l4re\_util\_cap\_last()

```
long l4re_util_cap_last (
    void )
```

Return last capability index the allocator can return.

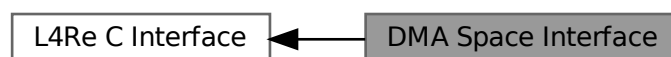
#### Returns

last/biggest capability index the allocator can return

### 13.9.3 DMA Space Interface

DMA Space C interface.

Collaboration diagram for DMA Space Interface:



## Typedefs

- typedef [l4\\_cap\\_idx\\_t](#) **l4re\_dma\_space\_t**  
*DMA space capability type.*

## Functions

- long `l4re_dma_space_map` (`l4re_dma_space_t` dma, `l4re_ds_t` src, `l4re_ds_offset_t` offset, `l4_size_t` \*size, unsigned long attrs, enum `l4re_dma_space_direction` dir, `l4re_dma_space_dma_addr_t` \*dma\_addr) `L4_NOTHROW`  
*Map the given part of this data space into the DMA address space.*
- long `l4re_dma_space_unmap` (`l4re_dma_space_t` dma, `l4re_dma_space_dma_addr_t` dma\_addr, `l4_size_t` size, unsigned long attrs, enum `l4re_dma_space_direction` dir) `L4_NOTHROW`  
*Unmap the given part of this data space from the DMA address space.*
- long `l4re_dma_space_associate` (`l4re_dma_space_t` dma, `l4_cap_idx_t` dma\_task, unsigned long attr) `L4_NOTHROW`  
*Associate a (kernel) DMA space for a device to this Dma\_space.*
- long `l4re_dma_space_disassociate` (`l4re_dma_space_t` dma)  
*Disassociate the (kernel) DMA space from this Dma\_space.*

### 13.9.3.1 Detailed Description

DMA Space C interface.

### 13.9.3.2 Typedef Documentation

#### 13.9.3.2.1 `l4re_dma_space_t`

```
typedef l4_cap_idx_t l4re_dma_space_t
```

DMA space capability type.

Managed DMA Address Space.

A managed `Dma_space` represents the [L4Re](#) abstraction of an DMA address space of one or several devices. Devices are assigned to a managed `Dma_space` by binding the `Dma_space` to the respective DMA domain (see [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#)), which might link the `Dma_space` with a kernel [DMA space](#). Note that several DMA domains can be bound to the same `Dma_space`. Whenever a device needs direct access to parts of an [L4Re::Dataspace](#), that part of the data space must be mapped to the managed `Dma_space` that is assigned to that device. Binding to DMA domains must happen before mapping. After the DMA accesses to the memory are finished the memory must be unmapped from the device's DMA address space.

Mapping to a managed DMA address space, using `map()`, makes the given parts of the data space visible to the associated device at the returned DMA address. As long as the memory is mapped into a DMA space it is 'pinned' and cannot be subject to dynamic memory management such as swapping. Additionally, `map()` is responsible for the necessary syncing operations before the DMA.

`unmap()` is the reverse operation to `map()` and unmaps the given data-space part for the DMA address space. `unmap()` is responsible for the necessary sync operations after the DMA.

Definition at line 59 of file [dma\\_space.h](#).

### 13.9.3.3 Function Documentation

#### 13.9.3.3.1 `l4re_dma_space_associate()`

```
long l4re_dma_space_associate (
    l4re_dma_space_t dma,
    l4_cap_idx_t dma_task,
    unsigned long attr )
```

Associate a (kernel) [DMA space](#) for a device to this `Dma_space`.

## Parameters

	<i>dma</i>	DMA space capability
in	<i>dma_task</i>	The (kernel) <a href="#">DMA space</a> used for the device that shall be associated with this DMA space. In case no IOMMU is present or configured, the <i>dma_task</i> might be an invalid capability when <a href="#">L4Re::Dma_space::Phys_space</a> is set in <i>attr</i> , in this case the CPUs physical memory is used as DMA address space.
in	<i>attr</i>	Attributes for this DMA space. See <a href="#">L4Re::Dma_space::Space_attrib</a> .

## Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EPERM</i>	No <a href="#">L4_CAP_FPAGE_W</a> right on the <i>Dma_space</i> capability.
<i>-L4_EINVAL</i>	
<i>-L4_ENOENT</i>	

## Precondition

requires capability rights: {RW}

13.9.3.3.2 [l4re\\_dma\\_space\\_disassociate\(\)](#)

```
long l4re_dma_space_disassociate (
    l4re\_dma\_space\_t dma )
```

Disassociate the (kernel) [DMA space](#) from this *Dma\_space*.

## Parameters

<i>dma</i>	DMA space capability
------------	----------------------

## Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EPERM</i>	No <a href="#">L4_CAP_FPAGE_W</a> right on the <i>Dma_space</i> capability.
<i>-L4_ENOENT</i>	

## Precondition

requires capability rights: {RW}

13.9.3.3.3 [l4re\\_dma\\_space\\_map\(\)](#)

```
long l4re_dma_space_map (
    l4re\_dma\_space\_t dma,
    l4re\_ds\_t src,
```



```

l4re_ds_offset_t offset,
l4_size_t * size,
unsigned long attrs,
enum l4re_dma_space_direction dir,
l4re_dma_space_dma_addr_t * dma_addr )

```

Map the given part of this data space into the DMA address space.

#### Parameters

	<i>dma</i>	DMA space capability
in	<i>src</i>	Source data space (that describes the memory). Caller needs write right to the data space.
in	<i>offset</i>	The offset (bytes) within <i>src</i> .
in, out	<i>size</i>	The size (bytes) of the region to be mapped for DMA, after successful mapping the size returned is the size mapped for DMA as a single block. This size might be smaller than the original input size, in this case the caller might call <code>map()</code> again with a new offset and the remaining size.
in	<i>attrs</i>	The attributes used for this DMA mapping (a combination of <code>Dma_space::Attribute</code> values).
in	<i>dir</i>	The direction of the DMA transfer issued with this mapping. The same value must later be passed to <code>unmap()</code> .
out	<i>dma_addr</i>	The DMA address to use for DMA with the associated device.

#### Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EPERM</i>	No <code>L4_CAP_FPAGE_W</code> right on <i>src</i> capability.
<i>-L4_EINVAL</i>	The <i>src</i> capability is invalid or does not refer to a valid dataspace.
<i>-L4_EEXIST</i>	The specified region overlaps an existing mapping.
<i>-L4_ENOMEM</i>	Not enough memory to allocate internal datastructures.
<i>-L4_ERANGE</i>	<i>offset</i> is larger than the size of the dataspace.

#### Precondition

requires capability rights: {R}

#### Note

`associate()` must be called prior to mapping memory. Usually this is done implicitly when binding the managed `Dma_space` to a DMA domain (see `L4vbus::Vbus::assign_dma_domain()`).

#### 13.9.3.3.4 l4re\_dma\_space\_unmap()

```

long l4re_dma_space_unmap (
    l4re_dma_space_t dma,
    l4re_dma_space_dma_addr_t dma_addr,
    l4_size_t size,
    unsigned long attrs,
    enum l4re_dma_space_direction dir )

```

Unmap the given part of this data space from the DMA address space.

**Parameters**

<i>dma</i>	DMA space capability
<i>dma_addr</i>	The DMA address (returned by <code>Dma_space::map()</code> ).
<i>size</i>	The size (bytes) of the memory region to unmap.
<i>attrs</i>	The attributes for the unmap (currently none).
<i>dir</i>	The direction of the finished DMA operation.

**Returns**

0 in the case of success, a negative error code otherwise.

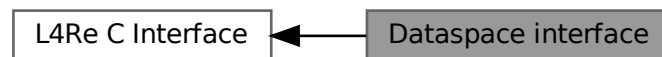
**Precondition**

requires capability rights: {R}

**13.9.4 Dataspace interface**

Dataspace C interface.

Collaboration diagram for Dataspace interface:

**Data Structures**

- struct [l4re\\_ds\\_stats\\_t](#)  
*Information about the data space.*

**Typedefs**

- typedef [l4\\_cap\\_idx\\_t](#) **l4re\_ds\_t**  
*Dataspace type.*

**Enumerations**

- enum [l4re\\_ds\\_map\\_flags](#) { }  
*Flags to specify the memory mapping type of a request.*

## Functions

- long [l4re\\_ds\\_clear](#) ([l4re\\_ds\\_t](#) ds, [l4re\\_ds\\_offset\\_t](#) offset, [l4re\\_ds\\_size\\_t](#) size) [L4\\_NOTHROW](#)  
*Clear parts of a dataspace.*
- long [l4re\\_ds\\_allocate](#) ([l4re\\_ds\\_t](#) ds, [l4re\\_ds\\_offset\\_t](#) offset, [l4re\\_ds\\_size\\_t](#) size) [L4\\_NOTHROW](#)  
*Allocate a range in the dataspace.*
- int [l4re\\_ds\\_copy\\_in](#) ([l4re\\_ds\\_t](#) ds, [l4re\\_ds\\_offset\\_t](#) dst\_offs, [l4re\\_ds\\_t](#) src, [l4re\\_ds\\_offset\\_t](#) src\_offs, [l4re\\_ds\\_size\\_t](#) size) [L4\\_NOTHROW](#)  
*Copy contents from another dataspace.*
- [l4re\\_ds\\_size\\_t](#) [l4re\\_ds\\_size](#) ([l4re\\_ds\\_t](#) ds) [L4\\_NOTHROW](#)  
*Get size of a dataspace.*
- [l4re\\_ds\\_flags\\_t](#) [l4re\\_ds\\_flags](#) ([l4re\\_ds\\_t](#) ds) [L4\\_NOTHROW](#)  
*Get flags of the dataspace.*
- int [l4re\\_ds\\_info](#) ([l4re\\_ds\\_t](#) ds, [l4re\\_ds\\_stats\\_t](#) \*stats) [L4\\_NOTHROW](#)  
*Get information on the dataspace.*
- int [l4re\\_ds\\_map\\_info](#) ([l4re\\_ds\\_t](#) ds, [l4\\_addr\\_t](#) \*start\_addr, [l4\\_addr\\_t](#) \*end\_addr) [L4\\_NOTHROW](#)  
*Get mapping range of dataspace.*

### 13.9.4.1 Detailed Description

Dataspace C interface.

### 13.9.4.2 Enumeration Type Documentation

#### 13.9.4.2.1 l4re\_ds\_map\_flags

```
enum l4re\_ds\_map\_flags
```

Flags to specify the memory mapping type of a request.

Enumerator

<a href="#">L4RE_DS_F_NORMAL</a>	request normal memory mapping
<a href="#">L4RE_DS_F_CACHEABLE</a>	request normal memory mapping
<a href="#">L4RE_DS_F_BUFFERABLE</a>	request bufferable (write buffered) mappings
<a href="#">L4RE_DS_F_UNCACHEABLE</a>	request uncacheable memory mappings
<a href="#">L4RE_DS_F_CACHING_MASK</a>	mask for caching flags
<a href="#">L4RE_DS_F_CACHING_SHIFT</a>	shift value for caching flags

Definition at line 58 of file [dataspace.h](#).

### 13.9.4.3 Function Documentation

#### 13.9.4.3.1 l4re\_ds\_allocate()

```
long l4re\_ds\_allocate (  
    l4re\_ds\_t ds,
```

```

l4re_ds_offset_t offset,
l4re_ds_size_t size )

```

Allocate a range in the dataspace.

#### Parameters

<i>ds</i>	Dataspace capability.
<i>offset</i>	Offset in the dataspace, in bytes.
<i>size</i>	Size of the range, in bytes.

#### Return values

<i>L4_EOK</i>	Success
<i>-L4_ERANGE</i>	Given range is outside the dataspace. (A dataspace provider may also silently ignore areas outside the dataspace.)
<i>-L4_ENOMEM</i>	Not enough memory available.
<i>&lt;0</i>	IPC errors

On success, at least the given range is guaranteed to be allocated. The dataspace manager may also allocate more memory due to page granularity.

The memory is allocated with the same rights as the dataspace capability.

#### 13.9.4.3.2 l4re\_ds\_clear()

```

long l4re_ds_clear (
    l4re_ds_t ds,
    l4re_ds_offset_t offset,
    l4re_ds_size_t size )

```

Clear parts of a dataspace.

#### Parameters

<i>ds</i>	Dataspace capability.
<i>offset</i>	Offset within dataspace (in bytes).
<i>size</i>	Size of region to clear (in bytes).

#### Return values

<i>&gt;=0</i>	Success.
<i>-L4_ERANGE</i>	Given range is outside the dataspace. (A dataspace provider may also silently ignore areas outside the dataspace.)
<i>-L4_EACCESS</i>	No <a href="#">L4_CAP_FPAGE_W</a> right on dataspace capability.
<i>&lt;0</i>	IPC errors

Zeroes out the memory. Depending on the type of memory the memory could also be deallocated and replaced by a shared zero-page.

**13.9.4.3.3 l4re\_ds\_copy\_in()**

```
int l4re_ds_copy_in (
    l4re_ds_t ds,
    l4re_ds_offset_t dst_offs,
    l4re_ds_t src,
    l4re_ds_offset_t src_offs,
    l4re_ds_size_t size )
```

Copy contents from another dataspace.

**Parameters**

<i>ds</i>	Destination dataspace.
<i>dst_offs</i>	Offset in destination dataspace.
<i>src</i>	Source dataspace to copy from.
<i>src_offs</i>	Offset in the source dataspace.
<i>size</i>	Size to copy (in bytes).

**Return values**

<i>L4_EOK</i>	Success
<i>-L4_EACCESS</i>	No <a href="#">L4_CAP_FPAGE_W</a> right on the destination dataspace.
<i>-L4_EINVAL</i>	Invalid parameter supplied.
<i>&lt;0</i>	IPC errors

The copy operation may use copy-on-write mechanisms. The operation may also fail if both dataspace managers do not cooperate.

**13.9.4.3.4 l4re\_ds\_flags()**

```
l4re_ds_flags_t l4re_ds_flags (
    l4re_ds_t ds )
```

Get flags of the dataspace.

**Parameters**

<i>ds</i>	Dataspace capability.
-----------	-----------------------

**Return values**

<i>&gt;=0</i>	Flags of the dataspace
<i>&lt;0</i>	IPC errors

See also

[L4Re::Dataspace::F::Flags](#)

### 13.9.4.3.5 l4re\_ds\_info()

```
int l4re_ds_info (
    l4re_ds_t ds,
    l4re_ds_stats_t * stats )
```

Get information on the dataspace.

#### Parameters

	<i>ds</i>	Dataspace capability.
out	<i>stats</i>	Dataspace information

#### Return values

0	Success
<0	IPC errors

### 13.9.4.3.6 l4re\_ds\_map\_info()

```
int l4re_ds_map_info (
    l4re_ds_t ds,
    l4_addr_t * start_addr,
    l4_addr_t * end_addr )
```

Get mapping range of dataspace.

#### Parameters

<i>ds</i>	Dataspace capability.
-----------	-----------------------

In case of a MMU-less system, the dataspace must be mapped at the correct address in the task because virtual and physical address must match. This method returns the start and end address of the physically contiguous buffer backing the dataspace.

On MMU-enabled system any page aligned address is permissible. On such systems the method is just a stub.

#### Parameters

out	<i>start_addr</i>	Start address of dataspace.
out	<i>end_addr</i>	End address (inclusive) of dataspace.

#### Return values

>0	Start/end address have been set and need to be obeyed.
0	No constraint of mapping address.
-L4_EPERM	Cannot infer mapping address. Dataspace not mappable.
<0	IPC errors.

### 13.9.4.3.7 l4re\_ds\_size()

```
l4re_ds_size_t l4re_ds_size (
    l4re_ds_t ds )
```

Get size of a dataspace.

#### Parameters

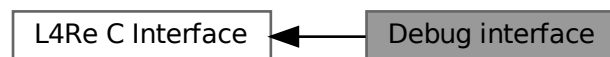
<i>ds</i>	Dataspace capability.
-----------	-----------------------

#### Returns

Size of the dataspace in bytes.

## 13.9.5 Debug interface

Collaboration diagram for Debug interface:



### Functions

- long [l4re\\_debug\\_obj\\_debug](#) ([l4\\_cap\\_idx\\_t](#) *srv*, unsigned long *function*) [L4\\_NOTHROW](#)  
Call debug function of [L4Re](#) service.

#### 13.9.5.1 Detailed Description

#### 13.9.5.2 Function Documentation

##### 13.9.5.2.1 l4re\_debug\_obj\_debug()

```
long l4re_debug_obj_debug (
    l4_cap_idx_t srv,
    unsigned long function )
```

Call debug function of [L4Re](#) service.

#### Parameters

<i>srv</i>	Object to call.
<i>function</i>	Function to call.

See also

[L4Re::Debug\\_obj::debug](#)

### 13.9.6 Event interface

Event C interface.

Collaboration diagram for Event interface:



#### Functions

- long [l4re\\_event\\_get\\_buffer](#) (const [l4\\_cap\\_idx\\_t](#) server, const [l4re\\_ds\\_t](#) ds) [L4\\_NOTHROW](#)  
*Get an event signal buffer.*
- long [l4re\\_event\\_get\\_num\\_streams](#) (const [l4\\_cap\\_idx\\_t](#) server) [L4\\_NOTHROW](#)  
*Get number of streams.*
- long [l4re\\_event\\_get\\_stream\\_info](#) (const [l4\\_cap\\_idx\\_t](#) server, int idx, [l4re\\_event\\_stream\\_info\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Get information on a stream.*
- long [l4re\\_event\\_get\\_stream\\_info\\_for\\_id](#) (const [l4\\_cap\\_idx\\_t](#) server, [l4\\_umword\\_t](#) stream\_id, [l4re\\_event\\_stream\\_info\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Get info for a stream given a stream id.*
- long [l4re\\_event\\_get\\_axis\\_info](#) (const [l4\\_cap\\_idx\\_t](#) server, [l4\\_umword\\_t](#) id, unsigned naxes, unsigned const \*axis, [l4re\\_event\\_absinfo\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Get Axis information for a stream.*

#### 13.9.6.1 Detailed Description

Event C interface.

#### 13.9.6.2 Function Documentation

##### 13.9.6.2.1 l4re\_event\_get\_axis\_info()

```

long l4re_event_get_axis_info (
    const l4\_cap\_idx\_t server,
    l4\_umword\_t id,
    unsigned naxes,
    unsigned const * axis,
    l4re\_event\_absinfo\_t * info )
  
```

Get Axis information for a stream.



## Parameters

	<i>server</i>	Server to talk to.
	<i>id</i>	Id of the stream to get information from.
	<i>naxes</i>	Number of axes in <i>axis</i> array.
in	<i>axis</i>	Array of axis IDs whose information should be retrieved.
out	<i>info</i>	Information buffer to store the retrieved axis infos.

## Return values

0	Success
<0	Error

## See also

[L4Re::Event::get\\_axis\\_info](#)**13.9.6.2.2 l4re\_event\_get\_buffer()**

```
long l4re_event_get_buffer (
    const l4_cap_idx_t server,
    const l4re_ds_t ds )
```

Get an event signal buffer.

## Parameters

<i>server</i>	Server to talk to.
<i>ds</i>	Buffer to event data.

## Returns

0 for success, <0 on error

## See also

[L4Re::Event::get\\_buffer](#)**13.9.6.2.3 l4re\_event\_get\_num\_streams()**

```
long l4re_event_get_num_streams (
    const l4_cap_idx_t server )
```

Get number of streams.

## Parameters

<i>server</i>	Server to talk to.
---------------	--------------------

**Returns**

0 for success, <0 on error

**See also**

[L4Re::Event::get\\_num\\_streams](#)

**13.9.6.2.4 l4re\_event\_get\_stream\_info()**

```
long l4re_event_get_stream_info (
    const l4_cap_idx_t server,
    int idx,
    l4re_event_stream_info_t * info )
```

Get information on a stream.

**Parameters**

	<i>server</i>	Server to talk to.
	<i>idx</i>	Index value.
out	<i>info</i>	Information buffer.

**Returns**

0 for success, <0 on error

**See also**

[L4Re::Event::get\\_stream\\_info](#)

**13.9.6.2.5 l4re\_event\_get\_stream\_info\_for\_id()**

```
long l4re_event_get_stream_info_for_id (
    const l4_cap_idx_t server,
    l4_umword_t stream_id,
    l4re_event_stream_info_t * info )
```

Get info for a stream given a stream id.

**Parameters**

	<i>server</i>	Server to talk to.
	<i>stream_id</i>	Stream ID.
out	<i>info</i>	Information buffer.

**Returns**

0 for success, <0 on error

**See also**

[L4Re::Event::get\\_stream\\_info\\_for\\_id](#)

**13.9.7 Initial Environment**

C interface of the initial environment that is provided to an [L4](#) task.

Collaboration diagram for Initial Environment:

**Data Structures**

- struct [l4re\\_env\\_cap\\_entry\\_t](#)  
*Entry in the [L4Re](#) environment array for the named initial objects.*

**Typedefs**

- typedef struct [l4re\\_env\\_cap\\_entry\\_t](#) [l4re\\_env\\_cap\\_entry\\_t](#)  
*Entry in the [L4Re](#) environment array for the named initial objects.*

**Functions**

- [l4re\\_env\\_t](#) \* [l4re\\_env](#) (void) [L4\\_NOTHROW](#)  
*Get [L4Re](#) initial environment.*
- [l4\\_kernel\\_info\\_t](#) const \* [l4re\\_kip](#) (void) [L4\\_NOTHROW](#)  
*Get Kernel Info Page.*
- [l4\\_cap\\_idx\\_t](#) [l4re\\_env\\_get\\_cap](#) (char const \*name) [L4\\_NOTHROW](#)  
*Get the capability selector for the object named name.*
- [l4\\_cap\\_idx\\_t](#) [l4re\\_env\\_get\\_cap\\_e](#) (char const \*name, [l4re\\_env\\_t](#) const \*e) [L4\\_NOTHROW](#)  
*Get the capability selector for the object named name.*
- [l4re\\_env\\_cap\\_entry\\_t](#) const \* [l4re\\_env\\_get\\_cap\\_l](#) (char const \*name, unsigned l, [l4re\\_env\\_t](#) const \*e) [L4\\_NOTHROW](#)  
*Get the full [l4re\\_env\\_cap\\_entry\\_t](#) for the object named name.*

### 13.9.7.1 Detailed Description

C interface of the initial environment that is provided to an [L4](#) task.

#### Include File

```
#include <l4/re/env.h>
```

For an explanation of the default task capabilities see [l4\\_default\\_caps\\_t](#).

For the C++ interface refer to [L4Re::Env](#).

### 13.9.7.2 Function Documentation

#### 13.9.7.2.1 l4re\_env()

```
l4re_env_t * l4re_env (
    void ) [inline]
```

Get [L4Re](#) initial environment.

#### Returns

Pointer to [L4Re](#) initial environment.

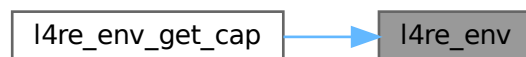
#### Examples

[examples/sys/aliens/main.c](#), [examples/sys/isr/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#),  
and [examples/sys/utcb-ipc/main.c](#).

Definition at line [190](#) of file [env.h](#).

Referenced by [l4re\\_env\\_get\\_cap\(\)](#).

Here is the caller graph for this function:



#### 13.9.7.2.2 l4re\_env\_get\_cap()

```
l4_cap_idx_t l4re_env_get_cap (
    char const * name ) [inline]
```

Get the capability selector for the object named *name*.

## Parameters

<i>name</i>	is the name of the object to lookup in the initial objects.
-------------	---

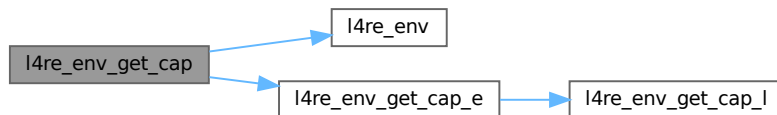
## Returns

A valid capability selector if the object exists or an invalid capability selector if not ([l4\\_is\\_invalid\\_cap\(\)](#)).

Definition at line 229 of file [env.h](#).

References [l4re\\_env\(\)](#), and [l4re\\_env\\_get\\_cap\\_e\(\)](#).

Here is the call graph for this function:



## 13.9.7.2.3 l4re\_env\_get\_cap\_e()

```

l4_cap_idx_t l4re_env_get_cap_e (
    char const * name,
    l4re_env_t const * e ) [inline]
  
```

Get the capability selector for the object named *name*.

## Parameters

<i>name</i>	is the name of the object to lookup in the initial objects.
<i>e</i>	is the environment structure to use for the operation.

## Returns

A valid capability selector if the object exists or an invalid capability selector if not ([l4\\_is\\_invalid\\_cap\(\)](#)).

Definition at line 216 of file [env.h](#).

References [l4re\\_env\\_cap\\_entry\\_t::cap](#), [L4\\_INVALID\\_CAP](#), and [l4re\\_env\\_get\\_cap\\_l\(\)](#).

Referenced by [l4re\\_env\\_get\\_cap\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.9.7.2.4 l4re\_env\_get\_cap\_l()

```

l4re_env_cap_entry_t const * l4re_env_get_cap_l (
    char const * name,
    unsigned l,
    l4re_env_t const * e ) [inline]
  
```

Get the full [l4re\\_env\\_cap\\_entry\\_t](#) for the object named *name*.

##### Parameters

<i>name</i>	is the name of the object to lookup in the initial objects.
<i>l</i>	is the length of the name string, thus <i>name</i> might not be zero terminated.
<i>e</i>	is the environment structure to use for the operation.

##### Returns

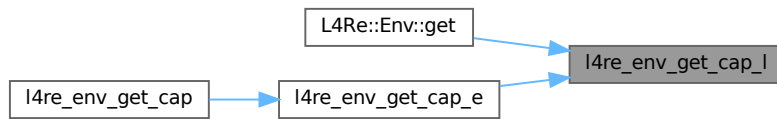
A pointer to an [l4re\\_env\\_cap\\_entry\\_t](#) if the object exists or NULL if not.

Definition at line 198 of file [env.h](#).

References [l4re\\_env\\_cap\\_entry\\_t::flags](#), and [l4re\\_env\\_cap\\_entry\\_t::name](#).

Referenced by [L4Re::Env::get\(\)](#), and [l4re\\_env\\_get\\_cap\\_e\(\)](#).

Here is the caller graph for this function:



#### 13.9.7.2.5 l4re\_kip()

```
l4_kernel_info_t const * l4re_kip (
    void ) [inline]
```

Get Kernel Info Page.

##### Returns

Pointer to Kernel Info Page (KIP) structure.

##### Examples

[examples/libs/shmc/prodcons.c](#), [examples/sys/aliens/main.c](#), and [examples/sys/ux-vhw/main.c](#).

Definition at line 194 of file [env.h](#).

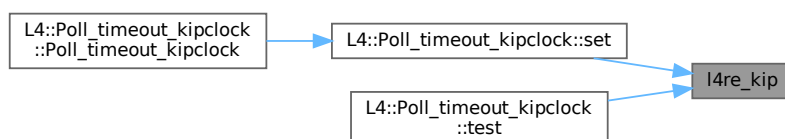
References [l4\\_kip\(\)](#).

Referenced by [L4::Poll\\_timeout\\_kipclock::set\(\)](#), and [L4::Poll\\_timeout\\_kipclock::test\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 13.9.8 Kumem allocator utility

Kumem allocator utility C interface.

Collaboration diagram for Kumem allocator utility:

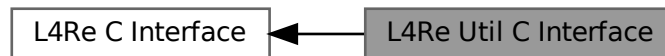


Kumem allocator utility C interface.

### 13.9.9 L4Re Util C Interface

Documentation of the [L4](#) Runtime Environment utility functionality in C.

Collaboration diagram for L4Re Util C Interface:



Documentation of the [L4](#) Runtime Environment utility functionality in C.

The interface functions closely align with the C++ functions and add no further functionalities.

For new programs it is advised to use the C++ interface.

### 13.9.10 Log interface

Log C interface.

Collaboration diagram for Log interface:





## Functions

- void [l4re\\_log\\_print](#) (char const \*string) [L4\\_NOTHROW](#)  
*Write a null terminated string to the default log.*
- void [l4re\\_log\\_printn](#) (char const \*string, int len) [L4\\_NOTHROW](#)  
*Write a string of a given length to the default log.*
- void [l4re\\_log\\_print\\_srv](#) (const [l4\\_cap\\_idx\\_t](#) logcap, char const \*string) [L4\\_NOTHROW](#)  
*Write a null terminated string to a log.*
- void [l4re\\_log\\_printn\\_srv](#) (const [l4\\_cap\\_idx\\_t](#) logcap, char const \*string, int len) [L4\\_NOTHROW](#)  
*Write a string of a given length to a log.*

### 13.9.10.1 Detailed Description

Log C interface.

### 13.9.10.2 Function Documentation

#### 13.9.10.2.1 l4re\_log\_print()

```
void l4re_log_print (
    char const * string ) [inline]
```

Write a null terminated string to the default log.

#### Parameters

<i>string</i>	Text to print, null terminated.
---------------	---------------------------------

#### See also

[L4Re::Log::print](#)

Definition at line 91 of file [log.h](#).

References [l4re\\_log\\_print\\_srv\(\)](#), and [l4re\\_env\\_t::log](#).

Here is the call graph for this function:



### 13.9.10.2.2 l4re\_log\_print\_srv()

```
void l4re_log_print_srv (
    const l4_cap_idx_t logcap,
    char const * string )
```

Write a null terminated string to a log.

#### Parameters

<i>logcap</i>	Log capability (service).
<i>string</i>	Text to print, null terminated.

#### See also

[L4Re::Log::print](#)

Referenced by [l4re\\_log\\_print\(\)](#).

Here is the caller graph for this function:



### 13.9.10.2.3 l4re\_log\_printn()

```
void l4re_log_printn (
    char const * string,
    int len ) [inline]
```

Write a string of a given length to the default log.

#### Parameters

<i>string</i>	Text to print, null terminated.
<i>len</i>	Length of string in bytes.

#### See also

[L4Re::Log::println](#)

Definition at line 97 of file [log.h](#).

References [l4re\\_log\\_printn\\_srv\(\)](#), and [l4re\\_env\\_t::log](#).

Here is the call graph for this function:



#### 13.9.10.2.4 l4re\_log\_printn\_srv()

```

void l4re_log_printn_srv (
    const l4_cap_idx_t logcap,
    char const * string,
    int len )
  
```

Write a string of a given length to a log.

##### Parameters

<i>logcap</i>	Log capability (service).
<i>string</i>	Text to print, null terminated.
<i>len</i>	Length of string in bytes.

##### See also

[L4Re::Log::printn](#)

Referenced by [l4re\\_log\\_printn\(\)](#).

Here is the caller graph for this function:



### 13.9.11 Memory allocator

Memory allocator C interface.

Collaboration diagram for Memory allocator:



## Enumerations

- enum [l4re\\_ma\\_flags](#)  
*Flags for requesting memory at the memory allocator.*

## Functions

- long [l4re\\_ma\\_alloc](#) (long size, [l4re\\_ds\\_t](#) const mem, unsigned long flags) [L4\\_NOTHROW](#)  
*Allocate memory.*
- long [l4re\\_ma\\_alloc\\_align](#) (long size, [l4re\\_ds\\_t](#) const mem, unsigned long flags, unsigned long align) [L4\\_NOTHROW](#)  
*Allocate memory.*
- long [l4re\\_ma\\_alloc\\_align\\_srv](#) ([l4\\_cap\\_idx\\_t](#) srv, long size, [l4re\\_ds\\_t](#) const mem, unsigned long flags, unsigned long align) [L4\\_NOTHROW](#)  
*Allocate memory.*

### 13.9.11.1 Detailed Description

Memory allocator C interface.

### 13.9.11.2 Enumeration Type Documentation

#### 13.9.11.2.1 l4re\_ma\_flags

```
enum l4re\_ma\_flags
```

Flags for requesting memory at the memory allocator.

See also

[L4Re::Mem\\_alloc::Mem\\_alloc\\_flags](#)

Definition at line 42 of file [mem\\_alloc.h](#).

### 13.9.11.3 Function Documentation

#### 13.9.11.3.1 l4re\_ma\_alloc()

```
long l4re\_ma\_alloc (
    long size,
    l4re\_ds\_t const mem,
    unsigned long flags ) [inline]
```

Allocate memory.

## Parameters

<i>size</i>	Size in bytes to be requested. Allocation granularity is (super)pages, however, the allocator will store the byte-granular given size as the size of the dataspace and consecutively will use this byte-granular size for servicing the dataspace. Allocators may optionally also implement a maximum allocation strategy: if <i>size</i> is a negative value and <i>flags</i> set the <code>Mem_alloc_flags::Continuous</code> bit, the allocator tries to allocate as much memory as possible leaving an amount of at least <code>-size</code> bytes within the associated quota.
<i>mem</i>	Capability slot where the capability to the dataspace is received.
<i>flags</i>	Special dataspace properties, see <a href="#">l4re_ma_flags</a>

## Return values

0	Success
-L4_ERANGE	Given size not supported.
-L4_ENOMEM	Not enough memory available.
<0	IPC error

## See also

[L4Re::Mem\\_alloc::alloc](#)

The memory allocator returns a dataspace.

## Note

This function is using the [L4Re::Env::env\(\)](#)->`mem_alloc()` service.

## Examples

[examples/libs/l4re/c/ma+rm.c](#).

Definition at line 146 of file [mem\\_alloc.h](#).

References [l4re\\_ma\\_alloc\\_align\\_srv\(\)](#), and [l4re\\_env\\_t::mem\\_alloc](#).

Here is the call graph for this function:



## 13.9.11.3.2 l4re\_ma\_alloc\_align()

```

long l4re_ma_alloc_align (
    long size,
    l4re_ds_t const mem,
    unsigned long flags,
    unsigned long align ) [inline]
  
```

Allocate memory.

## Parameters

<i>size</i>	Size in bytes to be requested. Allocation granularity is (super)pages, however, the allocator will store the byte-granular given size as the size of the dataspace and consecutively will use this byte-granular size for servicing the dataspace. Allocators may optionally also implement a maximum allocation strategy: if <i>size</i> is a negative value and <i>flags</i> set the <code>Mem_alloc_flags::Continuous</code> bit, the allocator tries to allocate as much memory as possible leaving an amount of at least <code>-size</code> bytes within the associated quota.
<i>mem</i>	Capability slot where the capability to the dataspace is received.
<i>flags</i>	Special dataspace properties, see <a href="#">l4re_ma_flags</a>
<i>align</i>	Log2 alignment of dataspace if supported by allocator, will be at least <code>L4_PAGESHIFT</code> , with <code>Super_pages</code> flag set at least <code>L4_SUPERPAGESHIFT</code>

## Return values

<code>0</code>	Success
<code>-L4_ERANGE</code>	Given size not supported.
<code>-L4_ENOMEM</code>	Not enough memory available.
<code>&lt;0</code>	IPC error

## See also

[L4Re::Mem\\_alloc::alloc](#) and  
[l4re\\_ma\\_alloc](#)

The memory allocator returns a dataspace.

## Note

This function is using the [L4Re::Env::env\(\)](#)->`mem_alloc()` service.

Definition at line 154 of file [mem\\_alloc.h](#).

References [l4re\\_ma\\_alloc\\_align\\_srv\(\)](#), and [l4re\\_env\\_t::mem\\_alloc](#).

Here is the call graph for this function:



## 13.9.11.3.3 l4re\_ma\_alloc\_align\_srv()

```

long l4re_ma_alloc_align_srv (
    l4_cap_idx_t srv,
    long size,
    l4re_ds_t const mem,
    unsigned long flags,
    unsigned long align )
  
```

Allocate memory.

## Parameters

<i>srv</i>	Memory allocator service.
<i>size</i>	Size to be requested.
<i>mem</i>	Capability slot to put the requested dataspace in
<i>flags</i>	Flags, see <a href="#">l4re_ma_flags</a>
<i>align</i>	Log2 alignment of dataspace if supported by allocator, will be at least L4_PAGESHIFT, with Super_pages flag set at least L4_SUPERPAGESHIFT, default 0

## Returns

0 on success, <0 on error

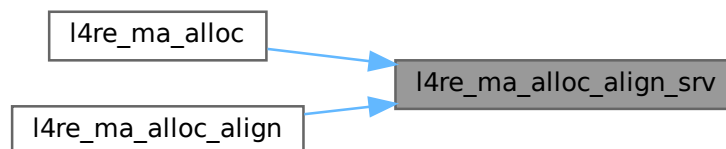
## See also

[L4Re::Mem\\_alloc::alloc](#)

The memory allocator returns a dataspace.

Referenced by [l4re\\_ma\\_alloc\(\)](#), and [l4re\\_ma\\_alloc\\_align\(\)](#).

Here is the caller graph for this function:



### 13.9.12 Namespace interface

Namespace C interface.

Collaboration diagram for Namespace interface:



## Typedefs

- typedef [l4\\_cap\\_idx\\_t](#) [l4re\\_namespace\\_t](#)  
*Namespace type.*

## Enumerations

- enum [l4re\\_ns\\_register\\_flags](#)  
*Namespace register flags.*

## Functions

- long [l4re\\_ns\\_query\\_to\\_srv](#) ([l4re\\_namespace\\_t](#) srv, char const \*name, [l4\\_cap\\_idx\\_t](#) const cap, int timeout) [L4\\_NOTHROW](#)  
*Query the name space for the object named by name.*
- long [l4re\\_ns\\_query\\_srv](#) ([l4re\\_namespace\\_t](#) srv, char const \*name, [l4\\_cap\\_idx\\_t](#) const cap) [L4\\_NOTHROW](#)  
*Query the name space for the object named by name.*
- long [l4re\\_ns\\_register\\_obj\\_srv](#) ([l4re\\_namespace\\_t](#) srv, char const \*name, [l4\\_cap\\_idx\\_t](#) const obj, unsigned flags) [L4\\_NOTHROW](#)  
*Register an object with a name.*

### 13.9.12.1 Detailed Description

Namespace C interface.

### 13.9.12.2 Enumeration Type Documentation

#### 13.9.12.2.1 [l4re\\_ns\\_register\\_flags](#)

```
enum l4re\_ns\_register\_flags
```

Namespace register flags.

See also

[L4Re::Namespace::Register\\_flags](#)

Definition at line 39 of file [namespace.h](#).

### 13.9.12.3 Function Documentation

#### 13.9.12.3.1 [l4re\\_ns\\_query\\_srv\(\)](#)

```
long l4re\_ns\_query\_srv (  
    l4re\_namespace\_t srv,  
    char const * name,  
    l4\_cap\_idx\_t const cap ) [inline]
```

Query the name space for the object named by name.



## Parameters

<i>srv</i>	Name space server to use for the query.
<i>name</i>	String to query.
<i>cap</i>	Capability slot where the received capability will be stored.

## Return values

0	Name could be fully resolved.
>0	Name could only be partly resolved. The number of remaining characters is returned.
-L4_ENOENT	Entry could not be found.
-L4_EAGAIN	Entry exists but no object is yet attached. Try again later.
<0	IPC errors, see <a href="#">l4_error_code_t</a> .

Definition at line 105 of file [namespace.h](#).

References [l4re\\_ns\\_query\\_to\\_srv\(\)](#).

Here is the call graph for this function:



## 13.9.12.3.2 l4re\_ns\_query\_to\_srv()

```

long l4re_ns_query_to_srv (
    l4re_namespace_t srv,
    char const * name,
    l4_cap_idx_t const cap,
    int timeout )
  
```

Query the name space for the object named by *name*.

## Parameters

<i>timeout</i>	Timeout of query in milliseconds. The client will only wait if a name already has been registered with the server but no object has been attached yet.
<i>srv</i>	Name space server to use for the query.
<i>name</i>	String to query.
<i>cap</i>	Capability slot where the received capability will be stored.

## Return values

<i>0</i>	Name could be fully resolved.
<i>&gt;0</i>	Name could only be partly resolved. The number of remaining characters is returned.
<i>-L4_ENOENT</i>	Entry could not be found.
<i>-L4_EAGAIN</i>	Entry exists but no object is yet attached. Try again later.
<i>&lt;0</i>	IPC errors, see <a href="#">l4_error_code_t</a> .

Referenced by [l4re\\_ns\\_query\\_srv\(\)](#).

Here is the caller graph for this function:



### 13.9.12.3.3 l4re\_ns\_register\_obj\_srv()

```

long l4re_ns_register_obj_srv (
    l4re_namespace_t srv,
    char const * name,
    l4_cap_idx_t const obj,
    unsigned flags )
  
```

Register an object with a name.

## Parameters

<i>srv</i>	Name space server to use for the query.
<i>name</i>	Name under which the object should be registered.
<i>obj</i>	Capability to object to register. An invalid capability may be given to only reserve the name for later use.
<i>flags</i>	Flags to assign to the entry, see <a href="#">L4Re::Namespace::Register_flags</a> . Note that the rights that are assigned to a capability are not only determined by the rights given in these flags but also by the rights with which the <code>obj</code> capability was mapped to the name space.

## Return values

<i>0</i>	Object was successfully registered with <i>name</i> .
<i>-L4_EEXIST</i>	Name already registered.
<i>-L4_EPERM</i>	Caller does not have <a href="#">L4_CAP_FPAGE_W</a> right on the invoked capability.
<i>-L4_ENOMEM</i>	Server has insufficient resources.
<i>-L4_EINVAL</i>	Invalid parameter.
<i>&lt;0</i>	IPC errors, see <a href="#">l4_error_code_t</a> .

**Precondition**

requires capability rights: {RW}

**13.9.13 Parent interface**

Collaboration diagram for Parent interface:

**13.9.14 Region map interface**

Region map C interface.

Collaboration diagram for Region map interface:

**Enumerations**

- enum `l4re_rm_flags_values` {  
`L4RE_RM_F_R` = `L4RE_DS_F_R` , `L4RE_RM_F_W` = `L4RE_DS_F_W` , `L4RE_RM_F_X` = `L4RE_DS_F_X`  
, `L4RE_RM_F_RX` = `L4RE_DS_F_RX` ,  
`L4RE_RM_F_RW` = `L4RE_DS_F_RW` , `L4RE_RM_F_RWX` = `L4RE_DS_F_RWX` , `L4RE_RM_F_NO_ALIAS`  
= `0x200` , `L4RE_RM_F_PAGER` = `0x400` ,  
`L4RE_RM_F_RESERVED` = `0x800` , `L4RE_RM_CACHING_SHIFT` = `4` , `L4RE_RM_F_CACHING` = `L4RE_DS_F_CACHING_MASK` , `L4RE_RM_REGION_FLAGS` = `0xffff` ,  
`L4RE_RM_F_CACHE_NORMAL` = `L4RE_DS_F_NORMAL` , `L4RE_RM_F_CACHE_BUFFERED` = `L4RE_DS_F_BUFFERABLE` , `L4RE_RM_F_CACHE_UNCACHED` = `L4RE_DS_F_UNCACHEABLE` ,  
`L4RE_RM_F_SEARCH_ADDR` = `0x020000` ,  
`L4RE_RM_F_IN_AREA` = `0x040000` , `L4RE_RM_F_EAGER_MAP` = `0x080000` , `L4RE_RM_F_NO_EAGER_MAP`  
= `0x100000` , `L4RE_RM_F_ATTACH_FLAGS` = `0x1f0000` }

*Flags for region operations.*

## Functions

- `int l4re_rm_reserve_area (l4_addr_t *start, unsigned long size, l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW`  
*Reserve the given area in the region map.*
- `int l4re_rm_free_area (l4_addr_t addr) L4_NOTHROW`  
*Free an area from the region map.*
- `int l4re_rm_attach (void **start, unsigned long size, l4re_rm_flags_t flags, l4re_ds_t mem, l4re_rm_offset_t offs, unsigned char align) L4_NOTHROW`  
*Attach a data space to a region.*
- `int l4re_rm_detach (void *addr) L4_NOTHROW`  
*Detach and unmap a region from the address space in the current task.*
- `int l4re_rm_detach_ds (void *addr, l4re_ds_t *ds) L4_NOTHROW`  
*Detach and unmap a region and return affected dataspace in the current task.*
- `int l4re_rm_detach_unmap (l4_addr_t addr, l4_cap_idx_t task) L4_NOTHROW`  
*Detach and unmap in specified task.*
- `int l4re_rm_detach_ds_unmap (void *addr, l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW`  
*Detach and unmap in specified task.*
- `int l4re_rm_find (l4_addr_t *addr, unsigned long *size, l4re_rm_offset_t *offset, l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW`  
*Find a region given an address and size.*
- `void l4re_rm_show_lists (void) L4_NOTHROW`  
*Dump region map internal data structures.*
- `int l4re_rm_reserve_area_srv (l4_cap_idx_t rm, l4_addr_t *start, unsigned long size, l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW`
- `int l4re_rm_free_area_srv (l4_cap_idx_t rm, l4_addr_t addr) L4_NOTHROW`
- `int l4re_rm_attach_srv (l4_cap_idx_t rm, void **start, unsigned long size, l4re_rm_flags_t flags, l4re_ds_t mem, l4re_rm_offset_t offs, unsigned char align) L4_NOTHROW`
- `int l4re_rm_detach_srv (l4_cap_idx_t rm, l4_addr_t addr, l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW`
- `int l4re_rm_find_srv (l4_cap_idx_t rm, l4_addr_t *addr, unsigned long *size, l4re_rm_offset_t *offset, l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW`
- `void l4re_rm_show_lists_srv (l4_cap_idx_t rm) L4_NOTHROW`  
*Dump region map internal data structures.*

### 13.9.14.1 Detailed Description

Region map C interface.

### 13.9.14.2 Enumeration Type Documentation

#### 13.9.14.2.1 l4re\_rm\_flags\_values

```
enum l4re_rm_flags_values
```

Flags for region operations.

#### Enumerator

L4RE_RM_F_R	Region is read-only.
L4RE_RM_F_NO_ALIAS	The region contains exclusive memory that is not mapped anywhere else.
L4RE_RM_F_PAGER	Region has a pager.

## Enumerator

L4RE_RM_F_RESERVED	Region is reserved (blocked)
L4RE_RM_CACHING_SHIFT	Start of region mapper cache bits.
L4RE_RM_F_CACHING	Mask of all region manager cache bits.
L4RE_RM_REGION_FLAGS	Mask of all region flags.
L4RE_RM_F_CACHE_NORMAL	Cache bits for normal cacheable memory.
L4RE_RM_F_CACHE_BUFFERED	Cache bits for buffered (write combining) memory.
L4RE_RM_F_CACHE_UNCACHED	Cache bits for uncached memory.
L4RE_RM_F_SEARCH_ADDR	Search for a suitable address range.
L4RE_RM_F_IN_AREA	Search only in area, or map into area.
L4RE_RM_F_EAGER_MAP	Eagerly map the attached data space in.
L4RE_RM_F_NO_EAGER_MAP	Prevent eager mapping of the attached data space.
L4RE_RM_F_ATTACH_FLAGS	Mask of all attach flags.

Definition at line 40 of file [rm.h](#).

## 13.9.14.3 Function Documentation

13.9.14.3.1 `l4re_rm_attach()`

```
int l4re_rm_attach (
    void ** start,
    unsigned long size,
    l4re_rm_flags_t flags,
    l4re_ds_t mem,
    l4re_rm_offset_t offs,
    unsigned char align ) [inline]
```

Attach a data space to a region.

## Parameters

<code>in, out</code>	<code>start</code>	Virtual start address where the region manager shall attach the data space. Will be rounded down to the nearest start of a page. If <a href="#">L4Re::Rm::F::Search_addr</a> is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If <a href="#">L4Re::Rm::F::In_area</a> is given the value is used as a selector for the area (see <a href="#">L4Re::Rm::reserve_area</a> ) to attach the data space to.
	<code>size</code>	Size of the data space to attach (in bytes). Will be rounded up to the nearest multiple of the page size.
	<code>flags</code>	The flags control how and with which rights the dataspace is attached to the region. See <a href="#">L4Re::Rm::F::Attach_flags</a> and <a href="#">L4Re::Rm::F::Region_flags</a> . The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the <code>F::Eager_map</code> flag is set this function may also return <a href="#">L4Re::Dataspace::map</a> error codes if the mapping fails.
	<code>mem</code>	Data space.
	<code>offs</code>	Offset into the data space to use.
	<code>align</code>	Alignment of the virtual region, log2-size, default: a page ( <a href="#">L4_PAGESHIFT</a> ). This is only meaningful if the <a href="#">L4Re::Rm::F::Search_addr</a> flag is used.
	<code>task</code>	Optional destination task of mapping if <code>F::Eager_map</code> flag was passed. If invalid, the mapping is established in the current task. This parameter is only useful if the region manager is for a foreign task.

## Return values

0	Success
-L4_ENOENT	No area could be found (see <a href="#">L4Re::Rm::F::In_area</a> )
-L4_EPERM	Operation not allowed.
-L4_EINVAL	
-L4_EADDRNOTAVAIL	The given address is not available.
<0	IPC errors

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

## Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see [L4Re::Rm::find](#)).

## See also

[L4Re::Rm::attach](#)

This function is using the `L4::Env::env()->rm()` service.

## Examples

[examples/libs/l4re/c/ma+rm.c](#).

Definition at line 287 of file [rm.h](#).

References [l4re\\_rm\\_attach\\_srv\(\)](#), and [l4re\\_env\\_t::rm](#).

Here is the call graph for this function:



### 13.9.14.3.2 l4re\_rm\_attach\_srv()

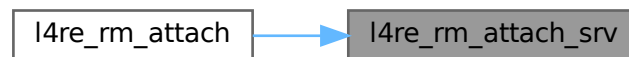
```
int l4re_rm_attach_srv (
    l4_cap_idx_t rm,
    void ** start,
    unsigned long size,
    l4re_rm_flags_t flags,
    l4re_ds_t mem,
    l4re_rm_offset_t offs,
    unsigned char align )
```

See also

[L4Re::Rm::attach](#)

Referenced by [l4re\\_rm\\_attach\(\)](#).

Here is the caller graph for this function:



### 13.9.14.3.3 l4re\_rm\_detach()

```
int l4re_rm_detach (
    void * addr ) [inline]
```

Detach and unmap a region from the address space in the current task.

Parameters

<i>addr</i>	Address of the region to detach.
-------------	----------------------------------

Return values

<a href="#">L4Re::Rm::Detach_result</a>	On success.
<code>-L4_ENOENT</code>	No region found.
<code>&lt;0</code>	IPC errors

Frees a region in the virtual address space given by *addr*. The corresponding part of the address space is now available again.

Also

See also

[L4Re::Rm::detach](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 297 of file `rm.h`.

References [L4\\_BASE\\_TASK\\_CAP](#), [l4re\\_rm\\_detach\\_srv\(\)](#), and [l4re\\_env\\_t::rm](#).

Here is the call graph for this function:



#### 13.9.14.3.4 l4re\_rm\_detach\_ds()

```
int l4re_rm_detach_ds (
    void * addr,
    l4re_ds_t * ds ) [inline]
```

Detach and unmap a region and return affected dataspace in the current task.

##### Parameters

	<i>addr</i>	Address of the region to detach.
out	<i>ds</i>	Returns dataspace that is affected.

##### Return values

<a href="#">L4Re::Rm::Detach_result</a>	On success.
<code>-L4_ENOENT</code>	No region found.
<code>&lt;0</code>	IPC errors

Frees a region in the virtual address space given by `addr`. The corresponding part of the address space is now available again.

Also

See also

[L4Re::Rm::detach](#)

This function is using the `L4::Env::env()->rm()` service.



## Examples

[examples/libs/l4re/c/ma+rm.c](#).

Definition at line 310 of file [rm.h](#).

References [L4\\_BASE\\_TASK\\_CAP](#), [l4re\\_rm\\_detach\\_srv\(\)](#), and [l4re\\_env\\_t::rm](#).

Here is the call graph for this function:



#### 13.9.14.3.5 l4re\_rm\_detach\_ds\_unmap()

```
int l4re_rm_detach_ds_unmap (
    void * addr,
    l4re_ds_t * ds,
    l4_cap_idx_t task ) [inline]
```

Detach and unmap in specified task.

## Parameters

	<i>addr</i>	Address of the region to detach.
out	<i>ds</i>	Returns dataspace that is affected.
	<i>task</i>	Task to unmap pages from, specify L4_INVALID_CAP to not unmap

## Returns

0 on success, <0 on error

Also

See also

[L4Re::Rm::detach](#)

This function is using the [L4::Env::env\(\)->rm\(\)](#) service.

Definition at line 317 of file [rm.h](#).

References [l4re\\_rm\\_detach\\_srv\(\)](#), and [l4re\\_env\\_t::rm](#).

Here is the call graph for this function:



#### 13.9.14.3.6 l4re\_rm\_detach\_srv()

```

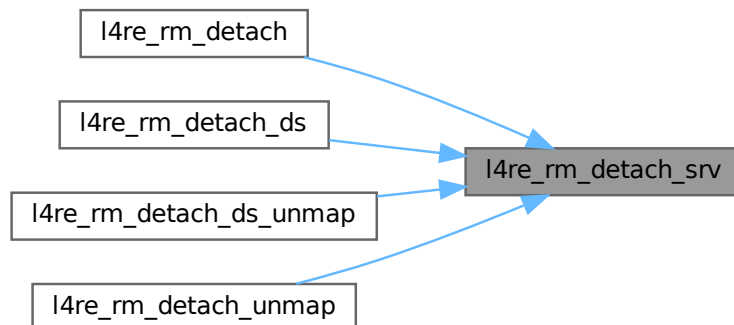
int l4re_rm_detach_srv (
    l4_cap_idx_t rm,
    l4_addr_t addr,
    l4re_ds_t * ds,
    l4_cap_idx_t task )
  
```

See also

[L4Re::Rm::detach](#)

Referenced by [l4re\\_rm\\_detach\(\)](#), [l4re\\_rm\\_detach\\_ds\(\)](#), [l4re\\_rm\\_detach\\_ds\\_unmap\(\)](#), and [l4re\\_rm\\_detach\\_unmap\(\)](#).

Here is the caller graph for this function:



#### 13.9.14.3.7 l4re\_rm\_detach\_unmap()

```

int l4re_rm_detach_unmap (
    l4_addr_t addr,
    l4_cap_idx_t task ) [inline]
  
```

Detach and unmap in specified task.

## Parameters

<i>addr</i>	Address of the region to detach.
<i>task</i>	Task to unmap pages from, specify L4_INVALID_CAP to not unmap

## Returns

0 on success, <0 on error

## Also

## See also

[L4Re::Rm::detach](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 304 of file [rm.h](#).

References [l4re\\_rm\\_detach\\_srv\(\)](#), and [l4re\\_env\\_t::rm](#).

Here is the call graph for this function:



## 13.9.14.3.8 l4re\_rm\_find()

```

int l4re_rm_find (
    l4_addr_t * addr,
    unsigned long * size,
    l4re_rm_offset_t * offset,
    l4re_rm_flags_t * flags,
    l4re_ds_t * m ) [inline]
  
```

Find a region given an address and size.

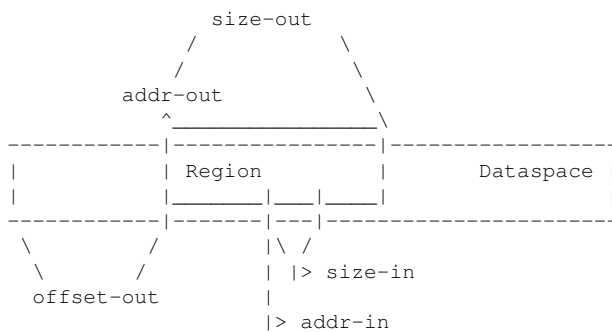
## Parameters

in, out	<i>addr</i>	Address to look for. Returns the start address of the found region.
in, out	<i>size</i>	Size of the area to look for (in bytes). Returns the size of the found region (in bytes).
out	<i>offset</i>	Offset at the beginning of the region within the associated dataspace.
out	<i>flags</i>	Region flags, see <code>F::Region_flags</code> (and <code>F::In_area</code> ).
out	<i>m</i>	Associated dataspace or paging service.

## Return values

0	Success
-L4_EPERM	Operation not allowed.
-L4_ENOENT	No region found.
<0	IPC errors

This function returns the properties of the region that contains the area described by the `addr` and `size` parameter. If no such region is found but a reserved area, the area is returned and `F::ln_area` is set in `flags`. Note, in the case of an area the `offset` and `m` return values are invalid.



## Note

The value of the size input parameter should be 1 to assure that a region can be determined unambiguously.

## See also

[L4Re::Rm::find](#)

Definition at line 324 of file [rm.h](#).

References [l4re\\_rm\\_find\\_srv\(\)](#), and [l4re\\_env\\_t::rm](#).

Here is the call graph for this function:



### 13.9.14.3.9 l4re\_rm\_find\_srv()

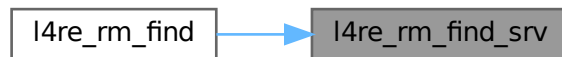
```
int l4re_rm_find_srv (
    l4_cap_idx_t rm,
    l4_addr_t * addr,
    unsigned long * size,
    l4re_rm_offset_t * offset,
    l4re_rm_flags_t * flags,
    l4re_ds_t * m )
```

See also

[L4Re::Rm::find](#)

Referenced by [l4re\\_rm\\_find\(\)](#).

Here is the caller graph for this function:



### 13.9.14.3.10 l4re\_rm\_free\_area()

```
int l4re_rm_free_area (
    l4_addr_t addr ) [inline]
```

Free an area from the region map.

Parameters

<i>addr</i>	An address within the area to free.
-------------	-------------------------------------

Return values

0	Success
-L4_ENOENT	No area found.
<0	IPC errors

Note

The data spaces that are attached to that area are not detached by this operation.

## See also

`reserve_area()` for more information about areas.

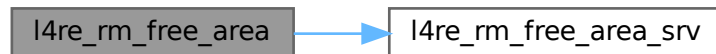
[L4Re::Rm::free\\_area](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 281 of file `rm.h`.

References [l4re\\_rm\\_free\\_area\\_srv\(\)](#), and [l4re\\_env\\_t::rm](#).

Here is the call graph for this function:



### 13.9.14.3.11 l4re\_rm\_free\_area\_srv()

```
int l4re_rm_free_area_srv (
    l4_cap_idx_t rm,
    l4_addr_t addr )
```

## See also

[L4Re::Rm::free\\_area](#)

Referenced by [l4re\\_rm\\_free\\_area\(\)](#).

Here is the caller graph for this function:



### 13.9.14.3.12 l4re\_rm\_reserve\_area()

```
int l4re_rm_reserve_area (
    l4_addr_t * start,
    unsigned long size,
    l4re_rm_flags_t flags,
    unsigned char align ) [inline]
```

Reserve the given area in the region map.

## Parameters

<code>in, out</code>	<code>start</code>	The virtual start address of the area to reserve. Returns the start address of the area.
	<code>size</code>	The size of the area to reserve (in bytes).
	<code>flags</code>	Flags for the reserved area (see <a href="#">L4Re::Rm::F::Region_flags</a> and <a href="#">L4Re::Rm::F::Attach_flags</a> ).
	<code>align</code>	Alignment of area if searched as bits (log2 value).

## Return values

<code>0</code>	Success
<code>-L4_EADDRNOTAVAIL</code>	The given area cannot be reserved.
<code>&lt;0</code>	IPC errors

This function reserves an area within the virtual address space managed by the region map. There are two kinds of areas available:

- Reserved areas (`flags = L4Re::Rm::F::Reserved`), where no data spaces can be attached
- Special purpose areas (`flags = 0`), where data spaces can be attached to the area via the [L4Re::Rm::F::In\\_area](#) flag and a start address within the area itself.

## Note

When searching for a free place in the virtual address space (with `flags = L4Re::Rm::F::Search\_addr`), the space between `start` and the end of the virtual address space is searched.

## See also

[L4Re::Rm::reserve\\_area](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 273 of file [rm.h](#).

References [l4re\\_rm\\_reserve\\_area\\_srv\(\)](#), and [l4re\\_env\\_t::rm](#).

Here is the call graph for this function:



#### 13.9.14.3.13 l4re\_rm\_reserve\_area\_srv()

```
int l4re_rm_reserve_area_srv (
    l4_cap_idx_t rm,
    l4_addr_t * start,
    unsigned long size,
    l4re_rm_flags_t flags,
    unsigned char align )
```

See also

[L4Re::Rm::reserve\\_area](#)

Referenced by [l4re\\_rm\\_reserve\\_area\(\)](#).

Here is the caller graph for this function:



#### 13.9.14.3.14 l4re\_rm\_show\_lists()

```
void l4re_rm_show_lists (
    void ) [inline]
```

Dump region map internal data structures.

This function is using the `L4::Env::env()->rm()` service.

Definition at line 332 of file [rm.h](#).

References [l4re\\_rm\\_show\\_lists\\_srv\(\)](#), and [l4re\\_env\\_t::rm](#).

Here is the call graph for this function:





### 13.9.15 Video API

Collaboration diagram for Video API:



#### Data Structures

- struct [l4re\\_video\\_color\\_component\\_t](#)  
*Color component structure.*
- struct [l4re\\_video\\_pixel\\_info\\_t](#)  
*Pixel\_info structure.*
- struct [l4re\\_video\\_goos\\_info\\_t](#)  
*Goos information structure.*
- struct [l4re\\_video\\_view\\_info\\_t](#)  
*View information structure.*
- struct [l4re\\_video\\_view\\_t](#)  
*C representation of a goos view.*

#### Typedefs

- typedef struct [l4re\\_video\\_color\\_component\\_t](#) [l4re\\_video\\_color\\_component\\_t](#)  
*Color component structure.*
- typedef struct [l4re\\_video\\_pixel\\_info\\_t](#) [l4re\\_video\\_pixel\\_info\\_t](#)  
*Pixel\_info structure.*
- typedef struct [l4re\\_video\\_view\\_info\\_t](#) [l4re\\_video\\_view\\_info\\_t](#)  
*View information structure.*
- typedef struct [l4re\\_video\\_view\\_t](#) [l4re\\_video\\_view\\_t](#)  
*C representation of a goos view.*

#### Enumerations

- enum [l4re\\_video\\_goos\\_info\\_flags\\_t](#) { [F\\_l4re\\_video\\_goos\\_auto\\_refresh](#) = 0x01 , [F\\_l4re\\_video\\_goos\\_pointer](#) = 0x02 , [F\\_l4re\\_video\\_goos\\_dynamic\\_views](#) = 0x04 , [F\\_l4re\\_video\\_goos\\_dynamic\\_buffers](#) = 0x08 }  
*Flags of information on the goos.*
- enum [l4re\\_video\\_view\\_info\\_flags\\_t](#) {  
[F\\_l4re\\_video\\_view\\_none](#) = 0x00 , [F\\_l4re\\_video\\_view\\_set\\_buffer](#) = 0x01 , [F\\_l4re\\_video\\_view\\_set\\_buffer\\_offset](#) = 0x02 , [F\\_l4re\\_video\\_view\\_set\\_bytes\\_per\\_line](#) = 0x04 ,  
[F\\_l4re\\_video\\_view\\_set\\_pixel](#) = 0x08 , [F\\_l4re\\_video\\_view\\_set\\_position](#) = 0x10 , [F\\_l4re\\_video\\_view\\_dyn\\_allocated](#) = 0x20 , [F\\_l4re\\_video\\_view\\_set\\_background](#) = 0x40 ,  
[F\\_l4re\\_video\\_view\\_set\\_flags](#) = 0x80 , [F\\_l4re\\_video\\_view\\_fully\\_dynamic](#) , [F\\_l4re\\_video\\_view\\_above](#) = 0x01000 , [F\\_l4re\\_video\\_view\\_flags\\_mask](#) = 0xff000 }  
*Flags of information on a view.*

## Functions

- `int l4re_video_goos_info (l4re_video_goos_t goos, l4re_video_goos_info_t *ginfo) L4_NOTHROW`  
*Get information on a goos.*
- `int l4re_video_goos_refresh (l4re_video_goos_t goos, int x, int y, int w, int h) L4_NOTHROW`  
*Flush a rectangle of pixels of the goos screen.*
- `int l4re_video_goos_create_buffer (l4re_video_goos_t goos, unsigned long size, l4_cap_idx_t buffer) L4_NOTHROW`  
*Create a new buffer (memory buffer) for pixel data.*
- `int l4re_video_goos_delete_buffer (l4re_video_goos_t goos, unsigned idx) L4_NOTHROW`  
*Delete a pixel buffer.*
- `int l4re_video_goos_get_static_buffer (l4re_video_goos_t goos, unsigned idx, l4_cap_idx_t buffer) L4_NOTHROW`  
*Get the data-space capability of the static pixel buffer.*
- `int l4re_video_goos_create_view (l4re_video_goos_t goos, l4re_video_view_t *view) L4_NOTHROW`  
*Create a new view (.*
- `int l4re_video_goos_delete_view (l4re_video_goos_t goos, l4re_video_view_t *view) L4_NOTHROW`  
*Delete a view.*
- `int l4re_video_goos_get_view (l4re_video_goos_t goos, unsigned idx, l4re_video_view_t *view) L4_NOTHROW`  
*Get a view for the given index.*
- `int l4re_video_view_refresh (l4re_video_view_t *view, int x, int y, int w, int h) L4_NOTHROW`  
*Flush the given rectangle of pixels of the given view.*
- `int l4re_video_view_get_info (l4re_video_view_t *view, l4re_video_view_info_t *info) L4_NOTHROW`  
*Retrieve information about the given view.*
- `int l4re_video_view_set_info (l4re_video_view_t *view, l4re_video_view_info_t *info) L4_NOTHROW`  
*Set properties of the view.*
- `int l4re_video_view_set_viewport (l4re_video_view_t *view, int x, int y, int w, int h, unsigned long bofs) L4_NOTHROW`  
*Set the viewport parameters of a view.*
- `int l4re_video_view_stack (l4re_video_view_t *view, l4re_video_view_t *pivot, int behind) L4_NOTHROW`  
*Change the stacking order in the stack of visible views.*

### 13.9.15.1 Detailed Description

### 13.9.15.2 Typedef Documentation

#### 13.9.15.2.1 l4re\_video\_view\_t

```
typedef struct l4re_video_view_t l4re_video_view_t
```

C representation of a goos view.

A view is a visible rectangle that provides a view to the contents of a buffer (frame buffer) memory object and is placed on a real screen.

### 13.9.15.3 Enumeration Type Documentation

#### 13.9.15.3.1 l4re\_video\_goos\_info\_flags\_t

```
enum l4re_video_goos_info_flags_t
```

Flags of information on the goos.

## Enumerator

F_l4re_video_goos_auto_refresh	The graphics display is automatically refreshed.
F_l4re_video_goos_pointer	We have a mouse pointer.
F_l4re_video_goos_dynamic_views	Supports dynamically allocated views.
F_l4re_video_goos_dynamic_buffers	Supports dynamically allocated buffers.

Definition at line 39 of file [goos.h](#).

## 13.9.15.3.2 l4re\_video\_view\_info\_flags\_t

```
enum l4re_video_view_info_flags_t
```

Flags of information on a view.

## Enumerator

F_l4re_video_view_none	everything for this view is static (the VESA-FB case)
F_l4re_video_view_set_buffer	buffer object for this view can be changed
F_l4re_video_view_set_buffer_offset	buffer offset can be set
F_l4re_video_view_set_bytes_per_line	bytes per line can be set
F_l4re_video_view_set_pixel	pixel type can be set
F_l4re_video_view_set_position	position on screen can be set
F_l4re_video_view_dyn_allocated	View is dynamically allocated.
F_l4re_video_view_set_background	Set view as background for session.
F_l4re_video_view_set_flags	Set view property flags.
F_l4re_video_view_above	Flag the view as stay on top.
F_l4re_video_view_flags_mask	Mask containing all possible property flags.

Definition at line 33 of file [view.h](#).

## 13.9.15.4 Function Documentation

## 13.9.15.4.1 l4re\_video\_goos\_create\_buffer()

```
int l4re_video_goos_create_buffer (
    l4re_video_goos_t goos,
    unsigned long size,
    l4_cap_idx_t buffer )
```

Create a new buffer (memory buffer) for pixel data.

## Parameters

<i>goos</i>	the target object for the operation.
<i>size</i>	the size in bytes for the pixel buffer.
<i>buffer</i>	a capability index to receive the data-space capability for the buffer.

**Returns**

$\geq 0$ : The index of the created buffer (used to assign views and for deletion).  $< 0$ : on error

**13.9.15.4.2 l4re\_video\_goos\_create\_view()**

```
int l4re_video_goos_create_view (
    l4re_video_goos_t goos,
    l4re_video_view_t * view )
```

Create a new view (.

**See also**

[l4re\\_video\\_view\\_t](#)

**Parameters**

	<i>goos</i>	the goos session to use.
out	<i>view</i>	structure initialized to the new view.

**13.9.15.4.3 l4re\_video\_goos\_delete\_buffer()**

```
int l4re_video_goos_delete_buffer (
    l4re_video_goos_t goos,
    unsigned idx )
```

Delete a pixel buffer.

**Parameters**

<i>goos</i>	the target goos object.
<i>idx</i>	the buffer index of the buffer to delete (the return value of <a href="#">l4re_video_goos_create_buffer()</a> )

**13.9.15.4.4 l4re\_video\_goos\_delete\_view()**

```
int l4re_video_goos_delete_view (
    l4re_video_goos_t goos,
    l4re_video_view_t * view )
```

Delete a view.

**Parameters**

<i>goos</i>	the goos session to use.
<i>view</i>	the view to delete, the given data-structure is invalid afterwards.

#### 13.9.15.4.5 l4re\_video\_goos\_get\_static\_buffer()

```
int l4re_video_goos_get_static_buffer (
    l4re_video_goos_t goos,
    unsigned idx,
    l4_cap_idx_t buffer )
```

Get the data-space capability of the static pixel buffer.

##### Parameters

<i>goos</i>	The target goos object.
<i>idx</i>	Index of the static buffer.
<i>buffer</i>	A capability index to receive the data-space capability.

This function allows access to static, preexisting pixel buffers. Such static buffers exist for static configurations, such as the VESA framebuffer.

#### 13.9.15.4.6 l4re\_video\_goos\_get\_view()

```
int l4re_video_goos_get_view (
    l4re_video_goos_t goos,
    unsigned idx,
    l4re_video_view_t * view )
```

Get a view for the given index.

##### Parameters

	<i>goos</i>	the target goos session.
	<i>idx</i>	the index of the view to retrieve.
out	<i>view</i>	structure initialized to the view with the given index.

This function allows to access static views as provided by the VESA framebuffer (the monitor). However, it also allows to access dynamic views created with [l4re\\_video\\_goos\\_create\\_view\(\)](#).

#### 13.9.15.4.7 l4re\_video\_goos\_info()

```
int l4re_video_goos_info (
    l4re_video_goos_t goos,
    l4re_video_goos_info_t * ginfo )
```

Get information on a goos.

##### Parameters

	<i>goos</i>	Goos object
out	<i>ginfo</i>	Pointer to goos information structure.

**Returns**

0 for success, <0 on error

- [-L4\\_ENODEV](#)
- IPC errors

**13.9.15.4.8 l4re\_video\_goos\_refresh()**

```
int l4re_video_goos_refresh (
    l4re_video_goos_t goos,
    int x,
    int y,
    int w,
    int h )
```

Flush a rectangle of pixels of the goos screen.

**Parameters**

<i>goos</i>	the target object of the operation.
<i>x</i>	the x-coordinate of the upper left corner of the rectangle
<i>y</i>	the y-coordinate of the upper left corner of the rectangle
<i>w</i>	the width of the rectangle to be flushed
<i>h</i>	the height of the rectangle

**13.9.15.4.9 l4re\_video\_view\_get\_info()**

```
int l4re_video_view_get_info (
    l4re_video_view_t * view,
    l4re_video_view_info_t * info )
```

Retrieve information about the given *view*.

**Parameters**

	<i>view</i>	the target view for the operation.
out	<i>info</i>	a buffer receiving the information about the view.

**13.9.15.4.10 l4re\_video\_view\_refresh()**

```
int l4re_video_view_refresh (
    l4re_video_view_t * view,
    int x,
    int y,
    int w,
    int h )
```

Flush the given rectangle of pixels of the given *view*.

## Parameters

<i>view</i>	the target view of the operation.
<i>x</i>	x-coordinate of the upper left corner
<i>y</i>	y-coordinate of the upper left corner
<i>w</i>	the width of the rectangle
<i>h</i>	the height of the rectangle

**13.9.15.4.11 l4re\_video\_view\_set\_info()**

```
int l4re_video_view_set_info (
    l4re_video_view_t * view,
    l4re_video_view_info_t * info )
```

Set properties of the view.

## Parameters

<i>view</i>	the target view of the operation.
<i>info</i>	the parameters to be set on the view.

Which parameters can be manipulated on a given view can be figured out with [l4re\\_video\\_view\\_get\\_info\(\)](#) and this depends on the concrete instance the view object.

**13.9.15.4.12 l4re\_video\_view\_set\_viewport()**

```
int l4re_video_view_set_viewport (
    l4re_video_view_t * view,
    int x,
    int y,
    int w,
    int h,
    unsigned long bofs )
```

Set the viewport parameters of a view.

## Parameters

<i>view</i>	the target view of the operation.
<i>x</i>	the x-coordinate of the upper left corner on the screen.
<i>y</i>	the y-coordinate of the upper left corner on the screen.
<i>w</i>	the width of the view.
<i>h</i>	the height of the view.
<i>bofs</i>	the offset (in bytes) of the upper left pixel in the memory buffer

This function is a convenience wrapper for [l4re\\_video\\_view\\_set\\_info\(\)](#), just setting the often changed parameters of a dynamic view. With this function a view can be placed on the real screen and at the same time on its backing buffer.

#### 13.9.15.4.13 l4re\_video\_view\_stack()

```
int l4re_video_view_stack (
    l4re_video_view_t * view,
    l4re_video_view_t * pivot,
    int behind )
```

Change the stacking order in the stack of visible views.

##### Parameters

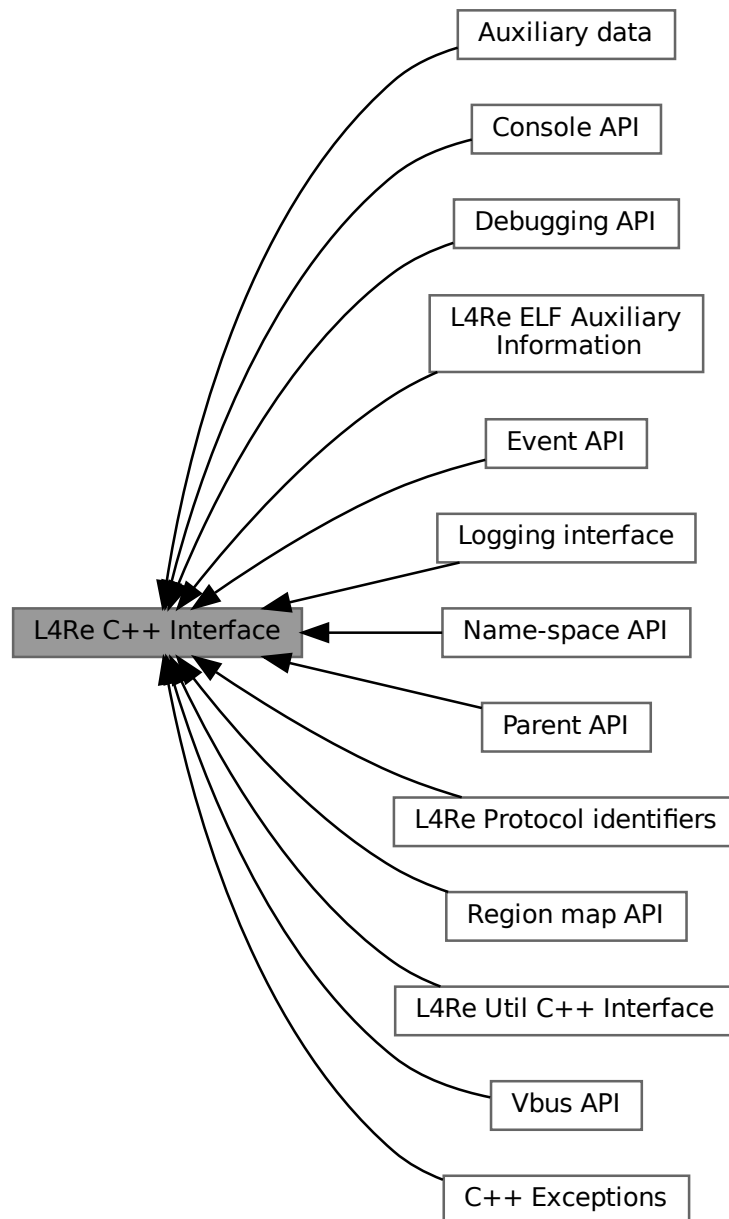
<i>view</i>	the target view for the operation.
<i>pivot</i>	the neighbor view, relative to which <i>view</i> shall be stacked. a NULL value allows top ( <i>behind</i> = 1) and bottom ( <i>behind</i> = 0) placement of the view.
<i>behind</i>	describes the placement of the view relative to the <i>pivot</i> view.

## 13.10 L4Re C++ Interface

Documentation of the [L4](#) Runtime Environment C++ API.



Collaboration diagram for L4Re C++ Interface:



## Modules

- [Auxiliary data](#)
- [C++ Exceptions](#)
- [Console API](#)  
*Console interface.*
- [Debugging API](#)  
*Debugging Interface.*

- [Event API](#)  
*Event API.*
- [L4Re ELF Auxiliary Information](#)  
*API for embedding auxiliary information into binary programs.*
- [L4Re Protocol identifiers](#)  
*Fix L4Re Protocol Constants.*
- [L4Re Util C++ Interface](#)  
*Documentation of the L4 Runtime Environment utility functionality in C++.*
- [Logging interface](#)  
*Interface for log output.*
- [Name-space API](#)  
*API for name spaces that store capabilities.*
- [Parent API](#)  
*Parent interface.*
- [Region map API](#)  
*Virtual address-space management.*
- [Vbus API](#)  
*C++ interface of the Vbus API.*

### 13.10.1 Detailed Description

Documentation of the [L4](#) Runtime Environment C++ API.

### 13.10.2 Auxiliary data

Collaboration diagram for Auxiliary data:



### Data Structures

- struct [l4re\\_aux\\_t](#)  
*Auxiliary descriptor.*

### Typedefs

- typedef struct [l4re\\_aux\\_t](#) [l4re\\_aux\\_t](#)  
*Auxiliary descriptor.*

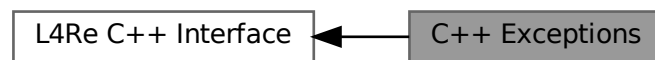
## Enumerations

- enum [l4re\\_aux\\_ldr\\_flags\\_t](#)  
*Flags for program loading.*

### 13.10.2.1 Detailed Description

## 13.10.3 C++ Exceptions

Collaboration diagram for C++ Exceptions:



## Files

- file [exceptions](#)  
*Base exceptions.*
- file [std\\_exc\\_io](#)  
*Base exceptions std stream operator.*

## Data Structures

- class [L4::Exception\\_tracer](#)  
*Back-trace support for exceptions.*
- class [L4::Base\\_exception](#)  
*Base class for all exceptions, thrown by the [L4Re](#) framework.*
- class [L4::Runtime\\_error](#)  
*Exception for an abstract runtime error.*
- class [L4::Out\\_of\\_memory](#)  
*Exception signalling insufficient memory.*
- class [L4::Element\\_already\\_exists](#)  
*Exception for duplicate element insertions.*
- class [L4::Unknown\\_error](#)  
*Exception for an unknown condition.*
- class [L4::Element\\_not\\_found](#)  
*Exception for a failed lookup (element not found).*
- class [L4::Invalid\\_capability](#)  
*Indicates that an invalid object was invoked.*
- class [L4::Com\\_error](#)  
*Error conditions during IPC.*
- class [L4::Bounds\\_error](#)  
*Access out of bounds.*

## Macros

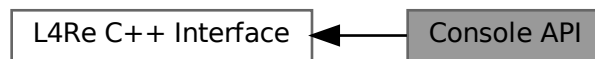
- `#define L4_CXX_EXCEPTION_BACKTRACE 20`  
*Number of instruction pointers in backtrace.*

### 13.10.3.1 Detailed Description

### 13.10.4 Console API

[Console](#) interface.

Collaboration diagram for Console API:



## Data Structures

- class [L4Re::Console](#)  
*[Console](#) class.*

### 13.10.4.1 Detailed Description

[Console](#) interface.

### 13.10.5 Debugging API

Debugging Interface.

Collaboration diagram for Debugging API:



## Data Structures

- class [L4Re::Debug\\_obj](#)  
*Debug interface.*

### 13.10.5.1 Detailed Description

Debugging Interface.

The debugging interface can be provided to retrieve, or log debugging information for an object. Each class may realize the debug interface to provide debugging functionality. For example, the region map objects provide a facility to dump the currently established memory regions.

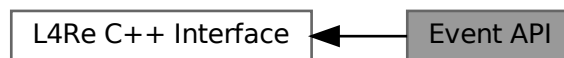
See also

[L4::Debug\\_obj](#) for more information.

## 13.10.6 Event API

[Event](#) API.

Collaboration diagram for Event API:



## Data Structures

- class [L4Re::Event](#)  
*Event class.*
- struct [L4Re::Default\\_event\\_payload](#)  
*Default event stream payload.*
- class [L4Re::Event\\_buffer\\_t< PAYLOAD >](#)  
*Event buffer class.*

### 13.10.6.1 Detailed Description

[Event](#) API.

On top of a shared [L4Re::Dataspace](#) (and optionally using [L4::Triggerable](#)), the event API implements asynchronous event transmission from an event provider (server) to an event receiver (client). Events are put into an [Event\\_buffer\\_t](#) residing on the shared [L4Re::Dataspace](#).

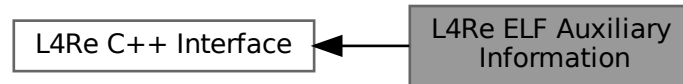
This interface is usually not used directly. Instead use [L4Re::Util::Event\\_t](#) for clients. An example server portion is implemented in [L4Re::Util::Event\\_svr](#).

This interface is usually used with [L4Re::Default\\_event\\_payload](#) which delivers HID events modeled on the Linux evdev API, and the interface's methods allow further querying of information about the HID event streams.

### 13.10.7 L4Re ELF Auxiliary Information

API for embedding auxiliary information into binary programs.

Collaboration diagram for L4Re ELF Auxiliary Information:



#### Data Structures

- struct [l4re\\_elf\\_aux\\_t](#)  
*Generic header for each auxiliary vector element.*
- struct [l4re\\_elf\\_aux\\_vma\\_t](#)  
*Auxiliary vector element for a reserved virtual memory area.*
- struct [l4re\\_elf\\_aux\\_mword\\_t](#)  
*Auxiliary vector element for a single unsigned data word.*

#### Macros

- `#define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".ro.l4re_elf_aux"), aligned(sizeof(l4_umword_t))))`  
*Define an auxiliary vector element.*
- `#define L4RE_ELF_AUX_ELEM_T(type, id, tag, val...) static L4RE_ELF_AUX_ELEM type id = {tag, sizeof(type), val}`  
*Define an auxiliary vector element.*

#### Typedefs

- typedef struct [l4re\\_elf\\_aux\\_t](#) [l4re\\_elf\\_aux\\_t](#)  
*Generic header for each auxiliary vector element.*
- typedef struct [l4re\\_elf\\_aux\\_vma\\_t](#) [l4re\\_elf\\_aux\\_vma\\_t](#)  
*Auxiliary vector element for a reserved virtual memory area.*
- typedef struct [l4re\\_elf\\_aux\\_mword\\_t](#) [l4re\\_elf\\_aux\\_mword\\_t](#)  
*Auxiliary vector element for a single unsigned data word.*

#### Enumerations

- enum {  
[L4RE\\_ELF\\_AUX\\_T\\_NONE](#) = 0 , [L4RE\\_ELF\\_AUX\\_T\\_VMA](#) , [L4RE\\_ELF\\_AUX\\_T\\_STACK\\_SIZE](#) ,  
[L4RE\\_ELF\\_AUX\\_T\\_STACK\\_ADDR](#) ,  
[L4RE\\_ELF\\_AUX\\_T\\_KIP\\_ADDR](#) , [L4RE\\_ELF\\_AUX\\_T\\_EX\\_REGS\\_FLAGS](#) }

### 13.10.7.1 Detailed Description

API for embedding auxiliary information into binary programs.

This API allows information for the binary loader to be embedded into a binary application. This information can be reserved areas in the virtual memory of an application and things such as the stack size to be allocated for the first application thread.

### 13.10.7.2 Macro Definition Documentation

#### 13.10.7.2.1 L4RE\_ELF\_AUX\_ELEM

```
#define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".ro14re_elf_aux"), aligned(sizeof(l4_umword_t))))
```

Define an auxiliary vector element.

This is the generic method for defining auxiliary vector elements. A more convenient way is to use `L4RE_ELF_AUX_ELEM_T`.

Usage:

```
L4RE_ELF_AUX_ELEM l4re_elf_aux_vma_t decl_name =
    { L4RE_ELF_AUX_T_VMA, sizeof(l4re_elf_aux_vma_t), 0x2000, 0x4000 };
```

Definition at line 52 of file [elf\\_aux.h](#).

#### 13.10.7.2.2 L4RE\_ELF\_AUX\_ELEM\_T

```
#define L4RE_ELF_AUX_ELEM_T(
    type,
    id,
    tag,
    val... )    static L4RE_ELF_AUX_ELEM type id = {tag, sizeof(type), val}
```

Define an auxiliary vector element.

#### Parameters

<i>type</i>	is the data type for the element (e.g., <a href="#">l4re_elf_aux_vma_t</a> )
<i>id</i>	is the identifier (variable name) for the declaration (the variable is defined with <code>static</code> storage class)
<i>tag</i>	is the tag value for the element e.g., <a href="#">L4RE_ELF_AUX_T_VMA</a>
<i>val</i>	are the values to be set in the descriptor

Usage:

```
L4RE_ELF_AUX_ELEM_T(l4re_elf_aux_vma_t, decl_name, L4RE_ELF_AUX_T_VMA, 0x2000, 0x4000 );
```

Definition at line 67 of file [elf\\_aux.h](#).

### 13.10.7.3 Enumeration Type Documentation

#### 13.10.7.3.1 anonymous enum

anonymous enum

## Enumerator

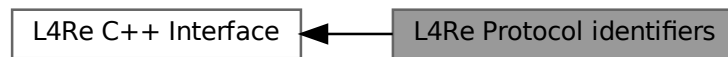
L4RE_ELF_AUX_T_NONE	Tag for an invalid element in the auxiliary vector.
L4RE_ELF_AUX_T_VMA	Tag for descriptor for a reserved virtual memory area.
L4RE_ELF_AUX_T_STACK_SIZE	Tag for descriptor that defines the stack size for the first application thread.
L4RE_ELF_AUX_T_STACK_ADDR	Tag for descriptor that defines the stack address for the first application thread.
L4RE_ELF_AUX_T_KIP_ADDR	Tag for descriptor that defines the KIP address for the binaries address space.
L4RE_ELF_AUX_T_EX_REGS_FLAGS	Tag for descriptor to override ex_regs() flags.

Definition at line 70 of file [elf\\_aux.h](#).

### 13.10.8 L4Re Protocol identifiers

Fix [L4Re](#) Protocol Constants.

Collaboration diagram for L4Re Protocol identifiers:



### Enumerations

- enum [L4Re::Dataspace\\_::Opcodes](#)  
*Data-space communication-protocol opcodes.*
- enum [L4Re::Event\\_::Opcodes](#)  
*Event communication-protocol opcodes.*
- enum [L4Re::Inhibitor\\_::Opcodes](#)  
*Inhibitor communication-protocol opcodes.*
- enum [L4Re::Log\\_::Opcodes](#)  
*Logging-service communication-protocol opcodes.*
- enum [L4Re::Mem\\_alloc\\_::Opcodes](#)  
*Memory-allocator communication-protocol opcodes.*
- enum [L4Re::Namespace\\_::Opcodes](#)  
*Name-space communication-protocol opcodes.*
- enum [L4Re::Parent\\_::Opcodes](#)  
*Parent communication-protocol opcodes.*
- enum [L4Re::Rm\\_::Opcodes](#)  
*Region-map communication-protocol opcodes.*
- enum [L4Re::Video::Goos\\_::Opcodes](#)  
*Frame buffer communication-protocol opcodes.*



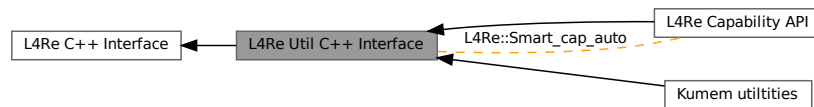
### 13.10.8.1 Detailed Description

Fix [L4Re](#) Protocol Constants.

### 13.10.9 L4Re Util C++ Interface

Documentation of the [L4](#) Runtime Environment utility functionality in C++.

Collaboration diagram for L4Re Util C++ Interface:



#### Modules

- [Kumem utilities](#)
- [L4Re Capability API](#)

#### Data Structures

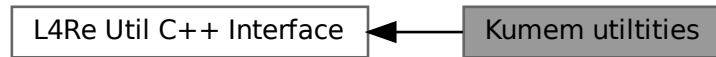
- class [L4Re::Smart\\_cap\\_auto< Unmap\\_flags >](#)  
*Helper for Unique\_cap and Unique\_del\_cap.*
- class [L4Re::Util::Cap\\_alloc\\_base](#)  
*Capability allocator.*
- class [L4Re::Util::Br\\_manager](#)  
*Buffer-register (BR) manager for L4::Server.*
- class [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE, Dbg >](#)  
*Internal reference-counting cap allocator.*
- class [L4Re::Util::Event\\_buffer\\_t< PAYLOAD >](#)  
*Event\_buffer utility class.*
- class [L4Re::Util::Event\\_buffer\\_consumer\\_t< PAYLOAD >](#)  
*An event buffer consumer.*
- class [L4Re::Util::Vcon\\_svr< SVR >](#)  
*Console server template class.*
- class [L4Re::Util::Video::Goos\\_svr](#)  
*Goos server class.*

### 13.10.9.1 Detailed Description

Documentation of the [L4](#) Runtime Environment utility functionality in C++.

### 13.10.9.2 Kumem utilities

Collaboration diagram for Kumem utilities:



#### Functions

- `int L4Re::Util::kumem_alloc (l4_addr_t *mem, unsigned pages_order, L4::Cap< L4::Task > task=L4Re::Env::env() ->task(), L4::Cap< L4Re::Rm > rm=L4Re::Env::env() ->rm()) noexcept`  
*Allocate state area.*

#### 13.10.9.2.1 Detailed Description

#### 13.10.9.2.2 Function Documentation

##### 13.10.9.2.2.1 kumem\_alloc()

```

int L4Re::Util::kumem_alloc (
    l4_addr_t * mem,
    unsigned pages_order,
    L4::Cap< L4::Task > task = L4Re::Env::env() ->task(),
    L4::Cap< L4Re::Rm > rm = L4Re::Env::env() ->rm() ) [noexcept]

```

Allocate state area.

#### Parameters

out	<i>mem</i>	Pointer to memory that has been allocated.
	<i>pages_order</i>	Size to allocate, in log2 pages.
	<i>task</i>	Task to use for allocation.
	<i>rm</i>	Region manager to use for allocation.

#### Return values

0	for success
<0	error code on failure

**Note**

The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page. A portable implementation should not depend on allocations greater than 16KiB to succeed.

References [L4Re::Util::kumem\\_alloc\(\)](#).

Referenced by [L4Re::Util::kumem\\_alloc\(\)](#).

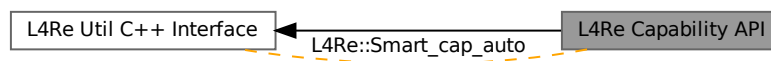
Here is the call graph for this function:



Here is the caller graph for this function:

**13.10.9.3 L4Re Capability API**

Collaboration diagram for L4Re Capability API:



## Data Structures

- class [L4Re::Cap\\_alloc](#)  
*Capability allocator interface.*
- class [L4Re::Smart\\_cap\\_auto< Unmap\\_flags >](#)  
*Helper for [Unique\\_cap](#) and [Unique\\_del\\_cap](#).*
- class [L4Re::Smart\\_count\\_cap< Unmap\\_flags >](#)  
*Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).*
- class [L4Re::Util::Smart\\_cap\\_auto< Unmap\\_flags >](#)  
*Helper for [Unique\\_cap](#) and [Unique\\_del\\_cap](#).*
- class [L4Re::Util::Smart\\_count\\_cap< Unmap\\_flags >](#)  
*Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).*
- struct [L4Re::Util::Ref\\_cap< T >](#)  
*Automatic capability that implements automatic free and unmap of the capability selector.*
- struct [L4Re::Util::Ref\\_del\\_cap< T >](#)  
*Automatic capability that implements automatic free and unmap+delete of the capability selector.*

## Functions

- template<typename T >  
[Ref\\_cap< T >::Cap L4Re::Util::make\\_ref\\_cap \(\)](#)  
*Allocate a capability slot and wrap it in a [Ref\\_cap](#).*
- template<typename T >  
[Ref\\_del\\_cap< T >::Cap L4Re::Util::make\\_ref\\_del\\_cap \(\)](#)  
*Allocate a capability slot and wrap it in a [Ref\\_del\\_cap](#).*
- virtual [L4Re::Cap\\_alloc::~~Cap\\_alloc \(\)=0](#)  
*Destructor.*

## Variables

- [\\_Cap\\_alloc](#) & [L4Re::Util::cap\\_alloc](#)  
*Capability allocator.*

### 13.10.9.3.1 Detailed Description

### 13.10.9.3.2 Function Documentation

#### 13.10.9.3.2.1 [make\\_ref\\_cap\(\)](#)

```
template<typename T >
Ref\_cap< T >::Cap L4Re::Util::make\_ref\_cap \( \)
```

Allocate a capability slot and wrap it in a [Ref\\_cap](#).

#### Template Parameters

<a href="#">T</a>	Type of capability the slot is used for.
-------------------	--

Definition at line 206 of file [cap\\_alloc](#).

References [L4Re::Util::cap\\_alloc](#).

### 13.10.9.3.2.2 make\_ref\_del\_cap()

```
template<typename T >
Ref_del_cap< T >::Cap L4Re::Util::make_ref_del_cap ( )
```

Allocate a capability slot and wrap it in a [Ref\\_del\\_cap](#).

#### Template Parameters

<i>T</i>	Type of capability the slot is used for.
----------	--

Definition at line 215 of file [cap\\_alloc](#).

References [L4Re::Util::cap\\_alloc](#).

### 13.10.9.3.3 Variable Documentation

#### 13.10.9.3.3.1 cap\_alloc

```
_Cap_alloc& L4Re::Util::cap_alloc [extern]
```

Capability allocator.

This is the instance of the capability allocator that is used by usual applications.

The capability allocator uses the [Counting\\_cap\\_alloc](#), a reference-counting thread-safe capability allocator, that keeps a reference counter for each managed capability selector.

#### Examples

[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), [examples/libs/l4re/c++/shared\\_ds/d](#)  
and [examples/libs/l4re/streammap/client.cc](#).

Referenced by [L4Re::Util::Br\\_manager::alloc\\_buffer\\_demand\(\)](#), [L4Re::Util::Smart\\_count\\_cap< Unmap\\_flags >::copy\(\)](#), [L4Re::Util::Smart\\_cap\\_auto< Unmap\\_flags >::free\(\)](#), [L4Re::Util::Smart\\_count\\_cap< Unmap\\_flags >::free\(\)](#), [L4Re::Util::make\\_ref\\_cap\(\)](#), [L4Re::Util::make\\_ref\\_del\\_cap\(\)](#), [L4Re::Util::make\\_shared\\_cap\(\)](#), [L4Re::Util::make\\_shared\\_del\\_cap\(\)](#), [L4Re::Util::make\\_unique\\_cap\(\)](#), [L4Re::Util::make\\_unique\\_del\\_cap\(\)](#), [L4Re::Util::Br\\_manager::realloc\\_rcv\\_cap\(\)](#), and [L4Re::Util::Object\\_registry::unregister\\_obj\(\)](#).

## 13.10.10 Logging interface

Interface for log output.

Collaboration diagram for Logging interface:



## Data Structures

- class [L4Re::Log](#)  
*Log interface class.*

### 13.10.10.1 Detailed Description

Interface for log output.

The logging interface provides a facility sending log output. One purpose of the interface is to serialize the output and provide the possibility to tag output sent to a specific log object.

### 13.10.11 Name-space API

API for name spaces that store capabilities.

Collaboration diagram for Name-space API:



## Data Structures

- class [L4Re::Namespace](#)  
*Name-space interface.*

### 13.10.11.1 Detailed Description

API for name spaces that store capabilities.

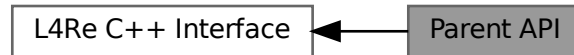
This is a basic abstraction for managing a mapping from human-readable names to capabilities. In particular, a name can also be mapped to a capability that refers to another name space object. By this means name spaces can be constructed hierarchically.

Name spaces play a central role in [L4Re](#), because the implementation of the name space objects determines the policy which capabilities (which objects) are accessible to a client of a name space.

### 13.10.12 Parent API

[Parent](#) interface.

Collaboration diagram for Parent API:



#### Data Structures

- class [L4Re::Parent](#)  
*[Parent](#) interface.*

#### 13.10.12.1 Detailed Description

[Parent](#) interface.

The parent interface provides means for an [L4](#) task to signal changes in its execution state. The main purpose is to signal program termination to the program that started it, so that its resources can be reclaimed. In a typical [L4Re](#) system, this program will be Moe or Ned.

See also

[L4Re::Parent](#) for information about the concrete interface.

### 13.10.13 Region map API

Virtual address-space management.

Collaboration diagram for Region map API:



## Data Structures

- class [L4Re::Rm](#)  
*Region map.*

### 13.10.13.1 Detailed Description

Virtual address-space management.

A region map object implements two protocols. The first protocol is the kernel page-fault protocol, to resolve page faults for threads running in an [L4](#) task. The second protocol is the region map protocol itself, which allows managing the virtual memory address space of an [L4](#) task.

There are two basic concepts provided by the region map abstraction:

- **Areas** are reserved ranges in the virtual memory address space.
- **Regions** are ranges that are backed by (part of) a dataspace, i.e. accessing them results in access to the physical memory the dataspace manages.

Note that regions may live outside of areas and that an area does not necessarily contain any region.

Areas can be reserved for special use or for attaching a dataspace at a later time. When attaching a dataspace, the user can instruct the region map to search for an appropriate range to attach to. Regions are skipped in this search since they already have dataspace attached to them, and, depending on [L4Re::Rm::F::In\\_area](#), areas are skipped because they are reserved. Amongst others, areas can be used to attach several dataspace inside a certain range of addresses without interference from other threads.

When a region map receives a page fault IPC, the region map will check if the faulting virtual address lies in a region. If yes, it will answer the page fault IPC with a mapping from the backing dataspace. If not, an error is returned.

Depending on the system type, an attached dataspace might or might not be mapped eagerly. MMU-based systems resort to lazy mapping while systems without MMU do eager mappings by default. The [L4Re::Rm::F::Eager\\_map](#) and [L4Re::Rm::F::No\\_eager\\_map](#) flags can be used to force the respective behaviour, independent of the underlying system. In case both flags are given, the [L4Re::Rm::F::No\\_eager\\_map](#) flag wins.

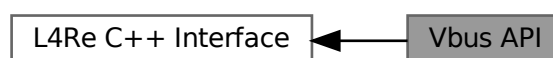
See also

[L4Re::Dataspace](#), [L4Re::Rm](#),  
[Memory management - Data Spaces and the Region Map](#)

### 13.10.14 Vbus API

C++ interface of the Vbus API.

Collaboration diagram for Vbus API:





## Data Structures

- class [L4vbus::Pm< DEC >](#)  
*Power-management API mixin.*
- class [L4vbus::Device](#)  
*Device on a [L4vbus::Vbus](#).*
- class [L4vbus::Icu](#)  
*Vbus Interrupt controller API.*
- class [L4vbus::Vbus](#)  
*The virtual bus ([Vbus](#)) interface.*
- class [L4vbus::Gpio\\_pin](#)  
*A GPIO pin.*
- class [L4vbus::Gpio\\_module](#)  
*A [Gpio\\_module](#) groups multiple GPIO pins together.*
- class [L4vbus::Pci\\_host\\_bridge](#)  
*A Pci host bridge.*
- class [L4vbus::Pci\\_dev](#)  
*A PCI device.*

### 13.10.14.1 Detailed Description

C++ interface of the Vbus API.

The virtual bus (Vbus) is a hierarchical (tree) structure of device nodes where each device has a set of resources attached to it. Each virtual bus provides an Icu ([Interrupt controller](#)) for interrupt handling.

The Vbus interface allows a client to find and query devices present on his virtual bus. After obtaining a device handle for a specific device the client can enumerate its resources.

Refer to [L4 Vbus functions](#) for the C API.

#### Include File

```
#include <l4/vbus/vbus>
```

#### Include File

```
#include <l4/vbus/vbus_gpio>
```

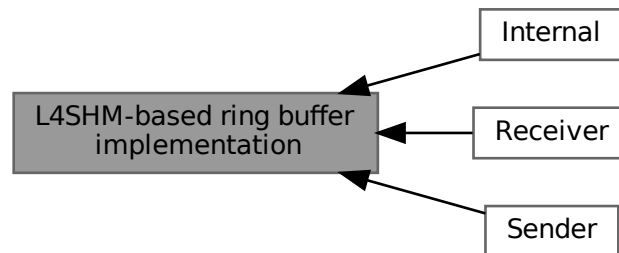
#### Include File

```
#include <l4/vbus/vbus_pci>
```

## 13.11 L4SHM-based ring buffer implementation

The library provides a non-locking (strictly 1:1) shared-memory-based ring buffer implementation based on the L4SHM library.

Collaboration diagram for L4SHM-based ring buffer implementation:



### Modules

- [Internal](#)
- [Receiver](#)
- [Sender](#)

### 13.11.1 Detailed Description

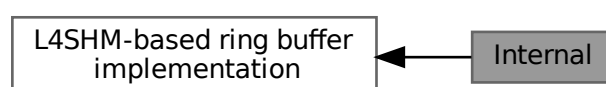
The library provides a non-locking (strictly 1:1) shared-memory-based ring buffer implementation based on the L4SHM library.

It requires an already allocated L4SHM area to be attached to sender and receiver. It will allocate an SHM chunk within this area and provides functions to produce data and consume data in FIFO order from the ring buffer.

The sender side of the buffer needs to be initialized *before* the receiver side, because allocation of the SHM chunk and the necessary signals is done on the sender side and the receiver initialization tries to attach to these objects.

### 13.11.2 Internal

Collaboration diagram for Internal:



## Data Structures

- struct [l4shmc\\_ringbuf\\_head\\_t](#)  
*Head field of a ring buffer.*
- struct [l4shmc\\_ringbuf\\_t](#)  
*Ring buffer.*

## Macros

- #define [L4SHMC\\_RINGBUF\\_HEAD](#)(ringbuf) (([l4shmc\\_ringbuf\\_head\\_t](#)\*)((ringbuf)->\_addr))  
*Get ring buffer head pointer.*
- #define [L4SHMC\\_RINGBUF\\_DATA](#)(ringbuf) ([L4SHMC\\_RINGBUF\\_HEAD](#)(ringbuf)->data)  
*Get ring buffer data pointer.*
- #define [L4SHMC\\_RINGBUF\\_DATA\\_SIZE](#)(ringbuf) ((ringbuf)->\_size - sizeof([l4shmc\\_ringbuf\\_head\\_t](#)))  
*Get size of data area.*

### 13.11.2.1 Detailed Description

### 13.11.2.2 Macro Definition Documentation

#### 13.11.2.2.1 L4SHMC\_RINGBUF\_DATA

```
#define L4SHMC_RINGBUF_DATA(  
    ringbuf ) (L4SHMC\_RINGBUF\_HEAD(ringbuf)->data)
```

Get ring buffer data pointer.

#### Parameters

<i>ringbuf</i>	<a href="#">l4shmc_ringbuf_t</a> struct
----------------	---

Definition at line [113](#) of file [ringbuf.h](#).

#### 13.11.2.2.2 L4SHMC\_RINGBUF\_DATA\_SIZE

```
#define L4SHMC_RINGBUF_DATA_SIZE(  
    ringbuf ) ((ringbuf)->_size - sizeof(l4shmc\_ringbuf\_head\_t))
```

Get size of data area.

#### Parameters

<i>ringbuf</i>	<a href="#">l4shmc_ringbuf_t</a> struct
----------------	---

Definition at line [122](#) of file [ringbuf.h](#).

### 13.11.2.2.3 L4SHMC\_RINGBUF\_HEAD

```
#define L4SHMC_RINGBUF_HEAD(  
    ringbuf ) ((l4shmc_ringbuf_head_t*)((ringbuf)->_addr))
```

Get ring buffer head pointer.

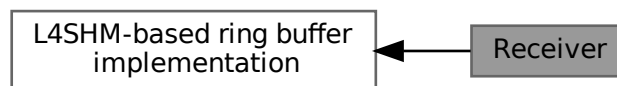
Parameters

<i>ringbuf</i>	<a href="#">l4shmc_ringbuf_t</a> struct
----------------	---

Definition at line 104 of file [ringbuf.h](#).

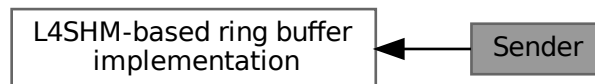
### 13.11.3 Receiver

Collaboration diagram for Receiver:



### 13.11.4 Sender

Collaboration diagram for Sender:



## 13.12 Server-Side IPC framework

Server-Side framework for implementing object-oriented servers.

## Namespaces

- namespace `L4::lpc_svr`  
*Helper classes for `L4::Server` instantiation.*

## Data Structures

- class `L4::Server_object`  
*Abstract server object to be used with `L4::Server` and `L4::Basic_registry`.*
- struct `L4::Server_object_t< IFACE, BASE >`  
*Base class (template) for server implementing server objects.*
- struct `L4::Server_object_x< Derived, IFACE, BASE >`  
*Helper class to implement `p_dispatch` based server objects.*
- struct `L4::lrq_handler_object`  
*Server object base class for handling IRQ messages.*
- class `L4::lpc_svr::Timeout`  
*Callback interface for `Timeout_queue`.*
- class `L4::lpc_svr::Timeout_queue`  
*Timeout queue to be used in `l4re` server loop.*
- class `L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >`  
*Loop hooks mixin for integrating a timeout queue into the server loop.*
- class `L4::lpc_svr::Server_iface`  
*Interface for server-loop related functions.*
- class `L4::Basic_registry`  
*This registry returns the corresponding server object based on the label of an `lpc_gate`.*
- struct `L4::lpc_svr::Ignore_errors`  
*Mix in for `LOOP_HOOKS` to ignore IPC errors.*
- struct `L4::lpc_svr::Default_timeout`  
*Mix in for `LOOP_HOOKS` to use a 0 send and a infinite receive timeout.*
- struct `L4::lpc_svr::Compound_reply`  
*Mix in for `LOOP_HOOKS` to always use compound reply and wait.*
- struct `L4::lpc_svr::Default_setup_wait`  
*Mix in for `LOOP_HOOKS` for `setup_wait` no op.*
- class `L4::lpc_svr::Br_manager_no_buffers`  
*Empty implementation of `Server_iface`.*
- struct `L4::lpc_svr::Default_loop_hooks`  
*Default `LOOP_HOOKS`.*
- class `L4::Server< LOOP_HOOKS >`  
*Basic server loop for handling client requests.*

## Enumerations

- enum `L4::lpc_svr::Reply_mode` { `L4::lpc_svr::Reply_compound` , `L4::lpc_svr::Reply_separate` }  
*Reply mode for server loop.*

### 13.12.1 Detailed Description

Server-Side framework for implementing object-oriented servers.

## 13.12.2 Enumeration Type Documentation

### 13.12.2.1 Reply\_mode

```
enum L4::Ipc_svr::Reply_mode
```

Reply mode for server loop.

The reply mode specifies if the server loop shall do a compound reply and wait operation ([Reply\\_compound](#)), which is the most performant method. Note, `setup_wait()` is called before the reply. The other way is to call reply and wait separately and call `setup_wait` in between.

The actual mode is determined by the return value of the `before_reply()` hook in the `LOOP_HOOKS` of [L4::Server](#).

Enumerator

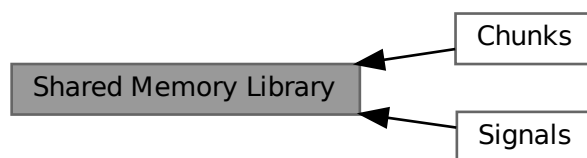
<code>Reply_compound</code>	<a href="#">Server</a> shall use a compound reply and wait (fast).
<code>Reply_separate</code>	<a href="#">Server</a> shall call reply and wait separately.

Definition at line 50 of file [ipc\\_server\\_loop](#).

## 13.13 Shared Memory Library

L4SHM provides a shared memory infrastructure that establishes a shared memory area between multiple parties and uses a fast notification mechanism.

Collaboration diagram for Shared Memory Library:



### Modules

- [Chunks](#)
- [Signals](#)

## Functions

- long [l4shmc\\_create](#) (char const \*shmc\_name)  
*Create a shared memory area.*
- long [l4shmc\\_attach](#) (char const \*shmc\_name, l4shmc\_area\_t \*shmarea)  
*Attach to a shared memory area.*
- long [l4shmc\\_get\\_client\\_nr](#) (l4shmc\_area\_t const \*shmarea)  
*Determine the client number of the shared memory region.*
- long [l4shmc\\_mark\\_client\\_initialized](#) (l4shmc\_area\_t \*shmarea)  
*Mark this shared memory client as 'initialized'.*
- long [l4shmc\\_get\\_initialized\\_clients](#) (l4shmc\_area\_t \*shmarea, l4\_umword\_t \*bitmask)  
*Fetch the `_clients_init_done` bitmask of the shared memory area.*
- long [l4shmc\\_connect\\_chunk\\_signal](#) (l4shmc\_chunk\_t \*chunk, l4shmc\_signal\_t \*signal)  
*Connect a signal with a chunk.*
- long [l4shmc\\_area\\_size](#) (l4shmc\_area\_t const \*shmarea)  
*Get size of shared memory area.*
- long [l4shmc\\_area\\_size\\_free](#) (l4shmc\_area\_t const \*shmarea)  
*Get free size of shared memory area.*
- long [l4shmc\\_area\\_overhead](#) (void)  
*Get memory overhead per area that is not available for chunks.*
- long [l4shmc\\_chunk\\_overhead](#) (void)  
*Get memory overhead required in addition to the chunk capacity for adding one chunk.*

### 13.13.1 Detailed Description

L4SHM provides a shared memory infrastructure that establishes a shared memory area between multiple parties and uses a fast notification mechanism.

A shared memory area consists of chunks and signals. A chunk is a defined chunk of memory within the memory area with a maximum size. A chunk is filled (written) by a producer and read by a consumer. When a producer has finished writing to the chunk it signals a data ready notification to the consumer.

A consumer attaches to a chunk and waits for the producer to fill the chunk. After reading out the chunk it marks the chunk free again.

A shared memory area can have multiple chunks.

The interface is divided in three roles.

- The master role, responsible for setting up a shared memory area.
- A producer, generating data into a chunk
- A consumer, receiving data.

A signal can be connected with a chunk or can be used independently (e.g. for multiple chunks).

## 13.13.2 Function Documentation

### 13.13.2.1 l4shmc\_area\_overhead()

```
long l4shmc_area_overhead (
    void )
```

Get memory overhead per area that is not available for chunks.

#### Returns

Size of the overhead in bytes.

### 13.13.2.2 l4shmc\_area\_size()

```
long l4shmc_area_size (
    l4shmc_area_t const * shmarea )
```

Get size of shared memory area.

#### Parameters

<i>shmarea</i>	Shared memory area.
----------------	---------------------

#### Return values

$>0$	Size of the shared memory area.
$<0$	Error.

### 13.13.2.3 l4shmc\_area\_size\_free()

```
long l4shmc_area_size_free (
    l4shmc_area_t const * shmarea )
```

Get free size of shared memory area.

To get the max size to pass to `l4shmc_add_chunk`, subtract [l4shmc\\_chunk\\_overhead\(\)](#).

#### Parameters

<i>shmarea</i>	Shared memory area.
----------------	---------------------

#### Returns

Size of the shared memory area.



### 13.13.2.4 l4shmc\_attach()

```
long l4shmc_attach (
    char const * shmc_name,
    l4shmc_area_t * shmarea )
```

Attach to a shared memory area.

#### Parameters

	<i>shmc_name</i>	Name of the shared memory area.
out	<i>shmarea</i>	Pointer to shared memory area descriptor to be filled with information for the shared memory area.

On success, the data in 'shmarea' contains a client number which can be used to mutual agree about client initialization:

- [l4shmc\\_get\\_client\\_nr\(\)](#) returns the client number stored in 'shmarea'. The first attached client will get 0 and this number is increased for each attached client.
- [l4shmc\\_mark\\_client\\_initialized\(\)](#) tells other clients that this client has finished its initialization.
- [l4shmc\\_get\\_initialized\\_clients\(\)](#) returns the bitmap of initialized clients attached to this shared memory.

#### Return values

0	Success.
<0	Error.

#### Examples

[examples/libs/shmc/prodcons.c](#).

### 13.13.2.5 l4shmc\_chunk\_overhead()

```
long l4shmc_chunk_overhead (
    void )
```

Get memory overhead required in addition to the chunk capacity for adding one chunk.

#### Returns

Size of the overhead in bytes.

### 13.13.2.6 l4shmc\_connect\_chunk\_signal()

```
long l4shmc_connect_chunk_signal (
    l4shmc_chunk_t * chunk,
    l4shmc_signal_t * signal )
```

Connect a signal with a chunk.

## Parameters

<i>chunk</i>	Chunk to attach the signal to.
<i>signal</i>	Signal to attach.

## Return values

0	Success.
<0	Error.

## Examples

[examples/libs/shmc/prodcons.c](#).

**13.13.2.7 l4shmc\_create()**

```
long l4shmc_create (
    char const * shmc_name )
```

Create a shared memory area.

## Parameters

<i>shmc_name</i>	Name of the shared memory area.
------------------	---------------------------------

## Return values

0	Success.
-L4_ENOMEM	The requested size is too big.
-L4_ENOENT	No valid capability with the name of the shared memory area found.
<0	Errors from l4re_rm_attach or l4re_ns_register_obj_srv.

## Examples

[examples/libs/shmc/prodcons.c](#).

**13.13.2.8 l4shmc\_get\_client\_nr()**

```
long l4shmc_get_client_nr (
    l4shmc_area_t const * shmarea )
```

Determine the client number of the shared memory region.

## Parameters

<i>shmarea</i>	The shared memory area.
----------------	-------------------------

**Returns**

client number.

**13.13.2.9 l4shmc\_get\_initialized\_clients()**

```
long l4shmc_get_initialized_clients (
    l4shmc_area_t * shmarea,
    l4_umword_t * bitmask )
```

Fetch the `_clients_init_done` bitmask of the shared memory area.

**Parameters**

	<i>shmarea</i>	The shared memory area.
out	<i>bitmask</i>	The bitmask describing which clients are initialized.

**Return values**

0	Success.
<0	Error.

**See also**

[l4shmc\\_mark\\_client\\_initialized\(\)](#), [l4shmc\\_get\\_client\\_nr\(\)](#)

**Examples**

[examples/libs/shmc/prodcons.c](#).

**13.13.2.10 l4shmc\_mark\_client\_initialized()**

```
long l4shmc_mark_client_initialized (
    l4shmc_area_t * shmarea )
```

Mark this shared memory client as 'initialized'.

The corresponding bit is set in the `_clients_init_done` bitmask. The bitmask can be fetched with [l4shmc\\_get\\_initialized\\_clients\(\)](#).

**Parameters**

<i>shmarea</i>	The shared memory area.
----------------	-------------------------

**Return values**

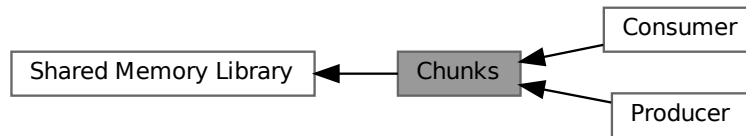
0	Success.
<0	Error.

## Examples

[examples/libs/shmc/prodcons.c](#).

### 13.13.3 Chunks

Collaboration diagram for Chunks:



## Modules

- [Consumer](#)
- [Producer](#)

## Functions

- long [l4shmc\\_add\\_chunk](#) (l4shmc\_area\_t \*shmarea, char const \*chunk\_name, [l4\\_umword\\_t](#) chunk\_capacity, l4shmc\_chunk\_t \*chunk)  
*Add a chunk in the shared memory area.*
- long [l4shmc\\_get\\_chunk](#) (l4shmc\_area\_t \*shmarea, char const \*chunk\_name, l4shmc\_chunk\_t \*chunk)  
*Get chunk out of shared memory area.*
- long [l4shmc\\_get\\_chunk\\_to](#) (l4shmc\_area\_t \*shmarea, char const \*chunk\_name, [l4\\_umword\\_t](#) timeout\_ms, l4shmc\_chunk\_t \*chunk)  
*Get chunk out of shared memory area, with timeout.*
- long [l4shmc\\_iterate\\_chunk](#) (l4shmc\_area\_t const \*shmarea, char const \*\*chunk\_name, long offs)  
*Iterate over names of all existing chunks.*
- void \* [l4shmc\\_chunk\\_ptr](#) (l4shmc\_chunk\_t const \*chunk)  
*Get data pointer to chunk.*
- long [l4shmc\\_chunk\\_capacity](#) (l4shmc\_chunk\_t const \*chunk)  
*Get capacity of a chunk.*
- l4shmc\_signal\_t \* [l4shmc\\_chunk\\_signal](#) (l4shmc\_chunk\_t const \*chunk)  
*Get the registered signal of a chunk.*

#### 13.13.3.1 Detailed Description

#### 13.13.3.2 Function Documentation

##### 13.13.3.2.1 l4shmc\_add\_chunk()

```

long l4shmc_add_chunk (
    l4shmc_area_t * shmarea,
    char const * chunk_name,
    l4\_umword\_t chunk_capacity,
    l4shmc_chunk_t * chunk )
  
```

Add a chunk in the shared memory area.

## Parameters

	<i>shmarea</i>	The shared memory area to put the chunk in.
	<i>chunk_name</i>	Name of the chunk.
	<i>chunk_capacity</i>	Capacity for payload of the chunk in bytes.
out	<i>chunk</i>	Chunk structure to fill in.

## Return values

0	Success.
<0	Error.

## Examples

[examples/libs/shmc/prodcons.c](#).

**13.13.3.2.2 l4shmc\_chunk\_capacity()**

```
long l4shmc_chunk_capacity (  
    l4shmc_chunk_t const * chunk ) [inline]
```

Get capacity of a chunk.

## Parameters

<i>chunk</i>	Chunk.
--------------	--------

## Returns

Capacity of the chunk in bytes.

**13.13.3.2.3 l4shmc\_chunk\_ptr()**

```
void * l4shmc_chunk_ptr (  
    l4shmc_chunk_t const * chunk ) [inline]
```

Get data pointer to chunk.

## Parameters

<i>chunk</i>	Chunk.
--------------	--------

## Returns

Chunk pointer.

## Examples

[examples/libs/shmc/prodcons.c](#).

#### 13.13.3.2.4 l4shmc\_chunk\_signal()

```
l4shmc_signal_t * l4shmc_chunk_signal (
    l4shmc_chunk_t const * chunk ) [inline]
```

Get the registered signal of a chunk.

##### Parameters

<i>chunk</i>	Chunk.
--------------	--------

##### Return values

0	No signal has been registered with this chunk.
<i>!=0</i>	Pointer to signal otherwise.

#### 13.13.3.2.5 l4shmc\_get\_chunk()

```
long l4shmc_get_chunk (
    l4shmc_area_t * shmbarea,
    char const * chunk_name,
    l4shmc_chunk_t * chunk ) [inline]
```

Get chunk out of shared memory area.

##### Parameters

	<i>shmbarea</i>	Shared memory area.
	<i>chunk_name</i>	Name of the chunk.
out	<i>chunk</i>	Chunk data structure to fill.

##### Return values

0	Success.
<0	Error.

##### Examples

[examples/libs/shmc/prodcons.c](#).

#### 13.13.3.2.6 l4shmc\_get\_chunk\_to()

```
long l4shmc_get_chunk_to (
    l4shmc_area_t * shmbarea,
    char const * chunk_name,
    l4_umword_t timeout_ms,
    l4shmc_chunk_t * chunk )
```

Get chunk out of shared memory area, with timeout.

## Parameters

	<i>shmbarea</i>	Shared memory area.
	<i>chunk_name</i>	Name of the chunk.
	<i>timeout_ms</i>	Timeout in milliseconds to wait for the chunk to appear in the shared memory area.
out	<i>chunk</i>	Chunk data structure to fill.

## Return values

0	Success.
<0	Error.

## 13.13.3.2.7 l4shmc\_iterate\_chunk()

```
long l4shmc_iterate_chunk (
    l4shmc_area_t const * shmbarea,
    char const ** chunk_name,
    long offs )
```

Iterate over names of all existing chunks.

## Parameters

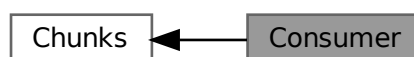
<i>shmbarea</i>	Shared memory area.
<i>chunk_name</i>	Where the name of the current chunk will be stored
<i>offs</i>	0 to start iteration, return value of previous call to <a href="#">l4shmc_iterate_chunk()</a> to get next chunk

## Return values

0	No more chunks available.
<0	Error.
>0	Iterator value for the next call.

## 13.13.3.3 Consumer

Collaboration diagram for Consumer:



## Functions

- long [l4shmc\\_chunk\\_try\\_to\\_take\\_for\\_reading](#) (l4shmc\_chunk\_t \*chunk)  
*Try to mark chunk busy reading.*
- long [l4shmc\\_enable\\_chunk](#) (l4shmc\_chunk\_t \*chunk)  
*Enable a signal connected with a chunk.*
- long [l4shmc\\_wait\\_chunk](#) (l4shmc\_chunk\_t \*chunk)  
*Wait on a specific chunk.*
- long [l4shmc\\_wait\\_chunk\\_to](#) (l4shmc\_chunk\_t \*chunk, [l4\\_timeout\\_t](#) timeout)  
*Check whether a specific chunk has an event pending, with timeout.*
- long [l4shmc\\_wait\\_chunk\\_try](#) (l4shmc\_chunk\_t \*chunk)  
*Check whether a specific chunk has an event pending.*
- long [l4shmc\\_chunk\\_consumed](#) (l4shmc\_chunk\_t \*chunk)  
*Mark a chunk as free.*
- long [l4shmc\\_is\\_chunk\\_ready](#) (l4shmc\_chunk\_t const \*chunk)  
*Check whether data is available.*
- long [l4shmc\\_chunk\\_size](#) (l4shmc\_chunk\_t const \*chunk)  
*Get current size of a chunk.*

### 13.13.3.3.1 Detailed Description

### 13.13.3.3.2 Function Documentation

#### 13.13.3.3.2.1 l4shmc\_chunk\_consumed()

```
long l4shmc_chunk_consumed (
    l4shmc_chunk_t * chunk ) [inline]
```

Mark a chunk as free.

#### Parameters

<i>chunk</i>	Chunk to mark as free.
--------------	------------------------

#### Return values

0	Success.
<0	Error.

#### Examples

[examples/libs/shmc/prodcons.c](#).

#### 13.13.3.3.2.2 l4shmc\_chunk\_size()

```
long l4shmc_chunk_size (
    l4shmc_chunk_t const * chunk ) [inline]
```

Get current size of a chunk.



## Parameters

<i>chunk</i>	Chunk.
--------------	--------

## Returns

Current size of the chunk in bytes.

## Examples

[examples/libs/shmc/prodcons.c](#).

**13.13.3.3.2.3 l4shmc\_chunk\_try\_to\_take\_for\_reading()**

```
long l4shmc_chunk_try_to_take_for_reading (  
    l4shmc_chunk_t * chunk ) [inline]
```

Try to mark chunk busy reading.

## Parameters

<i>chunk</i>	chunk to mark busy reading.
--------------	-----------------------------

## Return values

0	Chunk could be taken and can be read.
<0	Chunk could not be taken, try again.

**13.13.3.3.2.4 l4shmc\_enable\_chunk()**

```
long l4shmc_enable_chunk (  
    l4shmc_chunk_t * chunk )
```

Enable a signal connected with a chunk.

## Parameters

<i>chunk</i>	Chunk to enable.
--------------	------------------

## Return values

0	Success.
<0	Error.

A signal must be enabled before waiting when the consumer waits on any signal. Enabling is not needed if the consumer waits for a specific signal or chunk.

**13.13.3.3.2.5 l4shmc\_is\_chunk\_ready()**

```
long l4shmc_is_chunk_ready (
    l4shmc_chunk_t const * chunk ) [inline]
```

Check whether data is available.

**Parameters**

<i>chunk</i>	Chunk to check.
--------------	-----------------

**Return values**

<i>!=0</i>	Data is available.
<i>0</i>	No data available.

**13.13.3.3.2.6 l4shmc\_wait\_chunk()**

```
long l4shmc_wait_chunk (
    l4shmc_chunk_t * chunk ) [inline]
```

Wait on a specific chunk.

**Parameters**

<i>chunk</i>	Chunk to wait for.
--------------	--------------------

**Return values**

<i>0</i>	Success.
<i>&lt;0</i>	Error.

**Examples**

[examples/libs/shmc/prodcons.c](#).

**13.13.3.3.2.7 l4shmc\_wait\_chunk\_to()**

```
long l4shmc_wait_chunk_to (
    l4shmc_chunk_t * chunk,
    l4_timeout_t timeout )
```

Check whether a specific chunk has an event pending, with timeout.

**Parameters**

<i>chunk</i>	Chunk to check.
<i>timeout</i>	Timeout.

## Return values

0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

**13.13.3.3.2.8 l4shmc\_wait\_chunk\_try()**

```
long l4shmc_wait_chunk_try (
    l4shmc_chunk_t * chunk ) [inline]
```

Check whether a specific chunk has an event pending.

## Parameters

<i>chunk</i>	Chunk to check.
--------------	-----------------

## Return values

0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

**13.13.3.4 Producer**

Collaboration diagram for Producer:

**Functions**

- long [l4shmc\\_chunk\\_try\\_to\\_take](#) (l4shmc\_chunk\_t \*chunk)  
*Try to mark chunk busy.*
- long [l4shmc\\_chunk\\_try\\_to\\_take\\_for\\_writing](#) (l4shmc\_chunk\_t \*chunk)  
*Try to mark chunk busy writing.*
- long [l4shmc\\_chunk\\_try\\_to\\_take\\_for\\_overwriting](#) (l4shmc\_chunk\_t \*chunk)

*Try to mark the chunk busy writing after it was ready for reading.*

- long `l4shmc_chunk_ready` (`l4shmc_chunk_t *chunk`, `l4_umword_t size`)

*Mark chunk as filled (ready).*

- long `l4shmc_chunk_ready_sig` (`l4shmc_chunk_t *chunk`, `l4_umword_t size`)

*Mark chunk as filled (ready) and signal consumer.*

- long `l4shmc_is_chunk_clear` (`l4shmc_chunk_t const *chunk`)

*Check whether chunk is free.*

#### 13.13.3.4.1 Detailed Description

#### 13.13.3.4.2 Function Documentation

##### 13.13.3.4.2.1 `l4shmc_chunk_ready()`

```
long l4shmc_chunk_ready (
    l4shmc_chunk_t * chunk,
    l4_umword_t size ) [inline]
```

Mark chunk as filled (ready).

##### Parameters

<i>chunk</i>	chunk.
<i>size</i>	Size of data in the chunk, in bytes.

##### Return values

<i>0</i>	Success.
<i>&lt;0</i>	Error.

##### 13.13.3.4.2.2 `l4shmc_chunk_ready_sig()`

```
long l4shmc_chunk_ready_sig (
    l4shmc_chunk_t * chunk,
    l4_umword_t size ) [inline]
```

Mark chunk as filled (ready) and signal consumer.

##### Parameters

<i>chunk</i>	chunk.
<i>size</i>	Size of data in the chunk, in bytes.

##### Return values

<i>0</i>	Success.
<i>&lt;0</i>	Error.

### Examples

[examples/libs/shmc/prodcons.c](#).

#### 13.13.3.4.2.3 l4shmc\_chunk\_try\_to\_take()

```
long l4shmc_chunk_try_to_take (
    l4shmc_chunk_t * chunk ) [inline]
```

Try to mark chunk busy.

### Parameters

<i>chunk</i>	chunk to mark.
--------------	----------------

### Return values

0	Chunk could be taken.
<0	Chunk could not be taken, try again.

### Examples

[examples/libs/shmc/prodcons.c](#).

#### 13.13.3.4.2.4 l4shmc\_chunk\_try\_to\_take\_for\_overwriting()

```
long l4shmc_chunk_try_to_take_for_overwriting (
    l4shmc_chunk_t * chunk ) [inline]
```

Try to mark the chunk busy writing after it was ready for reading.

### Parameters

<i>chunk</i>	chunk to mark busy writing.
--------------	-----------------------------

This function is used by the producer to overwrite a message if the consumer did not read the message within an expected time. This function can only be used if the consumer uses [l4shmc\\_chunk\\_try\\_to\\_take\\_for\\_reading\(\)](#) before reading the chunk.

### Return values

0	Chunk could be taken and can be written.
<0	Chunk could not be taken, try again.

#### 13.13.3.4.2.5 l4shmc\_chunk\_try\_to\_take\_for\_writing()

```
long l4shmc_chunk_try_to_take_for_writing (
```

```
l4shmc_chunk_t * chunk ) [inline]
```

Try to mark chunk busy writing.

This function is actually an alias for [l4shmc\\_chunk\\_try\\_to\\_take\(\)](#).

#### Parameters

<i>chunk</i>	chunk to mark busy writing.
--------------	-----------------------------

#### Return values

0	Chunk could be taken and can be written.
<0	Chunk could not be taken, try again.

#### 13.13.3.4.2.6 l4shmc\_is\_chunk\_clear()

```
long l4shmc_is_chunk_clear (
    l4shmc_chunk_t const * chunk ) [inline]
```

Check whether chunk is free.

#### Parameters

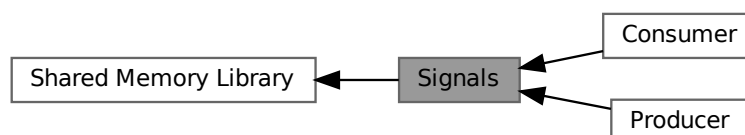
<i>chunk</i>	Chunk to check.
--------------	-----------------

#### Return values

!=0	Chunk is clear.
0	Chunk is not clear.

### 13.13.4 Signals

Collaboration diagram for Signals:



#### Modules

- [Consumer](#)
- [Producer](#)

## Functions

- long [l4shmc\\_add\\_signal](#) (l4shmc\_area\_t \*shmarea, char const \*signal\_name, l4shmc\_signal\_t \*signal)  
*Add a signal for the shared memory area.*
- long [l4shmc\\_attach\\_signal](#) (l4shmc\_area\_t \*shmarea, char const \*signal\_name, [l4\\_cap\\_idx\\_t](#) thread, l4shmc\_signal\_t \*signal)  
*Attach to signal.*
- long [l4shmc\\_get\\_signal](#) (l4shmc\_area\_t \*shmarea, char const \*signal\_name, l4shmc\_signal\_t \*signal)  
*Get signal object from the shared memory area.*
- [l4\\_cap\\_idx\\_t l4shmc\\_signal\\_cap](#) (l4shmc\_signal\_t const \*signal)  
*Get the signal capability of a signal.*
- long [l4shmc\\_check\\_magic](#) (l4shmc\_chunk\_t const \*chunk)  
*Check magic value of a chunk.*

### 13.13.4.1 Detailed Description

### 13.13.4.2 Function Documentation

#### 13.13.4.2.1 l4shmc\_add\_signal()

```
long l4shmc_add_signal (
    l4shmc_area_t * shmarea,
    char const * signal_name,
    l4shmc_signal_t * signal )
```

Add a signal for the shared memory area.

#### Parameters

	<i>shmarea</i>	The shared memory area.
	<i>signal_name</i>	Name of the signal.
out	<i>signal</i>	Signal structure to fill in.

#### Return values

0	Success.
<0	Error.

#### Examples

[examples/libs/shmc/prodcons.c](#).

#### 13.13.4.2.2 l4shmc\_attach\_signal()

```
long l4shmc_attach_signal (
    l4shmc_area_t * shmarea,
    char const * signal_name,
```

```
l4_cap_idx_t thread,  
l4shmc_signal_t * signal )
```

Attach to signal.



## Parameters

	<i>shmarea</i>	Shared memory area.
	<i>signal_name</i>	Name of the signal.
	<i>thread</i>	Thread capability index to attach the signal to.
out	<i>signal</i>	Signal data structure to fill.

## Return values

0	Success.
<0	Error.

## Examples

[examples/libs/shmc/prodcons.c](#).

**13.13.4.2.3 l4shmc\_check\_magic()**

```
long l4shmc_check_magic (
    l4shmc_chunk_t const * chunk ) [inline]
```

Check magic value of a chunk.

## Parameters

<i>chunk</i>	Chunk.
--------------	--------

## Return values

0	Magic value is not valid.
>0	Chunk is OK, the magic value is valid.

**13.13.4.2.4 l4shmc\_get\_signal()**

```
long l4shmc_get_signal (
    l4shmc_area_t * shmarea,
    char const * signal_name,
    l4shmc_signal_t * signal )
```

Get signal object from the shared memory area.

## Parameters

	<i>shmarea</i>	Shared memory area.
	<i>signal_name</i>	Name of the signal.
out	<i>signal</i>	Signal data structure to fill.

## Return values

0	Success.
<0	Error.

**13.13.4.2.5 l4shmc\_signal\_cap()**

```
l4_cap_idx_t l4shmc_signal_cap (
    l4shmc_signal_t const * signal ) [inline]
```

Get the signal capability of a signal.

## Parameters

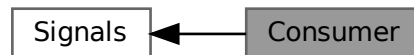
<i>signal</i>	Signal.
---------------	---------

## Returns

Capability of the signal object.

**13.13.4.3 Consumer**

Collaboration diagram for Consumer:

**Functions**

- long [l4shmc\\_enable\\_signal](#) (l4shmc\_signal\_t \*signal)  
*Enable a signal.*
- long [l4shmc\\_wait\\_any](#) (l4shmc\_signal\_t \*\*retsignal)  
*Wait on any signal.*
- long [l4shmc\\_wait\\_any\\_try](#) (l4shmc\_signal\_t \*\*retsignal)  
*Check whether any waited signal has an event pending.*
- long [l4shmc\\_wait\\_any\\_to](#) (l4\_timeout\_t timeout, l4shmc\_signal\_t \*\*retsignal)  
*Wait for any signal with timeout.*
- long [l4shmc\\_wait\\_signal](#) (l4shmc\_signal\_t \*signal)  
*Wait on a specific signal.*
- long [l4shmc\\_wait\\_signal\\_to](#) (l4shmc\_signal\_t \*signal, l4\_timeout\_t timeout)  
*Wait on a specific signal, with timeout.*
- long [l4shmc\\_wait\\_signal\\_try](#) (l4shmc\_signal\_t \*signal)  
*Check whether a specific signal has an event pending.*

#### 13.13.4.3.1 Detailed Description

#### 13.13.4.3.2 Function Documentation

##### 13.13.4.3.2.1 l4shmc\_enable\_signal()

```
long l4shmc_enable_signal (
    l4shmc_signal_t * signal )
```

Enable a signal.

##### Parameters

<i>signal</i>	Signal to enable.
---------------	-------------------

##### Return values

0	Success.
<0	Error.

A signal must be enabled before waiting when the consumer waits on any signal. Enabling is not needed if the consumer waits for a specific signal or chunk.

##### 13.13.4.3.2.2 l4shmc\_wait\_any()

```
long l4shmc_wait_any (
    l4shmc_signal_t ** retsignal ) [inline]
```

Wait on any signal.

##### Parameters

out	<i>retsignal</i>	Signal received.
-----	------------------	------------------

##### Return values

0	Success.
<0	Error.

##### 13.13.4.3.2.3 l4shmc\_wait\_any\_to()

```
long l4shmc_wait_any_to (
    l4_timeout_t timeout,
    l4shmc_signal_t ** retsignal )
```

Wait for any signal with timeout.

**Parameters**

	<i>timeout</i>	Timeout.
out	<i>retsignal</i>	Signal that has the event pending if any.

**Return values**

0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

**13.13.4.3.2.4 l4shmc\_wait\_any\_try()**

```
long l4shmc_wait_any_try (
    l4shmc_signal_t ** retsignal ) [inline]
```

Check whether any waited signal has an event pending.

**Parameters**

out	<i>retsignal</i>	Signal that has the event pending if any.
-----	------------------	---

**Return values**

0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

**13.13.4.3.2.5 l4shmc\_wait\_signal()**

```
long l4shmc_wait_signal (
    l4shmc_signal_t * signal ) [inline]
```

Wait on a specific signal.

**Parameters**

<i>signal</i>	Signal to wait for.
---------------	---------------------

**Return values**

0	Success.
<0	Error.

## Examples

[examples/libs/shmc/prodcons.c](#).

**13.13.4.3.2.6 l4shmc\_wait\_signal\_to()**

```
long l4shmc_wait_signal_to (
    l4shmc_signal_t * signal,
    l4_timeout_t timeout )
```

Wait on a specific signal, with timeout.

## Parameters

<i>signal</i>	Signal to wait for.
<i>timeout</i>	Timeout.

## Return values

0	Success.
<0	Error.

**13.13.4.3.2.7 l4shmc\_wait\_signal\_try()**

```
long l4shmc_wait_signal_try (
    l4shmc_signal_t * signal ) [inline]
```

Check whether a specific signal has an event pending.

## Parameters

<i>signal</i>	Signal to check.
---------------	------------------

## Return values

0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

#### 13.13.4.4 Producer

Collaboration diagram for Producer:



#### Functions

- long [l4shmc\\_trigger](#) (l4shmc\_signal\_t \*signal)  
*Trigger a signal.*

##### 13.13.4.4.1 Detailed Description

##### 13.13.4.4.2 Function Documentation

###### 13.13.4.4.2.1 l4shmc\_trigger()

```
long l4shmc_trigger (
    l4shmc_signal_t * signal ) [inline]
```

Trigger a signal.

#### Parameters

<i>signal</i>	Signal to trigger.
---------------	--------------------

#### Return values

0	Success.
<0	Error.

#### Examples

[examples/libs/shmc/prodcons.c](#).

## 13.14 Sigma0 API

Sigma0 API bindings.

Collaboration diagram for Sigma0 API:



## Modules

- [Internal constants](#)  
*Internal sigma0 definitions.*

## Files

- file [sigma0.h](#)  
*Sigma0 interface.*

## Enumerations

- enum [l4sigma0\\_return\\_flags\\_t](#) {  
    [L4SIGMA0\\_OK](#) , [L4SIGMA0\\_NOTALIGNED](#) , [L4SIGMA0\\_IPCERROR](#) , [L4SIGMA0\\_NOFPAGE](#) ,  
    [L4SIGMA0\\_4](#) , [L4SIGMA0\\_5](#) , [L4SIGMA0\\_SMALLERFPAGE](#) }  
*Return flags of libsigma0 functions.*

## Functions

- [l4\\_kernel\\_info\\_t](#) \* [l4sigma0\\_map\\_kip](#) ([l4\\_cap\\_idx\\_t](#) sigma0, void \*addr, unsigned log2\_size)  
*Map the kernel info page from sigma0 to addr.*
- int [l4sigma0\\_map\\_mem](#) ([l4\\_cap\\_idx\\_t](#) sigma0, [l4\\_addr\\_t](#) phys, [l4\\_addr\\_t](#) virt, [l4\\_addr\\_t](#) size)  
*Request a memory mapping from sigma0.*
- int [l4sigma0\\_map\\_iomem](#) ([l4\\_cap\\_idx\\_t](#) sigma0, [l4\\_addr\\_t](#) phys, [l4\\_addr\\_t](#) virt, [l4\\_addr\\_t](#) size, int cached)  
*Request IO memory from sigma0.*
- int [l4sigma0\\_map\\_anypage](#) ([l4\\_cap\\_idx\\_t](#) sigma0, [l4\\_addr\\_t](#) map\_area, unsigned log2\_map\_size, [l4\\_addr\\_t](#) \*base, unsigned sz)  
*Request an arbitrary free page of RAM.*
- void [l4sigma0\\_debug\\_dump](#) ([l4\\_cap\\_idx\\_t](#) sigma0)  
*Request sigma0 to dump internal debug information.*
- char const \* [l4sigma0\\_map\\_errstr](#) (int err)  
*Get user readable error messages for the return codes.*

### 13.14.1 Detailed Description

Sigma0 API bindings.

Convenience bindings for the Sigma0 protocol.

## 13.14.2 Enumeration Type Documentation

### 13.14.2.1 l4sigma0\_return\_flags\_t

```
enum l4sigma0_return_flags_t
```

Return flags of libsigma0 functions.

Enumerator

L4SIGMA0_OK	Ok.
L4SIGMA0_NOTALIGNED	Phys, virt or size not aligned.
L4SIGMA0_IPCERROR	IPC error.
L4SIGMA0_NOFPAGE	No fpage received.
L4SIGMA0_SMALLERFPAGE	Superpage requested but smaller flexpage received.

Definition at line 78 of file [sigma0.h](#).

## 13.14.3 Function Documentation

### 13.14.3.1 l4sigma0\_debug\_dump()

```
void l4sigma0_debug_dump (
    l4_cap_idx_t sigma0 )
```

Request sigma0 to dump internal debug information.

Parameters

<i>sigma0</i>	Capability selector for the sigma0 gate.
---------------	--

The debug information, such as internal memory maps, as well as statistics about the internal allocators is dumped to the kernel debugger.

### 13.14.3.2 l4sigma0\_map\_anypage()

```
int l4sigma0_map_anypage (
    l4_cap_idx_t sigma0,
    l4_addr_t map_area,
    unsigned log2_map_size,
    l4_addr_t * base,
    unsigned sz )
```

Request an arbitrary free page of RAM.

Parameters

	<i>sigma0</i>	Capability selector for the sigma0 gate.
--	---------------	--



## Parameters

	<i>map_area</i>	The base address of the local virtual memory area where the page should be mapped.
	<i>log2_map_size</i>	The size of the requested page log 2 (the size in bytes is $2^{\text{log2\_map\_size}}$ ). This must be at least the minimal page size. By specifying larger sizes the largest possible hardware page size will be used.
out	<i>base</i>	Physical address of the page received (i.e. the send base of the received mapping if any).
	<i>sz</i>	Size to map by the server in $2^{\text{sz}}$ bytes.

## Return values

0	Success.
-L4SIGMA0_IPCERROR	IPC error.
-L4SIGMA0_NOFPAGE	No fpage received.

This function requests arbitrary free memory from sigma0. It should be used whenever spare memory is needed, instead of requesting specific physical memory with [l4sigma0\\_map\\_mem\(\)](#).

See [l4sigma0\\_map\\_errstr\(\)](#) to get a description of the return value.

## 13.14.3.3 l4sigma0\_map\_errstr()

```
char const * l4sigma0_map_errstr (
    int err ) [inline]
```

Get user readable error messages for the return codes.

## Parameters

<i>err</i>	The error code reported by the <i>map</i> functions.
------------	--

## Returns

A string containing the error message.

Definition at line 210 of file [sigma0.h](#).

## 13.14.3.4 l4sigma0\_map\_iomem()

```
int l4sigma0_map_iomem (
    l4_cap_idx_t sigma0,
    l4_addr_t phys,
    l4_addr_t virt,
    l4_addr_t size,
    int cached )
```

Request IO memory from sigma0.

## Parameters

<i>sigma0</i>	Capability selector for the sigma0 gate.
<i>phys</i>	The physical address to be requested (page aligned).
<i>virt</i>	The virtual address where the memory should be mapped to (page aligned).
<i>size</i>	The size of the IO memory area to be mapped (multiple of page size)
<i>cached</i>	Requests cacheable IO memory if 1 and uncached if 0.

## Return values

0	Success.
-L4SIGMA0_NOTALIGNED	<i>phys</i> , <i>virt</i> , or <i>size</i> are not aligned.
-L4SIGMA0_IPCERROR	IPC error.
-L4SIGMA0_NOFPAGE	No fpage received.

This function is similar to [l4sigma0\\_map\\_mem\(\)](#), the difference is that it requests IO memory. IO memory is everything that is not known to be normal RAM. Also ACPI tables or the BIOS memory is treated as IO memory.

See [l4sigma0\\_map\\_errstr\(\)](#) to get a description of the return value.

## 13.14.3.5 l4sigma0\_map\_kip()

```
l4_kernel_info_t * l4sigma0_map_kip (
    l4_cap_idx_t sigma0,
    void * addr,
    unsigned log2_size )
```

Map the kernel info page from sigma0 to addr.

## Parameters

<i>sigma0</i>	Capability selector for the sigma0 gate.
<i>addr</i>	Start of the receive window to receive KIP in.
<i>log2_size</i>	Size of the receive window to receive KIP in.

## Returns

Address KIP was mapped to, 0 indicates an error.

## 13.14.3.6 l4sigma0\_map\_mem()

```
int l4sigma0_map_mem (
    l4_cap_idx_t sigma0,
    l4_addr_t phys,
    l4_addr_t virt,
    l4_addr_t size )
```

Request a memory mapping from sigma0.

## Parameters

<i>sigma0</i>	Capability selector for the sigma0 gate.
<i>phys</i>	The physical address of the requested page (must be at least aligned to the minimum page size).
<i>virt</i>	The virtual address where the paged should be mapped in the local address space (must be at least aligned to the minimum page size).
<i>size</i>	The size of the requested page, this must be a multiple of the minimum page size.

## Return values

<i>0</i>	Success.
<i>-L4SIGMA0_NOTALIGNED</i>	<i>phys</i> , <i>virt</i> , or <i>size</i> are not aligned.
<i>-L4SIGMA0_IPCERROR</i>	IPC error.
<i>-L4SIGMA0_NOFPAGE</i>	No fpage received.

This function only maps normal RAM. To map other memory, use [l4sigma0\\_map\\_iomem\(\)](#). See also there for the distinction between both memory types.

This is the direct method to request memory from sigma0. There is also the indirect method where sigma0 will answer page faults with a mapping that is one-to-one between the faulting virtual page and the backing physical page. See [L4::Pager::page\\_fault\(\)](#). For an overview of the memory hierarchy, see [Memory management - Data Spaces and the Region Map](#).

See [l4sigma0\\_map\\_errstr\(\)](#) to get a description of the return value.

### 13.14.4 Internal constants

Internal sigma0 definitions.

Collaboration diagram for Internal constants:



## Macros

- `#define SIGMA0_REQ_MAGIC ~0xFFUL`  
*Request magic.*
- `#define SIGMA0_REQ_MASK ~0xFFUL`  
*Request mask.*
- `#define SIGMA0_REQ_ID_MASK 0xF0`  
*ID mask.*
- `#define SIGMA0_REQ_ID_FPAGE_RAM 0x60`

- *RAM.*
- `#define SIGMA0_REQ_ID_FPAGE_IOMEM 0x70`
- *I/O memory.*
- `#define SIGMA0_REQ_ID_FPAGE_IOMEM_CACHED 0x80`
- *Cached I/O memory.*
- `#define SIGMA0_REQ_ID_FPAGE_ANY 0x90`
- *Any.*
- `#define SIGMA0_REQ_ID_KIP 0xA0`
- *KIP.*
- `#define SIGMA0_REQ_ID_DEBUG_DUMP 0xC0`
- *Debug dump.*
- `#define SIGMA0_IS_MAGIC_REQ(d1) ((d1 & SIGMA0_REQ_MASK) == SIGMA0_REQ_MAGIC)`
- *Check if magic.*
- `#define SIGMA0_REQ(x) (SIGMA0_REQ_MAGIC + SIGMA0_REQ_ID_ ## x)`
- *Construct.*
- `#define SIGMA0_REQ_FPAGE_RAM (SIGMA0_REQ(FPAGE_RAM))`
- *RAM.*
- `#define SIGMA0_REQ_FPAGE_IOMEM (SIGMA0_REQ(FPAGE_IOMEM))`
- *I/O memory.*
- `#define SIGMA0_REQ_FPAGE_IOMEM_CACHED (SIGMA0_REQ(FPAGE_IOMEM_CACHED))`
- *Cache I/O memory.*
- `#define SIGMA0_REQ_FPAGE_ANY (SIGMA0_REQ(FPAGE_ANY))`
- *Any.*
- `#define SIGMA0_REQ_KIP (SIGMA0_REQ(KIP))`
- *KIP.*
- `#define SIGMA0_REQ_DEBUG_DUMP (SIGMA0_REQ(DEBUG_DUMP))`
- *Debug dump.*

#### 13.14.4.1 Detailed Description

Internal sigma0 definitions.

## 13.15 Small C++ Template Library

### Namespaces

- namespace `cxx`
- *Our C++ library.*

## Data Structures

- class [L4::Alloc\\_list](#)  
*A simple list-based allocator.*
- class [cxx::List\\_item](#)  
*Basic list item.*
- struct [cxx::Pair< First, Second >](#)  
*Pair of two values.*
- class [cxx::Base\\_slab< Obj\\_size, Slab\\_size, Max\\_free, Alloc >](#)  
*Basic slab allocator.*
- class [cxx::Slab< Type, Slab\\_size, Max\\_free, Alloc >](#)  
*Slab allocator for object of type `Type`.*
- class [cxx::Base\\_slab\\_static< Obj\\_size, Slab\\_size, Max\\_free, Alloc >](#)  
*Merged slab allocator (allocators for objects of the same size are merged together).*
- class [cxx::Slab\\_static< Type, Slab\\_size, Max\\_free, Alloc >](#)  
*Merged slab allocator (allocators for objects of the same size are merged together).*
- class [cxx::Nothrow](#)  
*Helper type to distinguish the `operator new` version that does not throw exceptions.*
- class [cxx::New\\_allocator< \\_Type >](#)  
*Standard allocator based on `operator new ()`.*
- class [L4::String](#)  
*A null-terminated string container class.*

## Functions

- `template<typename A , typename ... ARGS>`  
`constexpr A const & cxx::min (A const &a1, A const &a2, ARGS const &...a)`  
*Get the minimum of `a1` and `a2` up to `aN`.*
- `template<typename A , typename ... ARGS>`  
`constexpr A const & cxx::min (cxx::identity_t< A > const &a1, cxx::identity_t< A > const &a2, ARGS const &...a)`  
*Get the minimum of `a1` and `a2` up to `aN`.*
- `template<typename A , typename ... ARGS>`  
`constexpr A const & cxx::max (A const &a1, A const &a2, ARGS const &...a)`  
*Get the maximum of `a1` and `a2` up to `aN`.*
- `template<typename A , typename ... ARGS>`  
`constexpr A const & cxx::max (cxx::identity_t< A > const &a1, cxx::identity_t< A > const &a2, ARGS const &...a)`  
*Get the maximum of `a1` and `a2` up to `aN`.*
- `template<typename T1 >`  
`T1 cxx::clamp (T1 v, T1 lo, T1 hi)`  
*Limit `v` to the range given by `lo` and `hi`.*
- `void * operator new (size_t, void *mem, cxx::Nothrow const &) noexcept`  
*Simple placement new operator.*
- `void * operator new (size_t, cxx::Nothrow const &) noexcept`  
*New operator that does not throw exceptions.*
- `void operator delete (void *, cxx::Nothrow const &) noexcept`  
*Delete operator complementing the new operator not throwing exceptions.*

### 13.15.1 Detailed Description

### 13.15.2 Function Documentation

#### 13.15.2.1 clamp()

```
template<typename T1 >
T1 cxx::clamp (
    T1 v,
    T1 lo,
    T1 hi ) [inline]
```

Limit *v* to the range given by *lo* and *hi*.

##### Parameters

<i>v</i>	The value to clamp.
<i>lo</i>	The lower boundary to clamp <i>v</i> to.
<i>hi</i>	The upper boundary to clamp <i>v</i> to.

Definition at line 120 of file [minmax](#).

#### 13.15.2.2 max() [1/2]

```
template<typename A , typename ... ARGS>
constexpr A const & cxx::max (
    A const & a1,
    A const & a2,
    ARGS const &... a ) [constexpr]
```

Get the maximum of *a1* and *a2* up to *aN*.

##### Parameters

<i>a1</i>	The first value.
<i>a2</i>	The second value.
<i>...↔</i> <i>a</i>	Arbitrary number of additional parameters.

Matches with automatic argument type deduction.

Definition at line 89 of file [minmax](#).

#### 13.15.2.3 max() [2/2]

```
template<typename A , typename ... ARGS>
constexpr A const & cxx::max (
    cxx::identity_t< A > const & a1,
    cxx::identity_t< A > const & a2,
    ARGS const &... a ) [constexpr]
```

Get the maximum of *a1* and *a2* up to *aN*.

**Parameters**

<i>a1</i>	The first value.
<i>a2</i>	The second value.
<i>...↔</i> <i>a</i>	Arbitrary number of additional parameters.

Matches with explicit template type A.

Definition at line 104 of file [minmax](#).

**13.15.2.4 min() [1/2]**

```
template<typename A , typename ... ARGS>
constexpr A const & cxx::min (
    A const & a1,
    A const & a2,
    ARGS const &... a ) [constexpr]
```

Get the minimum of a1 and a2 upt to aN.

**Parameters**

<i>a1</i>	The first value.
<i>a2</i>	The second value.
<i>...↔</i> <i>a</i>	Arbitrary number of additional parameters.

Matches with automatic argument type deduction.

Definition at line 47 of file [minmax](#).

**13.15.2.5 min() [2/2]**

```
template<typename A , typename ... ARGS>
constexpr A const & cxx::min (
    cxx::identity_t< A > const & a1,
    cxx::identity_t< A > const & a2,
    ARGS const &... a ) [constexpr]
```

Get the minimum of a1 and a2 upt to aN.

**Parameters**

<i>a1</i>	The first value.
<i>a2</i>	The second value.
<i>...↔</i> <i>a</i>	Arbitrary number of additional parameters.

Matches with explicit template type A.

Definition at line 64 of file [minmax](#).

### 13.15.2.6 operator new()

```
void * operator new (  
    size_t ,  
    void * mem,  
    cxx::Nothrow const & ) [inline], [noexcept]
```

Simple placement new operator.

#### Parameters

<i>mem</i>	the address of the memory block to place the new object.
------------	--

#### Returns

the address given by *mem*.

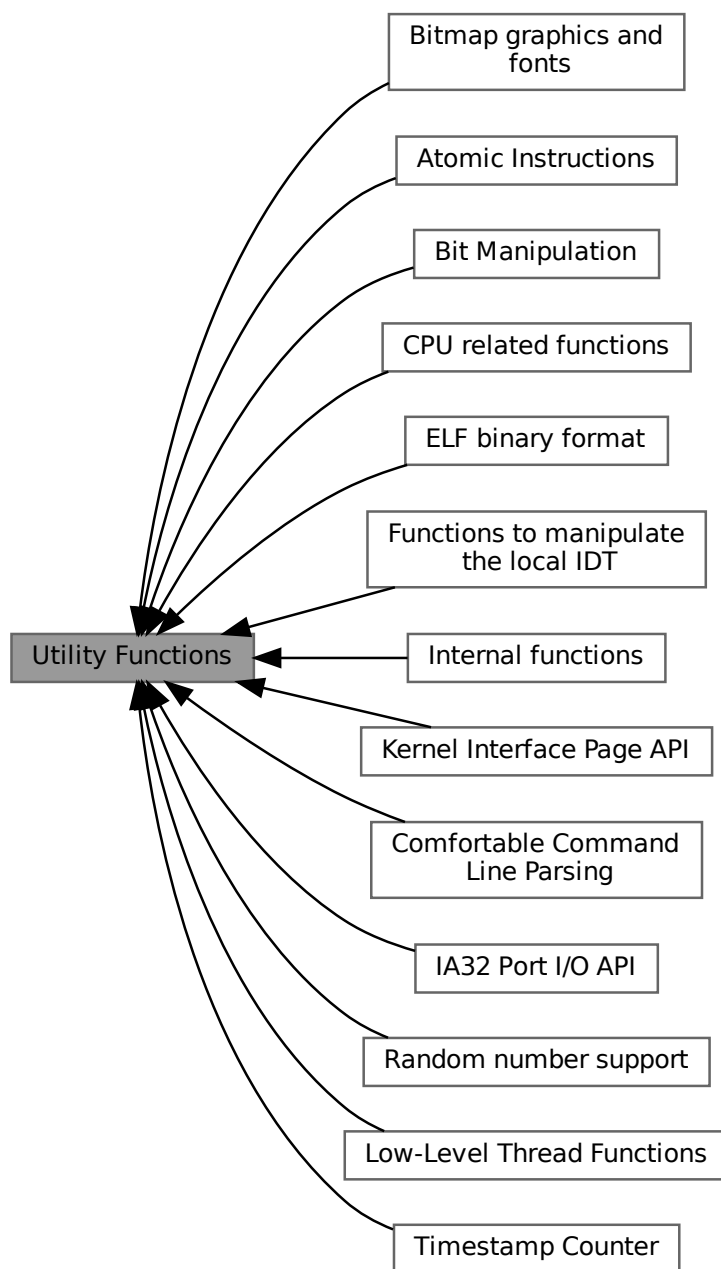
Definition at line 39 of file [std\\_alloc](#).

## 13.16 Utility Functions

Utilities, generic file.



Collaboration diagram for Utility Functions:



## Modules

- [Atomic Instructions](#)
- [Bit Manipulation](#)
- [Bitmap graphics and fonts](#)

*This library provides some functions for bitmap handling in frame buffers.*

- [CPU related functions](#)

- [Comfortable Command Line Parsing](#)
- [ELF binary format](#)

*Functions and types related to ELF binaries.*

- [Functions to manipulate the local IDT](#)
- [IA32 Port I/O API](#)
- [Internal functions](#)
- [Kernel Interface Page API](#)
- [Low-Level Thread Functions](#)
- [Random number support](#)
- [Timestamp Counter](#)

## Files

- file [rand.h](#)

*Simple Pseudo-Random Number Generator.*

## Functions

- long [l4util\\_splitlog2\\_hdl](#) ([l4\\_addr\\_t](#) start, [l4\\_addr\\_t](#) end, long(\*handler)([l4\\_addr\\_t](#) s, [l4\\_addr\\_t](#) e, int log2size))  
*Split a range into log2 base and size aligned chunks.*
- [l4\\_addr\\_t](#) [l4util\\_splitlog2\\_size](#) ([l4\\_addr\\_t](#) start, [l4\\_addr\\_t](#) end)  
*Return log2 base and size aligned length of a range.*
- [l4\\_timeout\\_s](#) [l4util\\_micros2l4to](#) ([l4\\_uint64\\_t](#) us) [L4\\_NOTHROW](#)  
*Calculate l4 timeouts.*
- void [l4\\_sleep](#) ([l4\\_uint32\\_t](#) ms) [L4\\_NOTHROW](#)  
*Suspend thread for a period of ms milliseconds.*
- void [l4\\_usleep](#) ([l4\\_uint64\\_t](#) us) [L4\\_NOTHROW](#)  
*Suspend thread for a period of us microseconds.*
- void [l4\\_sleep\\_forever](#) (void) [L4\\_NOTHROW](#) [L4\\_NORETURN](#)  
*Go sleep and never wake up.*
- void [l4\\_touch\\_ro](#) (const void \*addr, unsigned size) [L4\\_NOTHROW](#)  
*Touch data area to force mapping (read-only)*
- void [l4\\_touch\\_rw](#) (const void \*addr, unsigned size) [L4\\_NOTHROW](#)  
*Touch data areas to force mapping (read-write)*

### 13.16.1 Detailed Description

Utilities, generic file.

### 13.16.2 Function Documentation

#### 13.16.2.1 [l4\\_sleep\(\)](#)

```
void l4_sleep (
    l4\_uint32\_t ms )
```

Suspend thread for a period of *ms* milliseconds.

## Parameters

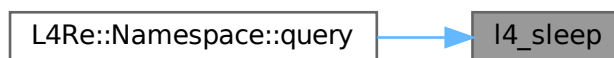
<i>ms</i>	Time in milliseconds
-----------	----------------------

## Examples

[examples/libs/libirq/async\\_isr.c](#), [examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/m](#)

Referenced by [L4Re::Namespace::query\(\)](#).

Here is the caller graph for this function:



## 13.16.2.2 I4\_touch\_ro()

```
void l4_touch_ro (
    const void * addr,
    unsigned size ) [inline]
```

Touch data area to force mapping (read-only)

## Parameters

<i>addr</i>	Start of memory area to touch.
<i>size</i>	Size of area to touch.

## Examples

[examples/sys/singlestep/main.c](#).

Definition at line 94 of file [util.h](#).

References [L4\\_PAGESIZE](#), and [I4\\_trunc\\_page\(\)](#).

Here is the call graph for this function:



### 13.16.2.3 l4\_touch\_rw()

```
void l4_touch_rw (
    const void * addr,
    unsigned size ) [inline]
```

Touch data areas to force mapping (read-write)

#### Parameters

<i>addr</i>	Start of memory area to touch.
<i>size</i>	Size of area to touch.

#### Examples

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line 107 of file [util.h](#).

References [L4\\_PAGESIZE](#), and [l4\\_trunc\\_page\(\)](#).

Here is the call graph for this function:



### 13.16.2.4 l4\_usleep()

```
void l4_usleep (
    l4_uint64_t us )
```

Suspend thread for a period of *us* microseconds.

#### Parameters

<i>us</i>	Time in microseconds
-----------	----------------------

#### Note

The timer resolution of [L4](#) kernels is usually 1ms.

## 13.16.2.5 l4util\_micros2l4to()

```
l4_timeout_s l4util_micros2l4to (
    l4_uint64_t us )
```

Calculate l4 timeouts.

## Parameters

<i>us</i>	time in microseconds. Special cases: <ul style="list-style-type: none"> <li>• 0 -&gt; timeout 0</li> <li>• ~0U -&gt; timeout NEVER</li> </ul>
-----------	---

## Returns

the corresponding l4\_timeout value

**Deprecated** Use l4\_timeout\_from\_us().

## 13.16.2.6 l4util\_splitlog2\_hdl()

```
long l4util_splitlog2_hdl (
    l4_addr_t start,
    l4_addr_t end,
    long(*) (l4_addr_t s, l4_addr_t e, int log2size) handler ) [inline]
```

Split a range into log2 base and size aligned chunks.

## Parameters

<i>start</i>	Start of range
<i>end</i>	End of range (inclusive) (e.g. 2-4 is len 3)
<i>handler</i>	Handler function that is called with start and end (both inclusive) of the chunk. On success, the handler must return 0, if it returns !=0 the function will immediately return with the return code of the handler.

## Returns

0 on success, != 0 otherwise

Definition at line 53 of file [splitlog2.h](#).

References [L4\\_EINVAL](#), and [l4util\\_splitlog2\\_size\(\)](#).

Here is the call graph for this function:



### 13.16.2.7 l4util\_splitlog2\_size()

```
l4_addr_t l4util_splitlog2_size (  
    l4_addr_t start,  
    l4_addr_t end ) [inline]
```

Return log2 base and size aligned length of a range.

#### Parameters

<i>start</i>	Start of range
<i>end</i>	End of range (inclusive) (e.g. 2-4 is len 3)

#### Returns

length of elements in log2size (length is  $1 \ll \log2size$ )

Definition at line 72 of file [splitlog2.h](#).

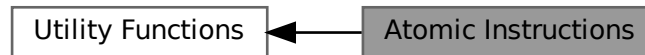
Referenced by [l4util\\_splitlog2\\_hdl\(\)](#).

Here is the caller graph for this function:



### 13.16.3 Atomic Instructions

Collaboration diagram for Atomic Instructions:



#### Files

- file [atomic.h](#)  
*atomic operations header and generic implementations*

#### Functions

- `int l4util_cmpxchg32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` cmp\_val, `l4_uint32_t` new\_val)  
*Atomic compare and exchange (32 bit version)*
- `int l4util_cmpxchg16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` cmp\_val, `l4_uint16_t` new\_val)  
*Atomic compare and exchange (16 bit version)*
- `int l4util_cmpxchg8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` cmp\_val, `l4_uint8_t` new\_val)  
*Atomic compare and exchange (8 bit version)*
- `int l4util_cmpxchg` (volatile `l4_umword_t` \*dest, `l4_umword_t` cmp\_val, `l4_umword_t` new\_val)  
*Atomic compare and exchange (machine wide fields)*
- `l4_uint32_t l4util_xchg32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)  
*Atomic exchange (32 bit version)*
- `l4_uint16_t l4util_xchg16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)  
*Atomic exchange (16 bit version)*
- `l4_uint8_t l4util_xchg8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)  
*Atomic exchange (8 bit version)*
- `l4_umword_t l4util_xchg` (volatile `l4_umword_t` \*dest, `l4_umword_t` val)  
*Atomic exchange (machine wide fields)*
- `void l4util_atomic_add` (volatile long \*dest, long val)  
*Atomic add.*
- `void l4util_atomic_inc` (volatile long \*dest)  
*Atomic increment.*

#### Atomic add/sub/and/or (8,16,32 bit version) without result

- `void l4util_add8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `void l4util_add16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `void l4util_add32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)
- `void l4util_sub8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `void l4util_sub16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `void l4util_sub32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)
- `void l4util_and8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `void l4util_and16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `void l4util_and32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)
- `void l4util_or8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `void l4util_or16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `void l4util_or32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)

**Atomic add/sub/and/or operations (8,16,32 bit) with result**

- [l4\\_uint8\\_t l4util\\_add8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest, [l4\\_uint8\\_t](#) val)
- [l4\\_uint16\\_t l4util\\_add16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest, [l4\\_uint16\\_t](#) val)
- [l4\\_uint32\\_t l4util\\_add32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)
- [l4\\_uint8\\_t l4util\\_sub8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest, [l4\\_uint8\\_t](#) val)
- [l4\\_uint16\\_t l4util\\_sub16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest, [l4\\_uint16\\_t](#) val)
- [l4\\_uint32\\_t l4util\\_sub32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)
- [l4\\_uint8\\_t l4util\\_and8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest, [l4\\_uint8\\_t](#) val)
- [l4\\_uint16\\_t l4util\\_and16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest, [l4\\_uint16\\_t](#) val)
- [l4\\_uint32\\_t l4util\\_and32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)
- [l4\\_uint8\\_t l4util\\_or8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest, [l4\\_uint8\\_t](#) val)
- [l4\\_uint16\\_t l4util\\_or16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest, [l4\\_uint16\\_t](#) val)
- [l4\\_uint32\\_t l4util\\_or32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)

**Atomic inc/dec (8,16,32 bit) without result**

- void [l4util\\_inc8](#) (volatile [l4\\_uint8\\_t](#) \*dest)
- void [l4util\\_inc16](#) (volatile [l4\\_uint16\\_t](#) \*dest)
- void [l4util\\_inc32](#) (volatile [l4\\_uint32\\_t](#) \*dest)
- void [l4util\\_dec8](#) (volatile [l4\\_uint8\\_t](#) \*dest)
- void [l4util\\_dec16](#) (volatile [l4\\_uint16\\_t](#) \*dest)
- void [l4util\\_dec32](#) (volatile [l4\\_uint32\\_t](#) \*dest)

**Atomic inc/dec (8,16,32 bit) with result**

- [l4\\_uint8\\_t l4util\\_inc8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest)
- [l4\\_uint16\\_t l4util\\_inc16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest)
- [l4\\_uint32\\_t l4util\\_inc32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest)
- [l4\\_uint8\\_t l4util\\_dec8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest)
- [l4\\_uint16\\_t l4util\\_dec16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest)
- [l4\\_uint32\\_t l4util\\_dec32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest)

**13.16.3.1 Detailed Description****13.16.3.2 Function Documentation****13.16.3.2.1 l4util\_add16()**

```
void l4util_add16 (
    volatile l4\_uint16\_t * dest,
    l4\_uint16\_t val ) [inline]
```

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 474 of file [atomic.h](#).



### 13.16.3.2.2 l4util\_add16\_res()

```
l4_uint16_t l4util_add16_res (
    volatile l4_uint16_t * dest,
    l4_uint16_t val ) [inline]
```

#### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

#### Returns

res

Definition at line 526 of file [atomic.h](#).

### 13.16.3.2.3 l4util\_add32()

```
void l4util_add32 (
    volatile l4_uint32_t * dest,
    l4_uint32_t val ) [inline]
```

#### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 478 of file [atomic.h](#).

### 13.16.3.2.4 l4util\_add32\_res()

```
l4_uint32_t l4util_add32_res (
    volatile l4_uint32_t * dest,
    l4_uint32_t val ) [inline]
```

#### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

#### Returns

res

Definition at line 530 of file [atomic.h](#).

#### 13.16.3.2.5 l4util\_add8()

```
void l4util_add8 (
    volatile l4_uint8_t * dest,
    l4_uint8_t val ) [inline]
```

##### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 470 of file [atomic.h](#).

#### 13.16.3.2.6 l4util\_add8\_res()

```
l4_uint8_t l4util_add8_res (
    volatile l4_uint8_t * dest,
    l4_uint8_t val ) [inline]
```

##### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

##### Returns

res

Definition at line 522 of file [atomic.h](#).

#### 13.16.3.2.7 l4util\_and16()

```
void l4util_and16 (
    volatile l4_uint16_t * dest,
    l4_uint16_t val ) [inline]
```

##### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 502 of file [atomic.h](#).

#### 13.16.3.2.8 l4util\_and16\_res()

```
l4_uint16_t l4util_and16_res (
    volatile l4_uint16_t * dest,
    l4_uint16_t val ) [inline]
```

## Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

## Returns

res

Definition at line 550 of file [atomic.h](#).

**13.16.3.2.9 l4util\_and32()**

```
void l4util_and32 (
    volatile l4_uint32_t * dest,
    l4_uint32_t val ) [inline]
```

## Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 506 of file [atomic.h](#).

**13.16.3.2.10 l4util\_and32\_res()**

```
l4_uint32_t l4util_and32_res (
    volatile l4_uint32_t * dest,
    l4_uint32_t val ) [inline]
```

## Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

## Returns

res

Definition at line 554 of file [atomic.h](#).

**13.16.3.2.11 l4util\_and8()**

```
void l4util_and8 (
    volatile l4_uint8_t * dest,
    l4_uint8_t val ) [inline]
```

## Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 498 of file [atomic.h](#).

**13.16.3.2.12 l4util\_and8\_res()**

```
l4_uint8_t l4util_and8_res (
    volatile l4_uint8_t * dest,
    l4_uint8_t val ) [inline]
```

## Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

## Returns

res

Definition at line 546 of file [atomic.h](#).

**13.16.3.2.13 l4util\_atomic\_add()**

```
void l4util_atomic_add (
    volatile long * dest,
    long val ) [inline]
```

Atomic add.

## Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add

Definition at line 482 of file [atomic.h](#).

**13.16.3.2.14 l4util\_atomic\_inc()**

```
void l4util_atomic_inc (
    volatile long * dest ) [inline]
```

Atomic increment.

## Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 425 of file [atomic.h](#).

**13.16.3.2.15 l4util\_cmpxchg()**

```
int l4util_cmpxchg (  
    volatile l4_umword_t * dest,  
    l4_umword_t cmp_val,  
    l4_umword_t new_val ) [inline]
```

Atomic compare and exchange (machine wide fields)

## Parameters

<i>dest</i>	destination operand
<i>cmp_val</i>	compare value
<i>new_val</i>	new value for dest

## Returns

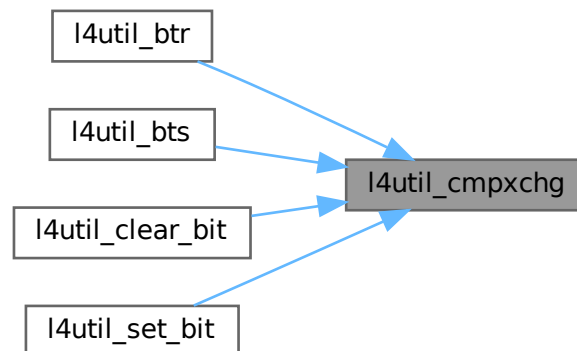
0 if comparison failed, 1 otherwise

Compare the value in *dest* with *cmp\_val*, if equal set *dest* to *new\_val*

Definition at line 381 of file [atomic.h](#).

Referenced by [l4util\\_btr\(\)](#), [l4util\\_bts\(\)](#), [l4util\\_clear\\_bit\(\)](#), and [l4util\\_set\\_bit\(\)](#).

Here is the caller graph for this function:



### 13.16.3.2.16 l4util\_cmpxchg16()

```
int l4util_cmpxchg16 (
    volatile l4_uint16_t * dest,
    l4_uint16_t cmp_val,
    l4_uint16_t new_val ) [inline]
```

Atomic compare and exchange (16 bit version)

#### Parameters

<i>dest</i>	destination operand
<i>cmp_val</i>	compare value
<i>new_val</i>	new value for dest

#### Returns

0 if comparison failed, !=0 otherwise

Compare the value in *dest* with *cmp\_val*, if equal set *dest* to *new\_val*

Definition at line 365 of file [atomic.h](#).

### 13.16.3.2.17 l4util\_cmpxchg32()

```
int l4util_cmpxchg32 (
    volatile l4_uint32_t * dest,
    l4_uint32_t cmp_val,
    l4_uint32_t new_val ) [inline]
```

Atomic compare and exchange (32 bit version)

#### Parameters

<i>dest</i>	destination operand
<i>cmp_val</i>	compare value
<i>new_val</i>	new value for dest

#### Returns

0 if comparison failed, !=0 otherwise

Compare the value in *dest* with *cmp\_val*, if equal set *dest* to *new\_val*

Definition at line 357 of file [atomic.h](#).

### 13.16.3.2.18 l4util\_cmpxchg8()

```
int l4util_cmpxchg8 (
    volatile l4_uint8_t * dest,
```

```
14_uint8_t cmp_val,  
14_uint8_t new_val ) [inline]
```

Atomic compare and exchange (8 bit version)

#### Parameters

<i>dest</i>	destination operand
<i>cmp_val</i>	compare value
<i>new_val</i>	new value for dest

#### Returns

0 if comparison failed, !=0 otherwise

Compare the value in *dest* with *cmp\_val*, if equal set *dest* to *new\_val*

Definition at line 373 of file [atomic.h](#).

#### 13.16.3.2.19 l4util\_dec16()

```
void l4util_dec16 (  
    volatile 14_uint16_t * dest ) [inline]
```

#### Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 433 of file [atomic.h](#).

#### 13.16.3.2.20 l4util\_dec16\_res()

```
14_uint16_t l4util_dec16_res (  
    volatile 14_uint16_t * dest ) [inline]
```

#### Parameters

<i>dest</i>	destination operand
-------------	---------------------

#### Returns

res

Definition at line 458 of file [atomic.h](#).

#### 13.16.3.2.21 l4util\_dec32()

```
void l4util_dec32 (  
    volatile 14_uint32_t * dest ) [inline]
```

## Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 437 of file [atomic.h](#).

**13.16.3.2.22 l4util\_dec32\_res()**

```
l4_uint32_t l4util_dec32_res (
    volatile l4_uint32_t * dest ) [inline]
```

## Parameters

<i>dest</i>	destination operand
-------------	---------------------

## Returns

res

Definition at line 462 of file [atomic.h](#).

**13.16.3.2.23 l4util\_dec8()**

```
void l4util_dec8 (
    volatile l4_uint8_t * dest ) [inline]
```

## Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 429 of file [atomic.h](#).

**13.16.3.2.24 l4util\_dec8\_res()**

```
l4_uint8_t l4util_dec8_res (
    volatile l4_uint8_t * dest ) [inline]
```

## Parameters

<i>dest</i>	destination operand
-------------	---------------------

## Returns

res

Definition at line 454 of file [atomic.h](#).



**13.16.3.2.25 l4util\_inc16()**

```
void l4util_inc16 (
    volatile l4_uint16_t * dest ) [inline]
```

**Parameters**

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 417 of file [atomic.h](#).

**13.16.3.2.26 l4util\_inc16\_res()**

```
l4_uint16_t l4util_inc16_res (
    volatile l4_uint16_t * dest ) [inline]
```

**Parameters**

<i>dest</i>	destination operand
-------------	---------------------

**Returns**

res

Definition at line 446 of file [atomic.h](#).

**13.16.3.2.27 l4util\_inc32()**

```
void l4util_inc32 (
    volatile l4_uint32_t * dest ) [inline]
```

**Parameters**

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 421 of file [atomic.h](#).

**13.16.3.2.28 l4util\_inc32\_res()**

```
l4_uint32_t l4util_inc32_res (
    volatile l4_uint32_t * dest ) [inline]
```

**Parameters**

<i>dest</i>	destination operand
-------------	---------------------

**Returns**

res

Definition at line 450 of file [atomic.h](#).**13.16.3.2.29 l4util\_inc8()**

```
void l4util_inc8 (
    volatile l4_uint8_t * dest ) [inline]
```

**Parameters**

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 413 of file [atomic.h](#).**13.16.3.2.30 l4util\_inc8\_res()**

```
l4_uint8_t l4util_inc8_res (
    volatile l4_uint8_t * dest ) [inline]
```

**Parameters**

<i>dest</i>	destination operand
-------------	---------------------

**Returns**

res

Definition at line 442 of file [atomic.h](#).**13.16.3.2.31 l4util\_or16()**

```
void l4util_or16 (
    volatile l4_uint16_t * dest,
    l4_uint16_t val ) [inline]
```

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 514 of file [atomic.h](#).**13.16.3.2.32 l4util\_or16\_res()**

```
l4_uint16_t l4util_or16_res (
```

```
volatile l4_uint16_t * dest,  
l4_uint16_t val ) [inline]
```

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

**Returns**

res

Definition at line 562 of file [atomic.h](#).

**13.16.3.2.33 l4util\_or32()**

```
void l4util_or32 (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t val ) [inline]
```

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 518 of file [atomic.h](#).

**13.16.3.2.34 l4util\_or32\_res()**

```
l4_uint32_t l4util_or32_res (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t val ) [inline]
```

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

**Returns**

res

Definition at line 566 of file [atomic.h](#).

**13.16.3.2.35 l4util\_or8()**

```
void l4util_or8 (  
    volatile l4_uint8_t * dest,  
    l4_uint8_t val ) [inline]
```

## Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 510 of file [atomic.h](#).

**13.16.3.2.36 l4util\_or8\_res()**

```
l4_uint8_t l4util_or8_res (  
    volatile l4_uint8_t * dest,  
    l4_uint8_t val ) [inline]
```

## Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

## Returns

res

Definition at line 558 of file [atomic.h](#).

**13.16.3.2.37 l4util\_sub16()**

```
void l4util_sub16 (  
    volatile l4_uint16_t * dest,  
    l4_uint16_t val ) [inline]
```

## Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 490 of file [atomic.h](#).

**13.16.3.2.38 l4util\_sub16\_res()**

```
l4_uint16_t l4util_sub16_res (  
    volatile l4_uint16_t * dest,  
    l4_uint16_t val ) [inline]
```

## Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

**Returns**

res

Definition at line 538 of file [atomic.h](#).

**13.16.3.2.39 l4util\_sub32()**

```
void l4util_sub32 (
    volatile l4_uint32_t * dest,
    l4_uint32_t val ) [inline]
```

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 494 of file [atomic.h](#).

**13.16.3.2.40 l4util\_sub32\_res()**

```
l4_uint32_t l4util_sub32_res (
    volatile l4_uint32_t * dest,
    l4_uint32_t val ) [inline]
```

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

**Returns**

res

Definition at line 542 of file [atomic.h](#).

**13.16.3.2.41 l4util\_sub8()**

```
void l4util_sub8 (
    volatile l4_uint8_t * dest,
    l4_uint8_t val ) [inline]
```

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 486 of file [atomic.h](#).

#### 13.16.3.2.42 l4util\_sub8\_res()

```
l4_uint8_t l4util_sub8_res (
    volatile l4_uint8_t * dest,
    l4_uint8_t val ) [inline]
```

##### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

##### Returns

res

Definition at line 534 of file [atomic.h](#).

#### 13.16.3.2.43 l4util\_xchg()

```
l4_umword_t l4util_xchg (
    volatile l4_umword_t * dest,
    l4_umword_t val ) [inline]
```

Atomic exchange (machine wide fields)

##### Parameters

<i>dest</i>	destination operand
<i>val</i>	new value for dest

##### Returns

old value at destination

Definition at line 407 of file [atomic.h](#).

#### 13.16.3.2.44 l4util\_xchg16()

```
l4_uint16_t l4util_xchg16 (
    volatile l4_uint16_t * dest,
    l4_uint16_t val ) [inline]
```

Atomic exchange (16 bit version)

##### Parameters

<i>dest</i>	destination operand
<i>val</i>	new value for dest

**Returns**

old value at destination

Definition at line 395 of file [atomic.h](#).

**13.16.3.2.45 l4util\_xchg32()**

```
l4_uint32_t l4util_xchg32 (
    volatile l4_uint32_t * dest,
    l4_uint32_t val ) [inline]
```

Atomic exchange (32 bit version)

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	new value for dest

**Returns**

old value at destination

Definition at line 389 of file [atomic.h](#).

**13.16.3.2.46 l4util\_xchg8()**

```
l4_uint8_t l4util_xchg8 (
    volatile l4_uint8_t * dest,
    l4_uint8_t val ) [inline]
```

Atomic exchange (8 bit version)

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	new value for dest

**Returns**

old value at destination

Definition at line 401 of file [atomic.h](#).

### 13.16.4 Bit Manipulation

Collaboration diagram for Bit Manipulation:



#### Files

- file [bitops\\_arch.h](#)  
*amd64 bit manipulation functions*
- file [bitops.h](#)  
*bit manipulation functions*
- file [bitops\\_arch.h](#)  
*x86 bit manipulation functions*

#### Functions

- void [l4util\\_set\\_bit](#) (int b, volatile [l4\\_umword\\_t](#) \*dest)  
*Set bit in memory.*
- void [l4util\\_clear\\_bit](#) (int b, volatile [l4\\_umword\\_t](#) \*dest)  
*Clear bit in memory.*
- void [l4util\\_complement\\_bit](#) (int b, volatile [l4\\_umword\\_t](#) \*dest)  
*Complement bit in memory.*
- int [l4util\\_test\\_bit](#) (int b, const volatile [l4\\_umword\\_t](#) \*dest)  
*Test bit (return value of bit)*
- int [l4util\\_bts](#) (int b, volatile [l4\\_umword\\_t](#) \*dest)  
*Bit test and set.*
- int [l4util\\_btr](#) (int b, volatile [l4\\_umword\\_t](#) \*dest)  
*Bit test and reset.*
- int [l4util\\_btc](#) (int b, volatile [l4\\_umword\\_t](#) \*dest)  
*Bit test and complement.*
- int [l4util\\_bsr](#) ([l4\\_umword\\_t](#) word)  
*Bit scan reverse.*
- int [l4util\\_bsf](#) ([l4\\_umword\\_t](#) word)  
*Bit scan forward.*
- int [l4util\\_find\\_first\\_set\\_bit](#) (const void \*dest, [l4\\_size\\_t](#) size)  
*Find the first set bit in a memory region.*
- int [l4util\\_find\\_first\\_zero\\_bit](#) (const void \*dest, [l4\\_size\\_t](#) size)  
*Find the first zero bit in a memory region.*
- int [l4util\\_next\\_power2](#) (unsigned long val)  
*Find the next power of 2 for a given number.*



### 13.16.4.1 Detailed Description

### 13.16.4.2 Function Documentation

#### 13.16.4.2.1 l4util\_bsf()

```
int l4util_bsf (  
    l4_umword_t word ) [inline]
```

Bit scan forward.

##### Parameters

<i>word</i>	value (machine size)
-------------	----------------------

##### Returns

index of least significant bit set in word, -1 if no bit is set (word == 0)

"bit scan forward", find least significant bit set in word.

Definition at line 318 of file [bitops.h](#).

#### 13.16.4.2.2 l4util\_bsr()

```
int l4util_bsr (  
    l4_umword_t word ) [inline]
```

Bit scan reverse.

##### Parameters

<i>word</i>	value (machine size)
-------------	----------------------

##### Returns

index of most significant set bit in word, -1 if no bit is set (word == 0)

"bit scan reverse", find most significant set bit in word (-> LOG2(word))

Definition at line 301 of file [bitops.h](#).

#### 13.16.4.2.3 l4util\_btc()

```
int l4util_btc (  
    int b,  
    volatile l4_umword_t * dest ) [inline]
```

Bit test and complement.

**Parameters**

<i>b</i>	bit position
<i>dest</i>	destination operand

**Returns**

Old value of bit *b*.

Complement bit *b* and return old value.

Definition at line 396 of file [bitops.h](#).

**13.16.4.2.4 l4util\_btr()**

```
int l4util_btr (
    int b,
    volatile l4_umword_t * dest ) [inline]
```

Bit test and reset.

**Parameters**

<i>b</i>	bit position
<i>dest</i>	destination operand

**Returns**

Old value of bit *b*.

Reset bit *b* and return old value.

Definition at line 280 of file [bitops.h](#).

References [l4util\\_cmpxchg\(\)](#).

Here is the call graph for this function:



#### 13.16.4.2.5 l4util\_bts()

```
int l4util_bts (
    int b,
    volatile l4_umword_t * dest ) [inline]
```

Bit test and set.

**Parameters**

<i>b</i>	bit position
<i>dest</i>	destination operand

**Returns**

Old value of bit *b*.

Set the *b* bit of *dest* to 1 and return the old value.

Definition at line 258 of file [bitops.h](#).

References [l4util\\_cmpxchg\(\)](#).

Here is the call graph for this function:

**13.16.4.2.6 l4util\_clear\_bit()**

```
void l4util_clear_bit (  
    int b,  
    volatile l4_umword_t * dest ) [inline]
```

Clear bit in memory.

**Parameters**

<i>b</i>	bit position
<i>dest</i>	destination operand

Definition at line 228 of file [bitops.h](#).

References [l4util\\_cmpxchg\(\)](#).

Here is the call graph for this function:



#### 13.16.4.2.7 l4util\_complement\_bit()

```
void l4util_complement_bit (
    int b,
    volatile l4_umword_t * dest ) [inline]
```

Complement bit in memory.

##### Parameters

<i>b</i>	bit position
<i>dest</i>	destination operand

Definition at line 361 of file [bitops.h](#).

#### 13.16.4.2.8 l4util\_find\_first\_set\_bit()

```
int l4util_find_first_set_bit (
    const void * dest,
    l4_size_t size ) [inline]
```

Find the first set bit in a memory region.

##### Parameters

<i>dest</i>	bit string
<i>size</i>	size of string in bits (must be a multiple of L4_MWORD_BITS!)

##### Returns

number of the first set bit, >= size if no bit is set

Definition at line 402 of file [bitops.h](#).

#### 13.16.4.2.9 l4util\_find\_first\_zero\_bit()

```
int l4util_find_first_zero_bit (
    const void * dest,
    l4_size_t size ) [inline]
```

Find the first zero bit in a memory region.

#### Parameters

<i>dest</i>	bit string
<i>size</i>	size of string in bits (must be a multiple of L4_MWORD_BITS!)

#### Returns

number of the first zero bit,  $\geq$  size if no bit is set

Definition at line 335 of file [bitops.h](#).

#### 13.16.4.2.10 l4util\_next\_power2()

```
int l4util_next_power2 (
    unsigned long val ) [inline]
```

Find the next power of 2 for a given number.

#### Parameters

<i>val</i>	initial value
------------	---------------

#### Returns

next-highest power of 2

Definition at line 375 of file [bitops.h](#).

#### 13.16.4.2.11 l4util\_set\_bit()

```
void l4util_set_bit (
    int b,
    volatile l4_umword_t * dest ) [inline]
```

Set bit in memory.

#### Parameters

<i>b</i>	bit position
<i>dest</i>	destination operand

Definition at line 209 of file [bitops.h](#).

References [l4util\\_cmpxchg\(\)](#).

Here is the call graph for this function:



#### 13.16.4.2.12 l4util\_test\_bit()

```

int l4util_test_bit (
    int b,
    const volatile l4_umword_t * dest ) [inline]
  
```

Test bit (return value of bit)

##### Parameters

<i>b</i>	bit position
<i>dest</i>	destination operand

##### Returns

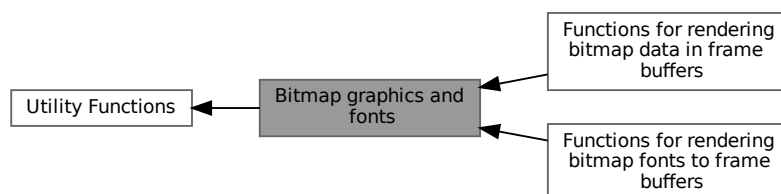
Value of bit *b*.

Definition at line 246 of file [bitops.h](#).

### 13.16.5 Bitmap graphics and fonts

This library provides some functions for bitmap handling in frame buffers.

Collaboration diagram for Bitmap graphics and fonts:



## Modules

- [Functions for rendering bitmap data in frame buffers](#)
- [Functions for rendering bitmap fonts to frame buffers](#)

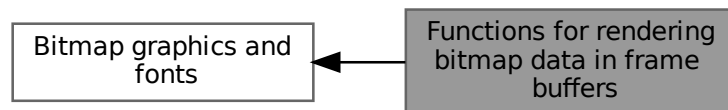
### 13.16.5.1 Detailed Description

This library provides some functions for bitmap handling in frame buffers.

Includes simple functions like filling or copying an area of the frame buffer going up to rendering text into the frame buffer using bitmap fonts.

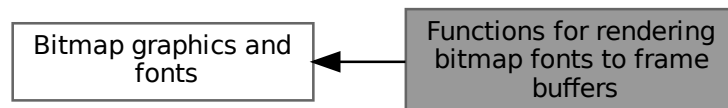
### 13.16.5.2 Functions for rendering bitmap data in frame buffers

Collaboration diagram for Functions for rendering bitmap data in frame buffers:



### 13.16.5.3 Functions for rendering bitmap fonts to frame buffers

Collaboration diagram for Functions for rendering bitmap fonts to frame buffers:



### 13.16.6 CPU related functions

Collaboration diagram for CPU related functions:





## Functions

- int [l4util\\_cpu\\_has\\_cpuid](#) (void)  
*Check whether the CPU supports the "cpuid" instruction.*
- unsigned int [l4util\\_cpu\\_capabilities](#) (void)  
*Returns the CPU capabilities if the "cpuid" instruction is available.*
- unsigned int [l4util\\_cpu\\_capabilities\\_nocheck](#) (void)  
*Returns the CPU capabilities.*
- void **[l4util\\_cpu\\_cpuid](#)** (unsigned long mode, unsigned long \*eax, unsigned long \*ebx, unsigned long \*ecx, unsigned long \*edx)  
*Generic CPUID access function.*

### 13.16.6.1 Detailed Description

### 13.16.6.2 Function Documentation

#### 13.16.6.2.1 [l4util\\_cpu\\_capabilities](#)()

```
unsigned int l4util_cpu_capabilities (
    void ) [inline]
```

Returns the CPU capabilities if the "cpuid" instruction is available.

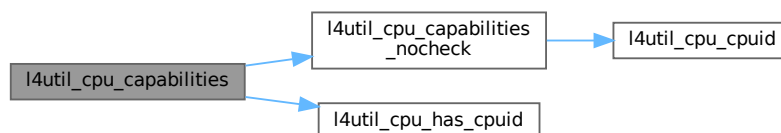
#### Returns

CPU capabilities if the "cpuid" instruction is available, 0 if the "cpuid" instruction is not supported.

Definition at line 97 of file [cpu.h](#).

References [l4util\\_cpu\\_capabilities\\_nocheck\(\)](#), and [l4util\\_cpu\\_has\\_cpuid\(\)](#).

Here is the call graph for this function:



### 13.16.6.2.2 l4util\_cpu\_capabilities\_nocheck()

```
unsigned int l4util_cpu_capabilities_nocheck (  
    void ) [inline]
```

Returns the CPU capabilities.

#### Returns

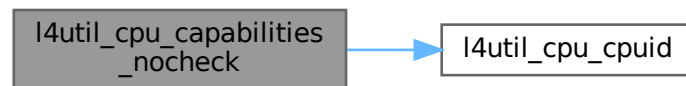
CPU capabilities.

Definition at line 86 of file [cpu.h](#).

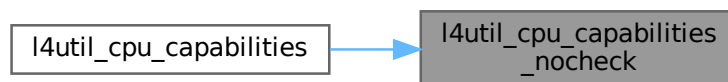
References [l4util\\_cpu\\_cpuid\(\)](#).

Referenced by [l4util\\_cpu\\_capabilities\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 13.16.6.2.3 l4util\_cpu\_has\_cpuid()

```
int l4util_cpu_has_cpuid (  
    void ) [inline]
```

Check whether the CPU supports the "cpuid" instruction.

**Returns**

1 if it has, 0 if it has not

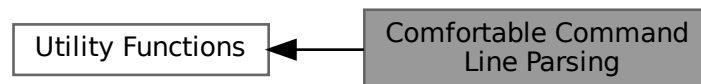
Definition at line 66 of file [cpu.h](#).

Referenced by [l4util\\_cpu\\_capabilities\(\)](#).

Here is the caller graph for this function:

**13.16.7 Comfortable Command Line Parsing**

Collaboration diagram for Comfortable Command Line Parsing:

**Typedefs**

- typedef void(\* [parse\\_cmd\\_fn\\_t](#)) (int)  
*Function type for PARSE\_CMD\_FN.*
- typedef void(\* [parse\\_cmd\\_fn\\_arg\\_t](#)) (int, const char \*, int)  
*Function type for PARSE\_CMD\_FN\_ARG.*

**Enumerations**

- enum [parse\\_cmd\\_type](#)  
*Types for parsing.*

**Functions**

- int [parse\\_cmdline](#) (int \*argc, const char \*\*\*argv, int arg0,...)  
*Parse the command-line for specified arguments and store the values into variables.*

### 13.16.7.1 Detailed Description

### 13.16.7.2 Function Documentation

#### 13.16.7.2.1 `parse_cmdline()`

```
int parse_cmdline (
    int * argc,
    const char *** argv,
    int arg0,
    ... )
```

Parse the command-line for specified arguments and store the values into variables.

This Functions gets the command-line, and a list of command-descriptors. Then, the command-line is parsed according to the given descriptors, storing strings, switches and numeric arguments at given addresses, and possibly calling specified functions. A default help descriptor is added. Its purpose is to present a short command overview in the case the given command-line does not fit to the descriptors.

Each command-descriptor has the following form:

*short option char, long option name, comment, type, val, addr.*

The *short option char* specifies the short form of the described option. The short form will be recognized after a single dash, or in a group of short options preceeded by a single dash. Specify ' ' if no short form should be used.

The *long option name* specifies the long form of the described option. The long form will be recognized after two dashes. Specify 0 if no long form should be used for this option.

The *comment* is a string that will be used when presenting the short command-line help.

The *type* specifies, if the option should be recognized as

- a number (PARSE\_CMD\_INT),
- a switch (PARSE\_CMD\_SWITCH),
- a string (PARSE\_CMD\_STRING),
- a function call (PARSE\_CMD\_FN, PARSE\_CMD\_FN\_ARG),
- an increment/decrement operator (PARSE\_CMD\_INC, PARSE\_CMD\_DEC).

If *type* is PARSE\_CMD\_INT, the option requires a second argument on the command-line after the option. This argument is parsed as a number. It can be preceeded by 0x to present a hex-value or by 0 to present an octal form. *addr* is interpreted as an int-pointer. The scanned argument from the command-line is stored in this pointer.

If *type* is PARSE\_CMD\_SWITCH, *addr* must be a pointer to int, and the value from *val* is stored at this pointer.

With PARSE\_CMD\_STRING, an additional argument is expected at the cmdline. *addr* must be a pointer to const char\*, and a pointer to the argument on the command line is stored at this pointer. The value in *val* is a default value, which is stored at *addr* if the corresponding option is not given on the command line.

With PARSE\_CMD\_FN\_ARG, *addr* is interpreted as a function pointer of type `parse_cmd_fn_t`. It will be called with *val* as argument if the corresponding option is found.

If *type* is PARSE\_CMD\_FN\_ARG, *addr* is as a function pointer of type `parse_cmd_fn_arg_t`, and handled similar to PARSE\_CMD\_FN. An additional argument is expected at the command line, however. It is given to the called function as 2nd argument, and parsed as an integer as with PARSE\_CMD\_INT as a third argument.

If *type* is PARSE\_CMD\_INC or PARSE\_CMD\_DEC, *addr* is interpreted as an int-pointer. The value of *val* is stored to this pointer first. For every occurrence of the option in the command line, the integer referenced by *addr* is incremented or decremented, respectively.

The list of command-descriptors is terminated by specifying a binary 0 for the short option char.

Note: The short option char 'h' and the long option name "help" must not be specified. They are used for the default help descriptor and produce a short command-options help when specified on the command-line.

**Parameters**

<i>argc</i>	pointer to number of command line parameters as passed to main
<i>argv</i>	pointer to array of command line parameters as passed to main
<i>arg0</i>	format list describing the command line options to parse for

**Returns**

0 if the command-line was successfully parsed, otherwise:

- -1 if the given descriptors are somehow wrong.
- -2 if not enough memory was available to hold temporary structs.
- -3 if the given command-line args did not meet the specified set.
- -4 if the help-option was given.

Upon return, *argc* and *argv* point to a list of arguments that were not scanned as arguments. See `getoptlong` for details on scanning.

**13.16.8 ELF binary format**

Functions and types related to ELF binaries.

Collaboration diagram for ELF binary format:

**Files**

- file [elf.h](#)  
*ELF definition.*

**Data Structures**

- struct [Elf32\\_Ehdr](#)  
*ELF32 header.*
- struct [Elf64\\_Ehdr](#)  
*ELF64 header.*
- struct [Elf32\\_Shdr](#)  
*ELF32 section header.*
- struct [Elf64\\_Shdr](#)  
*ELF64 section header.*

- struct [Elf32\\_Phdr](#)  
*ELF32 program header.*
- struct [Elf64\\_Phdr](#)  
*ELF64 program header.*
- struct [Elf32\\_Dyn](#)  
*ELF32 dynamic entry.*
- struct [Elf64\\_Dyn](#)  
*ELF64 dynamic entry.*
- struct [Elf32\\_Rel](#)  
*ELF32 relocation entry w/o addend.*
- struct [Elf32\\_Rela](#)  
*ELF32 relocation entry w/ addend.*
- struct [Elf64\\_Rel](#)  
*ELF64 relocation entry w/o addend.*
- struct [Elf64\\_Rela](#)  
*ELF64 relocation entry w/ addend.*
- struct [Elf32\\_Sym](#)  
*ELF32 symbol table entry.*
- struct [Elf64\\_Sym](#)  
*ELF64 symbol table entry.*
- struct [Elf32\\_Auxv](#)  
*Auxiliary vector (32-bit).*
- struct [Elf64\\_Auxv](#)  
*Auxiliary vector (64-bit).*

## Macros

- #define **ElfW**(type) \_ElfW(Elf, 32, type)  
*Use 64 or 32 bits types depending on the target architecture.*
- #define **ELF32\_R\_SYM**(i) ((i)>>8)  
*Symbol table index.*
- #define **ELF32\_R\_TYPE**(i) ((unsigned char)(i))
- #define **ELF32\_R\_INFO**(s, t) (((s)<<8)+(unsigned char)(t))  
*Create info from symbol table index + type.*
- #define **ELF64\_R\_SYM**(i) ((i)>>32)  
*Symbol table index.*
- #define **ELF64\_R\_TYPE**(i) ((i)&0xffffffffL)
- #define **ELF64\_R\_INFO**(s, t) (((s)<<32)+(t)&0xffffffffL)  
*Create info from symbol table index + type.*
- #define **ELF32\_ST\_BIND**(i) ((i)>>4)
- #define **ELF32\_ST\_TYPE**(i) ((i)&0xf)
- #define **ELF32\_ST\_INFO**(b, t) (((b)<<4)+((t)&0xf))  
*Make info from bind + type.*
- #define **ELF64\_ST\_BIND**(i) ((i)>>4)
- #define **ELF64\_ST\_TYPE**(i) ((i)&0xf)
- #define **ELF64\_ST\_INFO**(b, t) (((b)<<4)+((t)&0xf))  
*Make info from bind + type.*

## Typedefs

- typedef struct [Elf32\\_Auxv](#) **Elf32\_Auxv**  
*Auxiliary vector (32-bit).*
- typedef struct [Elf64\\_Auxv](#) **Elf64\_Auxv**  
*Auxiliary vector (64-bit).*

## Enumerations

- enum { [EI\\_NIDENT](#) = 16 }
- enum [Elf\\_ETs](#) {  
[ET\\_NONE](#) = 0 , [ET\\_REL](#) = 1 , [ET\\_EXEC](#) = 2 , [ET\\_DYN](#) = 3 ,  
[ET\\_CORE](#) = 4 , [ET\\_LOPROC](#) = 0xff00 , [ET\\_HIPROC](#) = 0xffff }  
*Object file type.*
- enum [Elf\\_EMs](#) {  
[EM\\_NONE](#) = 0 , [EM\\_M32](#) = 1 , [EM\\_SPARC](#) = 2 , [EM\\_386](#) = 3 ,  
[EM\\_68K](#) = 4 , [EM\\_88K](#) = 5 , [EM\\_860](#) = 7 , [EM\\_MIPS](#) = 8 ,  
[EM\\_MIPS\\_RS4\\_BE](#) = 10 , [EM\\_SPARC64](#) = 11 , [EM\\_PARISC](#) = 15 , [EM\\_VPP500](#) = 17 ,  
[EM\\_SPARC32PLUS](#) = 18 , [EM\\_960](#) = 19 , [EM\\_PPC](#) = 20 , [EM\\_V800](#) = 36 ,  
[EM\\_FR20](#) = 37 , [EM\\_RH32](#) = 38 , [EM\\_RCE](#) = 39 , [EM\\_ARM](#) = 40 ,  
[EM\\_ALPHA](#) = 41 , [EM\\_SH](#) = 42 , [EM\\_SPARCV9](#) = 43 , [EM\\_TRICORE](#) = 44 ,  
[EM\\_ARC](#) = 45 , [EM\\_H8\\_300](#) = 46 , [EM\\_H8\\_300H](#) = 47 , [EM\\_H8S](#) = 48 ,  
[EM\\_H8\\_500](#) = 49 , [EM\\_IA\\_64](#) = 50 , [EM\\_MIPS\\_X](#) = 51 , [EM\\_COLDFIRE](#) = 52 ,  
[EM\\_68HC12](#) = 53 , [EM\\_X86\\_64](#) = 62 , [EM\\_PDSP](#) = 63 , [EM\\_FX66](#) = 66 ,  
[EM\\_ST9PLUS](#) = 67 , [EM\\_ST7](#) = 68 , [EM\\_68HC16](#) = 69 , [EM\\_68HC11](#) = 70 ,  
[EM\\_68HC08](#) = 71 , [EM\\_68HC05](#) = 72 , [EM\\_SVX](#) = 73 , [EM\\_ST19](#) = 74 ,  
[EM\\_VAX](#) = 75 , [EM\\_CRIS](#) = 76 , [EM\\_JAVELIN](#) = 77 , [EM\\_FIREPATH](#) = 78 ,  
[EM\\_ZSP](#) = 79 , [EM\\_MMIX](#) = 80 , [EM\\_HUANY](#) = 81 , [EM\\_PRISM](#) = 82 ,  
[EM\\_AVR](#) = 83 , [EM\\_FR30](#) = 84 , [EM\\_D10V](#) = 85 , [EM\\_D30V](#) = 86 ,  
[EM\\_V850](#) = 87 , [EM\\_M32R](#) = 88 , [EM\\_MN10300](#) = 89 , [EM\\_MN10200](#) = 90 ,  
[EM\\_PJ](#) = 91 , [EM\\_OPENRISC](#) = 92 , [EM\\_ARC\\_A5](#) = 93 , [EM\\_XTENSA](#) = 94 ,  
[EM\\_ALTERA\\_NIOS2](#) = 113 , [EM\\_AARCH64](#) = 183 , [EM\\_TILEPRO](#) = 188 , [EM\\_MICROBLAZE](#) = 189 ,  
[EM\\_TILEGX](#) = 191 , [EM\\_RISCV](#) = 243 , [EM\\_NUM](#) = 244 }  
*Required architecture.*
- enum [Elf\\_EVs](#) { [EV\\_NONE](#) = 0 , [EV\\_CURRENT](#) = 1 }
- enum [Elf\\_EIs](#) {  
[EI\\_MAG0](#) = 0 , [EI\\_MAG1](#) = 1 , [EI\\_MAG2](#) = 2 , [EI\\_MAG3](#) = 3 ,  
[EI\\_CLASS](#) = 4 , [EI\\_DATA](#) = 5 , [EI\\_VERSION](#) = 6 , [EI\\_OSABI](#) = 7 ,  
[EI\\_ABIVERSION](#) = 8 , [EI\\_PAD](#) = 9 }
- enum [Elf\\_MAGs](#) { [ELFMAG0](#) = 0x7f , [ELFMAG1](#) = 'E' , [ELFMAG2](#) = 'L' , [ELFMAG3](#) = 'F' }
- enum [Elf\\_CIASSs](#) { [ELFCLASSNONE](#) = 0 , [ELFCLASS32](#) = 1 , [ELFCLASS64](#) = 2 , [ELFCLASSNUM](#) = 3 }
- enum [Elf\\_DATAs](#) { [ELFDATANONE](#) = 0 , [ELFDATA2LSB](#) = 1 , [ELFDATA2MSB](#) = 2 , [ELFDATANUM](#) = 3 }
- enum [Elf\\_OSABIs](#) {  
[ELFOSABI\\_NONE](#) = 0 , [ELFOSABI\\_SYSV](#) = 0 , [ELFOSABI\\_HPUX](#) = 1 , [ELFOSABI\\_NETBSD](#) = 2 ,  
[ELFOSABI\\_LINUX](#) = 3 , [ELFOSABI\\_SOLARIS](#) = 6 , [ELFOSABI\\_AIX](#) = 7 , [ELFOSABI\\_IRIX](#) = 8 ,  
[ELFOSABI\\_FREEBSD](#) = 9 , [ELFOSABI\\_TRU64](#) = 10 , [ELFOSABI\\_MODESTO](#) = 11 , [ELFOSABI\\_OPENBSD](#)  
= 12 ,  
[ELFOSABI\\_ARM](#) = 97 , [ELFOSABI\\_STANDALONE](#) = 255 }  
*Identify operating system and ABI to which the object is targeted.*

- enum `Elf_SHNs` {  
`SHN_UNDEF` = 0 , `SHN_LORESERVE` = 0xff00 , `SHN_LOPROC` = 0xff00 , `SHN_HIPROC` = 0xff1f ,  
`SHN_ABS` = 0xffff , `SHN_COMMON` = 0xffff2 , `SHN_HIRESERVE` = 0xffff }  
*Special section indexes.*
- enum `Elf_SHTs` {  
`SHT_NULL` = 0 , `SHT_PROGBITS` = 1 , `SHT_SYMTAB` = 2 , `SHT_STRTAB` = 3 ,  
`SHT_RELA` = 4 , `SHT_HASH` = 5 , `SHT_DYNAMIC` = 6 , `SHT_NOTE` = 7 ,  
`SHT_NOBITS` = 8 , `SHT_REL` = 9 , `SHT_SHLIB` = 10 , `SHT_DYNSYM` = 11 ,  
`SHT_INIT_ARRAY` = 14 , `SHT_FINI_ARRAY` = 15 , `SHT_PREINIT_ARRAY` = 16 , `SHT_GROUP` = 17 ,  
`SHT_SYMTAB_SHNDX` = 18 , `SHT_NUM` = 19 , `SHT_LOOS` = 0x60000000 , `SHT_HIOS` = 0x6fffffff ,  
`SHT_LOPROC` = 0x70000000 , `SHT_HIPROC` = 0x7fffffff , `SHT_LOUSER` = 0x80000000 , `SHT_HIUSER` =  
0xffffffff }  
*Section type.*
- enum `Elf_SHFs` {  
`SHF_WRITE` = 0x1 , `SHF_ALLOC` = 0x2 , `SHF_EXECINSTR` = 0x4 , `SHF_MERGE` = 0x10 ,  
`SHF_STRINGS` = 0x20 , `SHF_INFO_LINK` = 0x40 , `SHF_LINK_ORDER` = 0x80 , `SHF_OS_NONCONFORMING`  
= 0x100 ,  
`SHF_GROUP` = 0x200 , `SHF_TLS` = 0x400 , `SHF_MASKOS` = 0x0ff00000 , `SHF_MASKPROC` = 0xf0000000  
}  
*Section attribute flags.*
- enum `Elf_PTs` {  
`PT_NULL` = 0 , `PT_LOAD` = 1 , `PT_DYNAMIC` = 2 , `PT_INTERP` = 3 ,  
`PT_NOTE` = 4 , `PT_SHLIB` = 5 , `PT_PHDR` = 6 , `PT_TLS` = 7 ,  
`PT_NUM` = 8 , `PT_LOOS` = 0x60000000 , `PT_HIOS` = 0x6fffffff , `PT_LOPROC` = 0x70000000 ,  
`PT_HIPROC` = 0x7fffffff , `PT_GNU_EH_FRAME` = `PT_LOOS` + 0x474e550 , `PT_GNU_STACK` = `PT_LOOS`  
+ 0x474e551 , `PT_GNU_RELRO` = `PT_LOOS` + 0x474e552 ,  
`PT_L4_STACK` = `PT_LOOS` + 0x12 , `PT_L4_KIP` = `PT_LOOS` + 0x13 , `PT_L4_AUX` = `PT_LOOS` + 0x14 }  
*Segment types.*
- enum `Elf_PFs` {  
`PF_X` = 0x1 , `PF_W` = 0x2 , `PF_R` = 0x4 , `PF_MASKOS` = 0x0ff00000 ,  
`PF_MASKPROC` = 0x7fffffff }  
*Segment permissions.*
- enum `Elf_NTscore` {  
`NT_PRSTATUS` = 1 , `NT_FPREGSET` = 2 , `NT_PRPSINFO` = 3 , `NT_PRXREG` = 4 ,  
`NT_TASKSTRUCT` = 4 , `NT_PLATFORM` = 5 , `NT_AUXV` = 6 , `NT_GWINDOWS` = 7 ,  
`NT_ASRS` = 8 , `NT_PSTATUS` = 10 , `NT_PSINFO` = 13 , `NT_PRCRED` = 14 ,  
`NT_UTSNAME` = 15 , `NT_LWPSTATUS` = 16 , `NT_LWPSINFO` = 17 , `NT_PRFPXREG` = 20 }  
*Legal values for note segment descriptor types for core files.*
- enum `Elf_NTsobj` { `NT_VERSION` = 1 }  
*Legal values for the note segment descriptor types for object files.*
- enum `Elf_DTsc` {  
`DT_NULL` = 0 , `DT_NEEDED` = 1 , `DT_PLTRELSZ` = 2 , `DT_PLTGOT` = 3 ,  
`DT_HASH` = 4 , `DT_STRTAB` = 5 , `DT_SYMTAB` = 6 , `DT_RELA` = 7 ,  
`DT_RELASZ` = 8 , `DT_RELAENT` = 9 , `DT_STRSZ` = 10 , `DT_SYMENT` = 11 ,  
`DT_INIT` = 12 , `DT_FINI` = 13 , `DT_SONAME` = 14 , `DT_RPATH` = 15 ,  
`DT_SYMBOLIC` = 16 , `DT_REL` = 17 , `DT_RELSZ` = 18 , `DT_RELENT` = 19 ,  
`DT_PTRREL` = 20 , `DT_DEBUG` = 21 , `DT_TEXTREL` = 22 , `DT_JMPREL` = 23 ,  
`DT_BIND_NOW` = 24 , `DT_INIT_ARRAY` = 25 , `DT_FINI_ARRAY` = 26 , `DT_INIT_ARRAYSZ` = 27 ,  
`DT_FINI_ARRAYSZ` = 28 , `DT_RUNPATH` = 29 , `DT_FLAGS` = 30 , `DT_ENCODING` = 32 ,  
`DT_PREINIT_ARRAY` = 32 , `DT_PREINIT_ARRAYSZ` = 33 , `DT_NUM` = 34 , `DT_LOOS` = 0x6000000d ,  
`DT_HIOS` = 0x6ffff000 , `DT_LOPROC` = 0x70000000 , `DT_HIPROC` = 0x7fffffff }  
*Dynamic Array Tags.*
- enum `Elf_DFs` {  
`DF_ORIGIN` = 0x00000001 , `DF_SYMBOLIC` = 0x00000002 , `DF_TEXTREL` = 0x00000004 ,  
`DF_BIND_NOW` = 0x00000008 ,  
`DF_STATIC_TLS` = 0x00000010 }



*Values of Elf32\_Dyn.d\_un.d\_val, Elf64\_Dyn.d\_un.d\_val in the DT\_FLAGS entry.*

- enum `Elf_DF_1s` {  
`DF_1_NOW` = 0x00000001 , `DF_1_GLOBAL` = 0x00000002 , `DF_1_GROUP` = 0x00000004 ,  
`DF_1_NODELETE` = 0x00000008 ,  
`DF_1_LOADFLTR` = 0x00000010 , `DF_1_INITFIRST` = 0x00000020 , `DF_1_NOOPEN` = 0x00000040 ,  
`DF_1_ORIGIN` = 0x00000080 ,  
`DF_1_DIRECT` = 0x00000100 , `DF_1_TRANS` = 0x00000200 , `DF_1_INTERPOSE` = 0x00000400 ,  
`DF_1_NODEFLIB` = 0x00000800 ,  
`DF_1_NODUMP` = 0x00001000 , `DF_1_CONFALT` = 0x00002000 , `DF_1_ENDFILTEE` = 0x00004000 ,  
`DF_1_DISPRELDNE` = 0x00008000 ,  
`DF_1_DISPRELPND` = 0x00010000 }

*State flags selectable in the Elf32\_Dyn.d\_un.d\_val / Elf64\_Dyn.d\_un.d\_val element of the DT\_FLAGS\_1 entry in the dynamic section.*

- enum `Elf_DTF_1s`

*Flags for the feature selection in DT\_FEATURE\_1.*

- enum `Elf_DF_P1s` { `DF_P1_LAZYLOAD` = 0x00000001 , `DF_P1_GROUPPERM` = 0x00000002 }

*Flags in the DT\_POSFLAG\_1 entry effecting only the next DT\_\* entry.*

- enum `Elf_R_386_s` {  
`R_386_NONE` = 0 , `R_386_32` = 1 , `R_386_PC32` = 2 , `R_386_GOT32` = 3 ,  
`R_386_PLT32` = 4 , `R_386_COPY` = 5 , `R_386_GLOB_DAT` = 6 , `R_386_JMP_SLOT` = 7 ,  
`R_386_RELATIVE` = 8 , `R_386_GOTOFF` = 9 , `R_386_GOTPC` = 10 , `R_386_32PLT` = 11 ,  
`R_386_TLS_TPOFF` = 14 , `R_386_TLS_IE` = 15 , `R_386_TLS_GOTIE` = 16 , `R_386_TLS_LE` = 17 ,  
`R_386_TLS_GD` = 18 , `R_386_TLS_LDM` = 19 , `R_386_16` = 20 , `R_386_PC16` = 21 ,  
`R_386_8` = 22 , `R_386_PC8` = 23 , `R_386_TLS_GD_32` = 24 , `R_386_TLS_GD_PUSH` = 25 ,  
`R_386_TLS_GD_CALL` = 26 , `R_386_TLS_GD_POP` = 27 , `R_386_TLS_LDM_32` = 28 , `R_386_TLS_LDM_PUSH`  
= 29 ,  
`R_386_TLS_LDM_CALL` = 30 , `R_386_TLS_LDM_POP` = 31 , `R_386_TLS_LDO_32` = 32 , `R_386_TLS_IE_32`  
= 33 ,  
`R_386_TLS_LE_32` = 34 , `R_386_TLS_DTPMOD32` = 35 , `R_386_TLS_DTPOFF32` = 36 , `R_386_TLS_TPOFF32`  
= 37 ,  
`R_386_NUM` = 38 }

*Relocation types (processor specific).*

- enum `Elf_EF_ARM_s` { }

*ARM specific declarations.*

- enum `Elf_STT_ARM_s`

*Additional symbol types for Thumb.*

- enum `Elf_SHF_s_ARM` { `SHF_ARM_ENTRYSECT` = 0x10000000 , `SHF_ARM_COMDEF` = 0x80000000 }

*ARM-specific values for Elf32\_Shdr.sh\_flags / Elf64\_Shdr.sh\_flags.*

- enum `Elf_ARM_SBs` { `PF_ARM_SB` = 0x10000000 }

*ARM-specific program header flags.*

- enum `Elf_R_ARM_s` {  
`R_ARM_NONE` = 0 , `R_ARM_PC24` = 1 , `R_ARM_ABS32` = 2 , `R_ARM_REL32` = 3 ,  
`R_ARM_PC13` = 4 , `R_ARM_ABS16` = 5 , `R_ARM_ABS12` = 6 , `R_ARM_THM_ABS5` = 7 ,  
`R_ARM_ABS8` = 8 , `R_ARM_SBREL32` = 9 , `R_ARM_THM_PC22` = 10 , `R_ARM_THM_PC8` = 11 ,  
`R_ARM_AMP_VCALL9` = 12 , `R_ARM_SWI24` = 13 , `R_ARM_THM_SWI8` = 14 , `R_ARM_XPC25` = 15 ,  
`R_ARM_THM_XPC22` = 16 , `R_ARM_COPY` = 20 , `R_ARM_GLOB_DAT` = 21 , `R_ARM_JUMP_SLOT` = 22 ,  
`R_ARM_RELATIVE` = 23 , `R_ARM_GOTOFF` = 24 , `R_ARM_GOTPC` = 25 , `R_ARM_GOT32` = 26 ,  
`R_ARM_PLT32` = 27 , `R_ARM_ALU_PCREL_7_0` = 32 , `R_ARM_ALU_PCREL_15_8` = 33 , `R_ARM_↵`  
`ALU_PCREL_23_15` = 34 ,  
`R_ARM_LDR_SBREL_11_0` = 35 , `R_ARM_ALU_SBREL_19_12` = 36 , `R_ARM_ALU_SBREL_27_20` =  
37 , `R_ARM_GNU_VTENTRY` = 100 ,  
`R_ARM_GNU_VTINHERIT` = 101 , `R_ARM_THM_PC11` = 102 , `R_ARM_THM_PC9` = 103 , `R_ARM_↵`  
`RXPC25` = 249 ,  
`R_ARM_RSBREL32` = 250 , `R_ARM_THM_RPC22` = 251 , `R_ARM_RREL32` = 252 , `R_ARM_RABS22` =  
253 ,  
`R_ARM_RPC24` = 254 , `R_ARM_RBASE` = 255 , `R_ARM_NUM` = 256 }

*ARM relocations.*

- enum `Elf_R_AARCH64_s` { `R_AARCH64_NONE` = 0 , `R_AARCH64_RELATIVE` = 1027 }

*AARCH64 relocations.*

- enum `Elf_R_X86_64_s` {  
`R_X86_64_NONE` = 0 , `R_X86_64_64` = 1 , `R_X86_64_PC32` = 2 , `R_X86_64_GOT32` = 3 ,  
`R_X86_64_PLT32` = 4 , `R_X86_64_COPY` = 5 , `R_X86_64_GLOB_DAT` = 6 , `R_X86_64_JUMP_SLOT` = 7 ,  
`R_X86_64_RELATIVE` = 8 , `R_X86_64_GOTPCREL` = 9 , `R_X86_64_32` = 10 , `R_X86_64_32S` = 11 ,  
`R_X86_64_16` = 12 , `R_X86_64_PC16` = 13 , `R_X86_64_8` = 14 , `R_X86_64_PC8` = 15 ,  
`R_X86_64_DTPMOD64` = 16 , `R_X86_64_DTPOFF64` = 17 , `R_X86_64_TPOFF64` = 18 , `R_X86_64_TLSGD`  
= 19 ,  
`R_X86_64_TLSLD` = 20 , `R_X86_64_DTPOFF32` = 21 , `R_X86_64_GOTTPOFF` = 22 , `R_X86_64_TPOFF32`  
= 23 ,  
`R_X86_64_NUM` = 24 }

*AMD x86-64 relocations.*

- enum `Elf_STNs`  
*Symbol Table Entry.*
- enum `Elf_STBs` {  
`STB_LOCAL` = 0 , `STB_GLOBAL` = 1 , `STB_WEAK` = 2 , `STB_LOOS` = 10 ,  
`STB_HIOS` = 12 , `STB_LOPROC` = 13 , `STB_HIPROC` = 15 }

*Symbol Binding.*

- enum `Elf_STTs` {  
`STT_NOTYPE` = 0 , `STT_OBJECT` = 1 , `STT_FUNC` = 2 , `STT_SECTION` = 3 ,  
`STT_FILE` = 4 , `STT_LOOS` = 10 , `STT_HIOS` = 12 , `STT_LOPROC` = 13 ,  
`STT_HIPROC` = 15 }

*Symbol Types.*

- enum `Elf_ATs` {  
`AT_NULL` = 0 , `AT_IGNORE` = 1 , `AT_EXECFD` = 2 , `AT_PHDR` = 3 ,  
`AT_PHENT` = 4 , `AT_PHNUM` = 5 , `AT_PAGESZ` = 6 , `AT_BASE` = 7 ,  
`AT_FLAGS` = 8 , `AT_ENTRY` = 9 , `AT_NOTELF` = 10 , `AT_UID` = 11 ,  
`AT_EUID` = 12 , `AT_GID` = 13 , `AT_EGID` = 14 , `AT_L4_AUX` = 0xf0 ,  
`AT_L4_ENV` = 0xf1 }

*Legal values for `Elf32_Auxv.atype` / `Elf64_Auxv.atype`.*

## ELF types

- typedef `l4_uint32_t` `Elf32_Addr`  
*size 4 align 4*
- typedef `l4_uint32_t` `Elf32_Off`  
*size 4 align 4*
- typedef `l4_uint16_t` `Elf32_Half`  
*size 2 align 2*
- typedef `l4_uint32_t` `Elf32_Word`  
*size 4 align 4*
- typedef `l4_int32_t` `Elf32_Sword`  
*size 4 align 4*
- typedef `l4_uint64_t` `Elf64_Addr`  
*size 8 align 8*
- typedef `l4_uint64_t` `Elf64_Off`  
*size 8 align 8*
- typedef `l4_uint16_t` `Elf64_Half`  
*size 2 align 2*
- typedef `l4_uint32_t` `Elf64_Word`  
*size 4 align 4*

- typedef [l4\\_int32\\_t](#) Elf64\_Sword  
*size 4 align 4*
- typedef [l4\\_uint64\\_t](#) Elf64\_Xword  
*size 8 align 8*
- typedef [l4\\_int64\\_t](#) Elf64\_Sxword  
*size 8 align 8*

### 13.16.8.1 Detailed Description

Functions and types related to ELF binaries.

### 13.16.8.2 Macro Definition Documentation

#### 13.16.8.2.1 ELF32\_R\_TYPE

```
#define ELF32_R_TYPE(  
    i ) ((unsigned char)(i))
```

See also

[Elf\\_R\\_386s](#).

Definition at line [668](#) of file [elf.h](#).

#### 13.16.8.2.2 ELF32\_ST\_BIND

```
#define ELF32_ST_BIND(  
    i ) ((i)>>4)
```

See also

[Elf\\_STBs](#).

Definition at line [898](#) of file [elf.h](#).

#### 13.16.8.2.3 ELF32\_ST\_TYPE

```
#define ELF32_ST_TYPE(  
    i ) ((i)&0xf)
```

See also

[Elf\\_STTs](#).

Definition at line [901](#) of file [elf.h](#).

#### 13.16.8.2.4 ELF64\_R\_TYPE

```
#define ELF64_R_TYPE(  
    i ) ((i) & 0xffffffffL)
```

See also

[Elf\\_R\\_386s](#).

Definition at line [676](#) of file [elf.h](#).

#### 13.16.8.2.5 ELF64\_ST\_BIND

```
#define ELF64_ST_BIND(  
    i ) ((i) >> 4)
```

See also

[Elf\\_STBs](#)

Definition at line [907](#) of file [elf.h](#).

#### 13.16.8.2.6 ELF64\_ST\_TYPE

```
#define ELF64_ST_TYPE(  
    i ) ((i) & 0xf)
```

See also

[Elf\\_STTs](#)

Definition at line [910](#) of file [elf.h](#).

### 13.16.8.3 Enumeration Type Documentation

#### 13.16.8.3.1 anonymous enum

anonymous enum

Enumerator

EI_NIDENT	Number of characters.
-----------	-----------------------

Definition at line [121](#) of file [elf.h](#).

**13.16.8.3.2 Elf\_ARM\_SBs**

enum [Elf\\_ARM\\_SBs](#)

ARM-specific program header flags.

Enumerator

PF_ARM_SB	Segment contains the location addressed by the static base.
-----------	---

Definition at line [775](#) of file [elf.h](#).

**13.16.8.3.3 Elf\_ATs**

enum [Elf\\_ATs](#)

Legal values for [Elf32\\_Auxv.atype](#) / [Elf64\\_Auxv.atype](#).

Enumerator

AT_NULL	End of vector.
AT_IGNORE	Entry should be ignored.
AT_EXECD	File descriptor of program.
AT_PHDR	Program headers for program.
AT_PHEM	Size of program header entry.
AT_PHNUM	Number of program headers.
AT_PAGESZ	System page size.
AT_BASE	Base address of interpreter.
AT_FLAGS	Flags.
AT_ENTRY	Entry point of program.
AT_NOTELF	Program is not ELF.
AT_UID	Real UID.
AT_EUID	Effective UID.
AT_GID	Real GID.
AT_EGID	Effective GID.
AT_L4_AUX	<a href="#">L4Re</a> AUX section.
AT_L4_ENV	<a href="#">L4Re</a> ENV section.

Definition at line [944](#) of file [elf.h](#).

**13.16.8.3.4 Elf\_CLASSs**

enum [Elf\\_CLASSs](#)

File class or capacity.

**Enumerator**

ELFCLASSNONE	Invalid class.
ELFCLASS32	32-bit object
ELFCLASS64	64-bit object
ELFCLASSNUM	Mask for 32-bit or 64-bit class.

Definition at line 302 of file [elf.h](#).

**13.16.8.3.5 Elf\_DATAs**

enum [Elf\\_DATAs](#)

Data encoding.

**Enumerator**

ELFDATANONE	invalid data encoding
ELFDATA2LSB	0x01020304 => [ 0x04 0x03 0x02 0x01 ]
ELFDATA2MSB	0x01020304 => [ 0x01 0x02 0x03 0x04 ]
ELFDATANUM	Mask for valid data encoding.

Definition at line 311 of file [elf.h](#).

**13.16.8.3.6 Elf\_DF\_1s**

enum [Elf\\_DF\\_1s](#)

State flags selectable in the Elf32\_Dyn.d\_un.d\_val / Elf64\_Dyn.d\_un.d\_val element of the DT\_FLAGS\_1 entry in the dynamic section.

**Enumerator**

DF_1_NOW	Set RTLD_NOW for this object.
DF_1_GLOBAL	Set RTLD_GLOBAL for this object.
DF_1_GROUP	Set RTLD_GROUP for this object.
DF_1_NODELETE	Set RTLD_NODELETE for this object.
DF_1_LOADFLTR	Trigger filtee loading at runtime.
DF_1_INITFIRST	Set RTLD_INITFIRST for this object.
DF_1_NOOPEN	Set RTLD_NOOPEN for this object.
DF_1_ORIGIN	\$ORIGIN must be handled.
DF_1_DIRECT	Direct binding enabled.
DF_1_INTERPOSE	Object is used to interpose.

## Enumerator

DF_1_NODEFLIB	Ignore default lib search path.
DF_1_NODUMP	Object can't be dldump'ed.
DF_1_CONFALT	Configuration alternative created.
DF_1_ENDFILTEE	Filtee terminates filters search.
DF_1_DISPRELDNE	Disp reloc applied at build time.
DF_1_DISPRELPND	Disp reloc applied at run-time.

Definition at line 599 of file [elf.h](#).

**13.16.8.3.7 Elf\_DF\_P1s**

enum [Elf\\_DF\\_P1s](#)

Flags in the DT\_POSFLAG\_1 entry effecting only the next DT\_\* entry.

## Enumerator

DF_P1_LAZYLOAD	Lazyload following object.
DF_P1_GROUPPERM	Symbols from next object are not generally available.

Definition at line 628 of file [elf.h](#).

**13.16.8.3.8 Elf\_DFs**

enum [Elf\\_DFs](#)

Values of Elf32\_Dyn.d\_un.d\_val, Elf64\_Dyn.d\_un.d\_val in the DT\_FLAGS entry.

## Enumerator

DF_ORIGIN	Object may use DF_ORIGIN.
DF_SYMBOLIC	Symbol resolutions starts here.
DF_TEXTREL	Object contains text relocations.
DF_BIND_NOW	No lazy binding for this object.
DF_STATIC_TLS	Module uses the static TLS model.

Definition at line 586 of file [elf.h](#).

**13.16.8.3.9 Elf\_DTs**

enum [Elf\\_DTs](#)

Dynamic Array Tags.

See also

[Elf32\\_Dyn.d\\_tag](#), [Elf64\\_Dyn.d\\_tag](#).

#### Enumerator

DT_NULL	end of _DYNAMIC array
DT_NEEDED	name of a needed library
DT_PLTRELSZ	total size of relocation entry
DT_PLTGOT	address assoc with prog link table
DT_HASH	address of symbol hash table
DT_STRTAB	address of string table
DT_SYMTAB	address of symbol table
DT_RELA	address of relocation table
DT_RELASZ	total size of relocation table
DT_RELAENT	size of DT_RELA relocation entry
DT_STRSZ	size of the string table
DT_SYMENT	size of a symbol table entry
DT_INIT	address of initialization function
DT_FINI	address of termination function
DT_SONAME	name of the shared object
DT_RPATH	search library path
DT_SYMBOLIC	alter symbol resolution algorithm
DT_REL	address of relocation table
DT_RELSZ	total size of DT_REL relocation table
DT_RELENT	size of the DT_REL relocation entry
DT_PTRREL	type of relocation entry
DT_DEBUG	for debugging purposes
DT_TEXTREL	at least on entry changes r/o section
DT_JMPREL	address of relocation entries
DT_BIND_NOW	Process relocations of object.
DT_INIT_ARRAY	Array with addresses of init fct.
DT_FINI_ARRAY	Array with addresses of fini fct.
DT_INIT_ARRAYSZ	Size in bytes of DT_INIT_ARRAY.
DT_FINI_ARRAYSZ	Size in bytes of DT_FINI_ARRAY.
DT_RUNPATH	Library search path.
DT_FLAGS	Flags for the object being loaded.
DT_ENCODING	Start of encoded range.
DT_PREINIT_ARRAY	Array with addresses of preinit fct.
DT_PREINIT_ARRAYSZ	size in bytes of DT_PREINIT_ARRAY
DT_NUM	Number used.
DT_LOOS	Start of OS-specific.
DT_HIOS	End of OS-specific.
DT_LOPROC	processor-specific
DT_HIPROC	processor-specific

Definition at line 540 of file [elf.h](#).



**13.16.8.3.10 Elf\_EF\_ARM\_s**

enum [Elf\\_EF\\_ARM\\_s](#)

ARM specific declarations.

Processor specific flags for the ELF header e\_flags field.

**Enumerator**

EF_ARM_ALIGN8	8-bit structure alignment is in use
---------------	-------------------------------------

Definition at line [735](#) of file [elf.h](#).

**13.16.8.3.11 Elf\_EIs**

enum [Elf\\_EIs](#)

Identification Indices.

**See also**

[Elf32\\_Ehdr.e\\_ident](#), [Elf64\\_Ehdr.e\\_ident](#)

**Enumerator**

EI_MAG0	file id 0
EI_MAG1	file id 1
EI_MAG2	file id 2
EI_MAG3	file id 3
EI_CLASS	file class
EI_DATA	data encoding
EI_VERSION	file version
EI_OSABI	Operating system / ABI identification.
EI_ABIVERSION	ABI version.
EI_PAD	start of padding bytes

Definition at line [278](#) of file [elf.h](#).

**13.16.8.3.12 Elf\_EMs**

enum [Elf\\_EMs](#)

Required architecture.

**See also**

[Elf32\\_Ehdr.e\\_machine](#), [Elf64\\_Ehdr.e\\_machine](#)

## Enumerator

EM_NONE	no machine
EM_M32	AT&T WE 32100.
EM_SPARC	SPARC.
EM_386	Intel 80386.
EM_68K	Motorola 68000.
EM_88K	Motorola 88000.
EM_860	Intel 80860.
EM_MIPS	MIPS RS3000 big-endian.
EM_MIPS_RS4_BE	MIPS RS4000 big-endian.
EM_SPARC64	SPARC 64-bit.
EM_PARISC	HP PA-RISC.
EM_VPP500	Fujitsu VPP500.
EM_SPARC32PLUS	Sun's V8plus.
EM_960	Intel 80960.
EM_PPC	PowerPC.
EM_V800	NEC V800.
EM_FR20	Fujitsu FR20.
EM_RH32	TRW RH-32.
EM_RCE	Motorola RCE.
EM_ARM	Advanced RISC Machines ARM.
EM_ALPHA	Digital Alpha.
EM_SH	Hitachi SuperH.
EM_SPARCV9	SPARC v9 64-bit.
EM_TRICORE	Siemens Tricore embedded processor.
EM_ARC	Argonaut RISC Core, Argonaut Techn Inc.
EM_H8_300	Hitachi H8/300.
EM_H8_300H	Hitachi H8/300H.
EM_H8S	Hitachi H8/S.
EM_H8_500	Hitachi H8/500.
EM_IA_64	HP/Intel IA-64.
EM_MIPS_X	Stanford MIPS-X.
EM_COLDIRE	Motorola Coldfire.
EM_68HC12	Motorola M68HC12.
EM_X86_64	Advanced Micro Devices x86-64.
EM_PDSP	Sony DSP Processor.
EM_FX66	Siemens FX66 microcontroller.
EM_ST9PLUS	STMicroelectronics ST9+ 8/16 mc.
EM_ST7	STmicroelectronics ST7 8 bit mc.
EM_68HC16	Motorola MC68HC16 microcontroller.
EM_68HC11	Motorola MC68HC11 microcontroller.
EM_68HC08	Motorola MC68HC08 microcontroller.
EM_68HC05	Motorola MC68HC05 microcontroller.
EM_SVX	Silicon Graphics SVx.
EM_ST19	STMicroelectronics ST19 8 bit mc.
EM_VAX	Digital VAX.
EM_CRIS	Axis Communications 32-bit embedded processor.
EM_JAVELIN	Infineon Technologies 32-bit embedded processor.
EM_FIREPATH	Element 14 64-bit DSP Processor.

## Enumerator

EM_ZSP	LSI Logic 16-bit DSP Processor.
EM_MMIX	Donald Knuth's educational 64-bit processor.
EM_HUANY	Harvard University machine-independent object files.
EM_PRISM	SiTera Prism.
EM_AVR	Atmel AVR 8-bit microcontroller.
EM_FR30	Fujitsu FR30.
EM_D10V	Mitsubishi D10V.
EM_D30V	Mitsubishi D30V.
EM_V850	NEC v850.
EM_M32R	Mitsubishi M32R.
EM_MN10300	Matsushita MN10300.
EM_MN10200	Matsushita MN10200.
EM_PJ	picoJava
EM_OPENRISC	OpenRISC 32-bit embedded processor.
EM_ARC_A5	ARC Cores Tangent-A5.
EM_XTENSA	Tensilica Xtensa Architecture.
EM_ALTERA_NIOS2	Altera Nios II.
EM_AARCH64	ARM AARCH64.
EM_TILEPRO	Tilera TILEPro.
EM_MICROBLAZE	Xilinx MicroBlaze.
EM_TILEGX	Tilera TILE-Gx.
EM_RISCV	RISC-V.

Definition at line 187 of file [elf.h](#).

**13.16.8.3.13 Elf\_ETs**

```
enum Elf_ETs
```

Object file type.

See also

[Elf32\\_Ehdr.e\\_type](#), [Elf64\\_Ehdr.e\\_type](#)

## Enumerator

ET_NONE	no file type
ET_REL	relocatable file
ET_EXEC	executable file
ET_DYN	shared object file
ET_CORE	core file
ET_LOPROC	processor-specific
ET_HIPROC	processor-specific

Definition at line 172 of file [elf.h](#).

### 13.16.8.3.14 Elf\_EVs

enum [Elf\\_EVs](#)

Object file version.

See also

[Elf32\\_Ehdr.e\\_version](#), [Elf64\\_Ehdr.e\\_version](#)

Enumerator

EV_NONE	Invalid version.
EV_CURRENT	Current version.

Definition at line 270 of file [elf.h](#).

### 13.16.8.3.15 Elf\_MAGs

enum [Elf\\_MAGs](#)

Magic number.

Enumerator

ELFMAG0	e_ident[EI_MAG0]
ELFMAG1	e_ident[EI_MAG1]
ELFMAG2	e_ident[EI_MAG2]
ELFMAG3	e_ident[EI_MAG3]

Definition at line 293 of file [elf.h](#).

### 13.16.8.3.16 Elf\_NT\_s\_core

enum [Elf\\_NT\\_s\\_core](#)

Legal values for note segment descriptor types for core files.

Enumerator

NT_PRSTATUS	Contains copy of prstatus struct.
NT_FPREGSET	Contains copy of fpregset struct.
NT_PRPSINFO	Contains copy of prpsinfo struct.
NT_PRXREG	Contains copy of prxregset struct.
NT_TASKSTRUCT	Contains copy of task structure.
NT_PLATFORM	String from sysinfo(SI_PLATFORM)
NT_AUXV	Contains copy of auxv array.
NT_GWINDOWS	Contains copy of gwindows struct.

## Enumerator

NT_ASRS	Contains copy of asrset struct.
NT_PSTATUS	Contains copy of pstatus struct.
NT_PSINFO	Contains copy of psinfo struct.
NT_PRCRED	Contains copy of prcred struct.
NT_UTSNAME	Contains copy of utsname struct.
NT_LWPSTATUS	Contains copy of lwpstatus struct.
NT_LWPSINFO	Contains copy of lwpinfo struct.
NT_PRFPXREG	Contains copy of fpxregset struct.

Definition at line 491 of file [elf.h](#).

**13.16.8.3.17 Elf\_NTs\_obj**

enum [Elf\\_NTs\\_obj](#)

Legal values for the note segment descriptor types for object files.

## Enumerator

NT_VERSION	Contains a version string.
------------	----------------------------

Definition at line 512 of file [elf.h](#).

**13.16.8.3.18 Elf\_OSABIs**

enum [Elf\\_OSABIs](#)

Identify operating system and ABI to which the object is targeted.

## Enumerator

ELFOSABI_NONE	UNIX System V ABI.
ELFOSABI_SYSV	Alias.
ELFOSABI_HPUX	HP-UX.
ELFOSABI_NETBSD	NetBSD.
ELFOSABI_LINUX	Linux.
ELFOSABI_SOLARIS	Sun Solaris.
ELFOSABI_AIX	IBM AIX.
ELFOSABI_IRIX	SGI Irix.

## Enumerator

ELFOSABI_FREEBSD	FreeBSD.
ELFOSABI_TRU64	Compaq TRU64 UNIX.
ELFOSABI_MODESTO	Novell Modesto.
ELFOSABI_OPENBSD	OpenBSD.
ELFOSABI_ARM	ARM.
ELFOSABI_STANDALONE	Standalone (embedded) application.

Definition at line 320 of file [elf.h](#).

**13.16.8.3.19 ELF\_PFs**

enum [ELF\\_PFs](#)

Segment permissions.

## Enumerator

PF_X	Executable.
PF_W	Write.
PF_R	Read.
PF_MASKOS	OS-specific.
PF_MASKPROC	Processor-specific.

Definition at line 481 of file [elf.h](#).

**13.16.8.3.20 Elf\_PTs**

enum [Elf\\_PTs](#)

Segment types.

## Enumerator

PT_NULL	array is unused
PT_LOAD	loadable
PT_DYNAMIC	dynamic linking information
PT_INTERP	path to interpreter
PT_NOTE	auxiliary information
PT_SHLIB	reserved
PT_PHDR	location of the pht itself
PT_TLS	Thread-local storage segment.
PT_NUM	Number of defined types.
PT_LOOS	OS-specific.

## Enumerator

PT_HIOS	OS-specific.
PT_LOPROC	processor-specific
PT_HIPROC	processor-specific
PT_GNU_EH_FRAME	EH frame information.
PT_GNU_STACK	Flags for stack.
PT_GNU_RELRO	Read only after reloc.
PT_L4_STACK	Address of the stack.
PT_L4_KIP	Address of the KIP.
PT_L4_AUX	Address of the AUX structures.

Definition at line 455 of file [elf.h](#).

## 13.16.8.3.21 Elf\_R\_386\_s

enum [Elf\\_R\\_386\\_s](#)

Relocation types (processor specific).

## Enumerator

R_386_NONE	none
R_386_32	S + A.
R_386_PC32	S + A - P.
R_386_GOT32	G + A - P.
R_386_PLT32	L + A - P.
R_386_COPY	none
R_386_GLOB_DAT	S.
R_386_JMP_SLOT	S.
R_386_RELATIVE	B + A.
R_386_GOTOFF	S + A - GOT.
R_386_GOTPC	GOT + A - P.
R_386_TLS_TPOFF	Offset in static TLS block.
R_386_TLS_IE	Address of GOT entry for static TLS block offset.
R_386_TLS_GOTIE	GOT entry for static TLS block offset.
R_386_TLS_LE	Offset relative to static TLS block.
R_386_TLS_GD	Direct 32 bit for GNU version of general dynamic thread local data.
R_386_TLS_LDM	Direct 32 bit for GNU version of local dynamic thread local data in LE code.
R_386_TLS_GD_32	Direct 32 bit for general dynamic thread local data.
R_386_TLS_GD_PUSH	Tag for pushl in GD TLS code.
R_386_TLS_GD_CALL	Relocation for call to <code>__tls_get_addr()</code>
R_386_TLS_GD_POP	Tag for popl in GD TLS code.
R_386_TLS_LDM_32	Direct 32 bit for local dynamic thread local data in LE code.
R_386_TLS_LDM_PUSH	Tag for pushl in LDM TLS code.
R_386_TLS_LDM_CALL	Relocation for call to <code>__tls_get_addr()</code> in LDM code.
R_386_TLS_LDM_POP	Tag for popl in LDM TLS code.
R_386_TLS_LDO_32	Offset relative to TLS block.
R_386_TLS_IE_32	GOT entry for negated static TLS block offset.

## Enumerator

R_386_TLS_LE_32	Negated offset relative to static TLS block.
R_386_TLS_DTPMOD32	ID of module containing symbol.
R_386_TLS_DTPOFF32	Offset in TLS block.
R_386_TLS_TPOFF32	Negated offset in static TLS block.
R_386_NUM	Keep this the last entry.

Definition at line 682 of file [elf.h](#).

**13.16.8.3.22 Elf\_R\_AARCH64\_s**

enum [Elf\\_R\\_AARCH64\\_s](#)

AARCH64 relocations.

## Enumerator

R_AARCH64_NONE	No reloc.
----------------	-----------

Definition at line 830 of file [elf.h](#).

**13.16.8.3.23 Elf\_R\_ARM\_s**

enum [Elf\\_R\\_ARM\\_s](#)

ARM relocations.

## Enumerator

R_ARM_NONE	No reloc.
R_ARM_PC24	PC relative 26 bit branch.
R_ARM_ABS32	Direct 32 bit
R_ARM_REL32	PC relative 32 bit.
R_ARM_ABS16	Direct 16 bit.
R_ARM_ABS12	Direct 12 bit.
R_ARM_ABS8	Direct 8 bit.
R_ARM_COPY	Copy symbol at runtime.
R_ARM_GLOB_DAT	Create GOT entry.
R_ARM_JUMP_SLOT	Create PLT entry.
R_ARM_RELATIVE	Adjust by program base.
R_ARM_GOTOFF	32 bit offset to GOT
R_ARM_GOTPC	32 bit PC relative offset to GOT
R_ARM_GOT32	32 bit GOT entry
R_ARM_PLT32	32 bit PLT address
R_ARM_THM_PC11	thumb unconditional branch
R_ARM_THM_PC9	thumb conditional branch
R_ARM_NUM	Keep this the last entry.



Definition at line 782 of file [elf.h](#).

### 13.16.8.3.24 Elf\_R\_X86\_64\_s

enum [Elf\\_R\\_X86\\_64\\_s](#)

AMD x86-64 relocations.

#### Enumerator

R_X86_64_NONE	No reloc.
R_X86_64_64	Direct 64 bit
R_X86_64_PC32	PC relative 32 bit signed.
R_X86_64_GOT32	32 bit GOT entry
R_X86_64_PLT32	32 bit PLT address
R_X86_64_COPY	Copy symbol at runtime.
R_X86_64_GLOB_DAT	Create GOT entry.
R_X86_64_JUMP_SLOT	Create PLT entry.
R_X86_64_RELATIVE	Adjust by program base.
R_X86_64_GOTPCREL	32 bit signed PC relative offset to GOT
R_X86_64_32	Direct 32 bit zero extended.
R_X86_64_32S	Direct 32 bit sign extended.
R_X86_64_16	Direct 16 bit zero extended.
R_X86_64_PC16	16 bit sign extended pc relative
R_X86_64_8	Direct 8 bit sign extended
R_X86_64_PC8	8 bit sign extended pc relative
R_X86_64_DTPMOD64	ID of module containing symbol.
R_X86_64_DTPOFF64	Offset in module's TLS block.
R_X86_64_TPOFF64	Offset in initial TLS block.
R_X86_64_TLSGD	32 bit signed PC relative offset to two GOT entries for GD symbol
R_X86_64_TLSLD	32 bit signed PC relative offset to two GOT entries for LD symbol
R_X86_64_DTPOFF32	Offset in TLS block.
R_X86_64_GOTTPOFF	32 bit signed PC relative offset to GOT entry for IE symbol
R_X86_64_TPOFF32	Offset in initial TLS block.

Definition at line 837 of file [elf.h](#).

### 13.16.8.3.25 Elf\_SHF\_s\_ARM

enum [Elf\\_SHF\\_s\\_ARM](#)

ARM-specific values for [Elf32\\_Shdr.sh\\_flags](#) / [Elf64\\_Shdr.sh\\_flags](#).

#### Enumerator

SHF_ARM_ENTRYSECT	Section contains an entry point.
SHF_ARM_COMDEF	Section may be multiply defined in the input to a link step.

Definition at line 767 of file [elf.h](#).

### 13.16.8.3.26 Elf\_SHFs

enum [Elf\\_SHFs](#)

Section attribute flags.

#### Enumerator

SHF_WRITE	writeable during execution
SHF_ALLOC	section occupies virt memory
SHF_EXECINSTR	code section
SHF_MERGE	Might be merged.
SHF_STRINGS	Contains nul-terminated strings.
SHF_INFO_LINK	'sh_info' contains SHT index
SHF_LINK_ORDER	Preserve order after combining.
SHF_OS_NONCONFORMING	Non-standard OS-specific handling required.
SHF_GROUP	Section is member of a group.
SHF_TLS	Section hold thread-local data.
SHF_MASKOS	OS-specific.
SHF_MASKPROC	processor-specific mask

Definition at line 410 of file [elf.h](#).

### 13.16.8.3.27 Elf\_SHNs

enum [Elf\\_SHNs](#)

Special section indexes.

#### Enumerator

SHN_UNDEF	undefined section header entry
SHN_LORESERVE	lower bound of reserved indexes
SHN_LOPROC	lower bound of proc spec entr
SHN_HIPROC	upper bound of proc spec entr
SHN_ABS	absolute values for ref
SHN_COMMON	common symbols
SHN_HIRESERVE	upper bound of reserved indexes

Definition at line 339 of file [elf.h](#).

### 13.16.8.3.28 Elf\_SHTs

enum [Elf\\_SHTs](#)

Section type.

Enumerator

SHT_NULL	inactive section header
SHT_PROGBITS	information defined by program
SHT_SYMTAB	symbol table
SHT_STRTAB	string table
SHT_RELA	reloc entries w/ explicit addends
SHT_HASH	symbol hash table
SHT_DYNAMIC	information for dynamic linking
SHT_NOTE	information that marks the file
SHT_NOBITS	occupies no space in the file
SHT_REL	reloc entries w/o explicit addends
SHT_SHLIB	reserved + unspecified semantics
SHT_DYNSYM	symbol table (dynamic)
SHT_INIT_ARRAY	Array of constructors.
SHT_FINI_ARRAY	Array of destructors.
SHT_PREINIT_ARRAY	Array of pre-constructors.
SHT_GROUP	Section group.
SHT_SYMTAB_SHNDX	Extended section indices.
SHT_NUM	Number of defined types.
SHT_LOOS	Start OS-specific.
SHT_HIOS	End OS-specific.
SHT_LOPROC	Start processor-specific.
SHT_HIPROC	End processor-specific.
SHT_LOUSER	Start application-specific.
SHT_HIUSER	End application-specific.

Definition at line 381 of file [elf.h](#).

### 13.16.8.3.29 Elf\_STBs

enum [Elf\\_STBs](#)

Symbol Binding.

See also

[ELF32\\_ST\\_BIND](#), [ELF64\\_ST\\_BIND](#)

Enumerator

STB_LOCAL	not visible outside object file
STB_GLOBAL	visible to all objects being combined
STB_WEAK	resemble global symbols
STB_LOOS	OS-specific.
STB_HIOS	OS-specific.
STB_LOPROC	Processor-specific.
STB_HIPROC	Processor-specific.

Generated by Doxygen 1.8.17

Definition at line 917 of file [elf.h](#).

### 13.16.8.3.30 Elf\_STTs

enum [Elf\\_STTs](#)

Symbol Types.

See also

[ELF32\\_ST\\_TYPE](#), [ELF64\\_ST\\_TYPE](#)

Enumerator

STT_NOTYPE	symbol's type not specified
STT_OBJECT	associated with a data object
STT_FUNC	associated with a function or other code
STT_SECTION	associated with a section
STT_FILE	source file name associated with object
STT_LOOS	OS-specific.
STT_HIOS	OS-specific.
STT_LOPROC	processor-specific
STT_HIPROC	processor-specific

Definition at line 930 of file [elf.h](#).

## 13.16.9 Functions to manipulate the local IDT

Collaboration diagram for Functions to manipulate the local IDT:



### Data Structures

- struct [l4util\\_idt\\_desc\\_t](#)  
*IDT entry.*
- struct [l4util\\_idt\\_header\\_t](#)  
*Header of an IDT table.*

### 13.16.9.1 Detailed Description

## 13.16.10 IA32 Port I/O API

Collaboration diagram for IA32 Port I/O API:



### Functions

- `l4_uint8_t l4util_in8 (l4_uint16_t port)`  
*Read byte from I/O port.*
- `l4_uint16_t l4util_in16 (l4_uint16_t port)`  
*Read 16-bit-value from I/O port.*
- `l4_uint32_t l4util_in32 (l4_uint16_t port)`  
*Read 32-bit-value from I/O port.*
- `void l4util_ins8 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
*Read a block of 8-bit-values from I/O ports.*
- `void l4util_ins16 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
*Read a block of 16-bit-values from I/O ports.*
- `void l4util_ins32 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
*Read a block of 32-bit-values from I/O ports.*
- `void l4util_out8 (l4_uint8_t value, l4_uint16_t port)`  
*Write byte to I/O port.*
- `void l4util_out16 (l4_uint16_t value, l4_uint16_t port)`  
*Write 16-bit-value to I/O port.*
- `void l4util_out32 (l4_uint32_t value, l4_uint16_t port)`  
*Write 32-bit-value to I/O port.*
- `void l4util_outs8 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
*Write a block of bytes to I/O port.*
- `void l4util_outs16 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
*Write a block of 16-bit-values to I/O port.*
- `void l4util_outs32 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
*Write block of 32-bit-values to I/O port.*
- `void l4util_iodelay (void)`  
*delay I/O port access by writing to port 0x80*

### 13.16.10.1 Detailed Description

### 13.16.10.2 Function Documentation

#### 13.16.10.2.1 l4util\_in16()

```
l4_uint16_t l4util_in16 (
    l4_uint16_t port ) [inline]
```

Read 16-bit-value from I/O port.

**Parameters**

<i>port</i>	I/O port address
-------------	------------------

**Returns**

value

Definition at line 180 of file [port\\_io.h](#).

**13.16.10.2.2 l4util\_in32()**

```
l4_uint32_t l4util_in32 (  
    l4_uint16_t port ) [inline]
```

Read 32-bit-value from I/O port.

**Parameters**

<i>port</i>	I/O port address
-------------	------------------

**Returns**

value

Definition at line 188 of file [port\\_io.h](#).

**13.16.10.2.3 l4util\_in8()**

```
l4_uint8_t l4util_in8 (  
    l4_uint16_t port ) [inline]
```

Read byte from I/O port.

**Parameters**

<i>port</i>	I/O port address
-------------	------------------

**Returns**

value

Definition at line 172 of file [port\\_io.h](#).

Referenced by [l4util\\_irq\\_acknowledge\(\)](#).

Here is the caller graph for this function:



#### 13.16.10.2.4 l4util\_ins16()

```
void l4util_ins16 (
    l4_uint16_t port,
    l4_umword_t addr,
    l4_umword_t count ) [inline]
```

Read a block of 16-bit-values from I/O ports.

##### Parameters

<i>port</i>	I/O port address
<i>addr</i>	address of buffer
<i>count</i>	number of I/O operations

Definition at line 205 of file [port\\_io.h](#).

#### 13.16.10.2.5 l4util\_ins32()

```
void l4util_ins32 (
    l4_uint16_t port,
    l4_umword_t addr,
    l4_umword_t count ) [inline]
```

Read a block of 32-bit-values from I/O ports.

##### Parameters

<i>port</i>	I/O port address
<i>addr</i>	address of buffer
<i>count</i>	number of I/O operations

Definition at line 214 of file [port\\_io.h](#).

#### 13.16.10.2.6 l4util\_ins8()

```
void l4util_ins8 (
    l4_uint16_t port,
```

```
14_umword_t addr,  
14_umword_t count ) [inline]
```

Read a block of 8-bit-values from I/O ports.

#### Parameters

<i>port</i>	I/O port address
<i>addr</i>	address of buffer
<i>count</i>	number of I/O operations

Definition at line 196 of file [port\\_io.h](#).

#### 13.16.10.2.7 l4util\_out16()

```
void l4util_out16 (  
    14_uint16_t value,  
    14_uint16_t port ) [inline]
```

Write 16-bit-value to I/O port.

#### Parameters

<i>port</i>	I/O port address
<i>value</i>	value to write

Definition at line 229 of file [port\\_io.h](#).

#### 13.16.10.2.8 l4util\_out32()

```
void l4util_out32 (  
    14_uint32_t value,  
    14_uint16_t port ) [inline]
```

Write 32-bit-value to I/O port.

#### Parameters

<i>port</i>	I/O port address
<i>value</i>	value to write

Definition at line 235 of file [port\\_io.h](#).

#### 13.16.10.2.9 l4util\_out8()

```
void l4util_out8 (  
    14_uint8_t value,  
    14_uint16_t port ) [inline]
```



Write byte to I/O port.

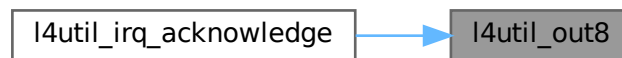
## Parameters

<i>port</i>	I/O port address
<i>value</i>	value to write

Definition at line 223 of file [port\\_io.h](#).

Referenced by [l4util\\_irq\\_acknowledge\(\)](#).

Here is the caller graph for this function:



#### 13.16.10.2.10 l4util\_outs16()

```
void l4util_outs16 (
    l4_uint16_t port,
    l4_umword_t addr,
    l4_umword_t count ) [inline]
```

Write a block of 16-bit-values to I/O port.

## Parameters

<i>port</i>	I/O port address
<i>addr</i>	address of buffer
<i>count</i>	number of I/O operations

Definition at line 250 of file [port\\_io.h](#).

#### 13.16.10.2.11 l4util\_outs32()

```
void l4util_outs32 (
    l4_uint16_t port,
    l4_umword_t addr,
    l4_umword_t count ) [inline]
```

Write block of 32-bit-values to I/O port.

## Parameters

<i>port</i>	I/O port address
<i>addr</i>	address of buffer
<i>count</i>	number of I/O operations

Definition at line 259 of file [port\\_io.h](#).

#### 13.16.10.2.12 l4util\_outs8()

```
void l4util_outs8 (
    l4_uint16_t port,
    l4_umword_t addr,
    l4_umword_t count ) [inline]
```

Write a block of bytes to I/O port.

##### Parameters

<i>port</i>	I/O port address
<i>addr</i>	address of buffer
<i>count</i>	number of I/O operations

Definition at line 241 of file [port\\_io.h](#).

### 13.16.11 Internal functions

Collaboration diagram for Internal functions:



#### Functions

- void **base64\_encode** (const char \*infile, unsigned int in\_size, char \*\*outfile)  
*base-64-encode string infile*
- void **base64\_decode** (const char \*infile, unsigned int in\_size, char \*\*outfile)  
*decode base-64-encoded string infile*

### 13.16.11.1 Detailed Description

## 13.16.12 Kernel Interface Page API

Collaboration diagram for Kernel Interface Page API:



### Files

- file [kip.h](#)

### Macros

- `#define l4util_kip_for_each_feature(s) l4_kip_for_each_feature(s)`  
*Cycle through kernel features given in the KIP.*

### Functions

- `int l4util_kip_kernel_is_ux (l4_kernel_info_t const *k)`  
*Return whether the kernel is running natively or under UX.*
- `int l4util_kip_kernel_has_feature (l4_kernel_info_t const *k, char const *str)`  
*Check if kernel supports a feature.*
- `unsigned long l4util_kip_kernel_abi_version (l4_kernel_info_t const *k)`  
*Return kernel ABI version.*

### 13.16.12.1 Detailed Description

### 13.16.12.2 Macro Definition Documentation

#### 13.16.12.2.1 l4util\_kip\_for\_each\_feature

```
#define l4util_kip_for_each_feature(
    s ) l4_kip_for_each_feature(s)
```

Cycle through kernel features given in the KIP.

Cycles through all KIP kernel feature strings. `s` must be a character pointer (`char const *`) initialized with [l4\\_kip\\_version\\_string\(\)](#).

**Deprecated** Use [l4\\_kip\\_for\\_each\\_feature\(\)](#).

Definition at line 68 of file [kip.h](#).

### 13.16.12.3 Function Documentation

#### 13.16.12.3.1 l4util\_kip\_kernel\_abi\_version()

```
unsigned long l4util_kip_kernel_abi_version (
    l4_kernel_info_t const * k )
```

Return kernel ABI version.

##### Parameters

<i>k</i>	Pointer to the kernel info page (KIP).
----------	--

##### Returns

Kernel ABI version.

#### 13.16.12.3.2 l4util\_kip\_kernel\_has\_feature()

```
int l4util_kip_kernel_has_feature (
    l4_kernel_info_t const * k,
    char const * str )
```

Check if kernel supports a feature.

##### Parameters

<i>k</i>	Pointer to the kernel info page (KIP).
<i>str</i>	Feature name to check.

##### Returns

1 if the kernel supports the feature, 0 if not.

Checks the feature field in the KIP for the given string.

**Deprecated** Use [l4\\_kip\\_kernel\\_has\\_feature\(\)](#).

#### 13.16.12.3.3 l4util\_kip\_kernel\_is\_ux()

```
int l4util_kip_kernel_is_ux (
    l4_kernel_info_t const * k )
```

Return whether the kernel is running natively or under UX.

#### Parameters

<i>k</i>	Pointer to the kernel info page (KIP).
----------	--

#### Returns

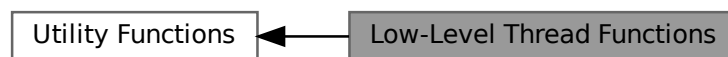
1 when running under UX, 0 if not running under UX.

#### Examples

[examples/sys/ux-vhw/main.c](#).

### 13.16.13 Low-Level Thread Functions

Collaboration diagram for Low-Level Thread Functions:



### 13.16.14 Random number support

Collaboration diagram for Random number support:



#### Functions

- [l4\\_uint32\\_t l4util\\_rand](#) (void)  
*Deliver next random number.*
- void [l4util\\_srand](#) ([l4\\_uint32\\_t](#) seed)  
*Initialize random number generator.*

### 13.16.14.1 Detailed Description

### 13.16.14.2 Function Documentation

#### 13.16.14.2.1 l4util\_rand()

```
l4_uint32_t l4util_rand (
    void )
```

Deliver next random number.

#### Returns

A new random number

#### 13.16.14.2.2 l4util\_srand()

```
void l4util_srand (
    l4_uint32_t seed )
```

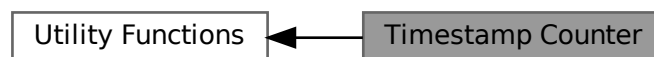
Initialize random number generator.

#### Parameters

<i>seed</i>	Value to initialize
-------------	---------------------

## 13.16.15 Timestamp Counter

Collaboration diagram for Timestamp Counter:



#### Files

- file [rdtsc.h](#)  
*Timestamp counter related functions.*
- file [rdtsc.h](#)  
*Timestamp counter related functions.*

## Functions

- [l4\\_cpu\\_time\\_t l4\\_rdtsc](#) (void)  
*Read current value of CPU-internal timestamp counter.*
- [l4\\_uint32\\_t l4\\_rdtsc\\_32](#) (void)  
*Read the lest significant 32 bit of the TSC.*
- [l4\\_uint64\\_t l4\\_rdpmc](#) (int ecx)  
*Return current value of CPU-internal performance measurement counter.*
- [l4\\_uint32\\_t l4\\_rdpmc\\_32](#) (int ecx)  
*Return the least significant 32 bit of a performance counter.*
- [l4\\_uint64\\_t l4\\_tsc\\_to\\_ns](#) ([l4\\_cpu\\_time\\_t](#) tsc)  
*Convert timestamp to ns value.*
- [l4\\_uint64\\_t l4\\_tsc\\_to\\_us](#) ([l4\\_cpu\\_time\\_t](#) tsc)  
*Convert timestamp into micro seconds value.*
- void [l4\\_tsc\\_to\\_s\\_and\\_ns](#) ([l4\\_cpu\\_time\\_t](#) tsc, [l4\\_uint32\\_t](#) \*s, [l4\\_uint32\\_t](#) \*ns)  
*Convert timestamp to s.ns value.*
- [l4\\_cpu\\_time\\_t l4\\_ns\\_to\\_tsc](#) ([l4\\_uint64\\_t](#) ns)  
*Convert nano seconds into CPU ticks.*
- void [l4\\_busy\\_wait\\_ns](#) ([l4\\_uint64\\_t](#) ns)  
*Wait busy for a small amount of time.*
- void [l4\\_busy\\_wait\\_us](#) ([l4\\_uint64\\_t](#) us)  
*Wait busy for a small amount of time.*
- [l4\\_uint32\\_t l4\\_calibrate\\_tsc](#) ([l4\\_kernel\\_info\\_t](#) const \*kip)  
*Determine scalers for timestamp calculations.*
- [l4\\_uint32\\_t l4\\_tsc\\_init](#) ([l4\\_kernel\\_info\\_t](#) const \*kip)  
*Initialize scaler for TSC calibrations from the kernel.*
- [l4\\_uint32\\_t l4\\_get\\_hz](#) (void)  
*Get CPU frequency in Hz.*

### 13.16.15.1 Detailed Description

### 13.16.15.2 Function Documentation

#### 13.16.15.2.1 l4\_busy\_wait\_ns()

```
void l4_busy_wait_ns (
    l4\_uint64\_t ns ) [inline]
```

Wait busy for a small amount of time.

#### Parameters

<a href="#">ns</a>	nano seconds to wait
--------------------	----------------------

#### Attention

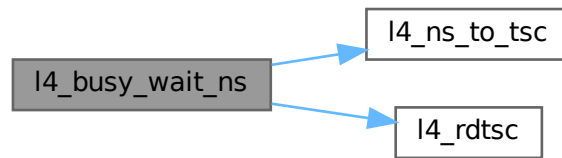
Not intended for any use!

Definition at line 264 of file [rdtsc.h](#).



References [l4\\_ns\\_to\\_tsc\(\)](#), and [l4\\_rdtsc\(\)](#).

Here is the call graph for this function:



#### 13.16.15.2.2 l4\_busy\_wait\_us()

```
void l4_busy_wait_us (
    l4_uint64_t us ) [inline]
```

Wait busy for a small amount of time.

##### Parameters

<i>us</i>	micro seconds to wait
-----------	-----------------------

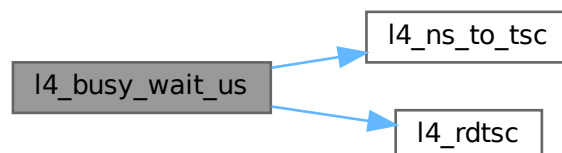
##### Attention

Not intended for any use!

Definition at line [274](#) of file [rdtsc.h](#).

References [l4\\_ns\\_to\\_tsc\(\)](#), and [l4\\_rdtsc\(\)](#).

Here is the call graph for this function:



### 13.16.15.2.3 l4\_calibrate\_tsc()

```
l4_uint32_t l4_calibrate_tsc (
    l4_kernel_info_t const * kip ) [inline]
```

Determine scalers for timestamp calculations.

Determine some scalers to be able to convert between real time and CPU ticks. Just calls [l4\\_tsc\\_init\(\)](#).

#### Examples

[examples/sys/aliens/main.c](#).

Definition at line 161 of file [rdtsc.h](#).

References [l4\\_tsc\\_init\(\)](#).

Here is the call graph for this function:



### 13.16.15.2.4 l4\_get\_hz()

```
l4_uint32_t l4_get_hz (
    void )
```

Get CPU frequency in Hz.

#### Returns

frequency in Hz

### 13.16.15.2.5 l4\_ns\_to\_tsc()

```
l4_cpu_time_t l4_ns_to_tsc (
    l4_uint64_t ns ) [inline]
```

Convert nano seconds into CPU ticks.

#### Parameters

<i>ns</i>	nano seconds
-----------	--------------

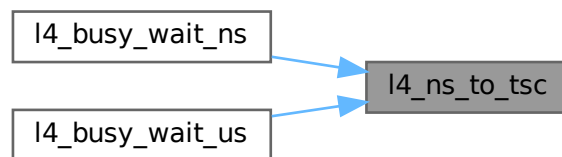
**Returns**

CPU ticks

Definition at line 250 of file [rdtsc.h](#).

Referenced by [l4\\_busy\\_wait\\_ns\(\)](#), and [l4\\_busy\\_wait\\_us\(\)](#).

Here is the caller graph for this function:

**13.16.15.2.6 l4\_rdpmc()**

```
l4_uint64_t l4_rdpmc (
    int ecx ) [inline]
```

Return current value of CPU-internal performance measurement counter.

**Parameters**

<code>ecx</code>	ECX value for the rdpmc instruction. For details see the Intel IA-32 Architectures Software Developer's Manual.
------------------	---

**Returns**

64-bit PMC

Definition at line 177 of file [rdtsc.h](#).

**13.16.15.2.7 l4\_rdpmc\_32()**

```
l4_uint32_t l4_rdpmc_32 (
    int ecx ) [inline]
```

Return the least significant 32 bit of a performance counter.

Useful for smaller differences, needs less cycles.

Definition at line 197 of file [rdtsc.h](#).

### 13.16.15.2.8 l4\_rdtsc()

```
l4_cpu_time_t l4_rdtsc (
    void ) [inline]
```

Read current value of CPU-internal timestamp counter.

#### Returns

64-bit timestamp

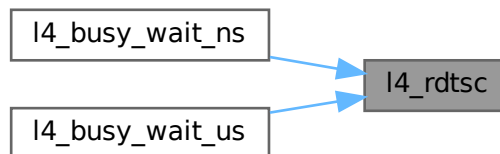
#### Examples

[examples/sys/aliens/main.c](#).

Definition at line 167 of file [rdtsc.h](#).

Referenced by [l4\\_busy\\_wait\\_ns\(\)](#), and [l4\\_busy\\_wait\\_us\(\)](#).

Here is the caller graph for this function:



### 13.16.15.2.9 l4\_rdtsc\_32()

```
l4_uint32_t l4_rdtsc_32 (
    void ) [inline]
```

Read the lest significant 32 bit of the TSC.

Useful for smaller differences, needs less cycles.

Definition at line 187 of file [rdtsc.h](#).

### 13.16.15.2.10 l4\_tsc\_init()

```
l4_uint32_t l4_tsc_init (
    l4_kernel_info_t const * kip )
```

Initialize scaler for TSC calibrations from the kernel.

Initialize the scalers needed by [l4\\_tsc\\_to\\_ns\(\)](#)/[l4\\_ns\\_to\\_tsc\(\)](#) and so on. Use the kernel-provided frequency.

## Parameters

<i>kip</i>	KIP pointer
------------	-------------

## Returns

0 on error (no scalars exported by kernel) otherwise returns ( $2^{32} / (\text{tsc per } \mu\text{sec})$ ). This value has the same semantics as the value returned by the `calibrate_delay_loop()` function of the Linux kernel.

Referenced by [l4\\_calibrate\\_tsc\(\)](#).

Here is the caller graph for this function:

**13.16.15.2.11 l4\_tsc\_to\_ns()**

```
l4_uint64_t l4_tsc_to_ns (
    l4_cpu_time_t tsc ) [inline]
```

Convert timestamp to ns value.

## Parameters

<i>tsc</i>	time value in CPU ticks
------------	-------------------------

## Returns

time value in ns

## Examples

[examples/sys/aliens/main.c](#).

Definition at line 207 of file [rdtsc.h](#).

**13.16.15.2.12 l4\_tsc\_to\_s\_and\_ns()**

```
void l4_tsc_to_s_and_ns (
    l4_cpu_time_t tsc,
    l4_uint32_t * s,
    l4_uint32_t * ns ) [inline]
```

Convert timestamp to s.ns value.

**Parameters**

	<i>tsc</i>	time value in CPU ticks
out	<i>s</i>	seconds
out	<i>ns</i>	nano seconds

Definition at line 235 of file [rdtsc.h](#).

**13.16.15.2.13 l4\_tsc\_to\_us()**

```
l4_uint64_t l4_tsc_to_us (
    l4_cpu_time_t tsc ) [inline]
```

Convert timestamp into micro seconds value.

**Parameters**

<i>tsc</i>	time value in CPU ticks
------------	-------------------------

**Returns**

time value in micro seconds

Definition at line 221 of file [rdtsc.h](#).

**13.17 vCPU Support Library**

vCPU handling functionality.

Collaboration diagram for vCPU Support Library:

**Modules**

- [Extended vCPU support](#)

*Extended vCPU handling functionality.*

## Data Structures

- class [L4vcpu::State](#)  
*C++ implementation of state word in the vCPU area.*
- class [L4vcpu::Vcpu](#)  
*C++ implementation of the vCPU save state area.*

## Functions

- void [l4vcpu\\_irq\\_disable](#) ([l4\\_vcpu\\_state\\_t](#) \*vcpu) [L4\\_NOTHROW](#)  
*Disable a vCPU for event delivery.*
- unsigned [l4vcpu\\_irq\\_disable\\_save](#) ([l4\\_vcpu\\_state\\_t](#) \*vcpu) [L4\\_NOTHROW](#)  
*Disable a vCPU for event delivery and return previous state.*
- void [l4vcpu\\_irq\\_enable](#) ([l4\\_vcpu\\_state\\_t](#) \*vcpu, [l4\\_utcb\\_t](#) \*utcb, [l4vcpu\\_event\\_hndl\\_t](#) do\_event\_work\_cb, [l4vcpu\\_setup\\_ipc\\_t](#) setup\_ipc) [L4\\_NOTHROW](#)  
*Enable a vCPU for event delivery.*
- void [l4vcpu\\_irq\\_restore](#) ([l4\\_vcpu\\_state\\_t](#) \*vcpu, unsigned s, [l4\\_utcb\\_t](#) \*utcb, [l4vcpu\\_event\\_hndl\\_t](#) do\_event\_work\_cb, [l4vcpu\\_setup\\_ipc\\_t](#) setup\_ipc) [L4\\_NOTHROW](#)  
*Restore a previously saved IRQ/event state.*
- void [l4vcpu\\_wait\\_for\\_event](#) ([l4\\_vcpu\\_state\\_t](#) \*vcpu, [l4\\_utcb\\_t](#) \*utcb, [l4vcpu\\_event\\_hndl\\_t](#) do\_event\_work\_cb, [l4vcpu\\_setup\\_ipc\\_t](#) setup\_ipc) [L4\\_NOTHROW](#)  
*Wait for event.*
- void [l4vcpu\\_print\\_state](#) (const [l4\\_vcpu\\_state\\_t](#) \*vcpu, const char \*prefix) [L4\\_NOTHROW](#)  
*Print the state of a vCPU.*
- int [l4vcpu\\_is\\_irq\\_entry](#) ([l4\\_vcpu\\_state\\_t](#) const \*vcpu) [L4\\_NOTHROW](#)  
*Return whether the entry reason was an IRQ/IPC message.*
- int [l4vcpu\\_is\\_page\\_fault\\_entry](#) ([l4\\_vcpu\\_state\\_t](#) const \*vcpu) [L4\\_NOTHROW](#)  
*Return whether the entry reason was a page fault.*

### 13.17.1 Detailed Description

vCPU handling functionality.

This library provides convenience functionality on top of the l4sys vCPU interface to ease programming. It wraps commonly used code and abstracts architecture depends parts as far as reasonable.

### 13.17.2 Function Documentation

#### 13.17.2.1 l4vcpu\_irq\_disable()

```
void l4vcpu_irq_disable (
    l4\_vcpu\_state\_t * vcpu ) [inline]
```

Disable a vCPU for event delivery.

#### Parameters

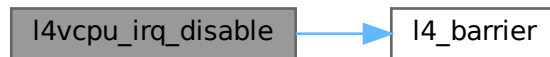
<i>vcpu</i>	Pointer to vCPU area.
-------------	-----------------------

Definition at line 212 of file [vcpu.h](#).

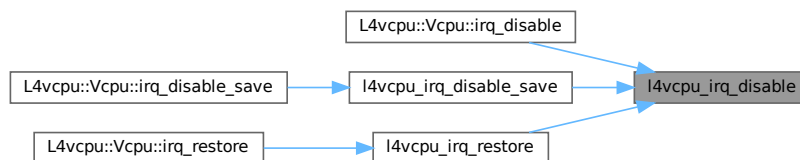
References [l4\\_barrier\(\)](#).

Referenced by [L4vcpu::Vcpu::irq\\_disable\(\)](#), [l4vcpu\\_irq\\_disable\\_save\(\)](#), and [l4vcpu\\_irq\\_restore\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 13.17.2.2 l4vcpu\_irq\_disable\_save()

```
unsigned l4vcpu_irq_disable_save (
    l4_vcpu_state_t * vcpu ) [inline]
```

Disable a vCPU for event delivery and return previous state.

#### Parameters

<i>vcpu</i>	Pointer to vCPU area.
-------------	-----------------------

#### Returns

IRQ state before disabling IRQs.

Definition at line 220 of file [vcpu.h](#).

References [l4vcpu\\_irq\\_disable\(\)](#).

Referenced by [L4vcpu::Vcpu::irq\\_disable\\_save\(\)](#).



Here is the call graph for this function:



Here is the caller graph for this function:



### 13.17.2.3 l4vcpu\_irq\_enable()

```

void l4vcpu_irq_enable (
    l4_vcpu_state_t * vcpu,
    l4_utcb_t * utcb,
    l4vcpu_event_hndl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc ) [inline]
  
```

Enable a vCPU for event delivery.

#### Parameters

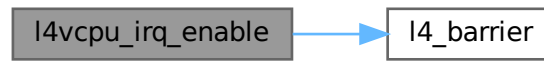
<i>vcpu</i>	Pointer to vCPU area.
<i>utcb</i>	Utc b pointer of the calling vCPU.
<i>do_event_work_cb</i>	Call-back function that is called in case an event (such as an interrupt) is pending.
<i>setup_ipc</i>	Function call-back that is called right before any IPC operation, and before event delivery is enabled.

Definition at line 243 of file [vcpu.h](#).

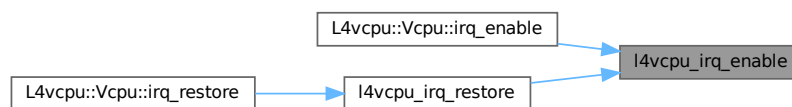
References [l4\\_barrier\(\)](#), [L4\\_IPC\\_BOTH\\_TIMEOUT\\_0](#), [L4\\_LIKELY](#), [L4\\_VCPU\\_F\\_IRQ](#), and [L4\\_VCPU\\_SF\\_IRQ\\_PENDING](#).

Referenced by [L4vcpu::Vcpu::irq\\_enable\(\)](#), and [l4vcpu\\_irq\\_restore\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 13.17.2.4 l4vcpu\_irq\_restore()

```

void l4vcpu_irq_restore (
    l4_vcpu_state_t * vcpu,
    unsigned s,
    l4_utcb_t * utcb,
    l4vcpu_event_hndl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc ) [inline]
  
```

Restore a previously saved IRQ/event state.

##### Parameters

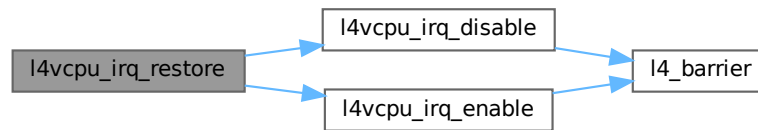
<i>vcpu</i>	Pointer to vCPU area.
<i>s</i>	IRQ state to be restored.
<i>utcb</i>	Utc pointer of the calling vCPU.
<i>do_event_work_cb</i>	Call-back function that is called in case an event (such as an interrupt) is pending after enabling.
<i>setup_ipc</i>	Function call-back that is called right before any IPC operation, and before event delivery is enabled.

Definition at line 268 of file [vcpu.h](#).

References [L4\\_VCPU\\_F\\_IRQ](#), [l4vcpu\\_irq\\_disable\(\)](#), and [l4vcpu\\_irq\\_enable\(\)](#).

Referenced by [L4vcpu::Vcpu::irq\\_restore\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 13.17.2.5 l4vcpu\_is\_irq\_entry()

```
int l4vcpu_is_irq_entry (
    l4_vcpu_state_t const * vcpu ) [inline]
```

Return whether the entry reason was an IRQ/IPC message.

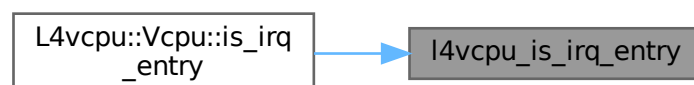
#### Parameters

<i>vcpu</i>	Pointer to vCPU area.
-------------	-----------------------

return 0 if not, !=0 otherwise.

Referenced by [L4vcpu::Vcpu::is\\_irq\\_entry\(\)](#).

Here is the caller graph for this function:



### 13.17.2.6 l4vcpu\_is\_page\_fault\_entry()

```
int l4vcpu_is_page_fault_entry (
    l4_vcpu_state_t const * vcpu ) [inline]
```

Return whether the entry reason was a page fault.

#### Parameters

<i>vcpu</i>	Pointer to vCPU area.
-------------	-----------------------

return 0 if not, !=0 otherwise.

Referenced by [L4vcpu::Vcpu::is\\_page\\_fault\\_entry\(\)](#).

Here is the caller graph for this function:



### 13.17.2.7 l4vcpu\_print\_state()

```
void l4vcpu_print_state (
    const l4_vcpu_state_t * vcpu,
    const char * prefix )
```

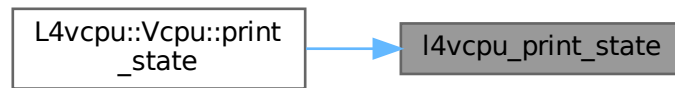
Print the state of a vCPU.

#### Parameters

<i>vcpu</i>	Pointer to vCPU area.
<i>prefix</i>	A prefix for each line printed.

Referenced by [L4vcpu::Vcpu::print\\_state\(\)](#).

Here is the caller graph for this function:



### 13.17.2.8 l4vcpu\_wait\_for\_event()

```

void l4vcpu_wait_for_event (
    l4_vcpu_state_t * vcpu,
    l4_utcb_t * utcb,
    l4vcpu_event_hndl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc ) [inline]
  
```

Wait for event.

#### Parameters

<i>vcpu</i>	Pointer to vCPU area.
<i>utcb</i>	Utc b pointer of the calling vCPU.
<i>do_event_work_cb</i>	Call-back function that is called when the vCPU awakes and needs to handle an event/IRQ.
<i>setup_ipc</i>	Function call-back that is called right before any IPC operation.

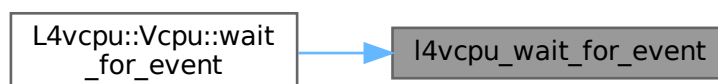
Note that event delivery remains disabled after this function returns.

Definition at line 281 of file [vcpu.h](#).

References [L4\\_IPC\\_NEVER](#).

Referenced by [L4vcpu::Vcpu::wait\\_for\\_event\(\)](#).

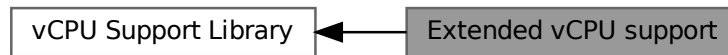
Here is the caller graph for this function:



### 13.17.3 Extended vCPU support

Extended vCPU handling functionality.

Collaboration diagram for Extended vCPU support:



#### Functions

- `int l4vcpu_ext_alloc (l4_vcpu_state_t **vcpu, l4_addr_t *ext_state, l4_cap_idx_t task, l4_cap_idx_t regmgr)`  
`L4_NOTHROW`

*Allocate state area for an extended vCPU.*

#### 13.17.3.1 Detailed Description

Extended vCPU handling functionality.

#### 13.17.3.2 Function Documentation

##### 13.17.3.2.1 l4vcpu\_ext\_alloc()

```

int l4vcpu_ext_alloc (
    l4_vcpu_state_t ** vcpu,
    l4_addr_t * ext_state,
    l4_cap_idx_t task,
    l4_cap_idx_t regmgr )
  
```

Allocate state area for an extended vCPU.

#### Parameters

out	<i>vcpu</i>	Allocated vcpu-state area.
out	<i>ext_state</i>	Allocated extended vcpu-state area.
	<i>task</i>	Task to use for allocation.
	<i>regmgr</i>	Region manager to use for allocation.

#### Returns

0 for success, error code otherwise

# Chapter 14

## Namespace Documentation

### 14.1 cxx Namespace Reference

Our C++ library.

#### Namespaces

- namespace [Bits](#)  
*Internal helpers for the cxx package.*

#### Data Structures

- class [Avl\\_map](#)  
*AVL tree based associative container.*
- class [Avl\\_set](#)  
*AVL set for simple comparable items.*
- class [Avl\\_tree](#)  
*A generic AVL tree.*
- class [Avl\\_tree\\_node](#)  
*Node of an AVL tree.*
- class [Base\\_slab](#)  
*Basic slab allocator.*
- class [Base\\_slab\\_static](#)  
*Merged slab allocator (allocators for objects of the same size are merged together).*
- class [Bitfield](#)  
*Definition for a member (part) of a bit field.*
- class [Bitmap](#)  
*A static bitmap.*
- class [Bitmap\\_base](#)  
*Basic bitmap abstraction.*
- class [H\\_list](#)  
*General double-linked list of unspecified [cxx::H\\_list\\_item](#) elements.*
- class [H\\_list\\_item\\_t](#)  
*Basic element type for a double-linked [H\\_list](#).*

- struct [H\\_list\\_t](#)  
*Double-linked list of typed [H\\_list\\_item\\_t](#) elements.*
- class [List](#)  
*Doubly linked list, with internal allocation.*
- class [List\\_alloc](#)  
*Standard list-based allocator.*
- class [List\\_item](#)  
*Basic list item.*
- struct [Lt\\_functor](#)  
*Generic comparator class that defaults to the less-than operator.*
- class [New\\_allocator](#)  
*Standard allocator based on `operator new ()`.*
- class [Nothrow](#)  
*Helper type to distinguish the `operator new` version that does not throw exceptions.*
- struct [Pair](#)  
*Pair of two values.*
- class [Pair\\_first\\_compare](#)  
*Comparison functor for [Pair](#).*
- struct [Ref\\_obj\\_list\\_item](#)  
*Item for list linked via [cxx::Ref\\_ptr](#) with default reference counting.*
- class [Ref\\_ptr](#)  
*A reference-counting pointer with automatic cleanup.*
- class [S\\_list](#)  
*Simple single-linked list.*
- class [Slab](#)  
*[Slab](#) allocator for object of type `Type`.*
- class [Slab\\_static](#)  
*Merged slab allocator (allocators for objects of the same size are merged together).*
- class [static\\_vector](#)  
*Simple encapsulation for a dynamically allocated array.*
- class [String](#)  
*Allocation free string class with explicit length field.*
- class [Weak\\_ref](#)  
*Typed weak reference to an object of type `T`.*
- class [Weak\\_ref\\_base](#)  
*Generic (base) weak reference to some object.*

## Typedefs

- typedef [H\\_list\\_item\\_t](#)< void > [H\\_list\\_item](#)  
*Untyped list item.*
- template<typename T >  
using [Ref\\_ptr\\_list\\_item](#) = [Bits::Smart\\_ptr\\_list\\_item](#)< T, [cxx::Ref\\_ptr](#)< T > >  
*Item for list linked with [cxx::Ref\\_ptr](#).*
- template<typename T >  
using [Ref\\_ptr\\_list](#) = [Bits::Smart\\_ptr\\_list](#)< [Ref\\_ptr\\_list\\_item](#)< T > >  
*Single-linked list where elements are connected via a [cxx::Ref\\_ptr](#).*
- template<typename T >  
using [Unique\\_ptr\\_list\\_item](#) = [Bits::Smart\\_ptr\\_list\\_item](#)< T, [cxx::unique\\_ptr](#)< T > >  
*Item for list linked with [cxx::unique\\_ptr](#).*
- template<typename T >  
using [Unique\\_ptr\\_list](#) = [Bits::Smart\\_ptr\\_list](#)< [Unique\\_ptr\\_list\\_item](#)< T > >  
*Single-linked list where elements are connected with a [cxx::unique\\_ptr](#).*



## Functions

- `template<typename A , typename ... ARGS>`  
`constexpr A const & min (A const &a1, A const &a2, ARGS const &...a)`  
*Get the minimum of a1 and a2 upt to aN.*
- `template<typename A , typename ... ARGS>`  
`constexpr A const & min (cxx::identity_t< A > const &a1, cxx::identity_t< A > const &a2, ARGS const &...a)`  
*Get the minimum of a1 and a2 upt to aN.*
- `template<typename A , typename ... ARGS>`  
`constexpr A const & max (A const &a1, A const &a2, ARGS const &...a)`  
*Get the maximum of a1 and a2 upt to aN.*
- `template<typename A , typename ... ARGS>`  
`constexpr A const & max (cxx::identity_t< A > const &a1, cxx::identity_t< A > const &a2, ARGS const &...a)`  
*Get the maximum of a1 and a2 upt to aN.*
- `template<typename T1 >`  
`T1 clamp (T1 v, T1 lo, T1 hi)`  
*Limit v to the range given by lo and hi.*
- `template<typename T >`  
`T access\_once (T const *a)`  
*Read the value at an address at most once.*
- `template<typename T , typename VAL >`  
`void write\_now (T *a, VAL &&val)`  
*Write a value at an address exactly once.*

### 14.1.1 Detailed Description

Our C++ library.

Small Low-Level C++ Library.

Strings.

Various kinds of C++ utilities.

### 14.1.2 Function Documentation

#### 14.1.2.1 `access_once()`

```
template<typename T >
T cxx::access_once (
    T const * a ) [inline]
```

Read the value at an address at most once.

The read might be omitted if the result is not used by any code unless `typename` contains `volatile`. If the read operation has side effects and must not be omitted, use different means like [L4drivers::Mmio\\_register\\_block](#) or similar.

The compiler is disallowed to reuse a previous read at the same address, for example:

```
val1 = *a;
val2 = access\_once(a); // compiler may not replace this by val2 = val1;
```

The compiler is also disallowed to repeat the read, for example:

```
val1 = access\_once(a);
val2 = val1; // compiler may not replace this by val2 = *a;
```

The above implies that the compiler is also disallowed to move the read out of or into loops.



## 14.2 cxx::Bits Namespace Reference

Internal helpers for the cxx package.

### Data Structures

- struct [Avl\\_map\\_get\\_key](#)  
*Key-getter for [Avl\\_map](#).*
- struct [Avl\\_set\\_get\\_key](#)  
*Internal, key-getter for [Avl\\_set](#) nodes.*
- class [Base\\_avl\\_set](#)  
*Internal: AVL set with internally managed nodes.*
- class [Basic\\_list](#)  
*Internal: Common functions for all head-based list implementations.*
- class [Bst](#)  
*Basic binary search tree (BST).*
- class [Bst\\_node](#)  
*Basic type of a node in a binary search tree (BST).*
- struct [Direction](#)  
*The direction to go in a binary search tree.*
- class [Smart\\_ptr\\_list](#)  
*List of smart-pointer-managed objects.*
- class [Smart\\_ptr\\_list\\_item](#)  
*List item for an arbitrary item in a [Smart\\_ptr\\_list](#).*

### 14.2.1 Detailed Description

Internal helpers for the cxx package.

## 14.3 L4 Namespace Reference

[L4](#) low-level kernel interface.

### Namespaces

- namespace [lpc](#)  
*IPC related functionality.*
- namespace [lpc\\_svr](#)  
*Helper classes for [L4::Server](#) instantiation.*
- namespace [Typeid](#)  
*Definition of interface data-type helpers.*
- namespace [Types](#)  
*[L4](#) basic type helpers for C++.*

## Data Structures

- class [Alloc\\_list](#)  
*A simple list-based allocator.*
- class [Arm\\_smccc](#)  
*Wrapper for function calls that follow the ARM SMC/HVC calling convention.*
- class [Base\\_exception](#)  
*Base class for all exceptions, thrown by the [L4Re](#) framework.*
- class [Basic\\_registry](#)  
*This registry returns the corresponding server object based on the label of an [lpc\\_gate](#).*
- class [Bounds\\_error](#)  
*Access out of bounds.*
- class [Cap](#)  
*C++ interface for capabilities.*
- class [Cap\\_base](#)  
*Base class for all kinds of capabilities.*
- class [Com\\_error](#)  
*Error conditions during IPC.*
- class [Debugger](#)  
*C++ kernel debugger API.*
- class [Element\\_already\\_exists](#)  
*Exception for duplicate element insertions.*
- class [Element\\_not\\_found](#)  
*Exception for a failed lookup (element not found).*
- struct [Epiface](#)  
*Base class for interface implementations.*
- struct [Epiface\\_t](#)  
*Epiface implementation for Kobject-based interface implementations.*
- struct [Epiface\\_t0](#)  
*Epiface mixin for generic Kobject-based interfaces.*
- class [Exception](#)  
*Exception interface.*
- class [Exception\\_tracer](#)  
*Back-trace support for exceptions.*
- class [Factory](#)  
*C++ Factory interface, see [Factory](#) for the C interface.*
- class [Icu](#)  
*C++ Icu interface, see [Interrupt controller](#) for the C interface.*
- class [Invalid\\_capability](#)  
*Indicates that an invalid object was invoked.*
- class [Io\\_pager](#)  
*Io\_pager interface.*
- class [Iommu](#)  
*Interface for IO-MMUs used for DMA remapping.*
- class [IOModifier](#)  
*Modifier class for the IO stream.*
- class [lpc\\_gate](#)  
*The C++ IPC gate interface, see [IPC-Gate API](#) for the C interface.*
- class [Irq](#)  
*C++ Irq interface, see [IRQs](#) for the C interface.*
- class [Irq\\_eoi](#)

- Interface for sending an unmask message to an object.*

  - struct [Irq\\_handler\\_object](#)

*Server object base class for handling IRQ messages.*
  - struct [Irq\\_mux](#)

*IRQ multiplexer for shared IRQs.*
  - struct [Irqep\\_t](#)

*Epiface implementation for interrupt handlers.*
  - class [Kobject](#)

*Base class for all kinds of kernel objects and remote objects, referenced by capabilities.*
  - class [Kobject\\_2t](#)

*Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject\\_t](#)).*
  - struct [Kobject\\_3t](#)

*Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject\\_t](#)).*
  - struct [Kobject\\_demand](#)

*Get the combined server-side resource requirements for all type T...*
  - class [Kobject\\_t](#)

*Helper class to create an [L4Re](#) interface class that is derived from a single base class.*
  - struct [Kobject\\_typeid](#)

*Meta object for handling access to type information of Kobjects.*
  - struct [Kobject\\_typeid< void >](#)

*Minimalistic ID for `void` interface.*
  - struct [Kobject\\_x](#)

*Generic [Kobject](#) inheritance template.*
  - class [Meta](#)

*Meta interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.*
  - class [Out\\_of\\_memory](#)

*Exception signalling insufficient memory.*
  - class [Pager](#)

*Pager interface including the [lo\\_pager](#) interface.*
  - class [Platform\\_control](#)

*[L4](#) C++ interface for controlling platform-wide properties, see [Platform Control C API](#) for the C interface.*
  - class [Poll\\_timeout\\_counter](#)

*Evaluate an expression for a maximum number of times.*
  - class [Poll\\_timeout\\_kipclock](#)

*A polling timeout based on the [L4Re](#) clock.*
  - struct [Proto\\_t](#)

*Data type for defining protocol numbers.*
  - class [Rcv\\_endpoint](#)

*Interface for kernel objects that allow to receive IPC from them.*
  - class [Registry\\_iface](#)

*Abstract interface for object registries.*
  - class [Runtime\\_error](#)

*Exception for an abstract runtime error.*
  - class [Scheduler](#)

*C++ interface of the [Scheduler](#) kernel object, see [Scheduler](#) for the C interface.*
  - struct [Semaphore](#)

*C++ Kernel-provided semaphore interface, see [Kernel-provided semaphore](#) for the C interface.*
  - class [Server](#)

*Basic server loop for handling client requests.*
  - class [Server\\_object](#)

- Abstract server object to be used with [L4::Server](#) and [L4::Basic\\_registry](#).*

  - struct [Server\\_object\\_t](#)

*Base class (template) for server implementing server objects.*
  - struct [Server\\_object\\_x](#)

*Helper class to implement p\_dispatch based server objects.*
  - class [Smart\\_cap](#)

*Smart capability class.*
  - class [String](#)

*A null-terminated string container class.*
  - class [Task](#)

*C++ interface of the [Task](#) kernel object, see [Task](#) for the C interface.*
  - class [Thread](#)

*C++ [L4](#) kernel thread interface, see [Thread](#) for the C interface.*
  - struct [Triggerable](#)

*Interface that allows an object to be triggered by some source.*
  - struct [Type\\_info](#)

*Dynamic Type Information for [L4Re](#) Interfaces.*
  - class [Uart](#)

*[Uart](#) driver abstraction.*
  - class [Uart\\_apb](#)

*Driver for the Advanced Peripheral Bus (APB) UART from the Cortex-M System Design Kit (apb).*
  - class [Unknown\\_error](#)

*[Exception](#) for an unknown condition.*
  - class [Vcon](#)

*C++ [L4 Vcon](#) interface, see [Virtual Console](#) for the C interface.*
  - class [Vm](#)

*Virtual machine host address space.*

## Typedefs

- typedef int **Opcode**

*Data type for RPC opcodes.*

## Enumerations

- enum { [PROTO\\_ANY](#) = 0 , [PROTO\\_EMPTY](#) = -19 }

## Functions

- void [throw\\_ipc\\_exception](#) ([L4::Cap](#)< void > const &o, [l4\\_msgtag\\_t](#) const &err, [l4\\_utcb\\_t](#) \*utcb)

*Throw an [L4](#) IPC error as exception.*
- void [throw\\_ipc\\_exception](#) (void const \*o, [l4\\_msgtag\\_t](#) const &err, [l4\\_utcb\\_t](#) \*utcb)

*Throw an [L4](#) IPC error as exception.*
- template<typename T >
[Type\\_info](#) const \* [kobject\\_typeid](#) () noexcept

*Get the [L4::Type\\_info](#) for the [L4Re](#) interface given in T.*
- template<typename T , typename F >
[Cap](#)< T > [cap\\_dynamic\\_cast](#) ([Cap](#)< F > const &c) noexcept

*dynamic\_cast for capabilities.*

- `template<typename T, typename F >`  
`Cap< T > cap_cast (Cap< F > const &c) noexcept`  
*static\_cast for capabilities.*
- `template<typename T, typename F >`  
`Cap< T > cap_reinterpret_cast (Cap< F > const &c) noexcept`  
*reinterpret\_cast for capabilities.*
- `template<typename T >`  
`constexpr T trunc_order (T val, unsigned char order)`  
*Round a value down so the given number of lsb is zero.*
- `template<typename T >`  
`constexpr T round_order (T val, unsigned char order)`  
*Round a value up so the given number of lsb is zero.*
- `template<typename T, typename F, typename SMART >`  
`Smart_cap< T, SMART > cap_cast (Smart_cap< F, SMART > const &c) noexcept`  
*static\_cast for (smart) capabilities.*
- `template<typename T, typename F, typename SMART >`  
`Smart_cap< T, SMART > cap_reinterpret_cast (Smart_cap< F, SMART > const &c) noexcept`  
*reinterpret\_cast for (smart) capabilities.*

## Variables

- `IOModifier` const **hex**  
*Modifies the stream to print numbers as hexadecimal values.*
- `IOModifier` const **dec**  
*Modifies the stream to print numbers as decimal values.*
- `BasicOStream` **cout**  
*Standard output stream.*
- `BasicOStream` **cerr**  
*Standard error stream.*

## 14.3.1 Detailed Description

[L4](#) low-level kernel interface.

## 14.3.2 Enumeration Type Documentation

### 14.3.2.1 anonymous enum

`anonymous enum`

Enumerator

<code>PROTO_ANY</code>	Default protocol used by <a href="#">Kobject_t</a> and <a href="#">Kobject_x</a> .
<code>PROTO_EMPTY</code>	Empty protocol for empty APIs.

Definition at line 55 of file [\\_\\_typeinfo.h](#).

### 14.3.3 Function Documentation

#### 14.3.3.1 `cap_cast()` [1/2]

```
template<typename T , typename F >
Cap< T > L4::cap_cast (
    Cap< F > const & c ) [inline], [noexcept]
```

`static_cast` for capabilities.

##### Template Parameters

<i>T</i>	The target type of the capability
<i>F</i>	The source type (and is usually implicitly set)

##### Parameters

<i>c</i>	The source capability that shall be casted
----------	--

##### Returns

A capability typed to the interface *T*.

The use of this cast operator is similar to the `static_cast<>()` for C++ pointers. It does the same type checking and adjustments like C++ does on pointers.

Example code:

```
L4::Cap<L4::Kobject> obj = ... ;
L4::Cap<L4::Icu> icu = L4::cap_cast<L4::Icu>(obj);
```

Definition at line 411 of file [capability.h](#).

#### 14.3.3.2 `cap_cast()` [2/2]

```
template<typename T , typename F , typename SMART >
Smart_cap< T, SMART > L4::cap_cast (
    Smart_cap< F, SMART > const & c ) [inline], [noexcept]
```

`static_cast` for (smart) capabilities.

##### Template Parameters

<i>T</i>	Type to cast the capability to.
<i>F</i>	(implicit) Type of the passed capability.
<i>SMART</i>	(implicit) Class implementing the <a href="#">Smart_cap</a> interface.



## Parameters

<i>c</i>	Capability to be casted.
----------	--------------------------

## Returns

A smart capability with new type *T*.

Definition at line 203 of file [smart\\_capability](#).

**14.3.3.3 cap\_dynamic\_cast()**

```
template<typename T , typename F >
Cap< T > L4::cap_dynamic_cast (
    Cap< F > const & c ) [inline], [noexcept]
```

`dynamic_cast` for capabilities.

## Template Parameters

<i>T</i>	The target type of the capability.
<i>F</i>	The source type (is usually implicitly set).

## Parameters

<i>c</i>	The source capability that shall be casted.
----------	---

## Return values

<i>Cap&lt;T&gt;</i>	Capability of target interface <i>T</i> .
<i>L4_INVALID_CAP</i>	<i>c</i> does not support the target interface <i>T</i> or the <a href="#">L4::Meta</a> interface.

The use of this cast operator is similar to the `dynamic_cast<>()` for C++ pointers. It also induces overhead, because it uses the meta interface ([L4::Meta](#)) to do runtime type checking.

## Example code:

```
L4::Cap<L4::Kobject> obj = ... ;
L4::Cap<L4::Icu> icu = L4::cap_dynamic_cast<L4::Icu>(obj);
```

Definition at line 126 of file [capability](#).

References [l4\\_error\(\)](#).

Here is the call graph for this function:



#### 14.3.3.4 cap\_reinterpret\_cast() [1/2]

```

template<typename T , typename F >
Cap< T > L4::cap_reinterpret_cast (
    Cap< F > const & c ) [inline], [noexcept]
  
```

reinterpret\_cast for capabilities.

##### Template Parameters

<i>T</i>	The target type of the capability
<i>F</i>	The source type (and is usually implicitly set)

##### Parameters

<i>c</i>	The source capability that shall be casted
----------	--

##### Returns

A capability typed to the interface T.

The use of this cast operator is similar to the `reinterpret_cast<>()` for C++ pointers. It does not do any type checking or type adjustment.

Example code:

```

L4::Cap<L4::Kobject> obj = ... ;
L4::Cap<L4::Icu> icu = L4::cap_reinterpret_cast<L4::Icu>(obj);
  
```

Definition at line 442 of file [capability.h](#).

#### 14.3.3.5 cap\_reinterpret\_cast() [2/2]

```

template<typename T , typename F , typename SMART >
Smart_cap< T, SMART > L4::cap_reinterpret_cast (
    Smart_cap< F, SMART > const & c ) [inline], [noexcept]
  
```

reinterpret\_cast for (smart) capabilities.

## Template Parameters

<i>T</i>	Type to cast the capability to.
<i>F</i>	(implicit) Type of the passed capability.
<i>SMART</i>	(implicit) Class implementing the <a href="#">Smart_cap</a> interface.

## Parameters

<i>c</i>	Capability to be casted.
----------	--------------------------

## Returns

A smart capability with new type *T*.

Definition at line [222](#) of file [smart\\_capability](#).

## 14.3.3.6 round\_order()

```
template<typename T >
constexpr T L4::round_order (
    T val,
    unsigned char order ) [constexpr]
```

Round a value up so the given number of lsb is zero.

## Template Parameters

<i>T</i>	The type of the value (shall be some integral type).
----------	--

## Parameters

<i>val</i>	The value to round up to the next multiple of 2 <sup>order</sup> .
<i>order</i>	order (2 <sup>order</sup> ) to round up to.

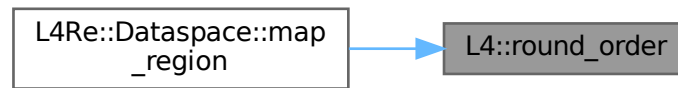
## Returns

*val* rounded up to the next 2<sup>order</sup>.

Definition at line [32](#) of file [consts](#).

Referenced by [L4Re::Dataspace::map\\_region\(\)](#).

Here is the caller graph for this function:



#### 14.3.3.7 trunc\_order()

```

template<typename T >
constexpr T L4::trunc_order (
    T val,
    unsigned char order ) [constexpr]
  
```

Round a value down so the given number of lsb is zero.

##### Template Parameters

<i>T</i>	The type of the value (shall be some integral type).
----------	--

##### Parameters

<i>val</i>	The value where the given lsb shall be masked.
<i>order</i>	the number of least significant bits (lsb) to mask.

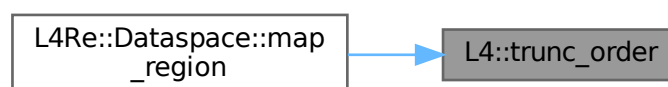
##### Returns

`val` with `order` lsb masked to zero.

Definition at line 18 of file [consts](#).

Referenced by [L4Re::Dataspace::map\\_region\(\)](#).

Here is the caller graph for this function:



## 14.4 L4::lpc Namespace Reference

IPC related functionality.

### Namespaces

- namespace [Msg](#)  
*IPC Message related functionality.*

### Data Structures

- struct [Array](#)  
*Array data type for dynamically sized arrays in RPCs.*
- struct [Array\\_in\\_buf](#)  
*Server-side copy in buffer for [Array](#).*
- struct [Array\\_ref](#)  
*Array reference data type for arrays located in the message.*
- struct [As\\_value](#)  
*Pass the argument as plain data value.*
- struct [Call](#)  
*RPC attribute for a standard RPC call.*
- struct [Call\\_t](#)  
*RPC attribute for an RPC call with required rights.*
- struct [Call\\_zero\\_send\\_timeout](#)  
*RPC attribute for an RPC call, with zero send timeout.*
- class [Cap](#)  
*Capability type for RPC interfaces (see [L4::Cap<T>](#)).*
- class [Gen\\_fpage](#)  
*Generic RPC base for [L4](#) flex-pages.*
- struct [In\\_out](#)  
*Mark an argument as in-out argument.*
- class [lostream](#)  
*Input/Output stream for IPC [un]marshalling.*
- class [Istream](#)  
*Input stream for IPC unmarshalling.*
- class [Msg\\_ptr](#)  
*Pointer to an element of type *T* in an [lpc::Istream](#).*
- struct [Opt](#)  
*Attribute for defining an optional RPC argument.*
- class [Ostream](#)  
*Output stream for IPC marshalling.*
- struct [Out](#)  
*Mark an argument as a output value in an RPC signature.*
- class [Rcv\\_fpage](#)  
*Rcv flex-page.*
- struct [Ret\\_array](#)  
*Dynamically sized output array of type *T*.*
- struct [Send\\_only](#)  
*RPC attribute for a send-only RPC.*

- class [Small\\_buf](#)  
*A receive item for receiving a single object capability.*
- class [Snd\\_fpage](#)  
*Send flex-page.*
- class [Str\\_cp\\_in](#)  
*Abstraction for extracting a zero-terminated string from an [lpc::Istream](#).*
- class [Varg](#)  
*Variably sized RPC argument.*
- class [Varg\\_list](#)  
*Self-contained list of variable-sized RPC parameters.*
- class [Varg\\_list\\_ref](#)  
*List of variable-sized RPC parameters as received by the server.*

## Typedefs

- typedef unsigned short **Array\_len\_default**  
*Default type for passing length of an array.*

## Functions

- template<typename T >  
Internal::Buf\_cp\_out< T > [buf\\_cp\\_out](#) (T const \*v, unsigned long size)  
*Insert an array into an [lpc::Ostream](#).*
- template<typename T >  
Internal::Buf\_cp\_in< T > [buf\\_cp\\_in](#) (T \*v, unsigned long &size)  
*Extract an array from an [lpc::Istream](#).*
- template<typename T >  
[Str\\_cp\\_in](#)< T > [str\\_cp\\_in](#) (T \*v, unsigned long &size)  
*Create a [Str\\_cp\\_in](#) for the given values.*
- template<typename T >  
[Msg\\_ptr](#)< T > [msg\\_ptr](#) (T \*&p)  
*Create an [Msg\\_ptr](#) to adjust the given pointer.*
- template<typename T >  
Internal::Buf\_in< T > [buf\\_in](#) (T \*&v, unsigned long &size)  
*Return a pointer to stream array data.*
- template<typename T >  
T [read](#) ([Istream](#) &s)  
*Read a value out of a stream.*
- template<typename T >  
[Cap](#)< T > [make\\_cap](#) ([L4::Cap](#)< T > cap, unsigned rights) noexcept  
*Make an [L4::lpc::Cap](#)<T> for the given capability and rights.*
- template<typename T >  
[Cap](#)< T > [make\\_cap\\_rw](#) ([L4::Cap](#)< T > cap) noexcept  
*Make an [L4::lpc::Cap](#)<T> for the given capability with [L4\\_CAP\\_FPAGE\\_RW](#) rights.*
- template<typename T >  
[Cap](#)< T > [make\\_cap\\_rws](#) ([L4::Cap](#)< T > cap) noexcept  
*Make an [L4::lpc::Cap](#)<T> for the given capability with [L4\\_CAP\\_FPAGE\\_RWS](#) rights.*
- template<typename T >  
[Cap](#)< T > [make\\_cap\\_full](#) ([L4::Cap](#)< T > cap) noexcept  
*Make an [L4::IPC::Cap](#)<T> for the given capability with full fpage and object-specific rights.*

### 14.4.1 Detailed Description

IPC related functionality.

### 14.4.2 Function Documentation

#### 14.4.2.1 buf\_cp\_in()

```
template<typename T >
Internal::Buf_cp_in< T > L4::Ipc::buf_cp_in (
    T * v,
    unsigned long & size )
```

Extract an array from an [lpc::Istream](#).

##### Parameters

	<i>v</i>	Pointer to the array that shall receive the values from the <a href="#">lpc::Istream</a> .
<i>in, out</i>	<i>size</i>	Input: the number of elements the array can take at most Output: the number of elements found in the stream.

[buf\\_cp\\_in\(\)](#) can be used to extract an array from an [lpc::Istream](#). This is the counterpart [buf\\_cp\\_out\(\)](#). The data from the received message is thereby copied to the given buffer and size is set to the number of elements found in the stream. To avoid the copy operation [buf\\_in\(\)](#) may be used instead.

##### See also

[buf\\_in\(\)](#) and [buf\\_cp\\_out\(\)](#).

Definition at line 170 of file [ipc\\_stream](#).

#### 14.4.2.2 buf\_cp\_out()

```
template<typename T >
Internal::Buf_cp_out< T > L4::Ipc::buf_cp_out (
    T const * v,
    unsigned long size )
```

Insert an array into an [lpc::Ostream](#).

##### Parameters

<i>v</i>	Pointer to the array that shall be inserted into an <a href="#">lpc::Ostream</a> .
<i>size</i>	Number of elements in the array.

This function inserts an array (e.g. a string) into an [lpc::Ostream](#). The data is copied to the stream. On insertion into the [lpc::Ostream](#) exactly the given number of elements of type T are copied to the message buffer, this means the source buffer is no longer referenced after insertion into the stream.

**See also**

The counterpart is either [buf\\_cp\\_in\(\)](#) or [buf\\_in\(\)](#).

Definition at line 111 of file [ipc\\_stream](#).

**14.4.2.3 buf\_in()**

```
template<typename T >
Internal::Buf_in< T > L4::Ipc::buf_in (
    T *& v,
    unsigned long & size )
```

Return a pointer to stream array data.

**Parameters**

out	<i>v</i>	Pointer to the array within the <a href="#">lpc::lstream</a> .
out	<i>size</i>	The number of elements found in the stream.

This routine provides a possibility to extract an array from an [lpc::lstream](#), without extra copy overhead. In contrast to [buf\\_cp\\_in\(\)](#) the data is not copied to a buffer, but a pointer to the array is returned. The user must make sure the UTCB is not used for other purposes while the returned pointer is still in use.

The mechanism is comparable to that of [Msg\\_ptr](#), however it handles arrays inserted with [buf\\_cp\\_out\(\)](#).

**See also**

[buf\\_cp\\_in\(\)](#) and [buf\\_cp\\_out\(\)](#).

Definition at line 321 of file [ipc\\_stream](#).

**14.4.2.4 make\_cap()**

```
template<typename T >
Cap< T > L4::Ipc::make_cap (
    L4::Cap< T > cap,
    unsigned rights ) [noexcept]
```

Make an L4::lpc::Cap<T> for the given capability and rights.

**Template Parameters**

<i>T</i>	(IMPLICIT) type of the referenced interface
----------	---

**Parameters**

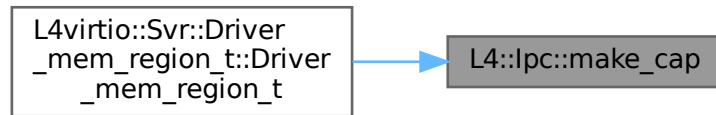
<i>cap</i>	source capability (L4::Cap<T>)
<i>rights</i>	rights mask that shall be applied on transfer.



Definition at line 649 of file [ipc\\_types](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_region\\_t< DATA >::Driver\\_mem\\_region\\_t\(\)](#).

Here is the caller graph for this function:



#### 14.4.2.5 make\_cap\_full()

```

template<typename T >
Cap< T > L4::Ipc::make_cap_full (
    L4::Cap< T > cap ) [noexcept]
  
```

Make an L4::IPC::Cap<T> for the given capability with full fpage and object-specific rights.

##### Template Parameters

<i>T</i>	(implicit) type of the referenced interface
----------	---

##### Parameters

<i>cap</i>	source capability (L4::Cap<T>)
------------	--------------------------------

##### See also

[L4\\_cap\\_fpage\\_rights](#)

[L4\\_obj\\_fpage\\_ctl](#)

##### Note

Full rights (including object-specific rights) are required when mapping an IPC gate where the receiver should become the server, i.e. where the receiver wants to call [L4::lpc\\_gate::bind\\_thread\(\)](#).

Definition at line 687 of file [ipc\\_types](#).

References [L4\\_CAP\\_FPAGE\\_RWSD](#), and [L4\\_FPAGE\\_C\\_OBJ\\_RIGHTS](#).

#### 14.4.2.6 make\_cap\_rw()

```
template<typename T >  
Cap< T > L4::Ipc::make_cap_rw (   
    L4::Cap< T > cap ) [noexcept]
```

Make an L4::Ipc::Cap<T> for the given capability with [L4\\_CAP\\_FPAGE\\_RW](#) rights.

## Template Parameters

<i>T</i>	(IMPLICIT) type of the referenced interface
----------	---

## Parameters

<i>cap</i>	source capability (L4::Cap<T>)
------------	--------------------------------

## Examples

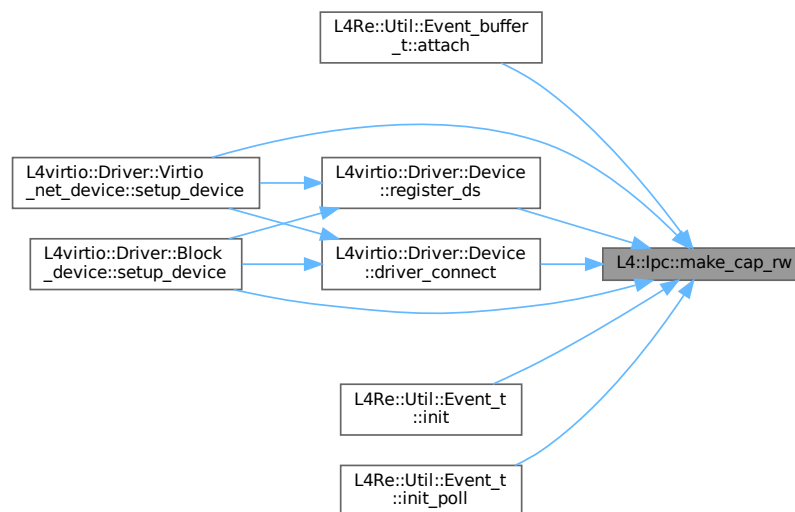
[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), and [examples/libs/l4re/c++/shared\\_](#)

Definition at line 659 of file [ipc\\_types](#).

References [L4\\_CAP\\_FPAGE\\_RW](#).

Referenced by [L4Re::Util::Event\\_buffer\\_t< PAYLOAD >::attach\(\)](#), [L4virtio::Driver::Device::driver\\_connect\(\)](#), [L4Re::Util::Event\\_t< PAYLOAD >::init\(\)](#), [L4Re::Util::Event\\_t< PAYLOAD >::init\\_poll\(\)](#), [L4virtio::Driver::Device::register\\_ds\(\)](#), [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#), and [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#).

Here is the caller graph for this function:



## 14.4.2.7 make\_cap\_rws()

```

template<typename T >
Cap< T > L4::Ipc::make_cap_rws (
    L4::Cap< T > cap ) [noexcept]

```

Make an `L4::Ipc::Cap<T>` for the given capability with [L4\\_CAP\\_FPAGE\\_RWS](#) rights.

## Template Parameters

<i>T</i>	(IMPLICIT) type of the referenced interface
----------	---

## Parameters

<i>cap</i>	source capability ( <code>L4::Cap&lt;T&gt;</code> )
------------	---

Definition at line 669 of file [ipc\\_types](#).

References [L4\\_CAP\\_FPAGE\\_RWS](#).

**14.4.2.8 msg\_ptr()**

```
template<typename T >
Msg_ptr< T > L4::Ipc::msg_ptr (
    T *& p )
```

Create an [Msg\\_ptr](#) to adjust the given pointer.

This function makes it more convenient to extract pointers to data in the message buffer itself from an [lpc::Istream](#). This may be used to avoid copy out of large data structures. (See [Msg\\_ptr](#).)

Definition at line 263 of file [ipc\\_stream](#).

**14.4.2.9 read()**

```
template<typename T >
T L4::Ipc::read (
    Istream & s ) [inline]
```

Read a value out of a stream.

## Parameters

<i>s</i>	An <a href="#">Istream</a> .
----------	------------------------------

## Returns

The value of type `T`.

The stream position is progressed accordingly.

Definition at line 1300 of file [ipc\\_stream](#).

**14.4.2.10 str\_cp\_in()**

```
template<typename T >
Str_cp_in< T > L4::Ipc::str_cp_in (
```

```

T * v,
unsigned long & size )

```

Create a [Str\\_cp\\_in](#) for the given values.

#### Parameters

	<i>v</i>	Pointer to the array that shall receive the values from the <a href="#">lpc::lstream</a> .
<i>in, out</i>	<i>size</i>	Input: the number of elements the array can take at most Output: the number of elements found in the stream.

This function makes it more convenient to extract arrays from an [lpc::lstream](#) (

See also

[Str\\_cp\\_in](#).)

Definition at line 224 of file [ipc\\_stream](#).

## 14.5 L4::lpc::Msg Namespace Reference

IPC Message related functionality.

### Data Structures

- struct [Clnt\\_val\\_ops](#)  
*Defines client-side handling of 'MTYPE' as RPC argument.*
- struct [Cls\\_buffer](#)  
*Marker type for receive buffer values.*
- struct [Cls\\_data](#)  
*Marker type for data values.*
- struct [Cls\\_item](#)  
*Marker type for item values.*
- struct [Dir\\_in](#)  
*Marker type for input values.*
- struct [Dir\\_out](#)  
*Marker type for output values.*
- struct [Do\\_in\\_data](#)  
*Marker for Input data.*
- struct [Do\\_in\\_items](#)  
*Marker for Input items.*
- struct [Do\\_out\\_data](#)  
*Marker for Output data.*
- struct [Do\\_out\\_items](#)  
*Marker for Output items.*
- struct [Do\\_rcv\\_buffers](#)  
*Marker for receive buffers.*
- struct [Elem< Array< A, LEN > & >](#)  
*[Array](#) as output argument.*

- struct [Elem< Array< A, LEN > >](#)  
*Array as input arguments.*
- struct [Elem< Array\\_ref< A, LEN > & >](#)  
*Array\_ref as output argument.*
- struct [Is\\_valid\\_rpc\\_type](#)  
*Type trait defining a valid RPC parameter type.*
- struct [Svr\\_arg\\_pack](#)  
*Server-side RPC arguments data structure used to provide arguments to the server-side implementation of an RPC function.*
- struct [Svr\\_val\\_ops](#)  
*Defines server-side handling for `MTYPE` server arguments.*

## Enumerations

- enum {  
[Word\\_bytes](#) = sizeof(l4\_umword\_t) , [Item\\_words](#) = 2 , [Item\\_bytes](#) = Word\_bytes \* Item\_words , [Mr\\_words](#) = L4\_UTCB\_GENERIC\_DATA\_SIZE ,  
[Mr\\_bytes](#) = Word\_bytes \* Mr\_words , [Br\\_bytes](#) = Word\_bytes \* L4\_UTCB\_GENERIC\_BUFFERS\_SIZE }

## Functions

- constexpr unsigned long [align\\_to](#) (unsigned long bytes, unsigned long align) noexcept  
*Pad bytes to the given alignment align (in bytes)*
- template<typename T >  
 constexpr unsigned long [align\\_to](#) (unsigned long bytes) noexcept  
*Pad bytes to the alignment of the type T.*
- template<typename T >  
 constexpr bool [check\\_size](#) (unsigned offset, unsigned limit) noexcept  
*Check if there is enough space for T from offset to limit.*
- template<typename T , typename CTYPE >  
 bool [check\\_size](#) (unsigned offset, unsigned limit, CTYPE cnt) noexcept  
*Check if there is enough space for an array of T from offset to limit.*
- template<typename T >  
 int [msg\\_add](#) (char \*msg, unsigned offs, unsigned limit, T v) noexcept  
*Add some data to a message at offs.*
- template<typename T >  
 int [msg\\_get](#) (char \*msg, unsigned offs, unsigned limit, T &v) noexcept  
*Get some data from a message at offs.*

### 14.5.1 Detailed Description

IPC Message related functionality.

### 14.5.2 Enumeration Type Documentation

#### 14.5.2.1 anonymous enum

anonymous enum

## Enumerator

Word_bytes	number of bytes for one message word
Item_words	number of message words for one message item
Item_bytes	number of bytes for one message item
Mr_words	number of message words available in the UTCB
Mr_bytes	number of bytes available in the UTCB message registers
Br_bytes	number of bytes available in the UTCB buffer registers

Definition at line 96 of file [ipc\\_basics](#).

### 14.5.3 Function Documentation

#### 14.5.3.1 align\_to() [1/2]

```
template<typename T >
constexpr unsigned long L4::Ipc::Msg::align_to (
    unsigned long bytes ) [constexpr], [noexcept]
```

Pad *bytes* to the alignment of the type *T*.

##### Template Parameters

<i>T</i>	The data type used for the alignment
----------	--------------------------------------

##### Parameters

<i>bytes</i>	The value to add the padding to
--------------	---------------------------------

##### Returns

*bytes* padded to achieve the alignment of *T*.

Definition at line 51 of file [ipc\\_basics](#).

References [align\\_to\(\)](#).

Here is the call graph for this function:



### 14.5.3.2 align\_to() [2/2]

```
constexpr unsigned long L4::Ipc::Msg::align_to (
    unsigned long bytes,
    unsigned long align ) [constexpr], [noexcept]
```

Pad bytes to the given alignment *align* (in bytes)

#### Parameters

<i>bytes</i>	The input value in bytes
<i>align</i>	The alignment value in bytes

#### Returns

the result after padding *bytes* to *align*.

Definition at line 41 of file [ipc\\_basics](#).

Referenced by [align\\_to\(\)](#).

Here is the caller graph for this function:



### 14.5.3.3 check\_size() [1/2]

```
template<typename T >
constexpr bool L4::Ipc::Msg::check_size (
    unsigned offset,
    unsigned limit ) [constexpr], [noexcept]
```

Check if there is enough space for T from offset to limit.

#### Template Parameters

<i>T</i>	The data type that shall be fitted at <i>offset</i>
----------	---

#### Parameters

<i>offset</i>	The current offset in bytes (must already be padded if desired).
<i>limit</i>	The limit in bytes that must not be exceeded after adding the size of <i>T</i> .



**Returns**

true if the limit will not be exceeded, false else.

Definition at line 64 of file [ipc\\_basics](#).

**14.5.3.4 check\_size()** [2/2]

```
template<typename T , typename CTYPE >
bool L4::IpC::Msg::check_size (
    unsigned offset,
    unsigned limit,
    CTYPE cnt ) [inline], [noexcept]
```

Check if there is enough space for an array of T from offset to limit.

**Template Parameters**

<i>T</i>	The data type that shall be fitted at <i>offset</i>
<i>CTYPE</i>	Type of the <i>cnt</i> parameter

**Parameters**

<i>offset</i>	The current offset in bytes (must already be padded if desired).
<i>limit</i>	The limit in bytes that must not be exceeded after adding <i>cnt</i> times the size of <i>T</i> .
<i>cnt</i>	The number of elements of type <i>T</i> that shall be put at <i>offset</i> .

**Returns**

true if the limit will not be exceeded, false else.

Definition at line 82 of file [ipc\\_basics](#).

References [L4\\_UNLIKELY](#).

**14.5.3.5 msg\_add()**

```
template<typename T >
int L4::IpC::Msg::msg_add (
    char * msg,
    unsigned offs,
    unsigned limit,
    T v ) [inline], [noexcept]
```

Add some data to a message at offs.

**Template Parameters**

<i>T</i>	The type of the data to add
----------	-----------------------------

## Parameters

<i>msg</i>	pointer to the start of the message
<i>offs</i>	The current offset within the message, this shall be padded to the alignment of <i>T</i> if <i>v</i> is added.
<i>limit</i>	The size limit in bytes that offset must not exceed.
<i>v</i>	The value to add to the message

## Returns

The new offset when successful, a negative value if the given limit will be exceeded.

Definition at line 125 of file [ipc\\_basics](#).

References [L4\\_MSGTOOLONG](#), and [L4\\_UNLIKELY](#).

### 14.5.3.6 msg\_get()

```
template<typename T >
int L4::Ipc::Msg::msg_get (
    char * msg,
    unsigned offs,
    unsigned limit,
    T & v ) [inline], [noexcept]
```

Get some data from a message at offs.

## Template Parameters

<i>T</i>	The type of the data to read
----------	------------------------------

## Parameters

<i>msg</i>	Pointer to the start of the message
<i>offs</i>	The current offset within the message, this shall be padded to the alignment of <i>T</i> if a <i>v</i> can be read.
<i>limit</i>	The size limit in bytes that offset must not exceed.
<i>v</i>	A reference to receive the value from the message

## Returns

The new offset when successful, a negative value if the given limit will be exceeded.

Definition at line 146 of file [ipc\\_basics](#).

References [L4\\_MSGTOOSHORT](#), and [L4\\_UNLIKELY](#).

## 14.6 L4::ipc\_svr Namespace Reference

Helper classes for [L4::Server](#) instantiation.

## Data Structures

- class [Br\\_manager\\_no\\_buffers](#)  
*Empty implementation of [Server\\_iface](#).*
- struct [Compound\\_reply](#)  
*Mix in for [LOOP\\_HOOKS](#) to always use compound reply and wait.*
- struct [Default\\_loop\\_hooks](#)  
*Default [LOOP\\_HOOKS](#).*
- struct [Default\\_setup\\_wait](#)  
*Mix in for [LOOP\\_HOOKS](#) for setup\_wait no op.*
- struct [Default\\_timeout](#)  
*Mix in for [LOOP\\_HOOKS](#) to use a 0 send and a infinite receive timeout.*
- struct [Direct\\_dispatch](#)  
*Direct dispatch helper, for forwarding dispatch calls to a registry *R*.*
- struct [Direct\\_dispatch< R \\* >](#)  
*Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry *R*.*
- struct [Exc\\_dispatch](#)  
*Dispatch helper wrapping try {} catch {} around the dispatch call.*
- struct [Ignore\\_errors](#)  
*Mix in for [LOOP\\_HOOKS](#) to ignore IPC errors.*
- class [Server\\_iface](#)  
*Interface for server-loop related functions.*
- class [Timeout](#)  
*Callback interface for [Timeout\\_queue](#).*
- class [Timeout\\_queue](#)  
*[Timeout](#) queue to be used in *l4re* server loop.*
- class [Timeout\\_queue\\_hooks](#)  
*Loop hooks mixin for integrating a timeout queue into the server loop.*

## Enumerations

- enum [Reply\\_mode](#) { [Reply\\_compound](#) , [Reply\\_separate](#) }  
*Reply mode for server loop.*

### 14.6.1 Detailed Description

Helper classes for [L4::Server](#) instantiation.

## 14.7 L4::Typeid Namespace Reference

Definition of interface data-type helpers.

## Data Structures

- struct [P\\_dispatch](#)  
*Use for protocol based dispatch stage.*
- struct [Raw\\_ipc](#)  
*RPCs list for passing raw incoming IPC to the server object.*
- struct [Rpc\\_nocode](#)  
*List of RPCs of an interface using a single operation without an opcode.*
- struct [Rpcs](#)  
*Standard list of RPCs of an interface.*
- struct [Rpcs\\_code](#)  
*List of RPCs of an interface using a special opcode type.*
- struct [Rpcs\\_sys](#)  
*List of RPCs typically used for kernel interfaces.*

### 14.7.1 Detailed Description

Definition of interface data-type helpers.

#### Note

These type helpers are intended for internal use, if you look for standard C++ type traits use the `<type_traits>` header for the standard C++ library or use `<l4/cxx/type_traits>`.

## 14.8 L4::Types Namespace Reference

[L4](#) basic type helpers for C++.

## Data Structures

- struct [Bool](#)  
*Boolean meta type.*
- struct [False](#)  
*False meta value.*
- class [Flags](#)  
*Template for defining typical [Flags](#) bitmaps.*
- struct [Flags\\_ops\\_t](#)  
*Mixin class to define a set of friend bitwise operators on *DT*.*
- struct [Flags\\_t](#)  
*Template type to define a flags type with bitwise operations.*
- struct [Int\\_for\\_size](#)  
*Metafunction to get an unsigned integral type for the given size.*
- struct [Int\\_for\\_type](#)  
*Metafunction to get an integral type of the same size as *T*.*
- struct [Same](#)  
*Compare two data types for equality.*
- struct [True](#)  
*True meta value.*

### 14.8.1 Detailed Description

[L4](#) basic type helpers for C++.

## 14.9 L4Re Namespace Reference

[L4Re](#) C++ Interfaces.

### Namespaces

- namespace [Util](#)  
*Documentation of the [L4](#) Runtime Environment utility functionality in C++.*
- namespace [Vfs](#)  
*Virtual file system for interfaces in POSIX libc.*

### Data Structures

- class [Cap\\_alloc](#)  
*Capability allocator interface.*
- class [Console](#)  
*[Console](#) class.*
- class [Dataspace](#)  
*Interface for memory-like objects.*
- class [Debug\\_obj](#)  
*Debug interface.*
- struct [Default\\_event\\_payload](#)  
*Default event stream payload.*
- class [Dma\\_space](#)  
*Managed DMA Address Space.*
- class [Env](#)  
*C++ interface of the initial environment that is provided to an [L4](#) task.*
- class [Event](#)  
*[Event](#) class.*
- class [Event\\_buffer\\_t](#)  
*[Event](#) buffer class.*
- class [Inhibitor](#)  
*Set of inhibitor locks, which inhibit specific actions when held.*
- class [Log](#)  
*[Log](#) interface class.*
- class [Mem\\_alloc](#)  
*Memory allocation interface.*
- struct [Mmio\\_space](#)  
*Interface for memory-like address space accessible via IPC.*
- class [Namespace](#)  
*Name-space interface.*
- class [Parent](#)  
*[Parent](#) interface.*
- struct [Random](#)

- *Low-bandwidth interface for random number generators.*
- class [Rm](#)  
*Region map.*
- class [Smart\\_cap\\_auto](#)  
*Helper for [Unique\\_cap](#) and [Unique\\_del\\_cap](#).*
- class [Smart\\_count\\_cap](#)  
*Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).*

## Typedefs

- `template<typename T >`  
using [Shared\\_cap](#) = [L4::Detail::Shared\\_cap\\_impl](#)< T, [Smart\\_count\\_cap](#)< [L4\\_FP\\_ALL\\_SPACES](#) > >  
*Shared capability that implements automatic free and unmap of the capability selector.*
- `template<typename T >`  
using [shared\\_cap](#) = [L4::Detail::Shared\\_cap\\_impl](#)< T, [Smart\\_count\\_cap](#)< [L4\\_FP\\_ALL\\_SPACES](#) > >  
*Shared capability that implements automatic free and unmap of the capability selector.*
- `template<typename T >`  
using [Shared\\_del\\_cap](#) = [L4::Detail::Shared\\_cap\\_impl](#)< T, [Smart\\_count\\_cap](#)< [L4\\_FP\\_DELETE\\_OBJ](#) > >  
*Shared capability that implements automatic free and unmap+delete of the capability selector.*
- `template<typename T >`  
using [shared\\_del\\_cap](#) = [L4::Detail::Shared\\_cap\\_impl](#)< T, [Smart\\_count\\_cap](#)< [L4\\_FP\\_DELETE\\_OBJ](#) > >  
*Shared capability that implements automatic free and unmap+delete of the capability selector.*
- `template<typename T >`  
using [Unique\\_cap](#) = [L4::Detail::Unique\\_cap\\_impl](#)< T, [Smart\\_cap\\_auto](#)< [L4\\_FP\\_ALL\\_SPACES](#) > >  
*Unique capability that implements automatic free and unmap of the capability selector.*
- `template<typename T >`  
using [unique\\_cap](#) = [L4::Detail::Unique\\_cap\\_impl](#)< T, [Smart\\_cap\\_auto](#)< [L4\\_FP\\_ALL\\_SPACES](#) > >  
*Unique capability that implements automatic free and unmap of the capability selector.*
- `template<typename T >`  
using [Unique\\_del\\_cap](#) = [L4::Detail::Unique\\_cap\\_impl](#)< T, [Smart\\_cap\\_auto](#)< [L4\\_FP\\_DELETE\\_OBJ](#) > >  
*Unique capability that implements automatic free and unmap+delete of the capability selector.*
- `template<typename T >`  
using [unique\\_del\\_cap](#) = [L4::Detail::Unique\\_cap\\_impl](#)< T, [Smart\\_cap\\_auto](#)< [L4\\_FP\\_DELETE\\_OBJ](#) > >  
*Unique capability that implements automatic free and unmap+delete of the capability selector.*

## Functions

- `void throw\_error (long err, char const *extra="")`  
*Generate C++ exception.*
- `long chksys (long err, char const *extra="", long ret=0)`  
*Generate C++ exception on error.*
- `long chksys (l4\_msgtag\_t const &t, char const *extra="", l4\_utcb\_t *utcb=l4\_utcb(), long ret=0)`  
*Generate C++ exception on error.*
- `long chksys (l4\_msgtag\_t const &t, l4\_utcb\_t *utcb, char const *extra="")`  
*Generate C++ exception on error.*
- `template<typename T >`  
`T chkcap (T &&cap, char const *extra="", long err=L4\_ENOMEM)`  
*Check for valid capability or raise C++ exception.*
- `l4\_msgtag\_t chkipc (l4\_msgtag\_t tag, char const *extra="", l4\_utcb\_t *utcb=l4\_utcb())`  
*Test a message tag for IPC errors.*

- `template<typename T >`  
[Shared\\_cap](#)< T > [make\\_shared\\_cap](#) (L4Re::Cap\_alloc \*ca)  
*Allocate a capability slot and wrap it in a Shared\_cap.*
- `template<typename T >`  
[Shared\\_del\\_cap](#)< T > [make\\_shared\\_del\\_cap](#) (L4Re::Cap\_alloc \*ca)  
*Allocate a capability slot and wrap it in a Shared\_del\_cap.*
- `template<typename T >`  
[Unique\\_cap](#)< T > [make\\_unique\\_cap](#) (L4Re::Cap\_alloc \*ca)  
*Allocate a capability slot and wrap it in an Unique\_cap.*
- `template<typename T >`  
[Unique\\_del\\_cap](#)< T > [make\\_unique\\_del\\_cap](#) (L4Re::Cap\_alloc \*ca)  
*Allocate a capability slot and wrap it in an Unique\_del\_cap.*

## 14.9.1 Detailed Description

[L4Re](#) C++ Interfaces.

[L4](#) Runtime Environment.

## 14.9.2 Typedef Documentation

### 14.9.2.1 Shared\_cap

```
template<typename T >
using L4Re::Shared\_cap = typedef L4::Detail::Shared_cap_impl<T, Smart\_count\_cap<L4\_FP\_ALL\_SPACES>
>
```

Shared capability that implements automatic free and unmap of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

#### Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Shared\\_cap](#).

Definition at line 44 of file [shared\\_cap](#).

### 14.9.2.2 shared\_cap

```
template<typename T >
using L4Re::shared\_cap = typedef L4::Detail::Shared_cap_impl<T, Smart\_count\_cap<L4\_FP\_ALL\_SPACES>
>
```

Shared capability that implements automatic free and unmap of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

#### Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Shared\\_cap](#).

Definition at line 47 of file [shared\\_cap](#).

### 14.9.2.3 Shared\_del\_cap

```
template<typename T >
using L4Re::Shared_del_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>
>
```

Shared capability that implements automatic free and unmap+delete of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to `Shared_cap` is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

#### Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Shared\\_del\\_cap](#).

Definition at line 80 of file [shared\\_cap](#).

### 14.9.2.4 shared\_del\_cap

```
template<typename T >
using L4Re::shared_del_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>
>
```

Shared capability that implements automatic free and unmap+delete of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--



This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to `Shared_cap` is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

#### Note

This type is intended for users who implement a custom capability allocator; otherwise use `L4Re::Util::Shared_del_cap`.

Definition at line 83 of file `shared_cap`.

### 14.9.2.5 Unique\_cap

```
template<typename T >
using L4Re::Unique_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>
>
```

Unique capability that implements automatic free and unmap of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

The ownership of the capability is managed in the same way as `unique_ptr`.

#### Note

This type is intended for users who implement a custom capability allocator; otherwise use `L4Re::Util::Unique_cap`.

Definition at line 42 of file `unique_cap`.

### 14.9.2.6 unique\_cap

```
template<typename T >
using L4Re::unique_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>
>
```

Unique capability that implements automatic free and unmap of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

The ownership of the capability is managed in the same way as `unique_ptr`.

#### Note

This type is intended for users who implement a custom capability allocator; otherwise use `L4Re::Util::Unique_cap`.

Definition at line 45 of file `unique_cap`.

### 14.9.2.7 Unique\_del\_cap

```
template<typename T >
using L4Re::Unique_del_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>
>
```

Unique capability that implements automatic free and unmap+delete of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

The main difference to Unique\_cap is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

#### Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Unique\\_del\\_cap](#).

Definition at line 75 of file [unique\\_cap](#).

### 14.9.2.8 unique\_del\_cap

```
template<typename T >
using L4Re::unique_del_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>
>
```

Unique capability that implements automatic free and unmap+delete of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

The main difference to Unique\_cap is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

#### Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Unique\\_del\\_cap](#).

Definition at line 78 of file [unique\\_cap](#).

## 14.9.3 Function Documentation

### 14.9.3.1 chkcap()

```
template<typename T >
T L4Re::chkcap (
    T && cap,
    char const * extra = "",
    long err = -L4_ENOMEM ) [inline]
```

Check for valid capability or raise C++ exception.

## Template Parameters

<i>T</i>	Type of object to check, must be capability-like ( <a href="#">L4::Cap</a> , <a href="#">L4Re::Util::Unique_cap</a> etc.)
----------	---

## Parameters

<i>cap</i>	Capability value to check.
<i>extra</i>	Optional text for exception.
<i>err</i>	Error value for exception or 0 if the capability value should be used.

This function checks whether the capability is valid. If the capability is invalid an C++ exception is generated, using *err* if *err* is not zero, otherwise the capability value is used. A valid capability will just be returned.

Definition at line 145 of file [error\\_helper](#).

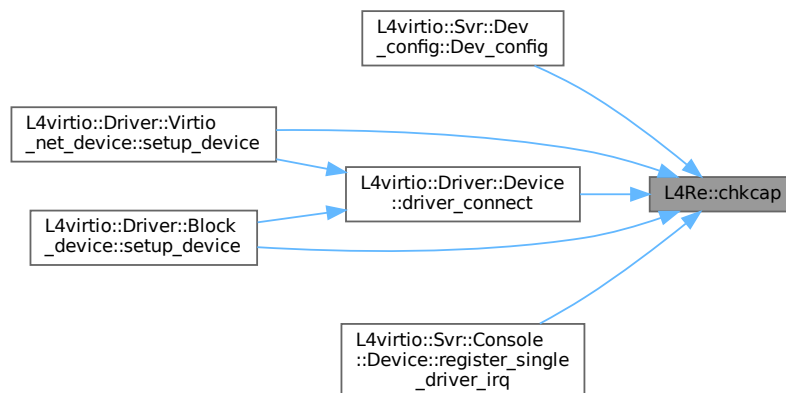
References [L4\\_UNLIKELY](#), and [throw\\_error\(\)](#).

Referenced by [L4virtio::Svr::Dev\\_config::Dev\\_config\(\)](#), [L4virtio::Driver::Device::driver\\_connect\(\)](#), [L4virtio::Svr::Console::Device::register\\_single\\_driver\\_irq\(\)](#), [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#), and [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 14.9.3.2 chkipc()

```
l4_msgtag_t L4Re::chkipc (
    l4_msgtag_t tag,
    char const * extra = "",
    l4_utcb_t * utcb = l4_utcb() ) [inline]
```

Test a message tag for IPC errors.

#### Parameters

<i>tag</i>	Message tag returned by the IPC.
<i>extra</i>	Exception message in case of error.
<i>utcb</i>	The UTCB used in the IPC operation.

#### Returns

On IPC error an exception is thrown, otherwise `tag` is returned.

#### Exceptions

<i>L4::Runtime_exception</i>	with the translated IPC error code
------------------------------	------------------------------------

This function does not check the message tag's label value.

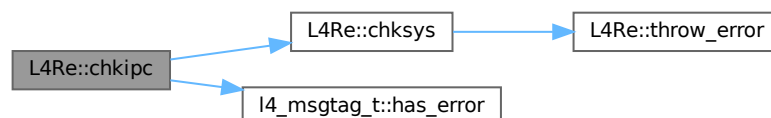
#### Note

This must be called on a message tag before the UTCB is changed.

Definition at line 176 of file [error\\_helper](#).

References [chksys\(\)](#), [l4\\_msgtag\\_t::has\\_error\(\)](#), and [L4\\_UNLIKELY](#).

Here is the call graph for this function:



### 14.9.3.3 chksys() [1/3]

```
long L4Re::chksys (
    l4_msgtag_t const & t,
    char const * extra = "",
    l4_utcb_t * utcb = l4_utcb(),
    long ret = 0 ) [inline]
```

Generate C++ exception on error.

## Parameters

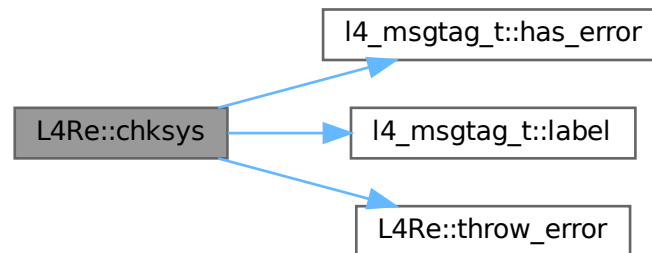
<i>t</i>	Message tag.
<i>extra</i>	Optional text for exception (default "")
<i>utcb</i>	Option UTCB
<i>ret</i>	Optional value for exception, default is error value (err)

This function throws an exception if the message tag contains an error or the label in the message tag is negative. Otherwise the label in the message tag is returned.

Definition at line 89 of file [error\\_helper](#).

References [l4\\_msgtag\\_t::has\\_error\(\)](#), [L4\\_UNLIKELY](#), [l4\\_msgtag\\_t::label\(\)](#), and [throw\\_error\(\)](#).

Here is the call graph for this function:



#### 14.9.3.4 chksys() [2/3]

```

long L4Re::chksys (
    l4_msgtag_t const & t,
    l4_utcb_t * utcb,
    char const * extra = "" ) [inline]

```

Generate C++ exception on error.

## Parameters

<i>t</i>	Message tag.
<i>utcb</i>	UTCB.
<i>extra</i>	Optional text for exception (default "")

This function throws an exception if the message tag contains an error or the label in the message tag is negative. Otherwise the label in the message tag is returned.

Definition at line 112 of file [error\\_helper](#).

References [chksys\(\)](#).

Here is the call graph for this function:



### 14.9.3.5 chksys() [3/3]

```

long L4Re::chksys (
    long err,
    char const * extra = "",
    long ret = 0 ) [inline]
  
```

Generate C++ exception on error.

#### Parameters

<i>err</i>	Error value, if negative exception will be thrown
<i>extra</i>	Optional text for exception (default "")
<i>ret</i>	Optional value for exception, default is error value (err)

This function throws an exception if the *err* is negative and otherwise returns *err*.

#### Examples

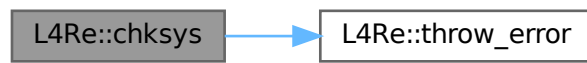
[examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#).

Definition at line 68 of file [error\\_helper](#).

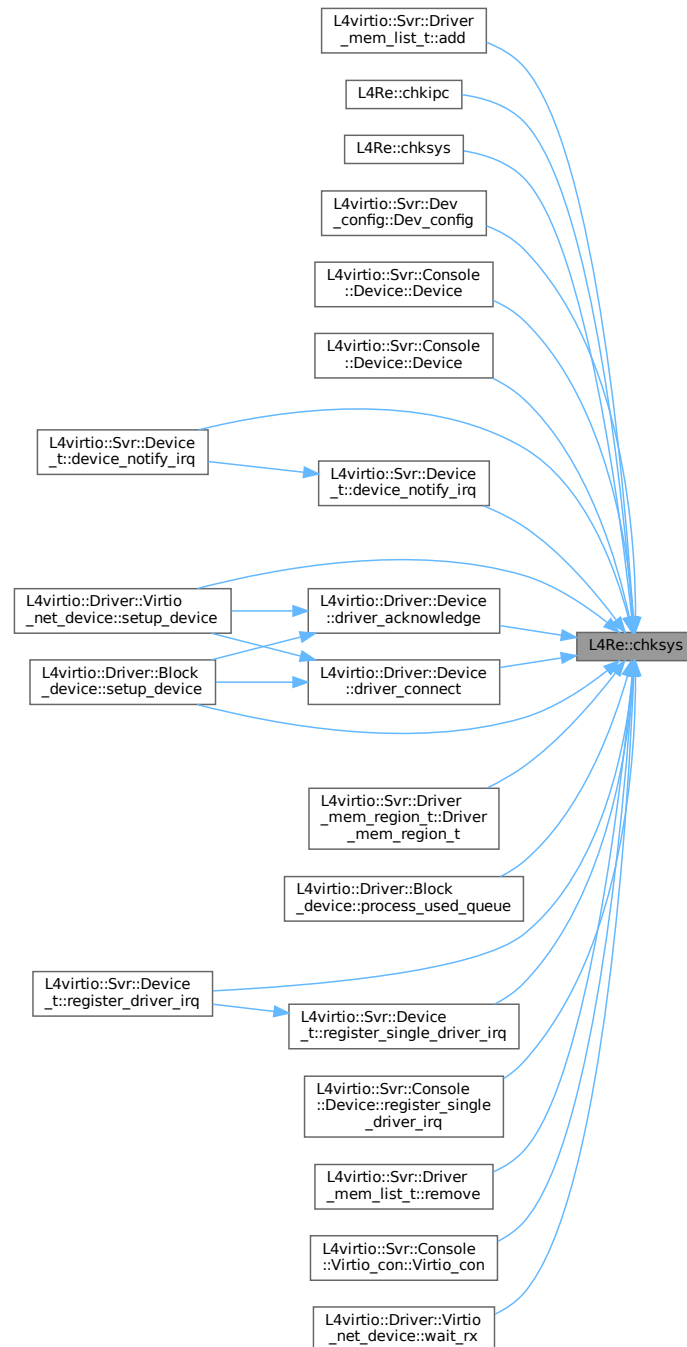
References [L4\\_UNLIKELY](#), and [throw\\_error\(\)](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_list\\_t< DATA >::add\(\)](#), [chkipc\(\)](#), [chksys\(\)](#), [L4virtio::Svr::Dev\\_config::Dev\\_config\(\)](#), [L4virtio::Svr::Console::Device::Device\(\)](#), [L4virtio::Svr::Console::Device::Device\(\)](#), [L4virtio::Svr::Device\\_t< DATA >::device\\_notify\\_irq\(\)](#), [L4virtio::Svr::Device\\_t< DATA >::device\\_notify\\_irq\(\)](#), [L4virtio::Driver::Device::driver\\_acknowledge\(\)](#), [L4virtio::Driver::Device::driver\\_c](#), [L4virtio::Svr::Driver\\_mem\\_region\\_t< DATA >::Driver\\_mem\\_region\\_t\(\)](#), [L4virtio::Driver::Block\\_device::process\\_used\\_queue\(\)](#), [L4virtio::Svr::Device\\_t< DATA >::register\\_driver\\_irq\(\)](#), [L4virtio::Svr::Device\\_t< DATA >::register\\_single\\_driver\\_irq\(\)](#), [L4virtio::Svr::Console::Device::register\\_single\\_driver\\_irq\(\)](#), [L4virtio::Svr::Driver\\_mem\\_list\\_t< DATA >::remove\(\)](#), [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#), [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#), [L4virtio::Svr::Console::Virtio\\_con::V](#) and [L4virtio::Driver::Virtio\\_net\\_device::wait\\_rx\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.9.3.6 make\_shared\_cap()

```

template<typename T >
Shared_cap< T > L4Re::make_shared_cap (
    L4Re::Cap_alloc * ca )

```

Allocate a capability slot and wrap it in a `Shared_cap`.



## Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

## Parameters

<i>ca</i>	Capability allocator to use.
-----------	------------------------------

## Note

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make\\_shared\\_cap<T>\(\)](#).

Definition at line 60 of file [shared\\_cap](#).

References [L4Re::Cap\\_alloc::alloc\(\)](#).

Here is the call graph for this function:



## 14.9.3.7 make\_shared\_del\_cap()

```

template<typename T >
Shared_del_cap< T > L4Re::make_shared_del_cap (
    L4Re::Cap_alloc * ca )
  
```

Allocate a capability slot and wrap it in a Shared\_del\_cap.

## Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

## Parameters

<i>ca</i>	Capability allocator to use.
-----------	------------------------------

## Note

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make\\_shared\\_del\\_cap<T>\(\)](#).

Definition at line 96 of file [shared\\_cap](#).

References [L4Re::Cap\\_alloc::alloc\(\)](#).

Here is the call graph for this function:



#### 14.9.3.8 make\_unique\_cap()

```

template<typename T >
Unique_cap< T > L4Re::make_unique_cap (
    L4Re::Cap_alloc * ca )
  
```

Allocate a capability slot and wrap it in an Unique\_cap.

##### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

##### Parameters

<i>ca</i>	Capability allocator to use.
-----------	------------------------------

##### Note

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make\\_unique\\_cap<T>\(\)](#).

Definition at line 58 of file [unique\\_cap](#).

References [L4Re::Cap\\_alloc::alloc\(\)](#).

Here is the call graph for this function:



### 14.9.3.9 make\_unique\_del\_cap()

```
template<typename T >
Unique_del_cap< T > L4Re::make_unique_del_cap (
    L4Re::Cap_alloc * ca )
```

Allocate a capability slot and wrap it in an Unique\_del\_cap.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

#### Parameters

<i>ca</i>	Capability allocator to use.
-----------	------------------------------

#### Note

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make\\_unique\\_del\\_cap<T>\(\)](#).

Definition at line 91 of file [unique\\_cap](#).

References [L4Re::Cap\\_alloc::alloc\(\)](#).

Here is the call graph for this function:



### 14.9.3.10 throw\_error()

```
void L4Re::throw_error (
    long err,
    char const * extra = "" ) [inline]
```

Generate C++ exception.

#### Parameters

<i>err</i>	Error value
<i>extra</i>	Optional text for exception (default "")

This function throws an [L4](#) exception. The exact exception type depends on the error value (err). This function does

never return.

### Examples

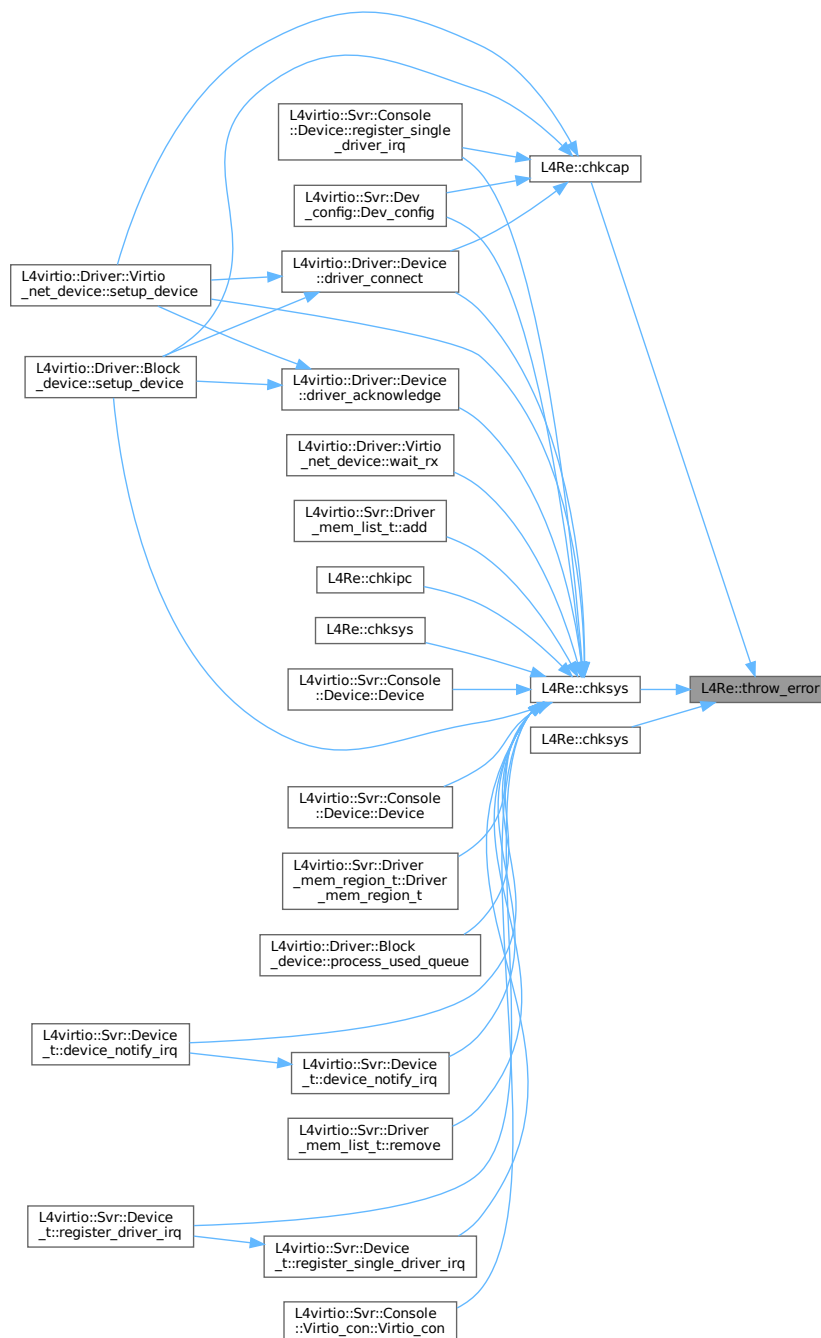
[examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#).

Definition at line 45 of file [error\\_helper](#).

References [L4\\_EEXIST](#), [L4\\_ENOENT](#), [L4\\_ENOMEM](#), and [L4\\_ERANGE](#).

Referenced by [chkcap\(\)](#), [chksys\(\)](#), and [chksys\(\)](#).

Here is the caller graph for this function:



## 14.10 L4Re::Util Namespace Reference

Documentation of the [L4](#) Runtime Environment utility functionality in C++.

### Data Structures

- class [Br\\_manager](#)  
*Buffer-register (BR) manager for [L4::Server](#).*
- struct [Br\\_manager\\_hooks](#)  
*Predefined server-loop hooks for a server loop using the [Br\\_manager](#).*
- struct [Br\\_manager\\_timeout\\_hooks](#)  
*Predefined server-loop hooks for a server with using the [Br\\_manager](#) and a timeout queue.*
- class [Cap\\_alloc\\_base](#)  
*Capability allocator.*
- struct [Counter](#)  
*Counter for [Counting\\_cap\\_alloc](#) with variable data width.*
- struct [Counter\\_atomic](#)  
*Thread safe version of counter for [Counting\\_cap\\_alloc](#).*
- class [Counting\\_cap\\_alloc](#)  
*Internal reference-counting cap allocator.*
- class [Dataspace\\_svr](#)  
*[Dataspace](#) server class.*
- class [Event\\_buffer\\_consumer\\_t](#)  
*An event buffer consumer.*
- class [Event\\_buffer\\_t](#)  
*Event\_buffer utility class.*
- class [Event\\_svr](#)  
*Convenience wrapper for implementing an event server.*
- class [Event\\_t](#)  
*Convenience wrapper for getting access to an event object.*
- class [Item\\_alloc\\_base](#)  
*Item allocator.*
- class [Object\\_registry](#)  
*A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread.*
- struct [Ref\\_cap](#)  
*Automatic capability that implements automatic free and unmap of the capability selector.*
- struct [Ref\\_del\\_cap](#)  
*Automatic capability that implements automatic free and unmap+delete of the capability selector.*
- class [Registry\\_server](#)  
*A server loop object which has a [Object\\_registry](#) included.*
- class [Smart\\_cap\\_auto](#)  
*Helper for [Unique\\_cap](#) and [Unique\\_del\\_cap](#).*
- class [Smart\\_count\\_cap](#)  
*Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).*
- class [Vcon\\_svr](#)  
*[Console](#) server template class.*

## Typedefs

- `template<typename T >`  
`using Shared\_cap = L4::Detail::Shared_cap_impl< T, Smart\_count\_cap< L4\_FP\_ALL\_SPACES > >`  
*Shared capability that implements automatic free and unmap of the capability selector.*
- `template<typename T >`  
`using shared\_cap = L4::Detail::Shared_cap_impl< T, Smart\_count\_cap< L4\_FP\_ALL\_SPACES > >`  
*Shared capability that implements automatic free and unmap of the capability selector.*
- `template<typename T >`  
`using Shared\_del\_cap = L4::Detail::Shared_cap_impl< T, Smart\_count\_cap< L4\_FP\_DELETE\_OBJ > >`  
*Shared capability that implements automatic free and unmap+delete of the capability selector.*
- `template<typename T >`  
`using shared\_del\_cap = L4::Detail::Shared_cap_impl< T, Smart\_count\_cap< L4\_FP\_DELETE\_OBJ > >`  
*Shared capability that implements automatic free and unmap+delete of the capability selector.*
- `template<typename T >`  
`using Unique\_cap = L4::Detail::Unique_cap_impl< T, Smart\_cap\_auto< L4\_FP\_ALL\_SPACES > >`  
*Unique capability that implements automatic free and unmap of the capability selector.*
- `template<typename T >`  
`using unique\_cap = L4::Detail::Unique_cap_impl< T, Smart\_cap\_auto< L4\_FP\_ALL\_SPACES > >`  
*Unique capability that implements automatic free and unmap of the capability selector.*
- `template<typename T >`  
`using Unique\_del\_cap = L4::Detail::Unique_cap_impl< T, Smart\_cap\_auto< L4\_FP\_DELETE\_OBJ > >`  
*Unique capability that implements automatic free and unmap+delete of the capability selector.*
- `template<typename T >`  
`using unique\_del\_cap = L4::Detail::Unique_cap_impl< T, Smart\_cap\_auto< L4\_FP\_DELETE\_OBJ > >`  
*Unique capability that implements automatic free and unmap+delete of the capability selector.*

## Functions

- `template<typename T >`  
`Ref\_cap< T >::Cap make\_ref\_cap ()`  
*Allocate a capability slot and wrap it in a [Ref\\_cap](#).*
- `template<typename T >`  
`Ref\_del\_cap< T >::Cap make\_ref\_del\_cap ()`  
*Allocate a capability slot and wrap it in a [Ref\\_del\\_cap](#).*
- `int kumem\_alloc (l4\_addr\_t *mem, unsigned pages_order, L4::Cap< L4::Task > task=L4Re::Env::env() ->task(), L4::Cap< L4Re::Rm > rm=L4Re::Env::env() ->rm()) noexcept`  
*Allocate state area.*
- `template<typename T >`  
`Shared\_cap< T > make\_shared\_cap ()`  
*Allocate a capability slot and wrap it in a [Shared\\_cap](#).*
- `template<typename T >`  
`Shared\_del\_cap< T > make\_shared\_del\_cap ()`  
*Allocate a capability slot and wrap it in a [Shared\\_del\\_cap](#).*
- `template<typename T >`  
`Unique\_cap< T > make\_unique\_cap ()`  
*Allocate a capability slot and wrap it in an [Unique\\_cap](#).*
- `template<typename T >`  
`Unique\_del\_cap< T > make\_unique\_del\_cap ()`  
*Allocate a capability slot and wrap it in an [Unique\\_del\\_cap](#).*

## Variables

- `_Cap_alloc` & `cap_alloc`  
*Capability allocator.*

### 14.10.1 Detailed Description

Documentation of the [L4](#) Runtime Environment utility functionality in C++.

### 14.10.2 Typedef Documentation

#### 14.10.2.1 Shared\_cap

```
template<typename T >
using L4Re::Util::Shared_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>
>
```

Shared capability that implements automatic free and unmap of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

Usage:

```
L4Re::Util::Shared_cap<L4Re::Dataspace> global_ds_cap;

{
    L4Re::Util::Shared_cap<L4Re::Dataspace>
        ds_cap = make_shared_cap<L4Re::Dataspace>();
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
```

Definition at line 59 of file [shared\\_cap](#).

#### 14.10.2.2 shared\_cap

```
template<typename T >
using L4Re::Util::shared_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>
>
```

Shared capability that implements automatic free and unmap of the capability selector.

**Template Parameters**

<i>T</i>	Type of the object the capability refers to.
----------	--

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

**Usage:**

```
L4Re::Util::Shared_cap<L4Re::Dataspace> global_ds_cap;

{
    L4Re::Util::Shared_cap<L4Re::Dataspace>
        ds_cap = make_shared_cap<L4Re::Dataspace>();
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
```

Definition at line 62 of file [shared\\_cap](#).

**14.10.2.3 Shared\_del\_cap**

```
template<typename T >
using L4Re::Util::Shared_del_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>
>
```

Shared capability that implements automatic free and unmap+delete of the capability selector.

**Template Parameters**

<i>T</i>	Type of the object the capability refers to.
----------	--

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to `Shared_cap` is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

**Usage:**

```
L4Re::Util::Shared_del_cap<L4Re::Dataspace> global_ds_cap;

{
    L4Re::Util::Shared_del_cap<L4Re::Dataspace>
        ds_cap = make_shared_del_cap<L4Re::Dataspace>();
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));
}
```



```

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
...
global_ds_cap = L4_INVALID_CAP;
// reference count dropped to 0 (data space shall be deleted).

```

Definition at line 109 of file [shared\\_cap](#).

#### 14.10.2.4 shared\_del\_cap

```

template<typename T >
using L4Re::Util::shared_del_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>
>

```

Shared capability that implements automatic free and unmap+delete of the capability selector.

##### Template Parameters

<b>T</b>	Type of the object the capability refers to.
----------	--

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to Shared\_cap is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

##### Usage:

```

L4Re::Util::Shared_del_cap<L4Re::Dataspace> global_ds_cap;

{
    L4Re::Util::Shared_del_cap<L4Re::Dataspace>
        ds_cap = make_shared_del_cap<L4Re::Dataspace>();
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
...
global_ds_cap = L4_INVALID_CAP;
// reference count dropped to 0 (data space shall be deleted).

```

Definition at line 112 of file [shared\\_cap](#).

#### 14.10.2.5 Unique\_cap

```

template<typename T >
using L4Re::Util::Unique_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>
>

```

Unique capability that implements automatic free and unmap of the capability selector.

## Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

The ownership of the capability is managed in the same way as `unique_ptr`.

## Usage:

```
{
    L4Re::Util::Unique_cap<L4Re::Dataspace>
        ds_cap = L4Re::Util::make_unique_cap<L4Re::Dataspace>();

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));

    ...

    // At the end of the scope ds_cap is unmapped and the capability
    // selector is freed.
}
```

Definition at line 54 of file [unique\\_cap](#).

## 14.10.2.6 unique\_cap

```
template<typename T >
using L4Re::Util::unique_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>
>
```

Unique capability that implements automatic free and unmap of the capability selector.

## Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

The ownership of the capability is managed in the same way as `unique_ptr`.

## Usage:

```
{
    L4Re::Util::Unique_cap<L4Re::Dataspace>
        ds_cap = L4Re::Util::make_unique_cap<L4Re::Dataspace>();

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));

    ...

    // At the end of the scope ds_cap is unmapped and the capability
    // selector is freed.
}
```

Definition at line 57 of file [unique\\_cap](#).

### 14.10.2.7 Unique\_del\_cap

```
template<typename T >
using L4Re::Util::Unique_del_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>
>
```

Unique capability that implements automatic free and unmap+delete of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

The main difference to Unique\_cap is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```
{
    L4Re::Util::Unique_del_cap<L4Re::Dataspace>
        ds_cap = make_unique_del_cap<L4Re::Dataspace>();

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));

    ...

    // At the end of the scope ds_cap is unmapped and the capability
    // selector is freed. Because the deletion flag is set the data space
    // shall also be deleted (even if there are other references to this
    // data space).
}
```

Definition at line 97 of file [unique\\_cap](#).

### 14.10.2.8 unique\_del\_cap

```
template<typename T >
using L4Re::Util::unique_del_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>
>
```

Unique capability that implements automatic free and unmap+delete of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

The main difference to Unique\_cap is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```
{
    L4Re::Util::Unique_del_cap<L4Re::Dataspace>
```

```

    ds_cap = make_unique_del_cap<L4Re::Dataspace>());

// use the dataspace cap
L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));

...

// At the end of the scope ds_cap is unmapped and the capability
// selector is freed. Because the deletion flag is set the data space
// shall also be deleted (even if there are other references to this
// data space).
}

```

Definition at line 100 of file [unique\\_cap](#).

### 14.10.3 Function Documentation

#### 14.10.3.1 make\_shared\_cap()

```

template<typename T >
Shared_cap< T > L4Re::Util::make_shared_cap ( )

```

Allocate a capability slot and wrap it in a Shared\_cap.

##### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

Definition at line 71 of file [shared\\_cap](#).

References [cap\\_alloc](#).

#### 14.10.3.2 make\_shared\_del\_cap()

```

template<typename T >
Shared_del_cap< T > L4Re::Util::make_shared_del_cap ( )

```

Allocate a capability slot and wrap it in a Shared\_del\_cap.

##### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

Definition at line 121 of file [shared\\_cap](#).

References [cap\\_alloc](#).

#### 14.10.3.3 make\_unique\_cap()

```

template<typename T >
Unique_cap< T > L4Re::Util::make_unique_cap ( )

```

Allocate a capability slot and wrap it in an Unique\_cap.

## Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

Definition at line 66 of file [unique\\_cap](#).

References [cap\\_alloc](#).

#### 14.10.3.4 make\_unique\_del\_cap()

```
template<typename T >
Unique_del_cap< T > L4Re::Util::make_unique_del_cap ( )
```

Allocate a capability slot and wrap it in an Unique\_del\_cap.

## Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

Definition at line 109 of file [unique\\_cap](#).

References [cap\\_alloc](#).

## 14.11 L4Re::Vfs Namespace Reference

Virtual file system for interfaces in POSIX libc.

## Data Structures

- class [Be\\_file](#)  
*Boiler plate class for implementing an open file for [L4Re::Vfs](#).*
- class [Be\\_file\\_system](#)  
*Boilerplate class for implementing a [L4Re::Vfs::File\\_system](#).*
- class [Directory](#)  
*Interface for a POSIX file that is a directory.*
- class [File](#)  
*The basic interface for an open POSIX file.*
- class [File\\_system](#)  
*Basic interface for an [L4Re::Vfs](#) file system.*
- class [Fs](#)  
*POSIX File-system related functionality.*
- class [Generic\\_file](#)  
*The common interface for an open POSIX file.*
- class [Mman](#)  
*Interface for POSIX memory management.*
- class [Ops](#)  
*Interface for the POSIX backends of an application.*
- class [Regular\\_file](#)  
*Interface for a POSIX file that provides regular file semantics.*
- class [Special\\_file](#)  
*Interface for a POSIX file that provides special file semantics.*

## Functions

- [L4Re::Vfs::Ops](#) \*vfs\_ops **asm** ("l4re\_env\_posix\_vfs\_ops")  
*Reference to the applications [L4Re::Vfs::Ops](#) singleton.*

### 14.11.1 Detailed Description

Virtual file system for interfaces in POSIX libc.

## 14.12 L4vbus Namespace Reference

C++ interface of the [Vbus](#) API.

### Data Structures

- class [Device](#)  
*Device on a [L4vbus::Vbus](#).*
- class [Gpio\\_module](#)  
*A [Gpio\\_module](#) groups multiple GPIO pins together.*
- class [Gpio\\_pin](#)  
*A GPIO pin.*
- class [Icu](#)  
*[Vbus](#) Interrupt controller API.*
- class [Pci\\_dev](#)  
*A PCI device.*
- class [Pci\\_host\\_bridge](#)  
*A Pci host bridge.*
- class [Pm](#)  
*Power-management API mixin.*
- class [Vbus](#)  
*The virtual bus ([Vbus](#)) interface.*

### 14.12.1 Detailed Description

C++ interface of the [Vbus](#) API.

The virtual bus ([Vbus](#)) is a hierarchical (tree) structure of device nodes where each device has a set of resources attached to it. Each virtual bus provides an [Icu](#) ([Interrupt controller](#)) for interrupt handling.

The [Vbus](#) interface allows a client to find and query devices present on his virtual bus. After obtaining a device handle for a specific device the client can enumerate its resources.

Refer to [L4 Vbus functions](#) for the C API.

#### Include File

```
#include <l4/vbus/vbus>
```

#### Include File

```
#include <l4/vbus/vbus_gpio>
```

#### Include File

```
#include <l4/vbus/vbus_pci>
```

## 14.13 L4virtio Namespace Reference

L4-VIRTIO Transport C++ API.

### Data Structures

- class [Device](#)  
*IPC interface for virtio over [L4](#) IPC.*
- class [Ptr](#)  
*Pointer used in virtio descriptors.*
- class [Virtqueue](#)  
*Low-level [Virtqueue](#).*

### 14.13.1 Detailed Description

L4-VIRTIO Transport C++ API.



## Chapter 15

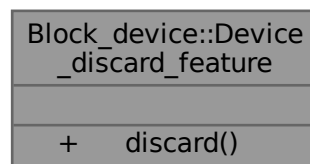
# Data Structure Documentation

### 15.1 Block\_device::Device\_discard\_feature Struct Reference

Partial interface for devices that offer discard functionality.

```
#include <device.h>
```

Collaboration diagram for Block\_device::Device\_discard\_feature:



#### Public Member Functions

- virtual int **discard** ([l4\\_uint64\\_t](#) offset, [Block\\_device::Inout\\_block](#) const &blocks, [Block\\_device::Inout\\_callback](#) const &cb, bool discard)=0  
*Issues one or more WRITE\_ZEROES or DISCARD commands.*

#### 15.1.1 Detailed Description

Partial interface for devices that offer discard functionality.

Definition at line [120](#) of file [device.h](#).

The documentation for this struct was generated from the following file:

- [l4/libblock-device/device.h](#)

## 15.2 Block\_device::Device\_mgr< DEV, FACTORY, SCHEDULER > Class Template Reference

Basic class that scans devices and handles client connections.

```
#include <block_device_mgr.h>
```

Collaboration diagram for Block\_device::Device\_mgr< DEV, FACTORY, SCHEDULER >:

Block_device::Device_mgr< DEV, FACTORY, SCHEDULER >	
+	check_clients()
+	shutdown_event()

### Public Member Functions

- void **check\_clients** ()  
*Remove clients where the client IPC gate is no longer valid.*
- void **shutdown\_event** (Shutdown\_type type)  
*Process a shutdown event on all connections.*

### 15.2.1 Detailed Description

```
template<typename DEV, typename FACTORY = Simple_factory<DEV>, typename SCHEDULER = Rr_scheduler<typename FACTORY::Device_type>>
class Block_device::Device_mgr< DEV, FACTORY, SCHEDULER >
```

Basic class that scans devices and handles client connections.

#### Template Parameters

<i>DEV</i>	Base class for all devices.
<i>FACTORY</i>	Class that creates clients and partitions. See Simple_factory for an example of the required interface.
<i>SCHEDULER</i>	Class that schedules VIRTIO block requests from all clients.

Definition at line 80 of file [block\\_device\\_mgr.h](#).

The documentation for this class was generated from the following file:

- I4/libblock-device/block\_device\_mgr.h

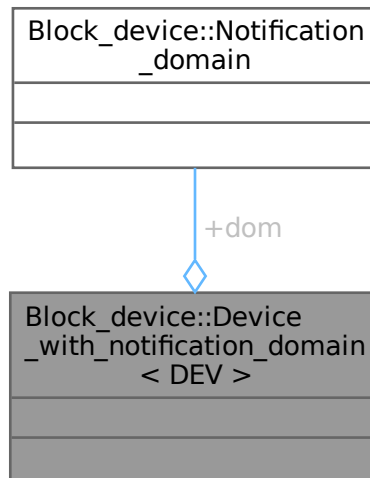
## 15.3 Block\_device::Device\_with\_notification\_domain< DEV > Struct Template Reference

Device with a per-device notification domain.

```
#include <device.h>
```

Inherits DEV.

Collaboration diagram for Block\_device::Device\_with\_notification\_domain< DEV >:



### 15.3.1 Detailed Description

```
template<typename DEV>
struct Block_device::Device_with_notification_domain< DEV >
```

Device with a per-device notification domain.

Definition at line 110 of file [device.h](#).

The documentation for this struct was generated from the following file:

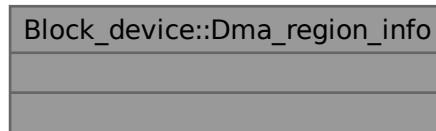
- `I4/libblock-device/device.h`

## 15.4 Block\_device::Dma\_region\_info Struct Reference

Base class used by the driver implementation to derive its own DMA mapping tracking structure.

```
#include <types.h>
```

Collaboration diagram for Block\_device::Dma\_region\_info:



### 15.4.1 Detailed Description

Base class used by the driver implementation to derive its own DMA mapping tracking structure.

Definition at line 44 of file [types.h](#).

The documentation for this struct was generated from the following file:

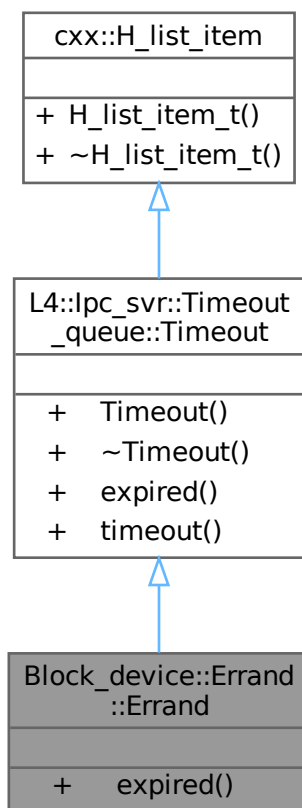
- l4/libblock-device/types.h

## 15.5 Block\_device::Errand::Errand Class Reference

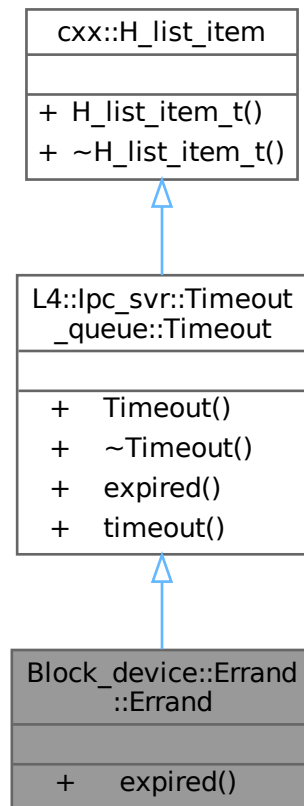
Wrapper for a small task executed asynchronously in the server loop.

```
#include <errand.h>
```

Inheritance diagram for Block\_device::Errand::Errand:



Collaboration diagram for `Block_device::Errand::Errand`:



### Public Member Functions

- void `expired()` final  
*callback function to be called when timeout happened*

### Public Member Functions inherited from `L4::lpc_svr::Timeout`

- `Timeout()`  
*Make a timeout.*
- virtual `~Timeout()`=0  
*Destroy a timeout.*
- `l4_kernel_clock_t timeout()` const  
*return absolute timeout of this callback.*

### Public Member Functions inherited from `cxx::H_list_item_t< ELEM_TYPE >`

- `H_list_item_t()`  
*Constructor.*
- `~H_list_item_t()` noexcept  
*Destructor.*

### 15.5.1 Detailed Description

Wrapper for a small task executed asynchronously in the server loop.

Errands are implemented as timeout tasks. They might be queued with the current timestamp, so that they are executed as soon as possible on the next iteration of the server loop or they might be scheduled with a timeout, which is particularly useful if the driver has to do a busy wait on the hardware.

Definition at line 108 of file [errand.h](#).

### 15.5.2 Member Function Documentation

#### 15.5.2.1 expired()

```
void Block_device::Errand::Errand::expired ( ) [inline], [final], [virtual]
```

callback function to be called when timeout happened

##### Note

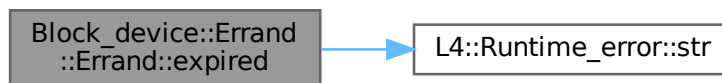
The timeout object is already dequeued when this function is called, this means the timeout may be safely queued again within the [expired\(\)](#) function.

Implements [L4::lpc\\_svr::Timeout](#).

Definition at line 113 of file [errand.h](#).

References [L4::Runtime\\_error::str\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

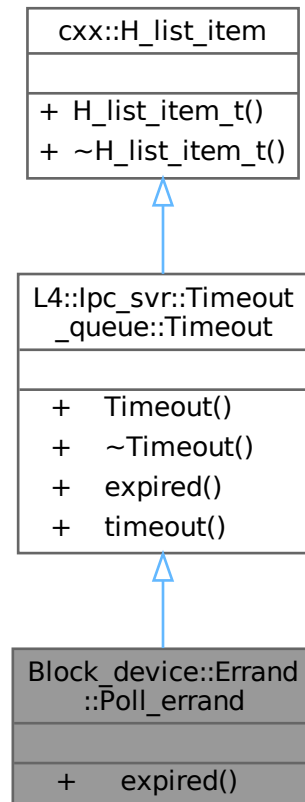
- `I4/libblock-device/errand.h`

## 15.6 Block\_device::Errand::Poll\_errand Class Reference

Wrapper for a regularly repeated task.

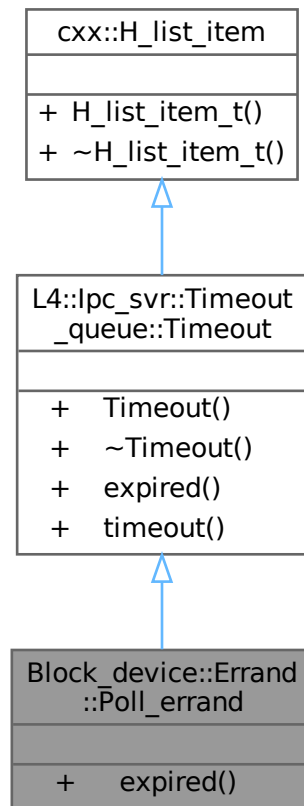
```
#include <errand.h>
```

Inheritance diagram for Block\_device::Errand::Poll\_errand:





Collaboration diagram for Block\_device::Errand::Poll\_errand:



### Public Member Functions

- void `expired()` final  
*callback function to be called when timeout happened*

### Public Member Functions inherited from `L4::lpc_svr::Timeout`

- `Timeout()`  
*Make a timeout.*
- virtual `~Timeout()`=0  
*Destroy a timeout.*
- `l4_kernel_clock_t timeout()` const  
*return absolute timeout of this callback.*

### Public Member Functions inherited from `cxx::H_list_item_t< ELEM_TYPE >`

- `H_list_item_t()`  
*Constructor.*
- `~H_list_item_t()` noexcept  
*Destructor.*

### 15.6.1 Detailed Description

Wrapper for a regularly repeated task.

Definition at line 42 of file [errand.h](#).

### 15.6.2 Member Function Documentation

#### 15.6.2.1 `expired()`

```
void Block_device::Errand::Poll_errand::expired ( ) [inline], [final], [virtual]
```

callback function to be called when timeout happened

##### Note

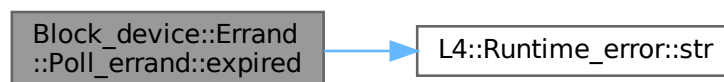
The timeout object is already dequeued when this function is called, this means the timeout may be safely queued again within the `expired()` function.

Implements [L4::lpc\\_svr::Timeout](#).

Definition at line 47 of file [errand.h](#).

References [L4::Runtime\\_error::str\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

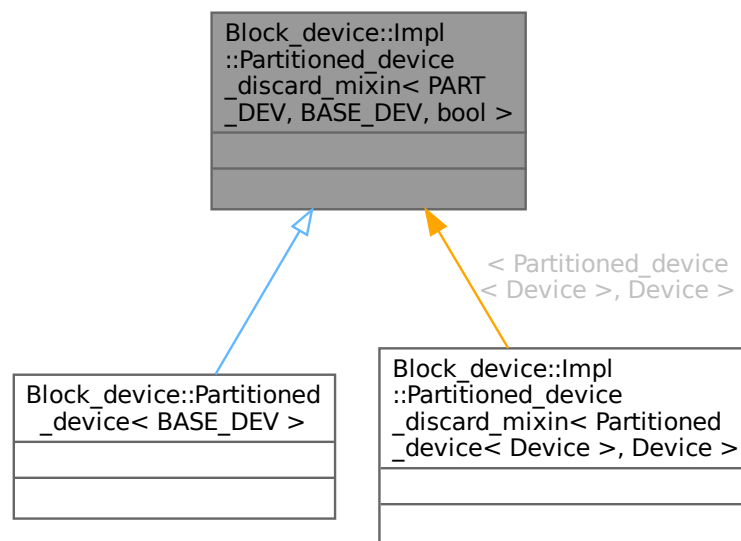
- `I4/libblock-device/errand.h`

## 15.7 Block\_device::Impl::Partitioned\_device\_discard\_mixin< PART\_DEV, BASE\_DEV, bool > Class Template Reference

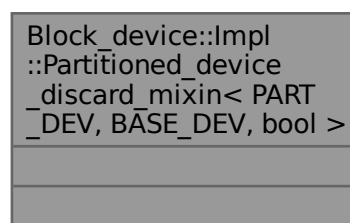
Dummy class used when the device class is not derived from [Device\\_discard\\_feature](#).

```
#include <part_device.h>
```

Inheritance diagram for Block\_device::Impl::Partitioned\_device\_discard\_mixin< PART\_DEV, BASE\_DEV, bool >:



Collaboration diagram for Block\_device::Impl::Partitioned\_device\_discard\_mixin< PART\_DEV, BASE\_DEV, bool >:



### 15.7.1 Detailed Description

```
template<typename PART_DEV, typename BASE_DEV, bool = std::is_base_of<Device_discard_feature,
BASE_DEV>::value>
class Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, bool >
```

Dummy class used when the device class is not derived from [Device\\_discard\\_feature](#).

Definition at line 29 of file [part\\_device.h](#).

The documentation for this class was generated from the following file:

- I4/libblock-device/part\_device.h

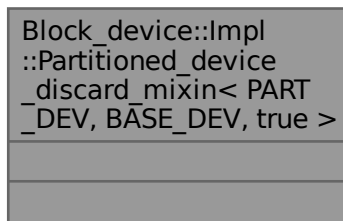
## 15.8 Block\_device::Impl::Partitioned\_device\_discard\_mixin< PART\_DEV, BASE\_DEV, true > Class Template Reference

Mixin implementing discard for partition devices.

```
#include <part_device.h>
```

Inherits [BASE\\_DEV](#).

Collaboration diagram for [Block\\_device::Impl::Partitioned\\_device\\_discard\\_mixin< PART\\_DEV, BASE\\_DEV, true >](#):



### 15.8.1 Detailed Description

```
template<typename PART_DEV, typename BASE_DEV>
class Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, true >
```

Mixin implementing discard for partition devices.

## Template Parameters

<i>PART_DEV</i>	Class of the partition device
<i>BASE_DEV</i>	Class implementing the Device interface.

Definition at line 38 of file [part\\_device.h](#).

The documentation for this class was generated from the following file:

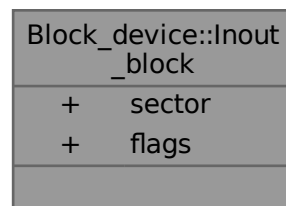
- l4/libblock-device/part\_device.h

## 15.9 Block\_device::Inout\_block Struct Reference

Description of an inout block to be sent to the device.

```
#include <types.h>
```

Collaboration diagram for Block\_device::Inout\_block:



### Data Fields

- [l4\\_uint64\\_t](#) **sector** = 0  
*Initial sector. Used only by DISCARD / WRITE\_ZEROES requests.*
- [l4\\_uint32\\_t](#) **flags** = 0  
*Flags from Inout\_flags.*

### 15.9.1 Detailed Description

Description of an inout block to be sent to the device.

Block may be scatter gather in which case they are chained via the next pointer.

Definition at line 67 of file [types.h](#).

The documentation for this struct was generated from the following file:

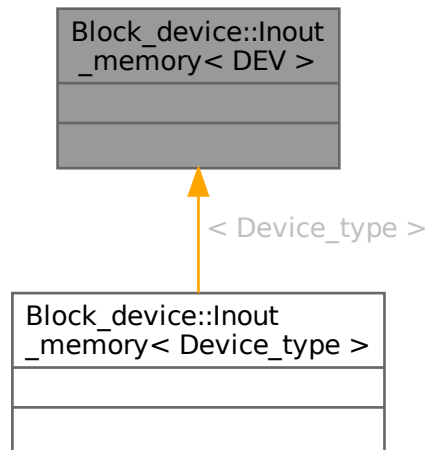
- l4/libblock-device/types.h

## 15.10 Block\_device::Inout\_memory< DEV > Class Template Reference

Helper class that temporarily allocates memory that can be used for in/out operations with the device.

```
#include <inout_memory.h>
```

Inheritance diagram for Block\_device::Inout\_memory< DEV >:



Collaboration diagram for Block\_device::Inout\_memory< DEV >:



### 15.10.1 Detailed Description

```
template<typename DEV>
class Block_device::Inout_memory< DEV >
```

Helper class that temporarily allocates memory that can be used for in/out operations with the device.

Definition at line 26 of file [inout\\_memory.h](#).

The documentation for this class was generated from the following file:

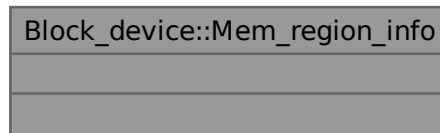
- `I4/libblock-device/inout_memory.h`

## 15.11 Block\_device::Mem\_region\_info Struct Reference

Additional info stored in each [L4virtio::Svr::Driver\\_mem\\_region\\_t](#) used for tracking dataspace-wide DMA mappings.

```
#include <types.h>
```

Collaboration diagram for Block\_device::Mem\_region\_info:



### 15.11.1 Detailed Description

Additional info stored in each [L4virtio::Svr::Driver\\_mem\\_region\\_t](#) used for tracking dataspace-wide DMA mappings.

Definition at line 53 of file [types.h](#).

The documentation for this struct was generated from the following file:

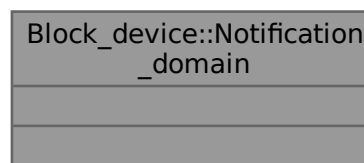
- `l4/libblock-device/types.h`

## 15.12 Block\_device::Notification\_domain Struct Reference

Opaque type for representing a notification domain.

```
#include <device.h>
```

Collaboration diagram for Block\_device::Notification\_domain:



### 15.12.1 Detailed Description

Opaque type for representing a notification domain.

Notification domains must be assigned to devices such that all devices that require a shared pool of resources to process their requests, also find themselves in the same notification domain. In particular, if two devices access common resources, then they must be in the same domain. An example of this are two partitions sharing the same parent device because processing of requests for one partition might depend on completion of request processing in another partition. On the other hand, independent disk devices will typically not share the same notification domain because their requests are completely independent of each other.

Definition at line 33 of file [device.h](#).

The documentation for this struct was generated from the following file:

- [l4/libblock-device/device.h](#)

## 15.13 Block\_device::Partition\_info Struct Reference

Information about a single partition.

```
#include <partition.h>
```

Collaboration diagram for Block\_device::Partition\_info:

Block_device::Partition_info	
+	guid
+	name
+	first
+	last
+	flags

### Data Fields

- char **guid** [37]  
*ID of the partition.*
- std::u16string **name**  
*UTF16 name of the partition.*
- [l4\\_uint64\\_t](#) **first**  
*First valid sector.*
- [l4\\_uint64\\_t](#) **last**  
*Last valid sector.*
- [l4\\_uint64\\_t](#) **flags**  
*Additional flags, depending on partition type.*



### 15.13.1 Detailed Description

Information about a single partition.

Definition at line 30 of file [partition.h](#).

The documentation for this struct was generated from the following file:

- I4/libblock-device/partition.h

## 15.14 Block\_device::Partition\_reader< DEV > Class Template Reference

Partition table reader for block devices.

```
#include <partition.h>
```

Inherits `cxx::Ref_obj`.

Collaboration diagram for Block\_device::Partition\_reader< DEV >:



### 15.14.1 Detailed Description

```
template<typename DEV>  
class Block_device::Partition_reader< DEV >
```

Partition table reader for block devices.

Definition at line 44 of file [partition.h](#).

The documentation for this class was generated from the following file:

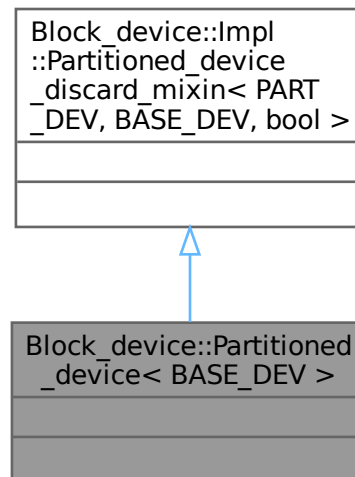
- I4/libblock-device/partition.h

## 15.15 Block\_device::Partitioned\_device< BASE\_DEV > Class Template Reference

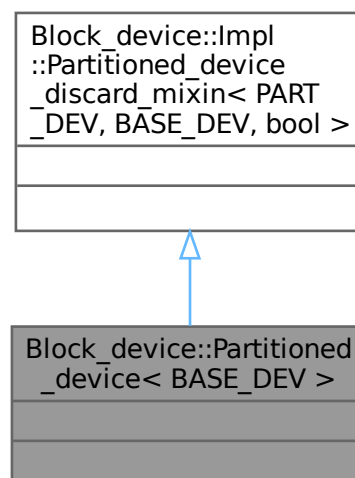
A partition device for the given device interface.

```
#include <part_device.h>
```

Inheritance diagram for Block\_device::Partitioned\_device< BASE\_DEV >:



Collaboration diagram for Block\_device::Partitioned\_device< BASE\_DEV >:



### 15.15.1 Detailed Description

```
template<typename BASE_DEV = Device>
class Block_device::Partitioned_device< BASE_DEV >
```

A partition device for the given device interface.

#### Template Parameters

<i>BASE_DEV</i>	Class defining the device interface. Attention: this is not the class implementing the device itself.
-----------------	---

Definition at line 92 of file [part\\_device.h](#).

The documentation for this class was generated from the following file:

- `l4/libblock-device/part_device.h`

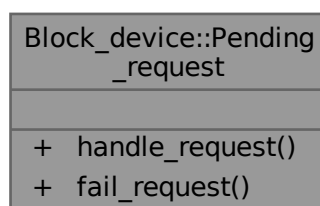
## 15.16 Block\_device::Pending\_request Struct Reference

Interface for pending requests.

```
#include <request.h>
```

Inherited by `Block_device::Virtio_client< DEV >::Generic_pending_request`.

Collaboration diagram for `Block_device::Pending_request`:



#### Public Member Functions

- virtual int [handle\\_request](#) ()=0  
*Callback used when the request is ready for processing.*
- virtual void [fail\\_request](#) ()=0  
*Callback used when a request is dropped from the queue.*

### 15.16.1 Detailed Description

Interface for pending requests.

Definition at line 15 of file [request.h](#).

### 15.16.2 Member Function Documentation

#### 15.16.2.1 fail\_request()

```
virtual void Block_device::Pending_request::fail_request ( ) [pure virtual]
```

Callback used when a request is dropped from the queue.

The function is called for notification only. The request will be destroyed.

#### 15.16.2.2 handle\_request()

```
virtual int Block_device::Pending_request::handle_request ( ) [pure virtual]
```

Callback used when the request is ready for processing.

Return values

<i>L4_EOK</i>	Request successfully issued. The callee has taken ownership of the request.
<i>-L4_EBUSY</i>	Device is still busy. The callee must not requeue the request as it will remain in the queue.
<	0 Other fatal error. The caller may dispose of the request.

The documentation for this struct was generated from the following file:

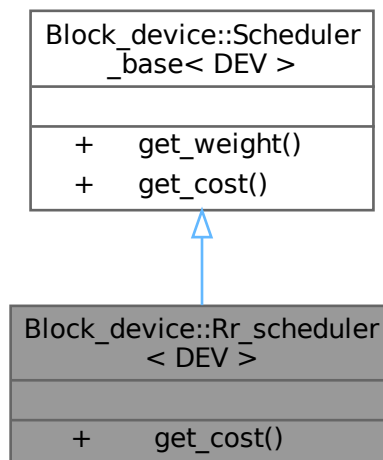
- [l4/libblock-device/request.h](#)

## 15.17 Block\_device::Rr\_scheduler< DEV > Struct Template Reference

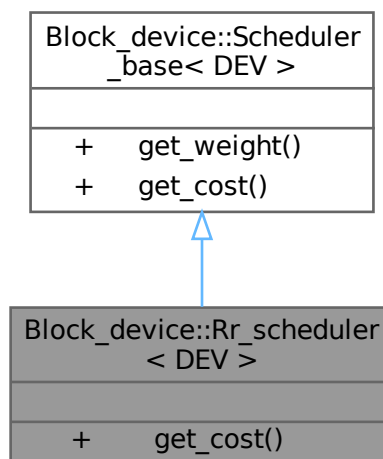
Round Robin scheduler class.

```
#include <scheduler.h>
```

Inheritance diagram for Block\_device::Rr\_scheduler< DEV >:



Collaboration diagram for Block\_device::Rr\_scheduler< DEV >:



### Public Member Functions

- `l4_size_t get_cost (Pending_request const &)` override  
*Return the cost of the pending request.*

## Public Member Functions inherited from [Block\\_device::Scheduler\\_base< DEV >](#)

- virtual [l4\\_size\\_t](#) **get\_weight** (Client\_type const \*)=0  
*Return the weight of the client.*

### 15.17.1 Detailed Description

```
template<typename DEV>
struct Block_device::Rr_scheduler< DEV >
```

Round Robin scheduler class.

All clients have fixed weight of 1 and all requests have fixed cost of 1, giving thus each client one scheduling chance per scheduling round.

Definition at line 340 of file [scheduler.h](#).

The documentation for this struct was generated from the following file:

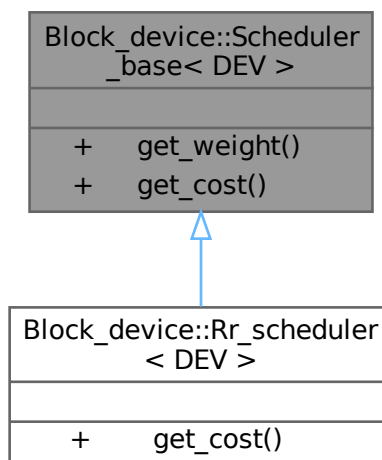
- [l4/libblock-device/scheduler.h](#)

## 15.18 [Block\\_device::Scheduler\\_base< DEV >](#) Class Template Reference

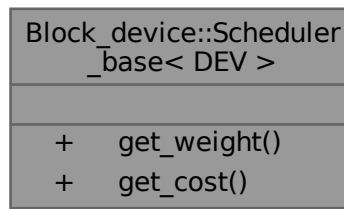
Scheduler base class.

```
#include <scheduler.h>
```

Inheritance diagram for [Block\\_device::Scheduler\\_base< DEV >](#):



Collaboration diagram for Block\_device::Scheduler\_base< DEV >:



### Public Member Functions

- virtual [l4\\_size\\_t](#) **get\_weight** (Client\_type const \*)=0  
*Return the weight of the client.*
- virtual [l4\\_size\\_t](#) **get\_cost** ([Pending\\_request](#) const &)=0  
*Return the cost of the pending request.*

### 15.18.1 Detailed Description

```
template<typename DEV>
class Block_device::Scheduler_base< DEV >
```

Scheduler base class.

Derive from this class and override [get\\_weight\(\)](#) and [get\\_cost\(\)](#) to implement the desired scheduling algorithm.

The interpretation of the weight function depends on the definition of the cost function. For example, if the cost of each request is fixed to be 1, the weight then says how many requests per scheduling round the client can process. If the weight of each client is also fixed to be 1, it will result in the Round Robin scheduler. If the request cost derives from the size of data the request operates on, the weight determines a data limit.

Definition at line [35](#) of file [scheduler.h](#).

The documentation for this class was generated from the following file:

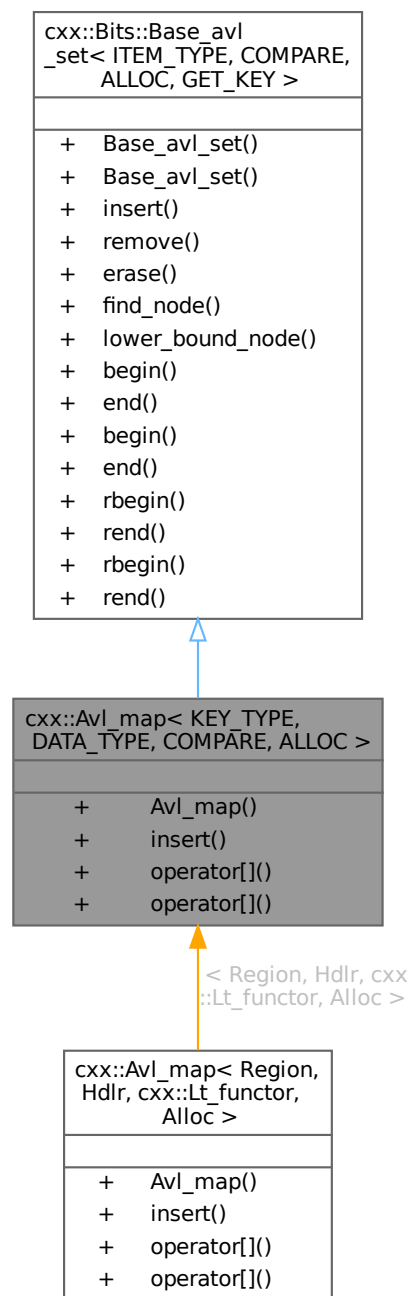
- [l4/libblock-device/scheduler.h](#)

## 15.19 cxx::Avl\_map< KEY\_TYPE, DATA\_TYPE, COMPARE, ALLOC > Class Template Reference

AVL tree based associative container.

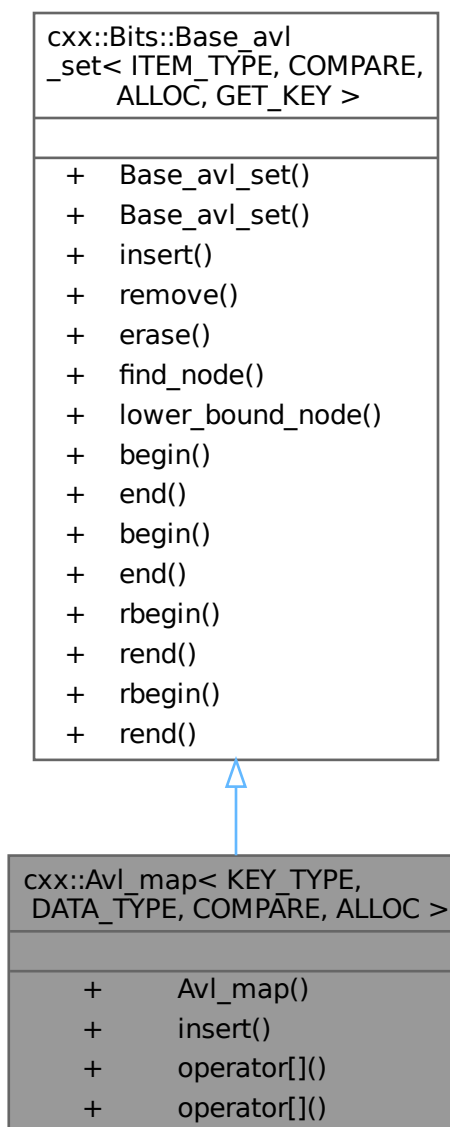
```
#include <avl_map>
```

Inheritance diagram for cxx::Avl\_map< KEY\_TYPE, DATA\_TYPE, COMPARE, ALLOC >:





Collaboration diagram for `cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >`:



## Public Types

- `typedef COMPARE< KEY_TYPE > Key_compare`  
*Type of the comparison functor.*
- `typedef KEY_TYPE Key_type`  
*Type of the key values.*
- `typedef DATA_TYPE Data_type`  
*Type of the data values.*
- `typedef Base_type::Node Node`  
*Return type for find.*
- `typedef Base_type::Node_allocator Node_allocator`  
*Type of the allocator.*

## Public Types inherited from

**cxx::Bits::Base\_avl\_set**< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >

- enum { E\_noent = 2 , E\_exist = 17 , E\_nomem = 12 , E\_inval = 22 }  
*Return status constants.*
- typedef ITEM\_TYPE **Item\_type**  
*Type for the items store in the set.*
- typedef GET\_KEY **Get\_key**  
*Key-getter type to derive the sort key of an internal node.*
- typedef GET\_KEY::Key\_type **Key\_type**  
*Type of the sort key used for the items.*
- typedef Type\_traits< Item\_type >::Const\_type **Const\_item\_type**  
*Type used for const items within the set.*
- typedef COMPARE **Item\_compare**  
*Type for the comparison functor.*
- typedef ALLOC< \_Node > **Node\_allocator**  
*Type for the node allocator.*
- typedef Avl\_set\_iter< \_Node, Item\_type, Fwd > **Iterator**  
*Forward iterator for the set.*
- typedef Avl\_set\_iter< \_Node, Const\_item\_type, Fwd > **Const\_iterator**  
*Constant forward iterator for the set.*
- typedef Avl\_set\_iter< \_Node, Item\_type, Rev > **Rev\_iterator**  
*Backward iterator for the set.*
- typedef Avl\_set\_iter< \_Node, Const\_item\_type, Rev > **Const\_rev\_iterator**  
*Constant backward iterator for the set.*

## Public Member Functions

- **Avl\_map** (Node\_allocator const &alloc=Node\_allocator())  
*Create an empty AVL-tree based map.*
- **cxx::Pair**< Iterator, int > **insert** (Key\_type const &key, Data\_type const &data)  
*Insert a <key, data> pair into the map.*
- Data\_type const & **operator[]** (Key\_type const &key) const  
*Get the data for the given key.*
- Data\_type & **operator[]** (Key\_type const &key)  
*Get or insert data for the given key.*

## Public Member Functions inherited from

**cxx::Bits::Base\_avl\_set**< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >

- **Base\_avl\_set** (Node\_allocator const &alloc=Node\_allocator())  
*Create a AVL-tree based set.*
- **Base\_avl\_set** (Base\_avl\_set const &o)  
*Create a copy of an AVL-tree based set.*
- **cxx::Pair**< Iterator, int > **insert** (Item\_type const &item)  
*Insert an item into the set.*
- int **remove** (Key\_type const &item)  
*Remove an item from the set.*
- int **erase** (Key\_type const &item)  
*Erase the item with the given key.*

- [Node find\\_node](#) ([Key\\_type](#) const &item) const  
*Lookup a node equal to `item`.*
- [Node lower\\_bound\\_node](#) ([Key\\_type](#) const &key) const  
*Find the first node greater or equal to `key`.*
- [Const\\_iterator begin](#) () const  
*Get the constant forward iterator for the first element in the set.*
- [Const\\_iterator end](#) () const  
*Get the end marker for the constant forward iterator.*
- [Iterator begin](#) ()  
*Get the mutable forward iterator for the first element of the set.*
- [Iterator end](#) ()  
*Get the end marker for the mutable forward iterator.*
- [Const\\_rev\\_iterator rbegin](#) () const  
*Get the constant backward iterator for the last element in the set.*
- [Const\\_rev\\_iterator rend](#) () const  
*Get the end marker for the constant backward iterator.*
- [Rev\\_iterator rbegin](#) ()  
*Get the mutable backward iterator for the last element of the set.*
- [Rev\\_iterator rend](#) ()  
*Get the end marker for the mutable backward iterator.*

### 15.19.1 Detailed Description

```
template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_↔
functor, template< typename B > class ALLOC = New_allocator>
class cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >
```

AVL tree based associative container.

#### Template Parameters

<i>KEY_TYPE</i>	Type of the key values.
<i>DATA_TYPE</i>	Type of the data values.
<i>COMPARE</i>	Type comparison functor for the key values.
<i>ALLOC</i>	Type of the allocator used for the nodes.

Definition at line 56 of file [avl\\_map](#).

### 15.19.2 Constructor & Destructor Documentation

#### 15.19.2.1 Avl\_map()

```
template<typename KEY_TYPE , typename DATA_TYPE , template< typename A > class COMPARE = Lt_↔
_functor, template< typename B > class ALLOC = New_allocator>
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::Avl_map (
    Node\_allocator const & alloc = Node\_allocator() ) [inline]
```

Create an empty AVL-tree based map.

## Parameters

<i>alloc</i>	The node allocator.
--------------	---------------------

Definition at line 91 of file [avl\\_map](#).

## 15.19.3 Member Function Documentation

### 15.19.3.1 insert()

```
template<typename KEY_TYPE , typename DATA_TYPE , template< typename A > class COMPARE = Lt↔
_func, template< typename B > class ALLOC = New_allocator>
cxx::Pair< Iterator, int > cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::insert (
    Key_type const & key,
    Data_type const & data ) [inline]
```

Insert a <key, data> pair into the map.

## Parameters

<i>key</i>	The key value.
<i>data</i>	The data value to insert.

## Returns

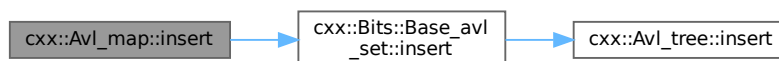
A pair of iterator (*first*) and return value (*second*). *second* will be 0 if the element was inserted into the set and `-#E_exist` if the key was already in the set and the set was therefore not updated. In both cases, *first* contains an iterator that points to the element. *second* may also be `-#E_nomem` when memory for the new node could not be allocated. *first* is then invalid.

Definition at line 110 of file [avl\\_map](#).

References [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >::insert\(\)](#).

Referenced by [cxx::Avl\\_map< KEY\\_TYPE, DATA\\_TYPE, COMPARE, ALLOC >::operator\[\]\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.19.3.2 `operator[]()` [1/2]

```

template<typename KEY_TYPE , typename DATA_TYPE , template< typename A > class COMPARE = Lt<
    _functor, template< typename B > class ALLOC = New_allocator>
Data_type & cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::operator[] (
    Key_type const & key ) [inline]
  
```

Get or insert data for the given key.

#### Parameters

<i>key</i>	The key value to use for lookup.
------------	----------------------------------

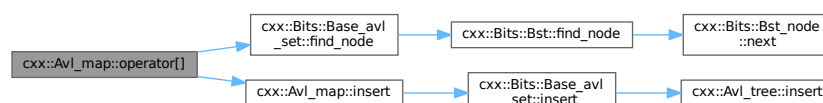
#### Returns

If the item already exists, a reference to the data item. Otherwise a new data item is default-constructed and inserted under the given key before a reference is returned.

Definition at line 134 of file `avl_map`.

References `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::find_node()`, and `cxx::Avl_map< KEY_TYPE,`

Here is the call graph for this function:



### 15.19.3.3 `operator[]()` [2/2]

```

template<typename KEY_TYPE , typename DATA_TYPE , template< typename A > class COMPARE = Lt<
    _functor, template< typename B > class ALLOC = New_allocator>
Data_type const & cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::operator[] (
    Key_type const & key ) const [inline]
  
```

Get the data for the given key.

**Parameters**

<i>key</i>	The key value to use for lookup.
------------	----------------------------------

**Precondition**

A `<key, data>` pair for the given key value must exist.

Definition at line 122 of file [avl\\_map](#).

References [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >::find\\_node\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

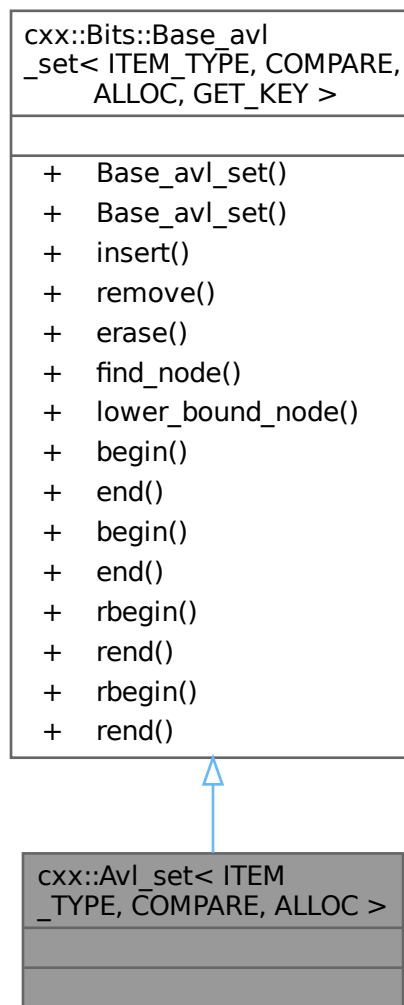
- [l4/cxx/avl\\_map](#)

## 15.20 `cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >` Class Template Reference

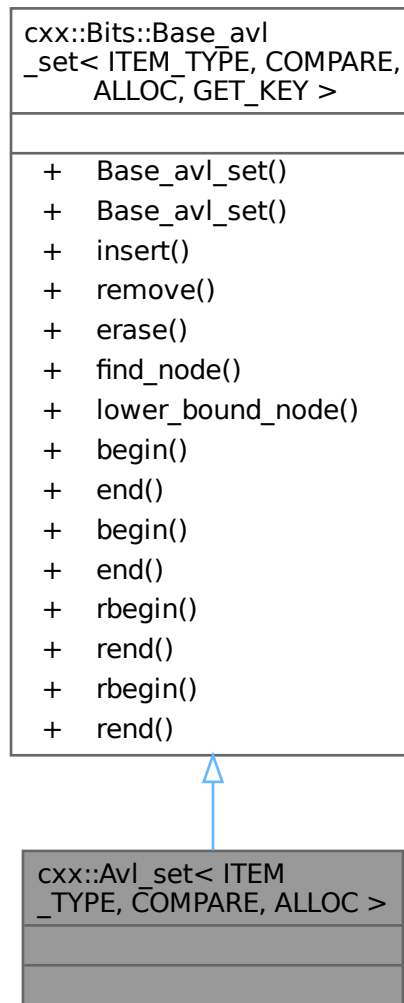
AVL set for simple compareable items.

```
#include <avl_set>
```

Inheritance diagram for cxx::Avl\_set< ITEM\_TYPE, COMPARE, ALLOC >:



Collaboration diagram for `cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >`:



#### Additional Inherited Members

#### Public Types inherited from

`cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >`

- enum { `E_noent` = 2 , `E_exist` = 17 , `E_nomem` = 12 , `E_inval` = 22 }

*Return status constants.*

- typedef `ITEM_TYPE` **Item\_type**

*Type for the items store in the set.*

- typedef `GET_KEY` **Get\_key**

*Key-getter type to derive the sort key of an internal node.*

- typedef `GET_KEY::Key_type` **Key\_type**

*Type of the sort key used for the items.*



- `typedef Type_traits< Item\_type >::Const_type Const_item_type`  
*Type used for const items within the set.*
- `typedef COMPARE Item_compare`  
*Type for the comparison functor.*
- `typedef ALLOC< _Node > Node_allocator`  
*Type for the node allocator.*
- `typedef Avl_set_iter< _Node, Item\_type, Fwd > Iterator`  
*Forward iterator for the set.*
- `typedef Avl_set_iter< _Node, Const\_item\_type, Fwd > Const_iterator`  
*Constant forward iterator for the set.*
- `typedef Avl_set_iter< _Node, Item\_type, Rev > Rev_iterator`  
*Backward iterator for the set.*
- `typedef Avl_set_iter< _Node, Const\_item\_type, Rev > Const_rev_iterator`  
*Constant backward iterator for the set.*

### Public Member Functions inherited from

#### `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >`

- `Base_avl_set (Node_allocator const &alloc=Node_allocator())`  
*Create a AVL-tree based set.*
- `Base_avl_set (Base_avl_set const &o)`  
*Create a copy of an AVL-tree based set.*
- `cxx::Pair< Iterator, int > insert (Item\_type const &item)`  
*Insert an item into the set.*
- `int remove (Key\_type const &item)`  
*Remove an item from the set.*
- `int erase (Key\_type const &item)`  
*Erase the item with the given key.*
- `Node find_node (Key\_type const &item) const`  
*Lookup a node equal to *item*.*
- `Node lower_bound_node (Key\_type const &key) const`  
*Find the first node greater or equal to *key*.*
- `Const_iterator begin () const`  
*Get the constant forward iterator for the first element in the set.*
- `Const_iterator end () const`  
*Get the end marker for the constant forward iterator.*
- `Iterator begin ()`  
*Get the mutable forward iterator for the first element of the set.*
- `Iterator end ()`  
*Get the end marker for the mutable forward iterator.*
- `Const_rev_iterator rbegin () const`  
*Get the constant backward iterator for the last element in the set.*
- `Const_rev_iterator rend () const`  
*Get the end marker for the constant backward iterator.*
- `Rev_iterator rbegin ()`  
*Get the mutable backward iterator for the last element of the set.*
- `Rev_iterator rend ()`  
*Get the end marker for the mutable backward iterator.*

### 15.20.1 Detailed Description

```
template<typename ITEM_TYPE, class COMPARE = Lt_functor<ITEM_TYPE>, template< typename A >
class ALLOC = New_allocator>
class cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >
```

AVL set for simple compareable items.

The AVL set can store any kind of items where a partial order is defined. The default relation is defined by the '<' operator.

#### Template Parameters

<i>ITEM_TYPE</i>	The type of the items to be stored in the set.
<i>COMPARE</i>	The relation to define the partial order, default is to use operator '<'.
<i>ALLOC</i>	The allocator to use for the nodes of the AVL set.

Definition at line [476](#) of file [avl\\_set](#).

The documentation for this class was generated from the following file:

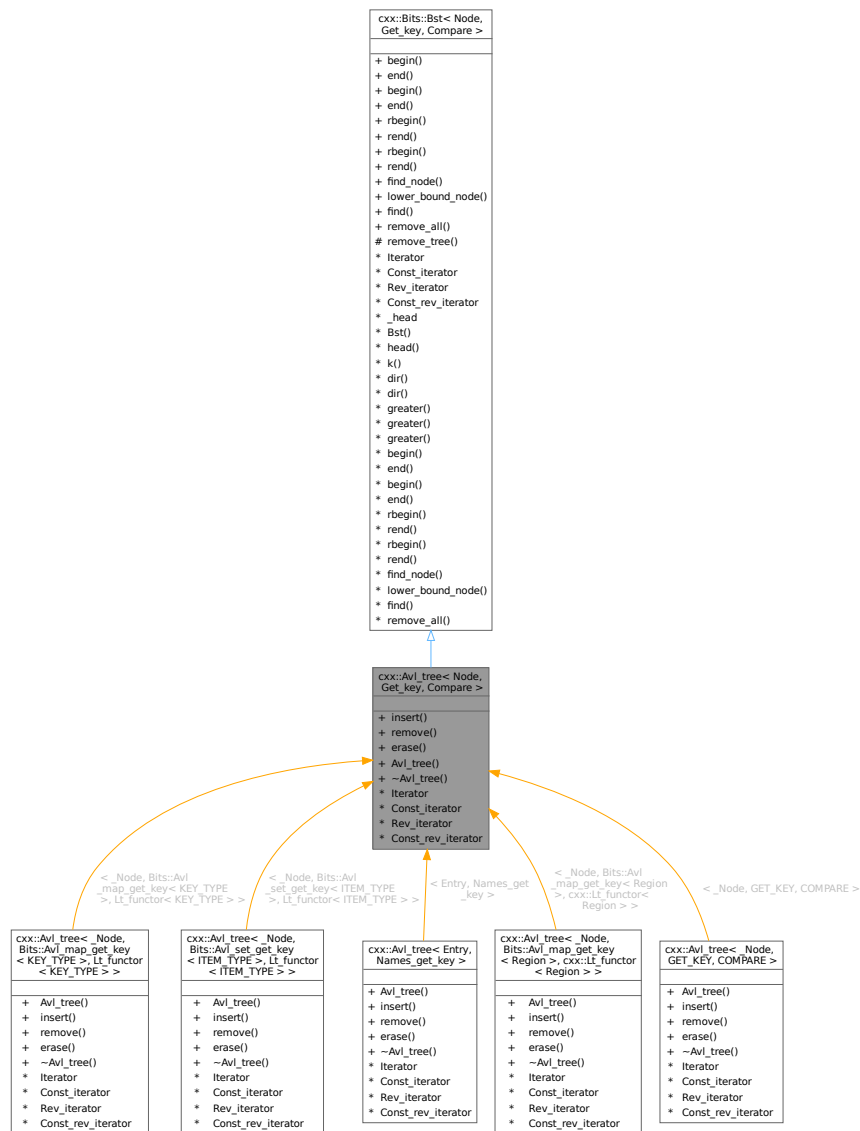
- [l4/cxx/avl\\_set](#)

## 15.21 cxx::Avl\_tree< Node, Get\_key, Compare > Class Template Reference

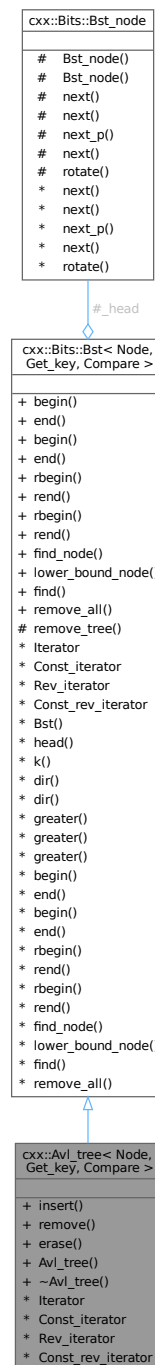
A generic AVL tree.

```
#include <avl_tree>
```

Inheritance diagram for cxx::Avl\_tree< Node, Get\_key, Compare >:



Collaboration diagram for `cxx::Avl_tree< Node, Get_key, Compare >`:



## Public Types inherited from `cxx::Bits::Bst< Node, Get_key, Compare >`

- `typedef Get_key::Key_type Key_type`  
*The type of key values used to generate the total order of the elements.*
- `typedef Type_traits< Key\_type >::Param_type Key_param_type`  
*The type for key parameters.*
- `typedef Fwd Fwd_iter_ops`

*Helper for building forward iterators for different wrapper classes.*

- typedef Rev **Rev\_iter\_ops**

*Helper for building reverse iterators for different wrapper classes.*

- typedef \_\_Bst\_iter< Node, Node, Fwd > **Iterator**

*Forward iterator.*

- typedef \_\_Bst\_iter< Node, Node const, Fwd > **Const\_iterator**

*Constant forward iterator.*

- typedef \_\_Bst\_iter< Node, Node, Rev > **Rev\_iterator**

*Backward iterator.*

- typedef \_\_Bst\_iter< Node, Node const, Rev > **Const\_rev\_iterator**

*Constant backward.*

### Public Member Functions

- **Pair**< Node \*, bool > **insert** (Node \*new\_node)

*Insert a new node into this AVL tree.*

- Node \* **remove** (Key\_param\_type key)

*Remove the node with key from the tree.*

- Node \* **erase** (Key\_param\_type key)

*An alias for [remove\(\)](#).*

- **Avl\_tree** ()=default

*Create an empty AVL tree.*

- ~**Avl\_tree** () noexcept

*Destroy the tree.*

### Public Member Functions inherited from [cxx::Bits::Bst](#)< Node, Get\_key, Compare >

- **Const\_iterator** **begin** () const

*Get the constant forward iterator for the first element in the set.*

- **Const\_iterator** **end** () const

*Get the end marker for the constant forward iterator.*

- **Iterator** **begin** ()

*Get the mutable forward iterator for the first element of the set.*

- **Iterator** **end** ()

*Get the end marker for the mutable forward iterator.*

- **Const\_rev\_iterator** **rbegin** () const

*Get the constant backward iterator for the last element in the set.*

- **Const\_rev\_iterator** **rend** () const

*Get the end marker for the constant backward iterator.*

- **Rev\_iterator** **rbegin** ()

*Get the mutable backward iterator for the last element of the set.*

- **Rev\_iterator** **rend** ()

*Get the end marker for the mutable backward iterator.*

- Node \* **find\_node** (Key\_param\_type key) const

*find the node with the given key.*

- Node \* **lower\_bound\_node** (Key\_param\_type key) const

*Find the first node with a key not less than the given key.*

- **Const\_iterator** **find** (Key\_param\_type key) const

*find the node with the given key.*

- template<typename FUNC >

void **remove\_all** (FUNC &&callback)

*Clear the tree.*

## Additional Inherited Members

### Protected Member Functions inherited from `cxx::Bits::Bst< Node, Get_key, Compare >`

- **Bst ()**  
*Create an empty tree.*
- **Node \* head () const**  
*Access the head node as object of type Node.*

### Static Protected Member Functions inherited from `cxx::Bits::Bst< Node, Get_key, Compare >`

- `template<typename FUNC >`  
`static void remove_tree (Bst_node *head, FUNC &&callback)`  
*Remove all elements in the subtree of head.*
- `static Key_type k (Bst_node const *n)`  
*Get the key value of n.*
- `static Dir dir (Key_param_type l, Key_param_type r)`  
*Get the direction to go from l to search for r.*
- `static Dir dir (Key_param_type l, Bst_node const *r)`  
*Get the direction to go from l to search for r.*
- `static bool greater (Key_param_type l, Key_param_type r)`  
*Is l greater than r.*
- `static bool greater (Key_param_type l, Bst_node const *r)`  
*Is l greater than r.*
- `static bool greater (Bst_node const *l, Bst_node const *r)`  
*Is l greater than r.*

### Protected Attributes inherited from `cxx::Bits::Bst< Node, Get_key, Compare >`

- `Bst_node * _head`  
*The head pointer of the tree.*

## 15.21.1 Detailed Description

```
template<typename Node, typename Get_key, typename Compare = Lt_functor<typename Get_key::Key↵
_type>>
class cxx::Avl_tree< Node, Get_key, Compare >
```

A generic AVL tree.

## Template Parameters

<i>Node</i>	The data type of the nodes (must inherit from <a href="#">Avl_tree_node</a> ).
<i>Get_key</i>	The meta function to get the key value from a node. The implementation uses <code>Get_key::key_of(ptr_to_node)</code> . The type of the key values must be defined in <code>Get_key::Key_type</code> .
<i>Compare</i>	Binary relation to establish a total order for the nodes of the tree. <code>Compare() (l, r)</code> must return true if the key <i>l</i> is smaller than the key <i>r</i> .

This implementation does not provide any memory management. It is the responsibility of the caller to allocate nodes before inserting them and to free them when they are removed or when the tree is destroyed. Conversely, the caller must also ensure that nodes are removed from the tree before they are destroyed.

## Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 111 of file [avl\\_tree](#).

## 15.21.2 Member Typedef Documentation

## 15.21.2.1 Iterator

```
template<typename Node , typename Get_key , typename Compare = Lt_functor<typename Get_key::Key_type>>
typedef Bst::Iterator cxx::Avl_tree< Node, Get_key, Compare >::Iterator
```

Forward iterator for the tree.

Definition at line 141 of file [avl\\_tree](#).

## 15.21.3 Member Function Documentation

15.21.3.1 `insert()`

```
template<typename Node , typename Get_key , class Compare >
Pair< Node *, bool > cxx::Avl_tree< Node, Get_key, Compare >::insert (
    Node * new_node )
```

Insert a new node into this AVL tree.

## Parameters

<i>new_node</i>	A pointer to the new node.
-----------------	----------------------------

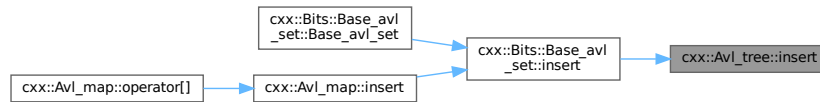
## Returns

A pair, with *second* set to `true` and *first* pointing to *new\_node*, on success. If there is already a node with the same key then *first* points to this node and *second* is 'false'.

Definition at line 231 of file [avl\\_tree](#).

Referenced by [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >::insert\(\)](#).

Here is the caller graph for this function:



### 15.21.3.2 remove()

```

template<typename Node , typename Get_key , class Compare >
Node * cxx::Avl\_tree< Node, Get_key, Compare >::remove (
    Key_param_type key ) [inline]
  
```

Remove the node with *key* from the tree.

#### Parameters

<i>key</i>	The key to the node to remove.
------------	--------------------------------

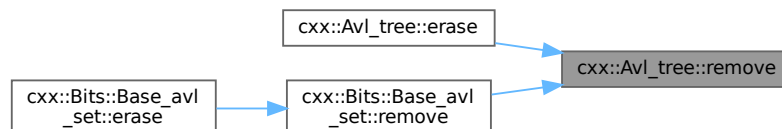
#### Returns

The pointer to the removed node on success, or 0 if no node with the *key* exists.

Definition at line 293 of file [avl\\_tree](#).

Referenced by [cxx::Avl\\_tree< Node, Get\\_key, Compare >::erase\(\)](#), and [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC,](#)

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- [I4/cxx/avl\\_tree](#)

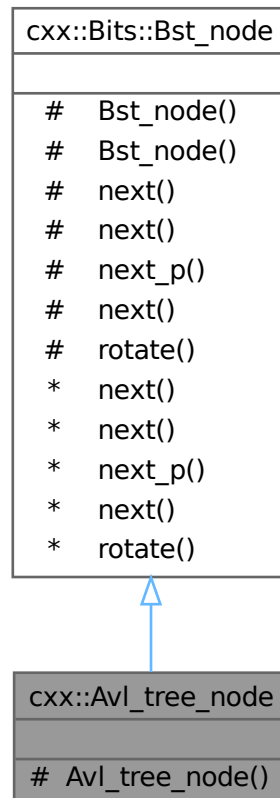


## 15.22 cxx::Avl\_tree\_node Class Reference

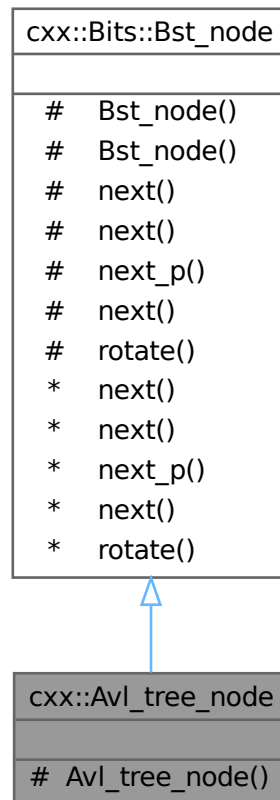
Node of an AVL tree.

```
#include <avl_tree>
```

Inheritance diagram for cxx::Avl\_tree\_node:



Collaboration diagram for `cxx::Avl_tree_node`:



### Protected Member Functions

- `Avl_tree_node ()`=default  
*Create an uninitialized node, this is what you should do.*

### Protected Member Functions inherited from `cxx::Bits::Bst_node`

- `Bst_node ()`  
*Create uninitialized node.*
- `Bst_node (bool)`  
*Create initialized node.*

### Additional Inherited Members

### Static Protected Member Functions inherited from `cxx::Bits::Bst_node`

- static `Bst_node * next (Bst_node const *p, Direction d)`

- Get next node in direction d.*
- static void **next** (Bst\_node \*p, Direction d, Bst\_node \*n)  
*Set next node of p in direction d to n.*
  - static Bst\_node \*\* **next\_p** (Bst\_node \*p, Direction d)  
*Get pointer to link in direction d.*
  - template<typename Node >  
static Node \* **next** (Bst\_node const \*p, Direction d)  
*Get next node in direction d as type Node.*
  - static void **rotate** (Bst\_node \*\*t, Direction idir)  
*Rotate subtree t in the opposite direction of idir.*

### 15.22.1 Detailed Description

Node of an AVL tree.

#### Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 40 of file [avl\\_tree](#).

The documentation for this class was generated from the following file:

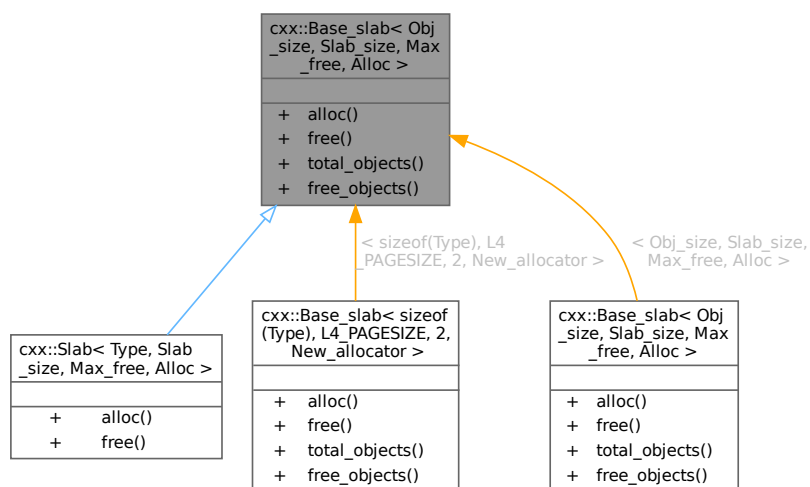
- [l4/cxx/avl\\_tree](#)

## 15.23 cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc > Class Template Reference

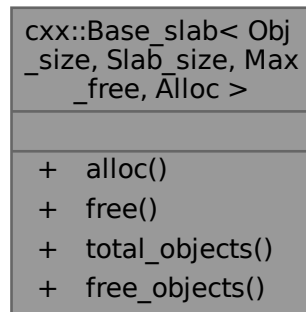
Basic slab allocator.

```
#include <slab_alloc>
```

Inheritance diagram for cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >:



Collaboration diagram for `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >`:



## Data Structures

- struct `Slab_i`  
*Type of a slab.*

## Public Types

- enum { `object_size` = `Obj_size` , `slab_size` = `Slab_size` , `objects_per_slab` = (`Slab_size` - `sizeof(Slab_head)`) / `object_size` , `max_free_slabs` = `Max_free` }
- typedef `Alloc< Slab_i >` **`Slab_alloc`**  
*Type of the backend allocator.*

## Public Member Functions

- void \* `alloc` () noexcept  
*Allocate a new object.*
- void `free` (void \* `_o`) noexcept  
*Free the given object (`_o`).*
- unsigned `total_objects` () const noexcept  
*Get the total number of objects managed by the slab allocator.*
- unsigned `free_objects` () const noexcept  
*Get the number of objects which can be allocated before a new empty slab needs to be added to the slab allocator.*

## 15.23.1 Detailed Description

```

template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc
= New_allocator>
class cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >

```

Basic slab allocator.

## Template Parameters

<i>Obj_size</i>	The size of the objects managed by the allocator (in bytes).
<i>Slab_size</i>	The size of a slab (in bytes).
<i>Max_free</i>	The maximum number of free slabs. When this limit is reached slabs are freed, provided that the backend allocator supports allocated memory to be freed.
<i>Alloc</i>	The backend allocator used to allocate slabs.

Definition at line 42 of file [slab\\_alloc](#).

## 15.23.2 Member Enumeration Documentation

### 15.23.2.1 anonymous enum

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
anonymous enum
```

## Enumerator

<code>object_size</code>	Size of an object.
<code>slab_size</code>	Size of a slab.
<code>objects_per_slab</code>	Objects per slab.
<code>max_free_slabs</code>	Maximum number of free slabs.

Definition at line 76 of file [slab\\_alloc](#).

## 15.23.3 Member Function Documentation

### 15.23.3.1 `alloc()`

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void * cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >::alloc \( \) [inline], [noexcept]
```

Allocate a new object.

## Returns

A pointer to the new object if the allocation succeeds, or 0 on failure to acquire memory from the backend allocator when the slab cache memory is already exhausted.

**Note**

The user is responsible for initializing the object.

Definition at line 218 of file [slab\\_alloc](#).

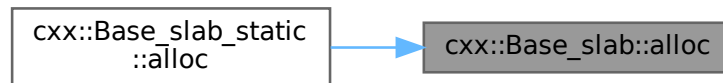
References [cxx::Base\\_slab< Obj\\_size, Slab\\_size, Max\\_free, Alloc >::free\(\)](#), [cxx::H\\_list< T, POLICY >::push\\_front\(\)](#), and [cxx::H\\_list< T, POLICY >::remove\(\)](#).

Referenced by [cxx::Base\\_slab\\_static< Obj\\_size, Slab\\_size, Max\\_free, Alloc >::alloc\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.23.3.2 free()**

```

template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::free (
    void * _o ) [inline], [noexcept]
  
```

Free the given object (`_o`).

**Precondition**

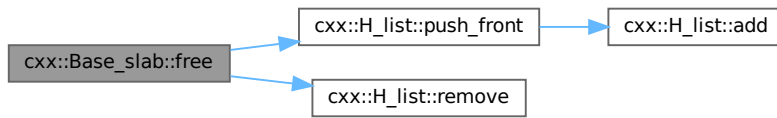
The object must have been allocated with this allocator.

Definition at line 257 of file [slab\\_alloc](#).

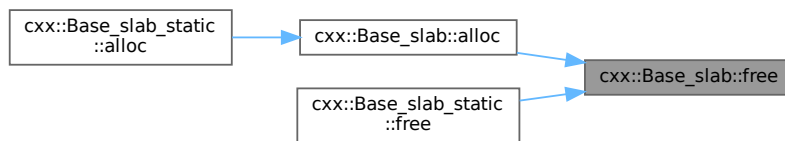
References [cxx::Base\\_slab< Obj\\_size, Slab\\_size, Max\\_free, Alloc >::max\\_free\\_slabs](#), [cxx::Base\\_slab< Obj\\_size, Slab\\_size, Max\\_free, Alloc >::free\(\)](#), [cxx::H\\_list< T, POLICY >::push\\_front\(\)](#), [cxx::H\\_list< T, POLICY >::remove\(\)](#), and [cxx::Base\\_slab< Obj\\_size, Slab\\_size, Max\\_free, Alloc >::alloc\(\)](#).

Referenced by [cxx::Base\\_slab< Obj\\_size, Slab\\_size, Max\\_free, Alloc >::alloc\(\)](#), and [cxx::Base\\_slab\\_static< Obj\\_size, Slab\\_size, Max\\_free, Alloc >::alloc\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.23.3.3 free\_objects()

```

template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::free_objects ( ) const [inline],
[noexcept]
  
```

Get the number of objects which can be allocated before a new empty slab needs to be added to the slab allocator.

#### Returns

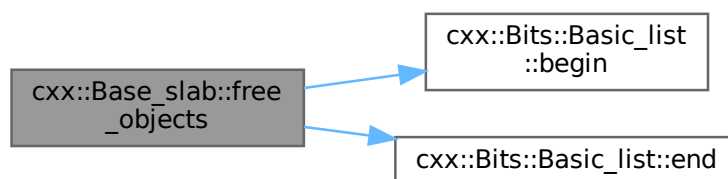
The number of free objects in the slab allocator.

Definition at line 319 of file [slab\\_alloc](#).

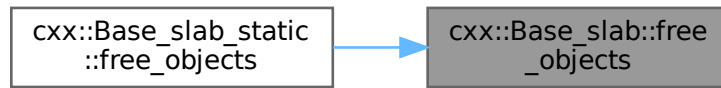
References [cxx::Bits::Basic\\_list< POLICY >::begin\(\)](#), [cxx::Bits::Basic\\_list< POLICY >::end\(\)](#), and [cxx::Base\\_slab< Obj\\_size, Slab\\_size, Max\\_free, Alloc >::free\\_objects\(\)](#).

Referenced by [cxx::Base\\_slab\\_static< Obj\\_size, Slab\\_size, Max\\_free, Alloc >::free\\_objects\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 15.23.3.4 total\_objects()

```

template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::total_objects ( ) const
[inline], [noexcept]
  
```

Get the total number of objects managed by the slab allocator.

##### Returns

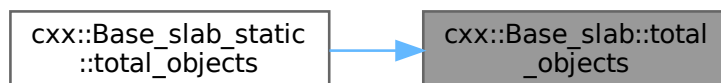
The number of objects managed by the allocator (including the free objects).

Definition at line 310 of file [slab\\_alloc](#).

References [cxx::Base\\_slab< Obj\\_size, Slab\\_size, Max\\_free, Alloc >::objects\\_per\\_slab](#).

Referenced by [cxx::Base\\_slab\\_static< Obj\\_size, Slab\\_size, Max\\_free, Alloc >::total\\_objects\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- `l4/cxx/slab_alloc`



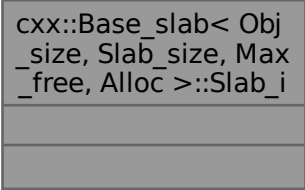
## 15.24 `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i` Struct Reference

Type of a slab.

```
#include <slab_alloc>
```

Inherits `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_store`, and `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_head`.

Collaboration diagram for `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i`:



```

classDiagram
    class Slab_i["cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i"]
    Slab_i --|> Slab_store["cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_store"]
    Slab_i --|> Slab_head["cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_head"]
  
```

### 15.24.1 Detailed Description

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc = New_allocator>
```

```
struct cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i
```

Type of a slab.

Definition at line 97 of file [slab\\_alloc](#).

The documentation for this struct was generated from the following file:

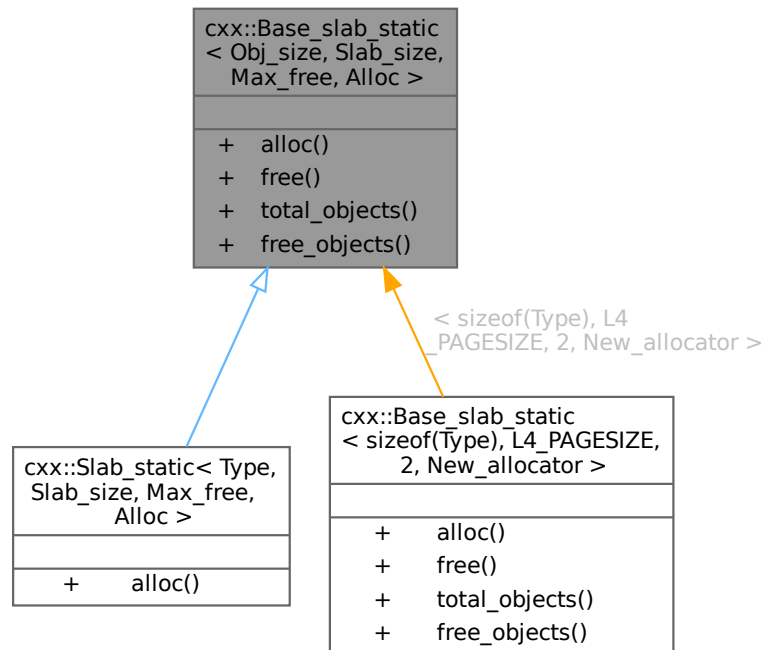
- `l4/cxx/slab_alloc`

## 15.25 `cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >` Class Template Reference

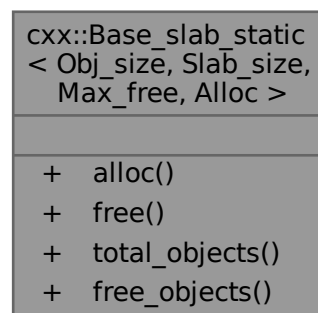
Merged slab allocator (allocators for objects of the same size are merged together).

```
#include <slab_alloc>
```

Inheritance diagram for `cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >`:



Collaboration diagram for `cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >`:



## Public Types

- enum { `object_size` = `Obj_size` , `slab_size` = `Slab_size` , `objects_per_slab` = `_A::objects_per_slab` , `max_free_slabs` = `Max_free` }

## Public Member Functions

- `void * alloc ()` noexcept  
*Allocate an object.*
- `void free (void *p)` noexcept  
*Free the given object (*p*).*
- `unsigned total\_objects ()` const noexcept  
*Get the total number of objects managed by the slab allocator.*
- `unsigned free\_objects ()` const noexcept  
*Get the number of free objects in the slab allocator.*

## 15.25.1 Detailed Description

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc
= New_allocator>
class cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >
```

Merged slab allocator (allocators for objects of the same size are merged together).

### Template Parameters

<i>Obj_size</i>	The size of an object managed by the slab allocator.
<i>Slab_size</i>	The size of a slab.
<i>Max_free</i>	The maximum number of free slabs.
<i>Alloc</i>	The allocator for the slabs.

This slab allocator class is useful for merging slab allocators with the same parameters (equal `Obj_size`, `Slab_size`, `Max_free`, and `Alloc` parameters) together and share the overhead for the slab caches among all equal-sized objects.

Definition at line 399 of file [slab\\_alloc](#).

## 15.25.2 Member Enumeration Documentation

### 15.25.2.1 anonymous enum

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
anonymous enum
```

### Enumerator

<code>object_size</code>	Size of an object.
<code>slab_size</code>	Size of a slab.
<code>objects_per_slab</code>	Number of objects per slab.
<code>max_free_slabs</code>	Maximum number of free slabs.

Definition at line 406 of file [slab\\_alloc](#).

## 15.25.3 Member Function Documentation

### 15.25.3.1 alloc()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void * cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >::alloc ( ) [inline],
[noexcept]
```

Allocate an object.

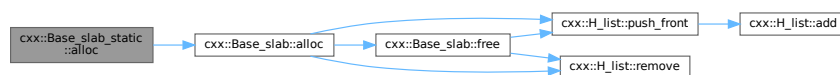
#### Note

The user is responsible for initializing the object.

Definition at line 423 of file [slab\\_alloc](#).

References [cxx::Base\\_slab< Obj\\_size, Slab\\_size, Max\\_free, Alloc >::alloc\(\)](#).

Here is the call graph for this function:



### 15.25.3.2 free()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >::free (
    void * p ) [inline], [noexcept]
```

Free the given object (p).

#### Parameters

<i>p</i>	The pointer to the object to free.
----------	------------------------------------

#### Precondition

*p* must be a pointer to an object allocated by this allocator.

Definition at line 431 of file [slab\\_alloc](#).

References [cxx::Base\\_slab< Obj\\_size, Slab\\_size, Max\\_free, Alloc >::free\(\)](#).

Here is the call graph for this function:



### 15.25.3.3 `free_objects()`

```

template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base\_slab\_static< Obj_size, Slab_size, Max_free, Alloc >::free_objects ( ) const
[inline], [noexcept]
  
```

Get the number of free objects in the slab allocator.

#### Returns

The number of free objects in all free and partially used slabs managed by this allocator.

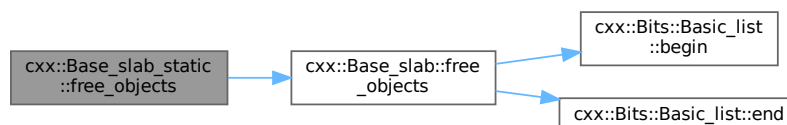
#### Note

The value is the merged value for all equal parameterized [Base\\_slab\\_static](#) instances.

Definition at line 451 of file [slab\\_alloc](#).

References [cxx::Base\\_slab< Obj\\_size, Slab\\_size, Max\\_free, Alloc >::free\\_objects\(\)](#).

Here is the call graph for this function:



### 15.25.3.4 total\_objects()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >::total_objects ( )
const [inline], [noexcept]
```

Get the total number of objects managed by the slab allocator.

#### Returns

The number of objects managed by the allocator (including the free objects).

#### Note

The value is the merged value for all equal parameterized [Base\\_slab\\_static](#) instances.

Definition at line 441 of file [slab\\_alloc](#).

References [cxx::Base\\_slab< Obj\\_size, Slab\\_size, Max\\_free, Alloc >::total\\_objects\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

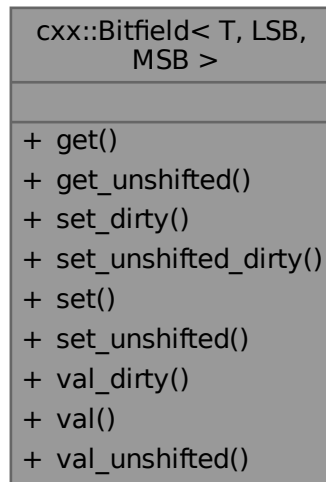
- [l4/cxx/slab\\_alloc](#)

## 15.26 cxx::Bitfield< T, LSB, MSB > Class Template Reference

Definition for a member (part) of a bit field.

```
#include <bitfield>
```

Collaboration diagram for cxx::Bitfield< T, LSB, MSB >:



## Data Structures

- class [Value](#)  
*Internal helper type.*
- class [Value\\_base](#)  
*Internal helper type.*
- class [Value\\_unshifted](#)  
*Internal helper type.*

## Public Types

- enum { [Bits](#) = MSB + 1 - LSB , [Lsb](#) = LSB , [Msb](#) = MSB }
- enum [Masks](#) : Base\_type { [Low\\_mask](#) = static\_cast<Base\_type>(~0ULL) >> (sizeof(Base\_type)\*8 - Bits) , [Mask](#) = Low\_mask << Lsb }
- *Masks for bitwise operation on internal parts of a bitfield.*
- typedef Best\_type< [Bits](#) >::Type [Bits\\_type](#)  
*Type to hold at least [Bits](#) bits.*
- typedef Best\_type< [Bits](#)+[Lsb](#) >::Type [Shift\\_type](#)  
*Type to hold at least [Bits](#) + [Lsb](#) bits.*
- typedef [Value](#)< Base\_type & > **Ref**  
*Reference type to access the bits inside a raw bit field.*
- typedef [Value](#)< Base\_type volatile & > **Ref\_volatile**  
*Volatile reference type to access the bits inside a raw bit field.*
- typedef [Value](#)< Base\_type const > **Val**  
*[Value](#) type to access the bits inside a raw bit field.*
- typedef [Value\\_unshifted](#)< Base\_type & > **Ref\_unshifted**  
*Reference type to access the bits inside a raw bit field (in place).*
- typedef [Value\\_unshifted](#)< Base\_type volatile & > **Ref\_unshifted\_volatile**  
*Volatile reference type to access the bits inside a raw bit field (in place).*
- typedef [Value\\_unshifted](#)< Base\_type const > **Val\_unshifted**  
*[Value](#) type to access the bits inside a raw bit field (in place).*

## Static Public Member Functions

- static constexpr [Bits\\_type](#) [get](#) ([Shift\\_type](#) val)  
*Get the bits out of val.*
- static constexpr [Base\\_type](#) [get\\_unshifted](#) ([Shift\\_type](#) val)  
*Get the bits in place out of val.*
- static constexpr [Base\\_type](#) [set\\_dirty](#) ([Base\\_type](#) dest, [Shift\\_type](#) val)  
*Set the bits corresponding to val.*
- static constexpr [Base\\_type](#) [set\\_unshifted\\_dirty](#) ([Base\\_type](#) dest, [Shift\\_type](#) val)  
*Set the bits corresponding to val.*
- static [Base\\_type](#) [set](#) ([Base\\_type](#) dest, [Bits\\_type](#) val)  
*Set the bits corresponding to val.*
- static [Base\\_type](#) [set\\_unshifted](#) ([Base\\_type](#) dest, [Shift\\_type](#) val)  
*Set the bits corresponding to val.*
- static constexpr [Base\\_type](#) [val\\_dirty](#) ([Shift\\_type](#) val)  
*Get the shifted bits for val.*
- static constexpr [Base\\_type](#) [val](#) ([Bits\\_type](#) val)  
*Get the shifted bits for val.*
- static constexpr [Base\\_type](#) [val\\_unshifted](#) ([Shift\\_type](#) val)  
*Get the shifted bits for val.*

### 15.26.1 Detailed Description

```
template<typename T, unsigned LSB, unsigned MSB>
class cxx::Bitfield< T, LSB, MSB >
```

Definition for a member (part) of a bit field.

#### Parameters

<i>T</i>	The underlying type of the bit field.
<i>LSB</i>	The least significant bit of our bits.
<i>MSB</i>	The most significant bit of our bits.

Definition at line 35 of file [bitfield](#).

### 15.26.2 Member Typedef Documentation

#### 15.26.2.1 Bits\_type

```
template<typename T , unsigned LSB, unsigned MSB>
typedef Best_type<Bits>::Type cxx::Bitfield< T, LSB, MSB >::Bits_type
```

Type to hold at least [Bits](#) bits.

This type can handle all values that can be stored in this part of the bit field.

Definition at line 85 of file [bitfield](#).



### 15.26.2.2 Shift\_type

```
template<typename T , unsigned LSB, unsigned MSB>
typedef Best_type<Bits+Lsb>::Type cxx::Bitfield< T, LSB, MSB >::Shift_type
```

Type to hold at least `Bits + Lsb` bits.

This type can handle all values that can be stored in this part of the bit field when they are at the target location (`Lsb` bits shifted to the left).

Definition at line 93 of file `bitfield`.

## 15.26.3 Member Enumeration Documentation

### 15.26.3.1 anonymous enum

```
template<typename T , unsigned LSB, unsigned MSB>
anonymous enum
```

#### Enumerator

Bits	Number of bits.
Lsb	index of the LSB
Msb	index of the MSB

Definition at line 63 of file `bitfield`.

### 15.26.3.2 Masks

```
template<typename T , unsigned LSB, unsigned MSB>
enum cxx::Bitfield::Masks : Base_type
```

Masks for bitwise operation on internal parts of a bitfield.

#### Enumerator

Low_mask	Mask value to get <code>Bits</code> bits.
Mask	Mask value to the bits out of a <code>T</code> .

Definition at line 71 of file `bitfield`.

## 15.26.4 Member Function Documentation

### 15.26.4.1 get()

```
template<typename T , unsigned LSB, unsigned MSB>
static constexpr Bits_type cxx::Bitfield< T, LSB, MSB >::get (
    Shift_type val ) [inline], [static], [constexpr]
```

Get the bits out of `val`.

## Parameters

<i>val</i>	The raw value of the whole bit field.
------------	---------------------------------------

## Returns

The bits form [Lsb](#) to [Msb](#) shifted to the right.

Definition at line 110 of file [bitfield](#).

References [cxx::Bitfield< T, LSB, MSB >::Low\\_mask](#), [cxx::Bitfield< T, LSB, MSB >::Lsb](#), and [cxx::Bitfield< T, LSB, MSB >::val\(\)](#).

Here is the call graph for this function:

15.26.4.2 `get_unshifted()`

```

template<typename T , unsigned LSB, unsigned MSB>
static constexpr Base_type cxx::Bitfield< T, LSB, MSB >::get_unshifted (
    Shift_type val ) [inline], [static], [constexpr]
  
```

Get the bits in place out of `val`.

## Parameters

<i>val</i>	The raw value of the whole bit field.
------------	---------------------------------------

## Returns

The bits from [Lsb](#) to [Msb](#) (unshifted).

This means other bits are masked out, however the result is not shifted to the right.

Definition at line 123 of file [bitfield](#).

References [cxx::Bitfield< T, LSB, MSB >::Mask](#), and [cxx::Bitfield< T, LSB, MSB >::val\(\)](#).

Here is the call graph for this function:



### 15.26.4.3 `set()`

```
template<typename T , unsigned LSB, unsigned MSB>
static Base_type cxx::Bitfield< T, LSB, MSB >::set (
    Base_type dest,
    Bits_type val ) [inline], [static]
```

Set the bits corresponding to `val`.

#### Parameters

<i>dest</i>	The current value of the whole bit field.
<i>val</i>	The value to set into the bits.

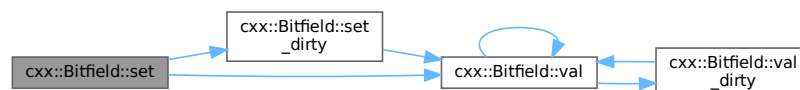
#### Returns

The new value of the whole bit field.

Definition at line 172 of file [bitfield](#).

References [cxx::Bitfield< T, LSB, MSB >::Low\\_mask](#), [cxx::Bitfield< T, LSB, MSB >::set\\_dirty\(\)](#), and [cxx::Bitfield< T, LSB, MSB >::val](#).

Here is the call graph for this function:



### 15.26.4.4 `set_dirty()`

```
template<typename T , unsigned LSB, unsigned MSB>
static constexpr Base_type cxx::Bitfield< T, LSB, MSB >::set_dirty (
    Base_type dest,
    Shift_type val ) [inline], [static], [constexpr]
```

Set the bits corresponding to `val`.

**Parameters**

<i>dest</i>	The current value of the whole bit field.
<i>val</i>	The value to set into the bits.

**Returns**

The new value of the whole bit field.

**Precondition**

`val` must not contain more than [Bits](#) bits.

**Note**

This function does not mask `val` to the right number of bits.

Definition at line [138](#) of file [bitfield](#).

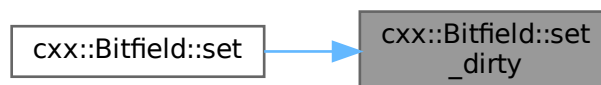
References [cxx::Bitfield< T, LSB, MSB >::Lsb](#), [cxx::Bitfield< T, LSB, MSB >::Mask](#), and [cxx::Bitfield< T, LSB, MSB >::val\(\)](#).

Referenced by [cxx::Bitfield< T, LSB, MSB >::set\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.26.4.5 set\_unshifted()**

```

template<typename T , unsigned LSB, unsigned MSB>
static Base_type cxx::Bitfield< T, LSB, MSB >::set_unshifted (
    Base_type dest,
    Shift_type val ) [inline], [static]
  
```

Set the bits corresponding to `val`.

## Parameters

<i>dest</i>	The current value of the whole bit field.
<i>val</i>	The value shifted <code>Lsb</code> bits to the left that shall be set into the bit field.

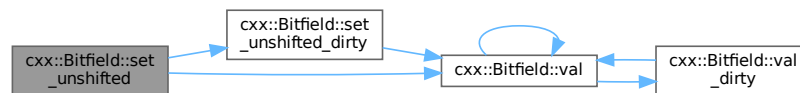
## Returns

the new value of the whole bit field.

Definition at line 184 of file `bitfield`.

References `cxx::Bitfield< T, LSB, MSB >::Mask`, `cxx::Bitfield< T, LSB, MSB >::set_unshifted_dirty()`, and `cxx::Bitfield< T, LSB, MSB >::val()`.

Here is the call graph for this function:

15.26.4.6 `set_unshifted_dirty()`

```

template<typename T , unsigned LSB, unsigned MSB>
static constexpr Base_type cxx::Bitfield< T, LSB, MSB >::set_unshifted_dirty (
    Base_type dest,
    Shift_type val ) [inline], [static], [constexpr]

```

Set the bits corresponding to `val`.

## Parameters

<i>dest</i>	The current value of the whole bit field.
<i>val</i>	The value shifted <code>Lsb</code> bits to the left that shall be set into the bits.

## Returns

The new value of the whole bit field.

## Precondition

`val` must not contain more than `Bits` bits shifted `Lsb` bits to the left.

**Note**

This function does not mask `val` to the right number of bits.

Definition at line 158 of file `bitfield`.

References `cxx::Bitfield< T, LSB, MSB >::Mask`, and `cxx::Bitfield< T, LSB, MSB >::val()`.

Referenced by `cxx::Bitfield< T, LSB, MSB >::set_unshifted()`.

Here is the call graph for this function:



Here is the caller graph for this function:

**15.26.4.7 val()**

```

template<typename T , unsigned LSB, unsigned MSB>
static constexpr Base_type cxx::Bitfield< T, LSB, MSB >::val (
    Bits_type val ) [inline], [static], [constexpr]
  
```

Get the shifted bits for `val`.

**Parameters**

<i>val</i>	The value to set into the bits.
------------	---------------------------------

**Returns**

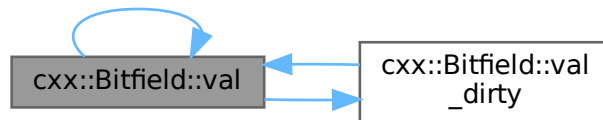
The raw bit field value.

Definition at line 207 of file `bitfield`.

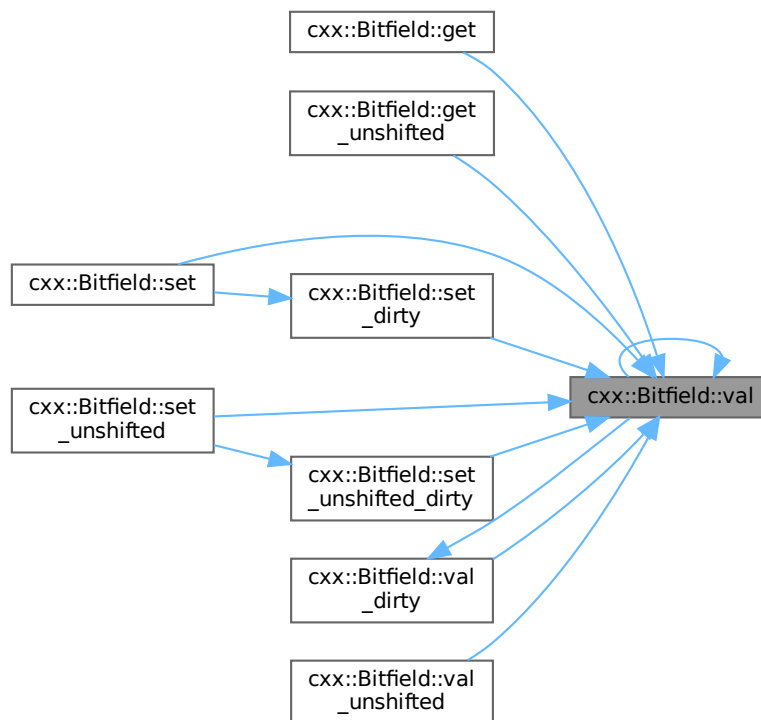
References [cxx::Bitfield< T, LSB, MSB >::Low\\_mask](#), [cxx::Bitfield< T, LSB, MSB >::val\(\)](#), and [cxx::Bitfield< T, LSB, MSB >::val\\_dirty](#)

Referenced by [cxx::Bitfield< T, LSB, MSB >::get\(\)](#), [cxx::Bitfield< T, LSB, MSB >::get\\_unshifted\(\)](#), [cxx::Bitfield< T, LSB, MSB >::set\(\)](#), [cxx::Bitfield< T, LSB, MSB >::set\\_dirty\(\)](#), [cxx::Bitfield< T, LSB, MSB >::set\\_unshifted\(\)](#), [cxx::Bitfield< T, LSB, MSB >::set\\_unshifted\\_dirty\(\)](#), [cxx::Bitfield< T, LSB, MSB >::val\(\)](#), [cxx::Bitfield< T, LSB, MSB >::val\\_dirty\(\)](#), and [cxx::Bitfield< T, LSB, MSB >::val\\_unshifted\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 15.26.4.8 val\_dirty()

```

template<typename T , unsigned LSB, unsigned MSB>
static constexpr Base_type cxx::Bitfield< T, LSB, MSB >::val_dirty (
    Shift_type val ) [inline], [static], [constexpr]
  
```

Get the shifted bits for `val`.



## Parameters

<code>val</code>	The value to set into the bits.
------------------	---------------------------------

## Returns

The raw bit field value.

## Precondition

`val` must not contain more than [Bits](#) bits.

## Note

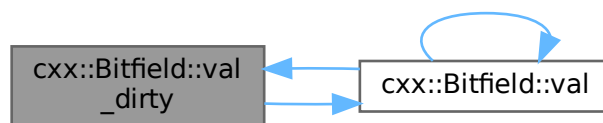
This function does not mask `val` to the right number of bits.

Definition at line 198 of file [bitfield](#).

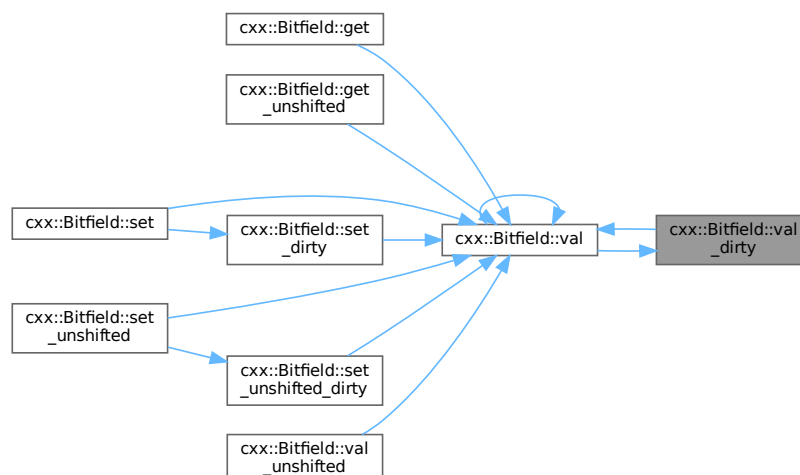
References [cxx::Bitfield< T, LSB, MSB >::Lsb](#), and [cxx::Bitfield< T, LSB, MSB >::val\(\)](#).

Referenced by [cxx::Bitfield< T, LSB, MSB >::val\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 15.26.4.9 `val_unshifted()`

```
template<typename T , unsigned LSB, unsigned MSB>
static constexpr Base_type cxx::Bitfield< T, LSB, MSB >::val_unshifted (
    Shift_type val ) [inline], [static], [constexpr]
```

Get the shifted bits for `val`.

##### Parameters

<code>val</code>	The value shifted <code>Lsb</code> bits to the left that shall be set into the bits.
------------------	--

##### Returns

The raw bit field value.

Definition at line 217 of file [bitfield](#).

References [cxx::Bitfield< T, LSB, MSB >::Mask](#), and [cxx::Bitfield< T, LSB, MSB >::val\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

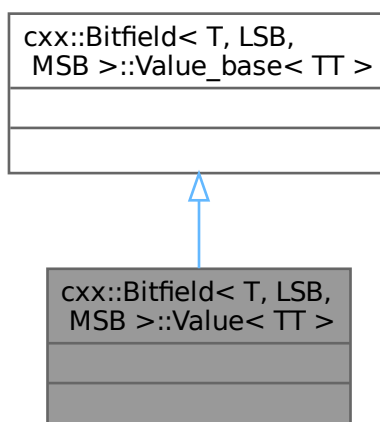
- `I4/cxx/bitfield`

## 15.27 `cxx::Bitfield< T, LSB, MSB >::Value< TT >` Class Template Reference

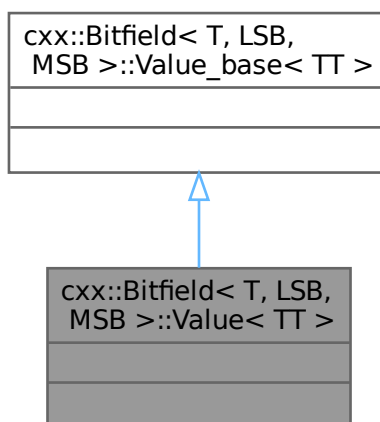
Internal helper type.

```
#include <bitfield>
```

Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >::Value< TT >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value< TT >`:



### 15.27.1 Detailed Description

```

template<typename T, unsigned LSB, unsigned MSB>
template<typename TT>
class cxx::Bitfield< T, LSB, MSB >::Value< TT >

```

Internal helper type.

Definition at line [239](#) of file [bitfield](#).

The documentation for this class was generated from the following file:

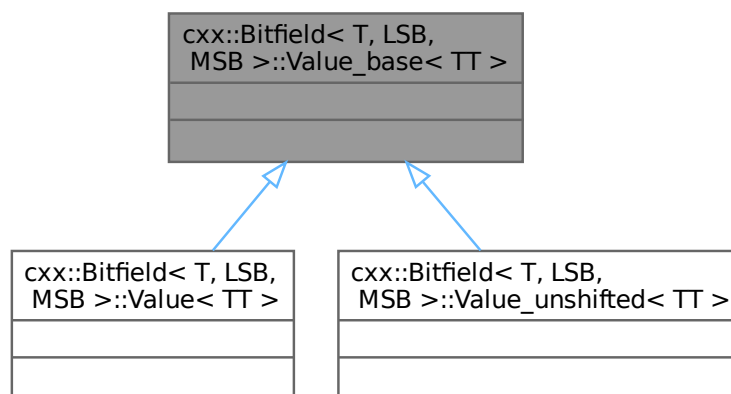
- l4/cxx/bitfield

## 15.28 `cxx::Bitfield< T, LSB, MSB >::Value_base< TT >` Class Template Reference

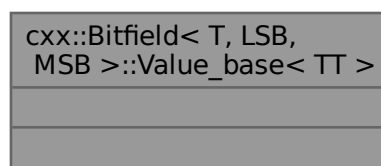
Internal helper type.

```
#include <bitfield>
```

Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >::Value_base< TT >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value_base< TT >`:



### 15.28.1 Detailed Description

```
template<typename T, unsigned LSB, unsigned MSB>
template<typename TT>
class cxx::Bitfield< T, LSB, MSB >::Value_base< TT >
```

Internal helper type.

Definition at line 221 of file [bitfield](#).

The documentation for this class was generated from the following file:

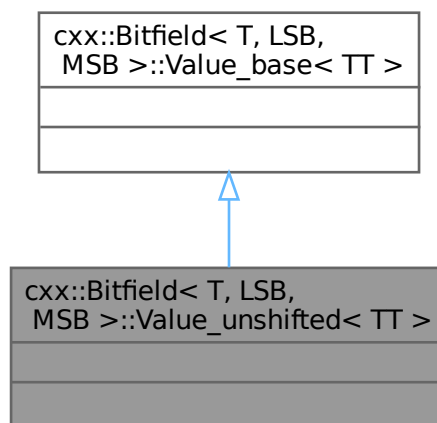
- `I4/cxx/bitfield`

## 15.29 `cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >` Class Template Reference

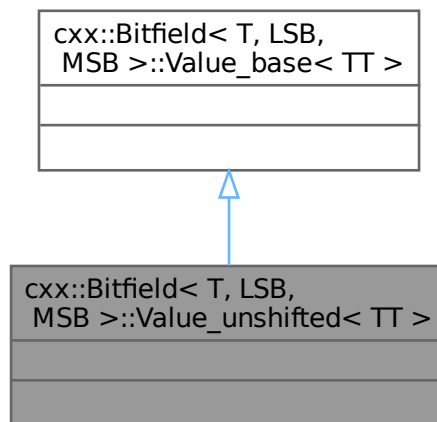
Internal helper type.

```
#include <bitfield>
```

Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >`:



### 15.29.1 Detailed Description

```

template<typename T, unsigned LSB, unsigned MSB>
template<typename TT>
class cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >

```

Internal helper type.

Definition at line 252 of file [bitfield](#).

The documentation for this class was generated from the following file:

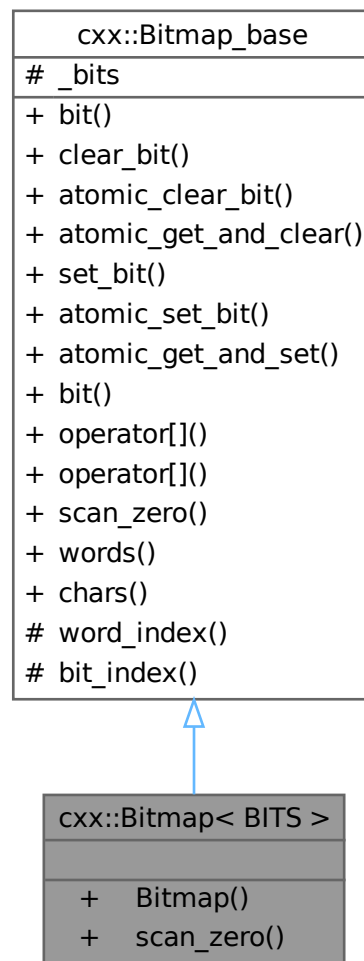
- `I4/cxx/bitfield`

## 15.30 `cxx::Bitmap< BITS >` Class Template Reference

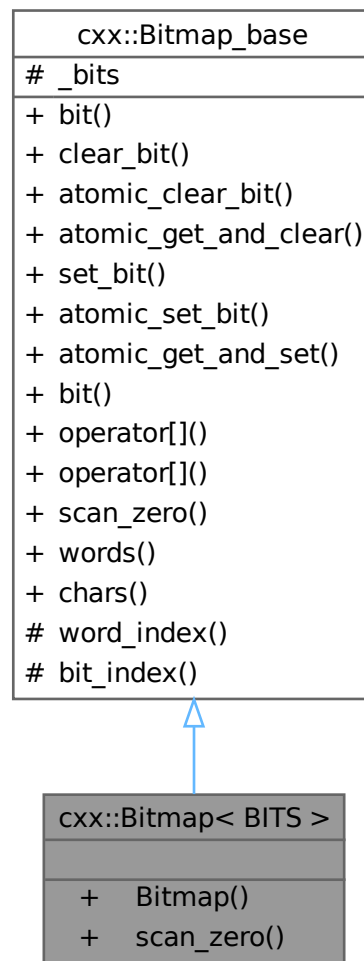
A static bitmap.

```
#include <bitmap>
```

Inheritance diagram for cxx::Bitmap< BITS >:



Collaboration diagram for `cxx::Bitmap< BITS >`:



## Public Member Functions

- **Bitmap** () noexcept  
*Create a bitmap with `BITS` bits.*
- long `scan_zero` (long start\_bit=0) const noexcept  
*Scan for the first zero bit.*

## Public Member Functions inherited from `cxx::Bitmap_base`

- void `bit` (long bit, bool on) noexcept  
*Set the value of bit `bit` to on.*
- void `clear_bit` (long bit) noexcept  
*Clear bit `bit`.*
- void `atomic_clear_bit` (long bit) noexcept



- Clear bit `bit` atomically.*
- `word_type atomic_get_and_clear` (long `bit`) noexcept  
*Clear bit `bit` atomically and return old state.*
- void `set_bit` (long `bit`) noexcept  
*Set bit `bit`.*
- void `atomic_set_bit` (long `bit`) noexcept  
*Set bit `bit` atomically.*
- `word_type atomic_get_and_set` (long `bit`) noexcept  
*Set bit `bit` atomically and return old state.*
- `word_type bit` (long `bit`) const noexcept  
*Get the truth value of a bit.*
- `word_type operator[]` (long `bit`) const noexcept  
*Get the bit at index `bit`.*
- `Bit operator[]` (long `bit`) noexcept  
*Get the lvalue for the bit at index `bit`.*
- long `scan_zero` (long `max_bit`, long `start_bit=0`) const noexcept  
*Scan for the first zero bit.*

### Additional Inherited Members

### Static Public Member Functions inherited from `cxx::Bitmap_base`

- static long `words` (long bits) noexcept  
*Get the number of `words` that are used for the bitmap.*
- static long `chars` (long bits) noexcept  
*Get the number of chars that are used for the bitmap.*

### Protected Types inherited from `cxx::Bitmap_base`

- enum { `W_bits` = sizeof(`word_type`) \* 8 , `C_bits` = 8 }
- typedef unsigned long `word_type`  
*Data type for each element of the bit buffer.*

### Static Protected Member Functions inherited from `cxx::Bitmap_base`

- static unsigned `word_index` (unsigned `bit`)  
*Get the word index for the given bit.*
- static unsigned `bit_index` (unsigned `bit`)  
*Get the bit index within `word_type` for the given bit.*

### Protected Attributes inherited from `cxx::Bitmap_base`

- `word_type * _bits`  
*Pointer to the buffer storing the bits.*

## 15.30.1 Detailed Description

```
template<int BITS>
class cxx::Bitmap< BITS >
```

A static bitmap.

## Template Parameters

<i>BITS</i>	The number of bits that shall be in the bitmap.
-------------	---

Definition at line 231 of file [bitmap](#).

## 15.30.2 Member Function Documentation

### 15.30.2.1 `scan_zero()`

```
template<int BITS>
long cxx::Bitmap< BITS >::scan_zero (
    long start_bit = 0 ) const [inline], [noexcept]
```

Scan for the first zero bit.

## Parameters

<i>start_bit</i>	Hint at the number of the first bit to look at. Zero bits below <i>start_bit</i> may or may not be taken into account by the implementation.
------------------	--

## Return values

<code>&gt;=</code>	0 Number of first zero bit found.
<code>-1</code>	All bits at <i>start_bit</i> or higher are set.

Compared to [Bitmap\\_base::scan\\_zero\(\)](#), the upper bound is set to BITS.

Definition at line 376 of file [bitmap](#).

References [cxx::Bitmap\\_base::scan\\_zero\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

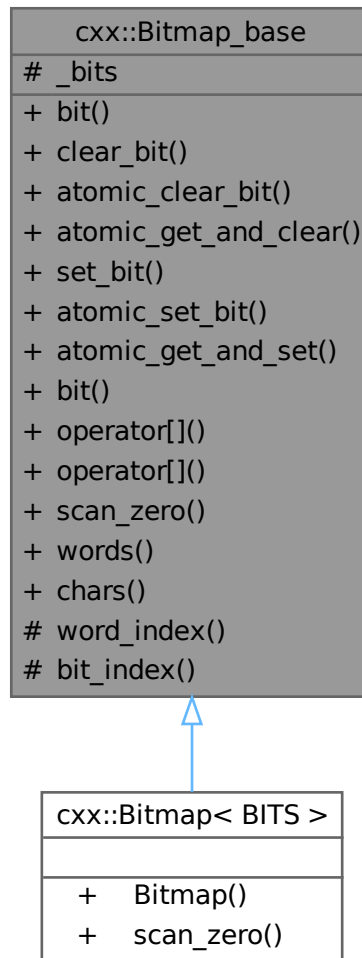
- `I4/cxx/bitmap`

## 15.31 cxx::Bitmap\_base Class Reference

Basic bitmap abstraction.

```
#include <bitmap>
```

Inheritance diagram for cxx::Bitmap\_base:



Collaboration diagram for `cxx::Bitmap_base`:

<code>cxx::Bitmap_base</code>
<code># _bits</code>
<code>+ bit()</code>
<code>+ clear_bit()</code>
<code>+ atomic_clear_bit()</code>
<code>+ atomic_get_and_clear()</code>
<code>+ set_bit()</code>
<code>+ atomic_set_bit()</code>
<code>+ atomic_get_and_set()</code>
<code>+ bit()</code>
<code>+ operator[]()</code>
<code>+ operator[]()</code>
<code>+ scan_zero()</code>
<code>+ words()</code>
<code>+ chars()</code>
<code># word_index()</code>
<code># bit_index()</code>

## Data Structures

- class [Bit](#)  
*A writeable bit in a bitmap.*
- class [Char](#)  
*Helper abstraction for a byte contained in the bitmap.*
- class [Word](#)  
*Helper abstraction for a word contained in the bitmap.*

## Public Member Functions

- void [bit](#) (long [bit](#), bool [on](#)) noexcept  
*Set the value of bit [bit](#) to [on](#).*
- void [clear\\_bit](#) (long [bit](#)) noexcept  
*Clear bit [bit](#).*
- void [atomic\\_clear\\_bit](#) (long [bit](#)) noexcept  
*Clear bit [bit](#) atomically.*
- [word\\_type](#) [atomic\\_get\\_and\\_clear](#) (long [bit](#)) noexcept  
*Clear bit [bit](#) atomically and return old state.*
- void [set\\_bit](#) (long [bit](#)) noexcept  
*Set bit [bit](#).*

- void `atomic_set_bit` (long `bit`) noexcept  
*Set bit `bit` atomically.*
- `word_type` `atomic_get_and_set` (long `bit`) noexcept  
*Set bit `bit` atomically and return old state.*
- `word_type` `bit` (long `bit`) const noexcept  
*Get the truth value of a bit.*
- `word_type` `operator[]` (long `bit`) const noexcept  
*Get the bit at index `bit`.*
- `Bit operator[]` (long `bit`) noexcept  
*Get the lvalue for the bit at index `bit`.*
- long `scan_zero` (long `max_bit`, long `start_bit`=0) const noexcept  
*Scan for the first zero bit.*

### Static Public Member Functions

- static long `words` (long `bits`) noexcept  
*Get the number of words that are used for the bitmap.*
- static long `chars` (long `bits`) noexcept  
*Get the number of chars that are used for the bitmap.*

### Protected Types

- enum { `W_bits` = sizeof(`word_type`) \* 8 , `C_bits` = 8 }
- typedef unsigned long `word_type`  
*Data type for each element of the bit buffer.*

### Static Protected Member Functions

- static unsigned `word_index` (unsigned `bit`)  
*Get the word index for the given bit.*
- static unsigned `bit_index` (unsigned `bit`)  
*Get the bit index within `word_type` for the given bit.*

### Protected Attributes

- `word_type` \* `_bits`  
*Pointer to the buffer storing the bits.*

## 15.31.1 Detailed Description

Basic bitmap abstraction.

This abstraction keeps a pointer to a memory area that is used as bitmap.

Definition at line 29 of file `bitmap`.

## 15.31.2 Member Enumeration Documentation

### 15.31.2.1 anonymous enum

```
anonymous enum [protected]
```

## Enumerator

W_bits	number of bits in word_type
C_bits	number of bits in char

Definition at line 37 of file [bitmap](#).

### 15.31.3 Member Function Documentation

#### 15.31.3.1 atomic\_clear\_bit()

```
void cxx::Bitmap_base::atomic_clear_bit (  
    long bit ) [inline], [noexcept]
```

Clear bit `bit` atomically.

Use this function for multi-threaded access to the bitmap.

## Parameters

<i>bit</i>	The number of the bit to clear.
------------	---------------------------------

Definition at line 280 of file [bitmap](#).

#### 15.31.3.2 atomic\_get\_and\_clear()

```
Bitmap_base::word_type cxx::Bitmap_base::atomic_get_and_clear (  
    long bit ) [inline], [noexcept]
```

Clear bit `bit` atomically and return old state.

Use this function for multi-threaded access to the bitmap.

## Parameters

<i>bit</i>	The number of the bit to clear.
------------	---------------------------------

Definition at line 290 of file [bitmap](#).

#### 15.31.3.3 atomic\_get\_and\_set()

```
Bitmap_base::word_type cxx::Bitmap_base::atomic_get_and_set (  
    long bit ) [inline], [noexcept]
```

Set bit `bit` atomically and return old state.

Use this function for multi-threaded access to the bitmap.

## Parameters

<i>bit</i>	The number of the bit to set.
------------	-------------------------------

Definition at line 319 of file [bitmap](#).

**15.31.3.4 `atomic_set_bit()`**

```
void cxx::Bitmap_base::atomic_set_bit (
    long bit ) [inline], [noexcept]
```

Set bit `bit` atomically.

Use this function for multi-threaded access to the bitmap.

## Parameters

<i>bit</i>	The number of the bit to set.
------------	-------------------------------

Definition at line 309 of file [bitmap](#).

**15.31.3.5 `bit()` [1/2]**

```
Bitmap_base::word_type cxx::Bitmap_base::bit (
    long bit ) const [inline], [noexcept]
```

Get the truth value of a bit.

## Parameters

<i>bit</i>	The number of the bit to read.
------------	--------------------------------

## Return values

0	Bit is not set.
!= 0	Bit is set.

Definition at line 329 of file [bitmap](#).

**15.31.3.6 `bit()` [2/2]**

```
void cxx::Bitmap_base::bit (
    long bit,
    bool on ) [inline], [noexcept]
```

Set the value of bit `bit` to `on`.

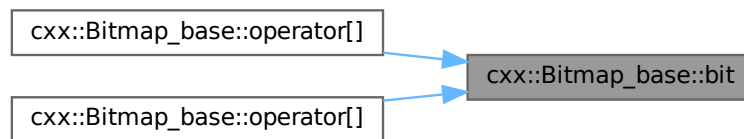
## Parameters

<i>bit</i>	The number of the bit.
<i>on</i>	The boolean value that shall be assigned to the bit.

Definition at line [262](#) of file [bitmap](#).

Referenced by [operator\[\]\(\)](#), and [operator\[\]\(\)](#).

Here is the caller graph for this function:



### 15.31.3.7 `bit_index()`

```
static unsigned cxx::Bitmap_base::bit_index (
    unsigned bit ) [inline], [static], [protected]
```

Get the bit index within `word_type` for the given bit.

## Parameters

<i>bit</i>	The bit index in the bitmap.
------------	------------------------------

## Returns

the bit index within `word_type` (`bit % W_bits`).

Definition at line [64](#) of file [bitmap](#).

References [W\\_bits](#).

### 15.31.3.8 `clear_bit()`

```
void cxx::Bitmap_base::clear_bit (
    long bit ) [inline], [noexcept]
```

Clear bit `bit`.



## Parameters

<i>bit</i>	The number of the bit to clear.
------------	---------------------------------

Definition at line 271 of file [bitmap](#).

**15.31.3.9 operator[]() [1/2]**

```
word_type cxx::Bitmap_base::operator[] (
    long bit ) const [inline], [noexcept]
```

Get the bit at index `bit`.

## Parameters

<i>bit</i>	The number of the bit to read.
------------	--------------------------------

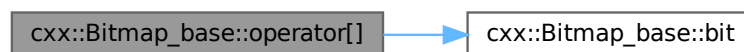
## Return values

0	Bit is not set.
!= 0	Bit is set.

Definition at line 192 of file [bitmap](#).

References [bit\(\)](#).

Here is the call graph for this function:

**15.31.3.10 operator[]() [2/2]**

```
Bit cxx::Bitmap_base::operator[] (
    long bit ) [inline], [noexcept]
```

Get the lvalue for the bit at index `bit`.

## Parameters

<i>bit</i>	The number.
------------	-------------

**Returns**

lvalue for `bit`

Definition at line 202 of file [bitmap](#).

References [bit\(\)](#).

Here is the call graph for this function:

**15.31.3.11 scan\_zero()**

```

long cxx::Bitmap_base::scan_zero (
    long max_bit,
    long start_bit = 0 ) const [inline], [noexcept]
  
```

Scan for the first zero bit.

**Parameters**

<i>max_bit</i>	Upper bound (exclusive) for the scanning operation.
<i>start_bit</i>	Hint at the number of the first bit to look at. Zero bits below <i>start_bit</i> may or may not be taken into account by the implementation.

**Return values**

<code>&gt;=</code>	0 Number of first zero bit found.
<code>-1</code>	All bits between <i>start_bit</i> and <i>max_bit</i> are set.

Definition at line 350 of file [bitmap](#).

Referenced by [cxx::Bitmap< BITS >::scan\\_zero\(\)](#).

Here is the caller graph for this function:



**15.31.3.12 set\_bit()**

```
void cxx::Bitmap_base::set_bit (
    long bit ) [inline], [noexcept]
```

Set bit *bit*.

**Parameters**

<i>bit</i>	The number of the bit to set.
------------	-------------------------------

Definition at line 300 of file [bitmap](#).

**15.31.3.13 word\_index()**

```
static unsigned cxx::Bitmap_base::word_index (
    unsigned bit ) [inline], [static], [protected]
```

Get the word index for the given bit.

**Parameters**

<i>bit</i>	The index of the bit in question.
------------	-----------------------------------

**Returns**

the index in [Bitmap\\_base::\\_bits](#) for the given bit (*bit* / *W\_bits*).

Definition at line 55 of file [bitmap](#).

References [W\\_bits](#).

The documentation for this class was generated from the following file:

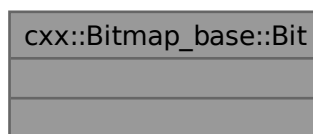
- [I4/cxx/bitmap](#)

**15.32 cxx::Bitmap\_base::Bit Class Reference**

A writeable bit in a bitmap.

```
#include <bitmap>
```

Collaboration diagram for cxx::Bitmap\_base::Bit:



### 15.32.1 Detailed Description

A writeable bit in a bitmap.

Definition at line 69 of file [bitmap](#).

The documentation for this class was generated from the following file:

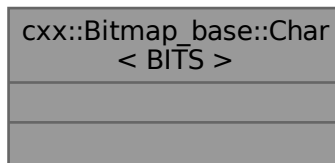
- I4/cxx/bitmap

## 15.33 cxx::Bitmap\_base::Char< BITS > Class Template Reference

Helper abstraction for a byte contained in the bitmap.

```
#include <bitmap>
```

Collaboration diagram for cxx::Bitmap\_base::Char< BITS >:



### 15.33.1 Detailed Description

```
template<long BITS>
class cxx::Bitmap_base::Char< BITS >
```

Helper abstraction for a byte contained in the bitmap.

Definition at line 106 of file [bitmap](#).

The documentation for this class was generated from the following file:

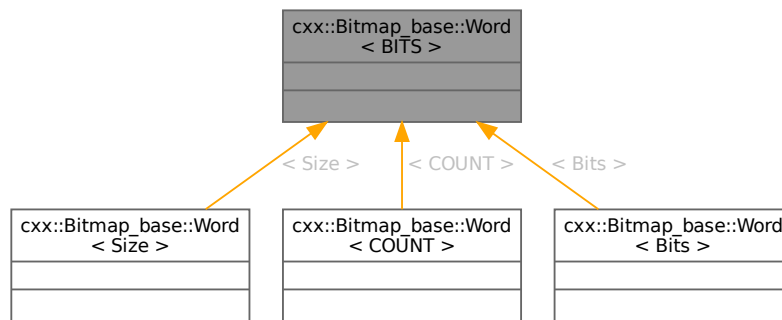
- I4/cxx/bitmap

## 15.34 cxx::Bitmap\_base::Word< BITS > Class Template Reference

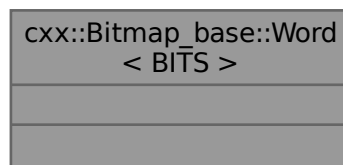
Helper abstraction for a word contained in the bitmap.

```
#include <bitmap>
```

Inheritance diagram for cxx::Bitmap\_base::Word< BITS >:



Collaboration diagram for cxx::Bitmap\_base::Word< BITS >:



### 15.34.1 Detailed Description

```
template<long BITS>
class cxx::Bitmap_base::Word< BITS >
```

Helper abstraction for a word contained in the bitmap.

Definition at line 90 of file [bitmap](#).

The documentation for this class was generated from the following file:

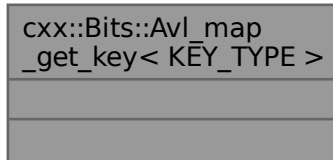
- `I4/cxx/bitmap`

## 15.35 `cxx::Bits::Avl_map_get_key< KEY_TYPE >` Struct Template Reference

Key-getter for [Avl\\_map](#).

```
#include <avl_map>
```

Collaboration diagram for `cxx::Bits::Avl_map_get_key< KEY_TYPE >`:



### 15.35.1 Detailed Description

```
template<typename KEY_TYPE>
struct cxx::Bits::Avl_map_get_key< KEY_TYPE >
```

Key-getter for [Avl\\_map](#).

Definition at line 36 of file [avl\\_map](#).

The documentation for this struct was generated from the following file:

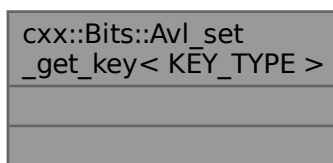
- [l4/cxx/avl\\_map](#)

## 15.36 `cxx::Bits::Avl_set_get_key< KEY_TYPE >` Struct Template Reference

Internal, key-getter for [Avl\\_set](#) nodes.

```
#include <avl_set>
```

Collaboration diagram for `cxx::Bits::Avl_set_get_key< KEY_TYPE >`:



### 15.36.1 Detailed Description

```
template<typename KEY_TYPE>
struct cxx::Bits::Avl_set_get_key< KEY_TYPE >
```

Internal, key-getter for [Avl\\_set](#) nodes.

Definition at line 109 of file [avl\\_set](#).

The documentation for this struct was generated from the following file:

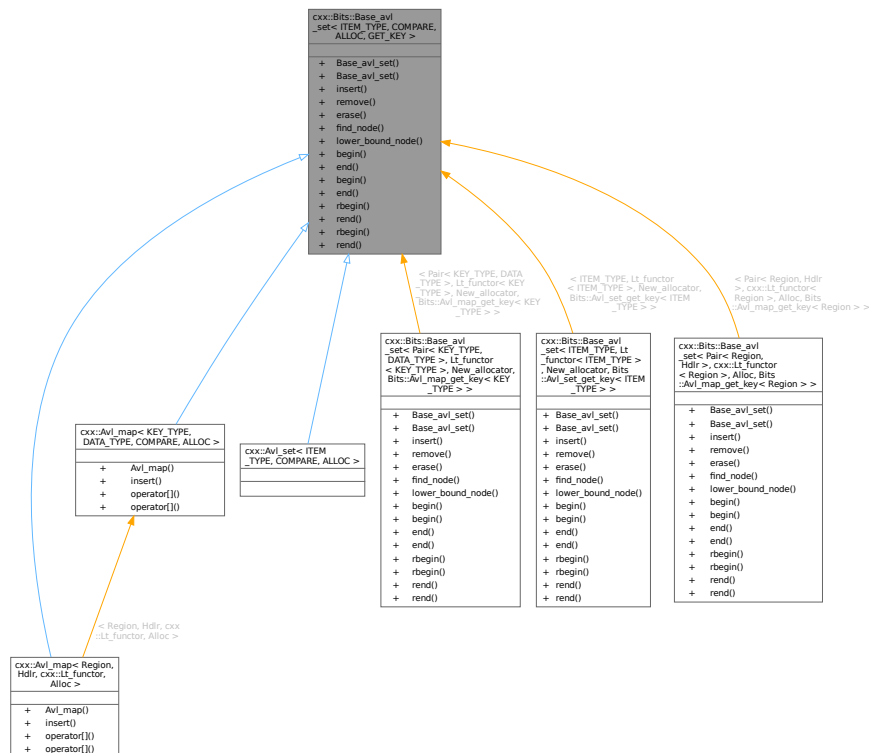
- [l4/cxx/avl\\_set](#)

## 15.37 cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY > Class Template Reference

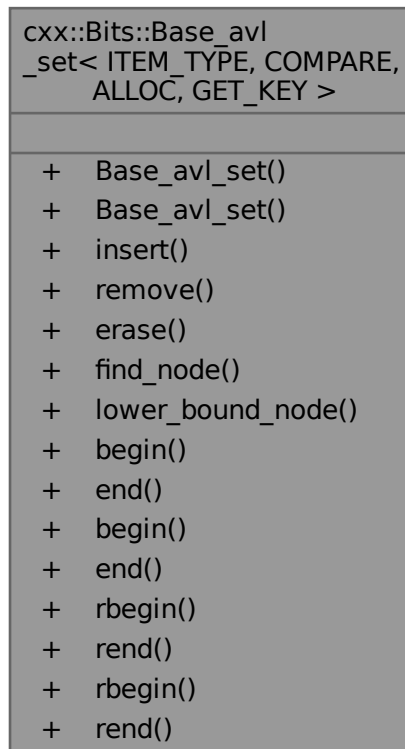
Internal: AVL set with internally managed nodes.

```
#include <avl_set>
```

Inheritance diagram for `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >`:



Collaboration diagram for `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >`:



## Data Structures

- class [Node](#)

*A smart pointer to a tree item.*

## Public Types

- enum { [E\\_noent](#) = 2 , [E\\_exist](#) = 17 , [E\\_nomem](#) = 12 , [E\\_inval](#) = 22 }
- Return status constants.*
- typedef `ITEM_TYPE` **Item\_type**
- Type for the items store in the set.*
- typedef `GET_KEY` **Get\_key**
- Key-getter type to derive the sort key of an internal node.*
- typedef `GET_KEY::Key_type` **Key\_type**
- Type of the sort key used for the items.*
- typedef `Type_traits< Item\_type >::Const_type` **Const\_item\_type**
- Type used for const items within the set.*
- typedef `COMPARE` **Item\_compare**
- Type for the comparison functor.*
- typedef `ALLOC< _Node >` **Node\_allocator**



*Type for the node allocator.*

- `typedef Avl_set_iter< _Node, Item\_type, Fwd > Iterator`

*Forward iterator for the set.*

- `typedef Avl_set_iter< _Node, Const\_item\_type, Fwd > Const_iterator`

*Constant forward iterator for the set.*

- `typedef Avl_set_iter< _Node, Item\_type, Rev > Rev_iterator`

*Backward iterator for the set.*

- `typedef Avl_set_iter< _Node, Const\_item\_type, Rev > Const_rev_iterator`

*Constant backward iterator for the set.*

## Public Member Functions

- `Base\_avl\_set (Node\_allocator const &alloc=Node\_allocator())`

*Create a AVL-tree based set.*

- `Base\_avl\_set (Base\_avl\_set const &o)`

*Create a copy of an AVL-tree based set.*

- `cxx::Pair< Iterator, int > insert (Item\_type const &item)`

*Insert an item into the set.*

- `int remove (Key\_type const &item)`

*Remove an item from the set.*

- `int erase (Key\_type const &item)`

*Erase the item with the given key.*

- `Node find\_node (Key\_type const &item) const`

*Lookup a node equal to `item`.*

- `Node lower\_bound\_node (Key\_type const &key) const`

*Find the first node greater or equal to `key`.*

- `Const\_iterator begin () const`

*Get the constant forward iterator for the first element in the set.*

- `Const\_iterator end () const`

*Get the end marker for the constant forward iterator.*

- `Iterator begin ()`

*Get the mutable forward iterator for the first element of the set.*

- `Iterator end ()`

*Get the end marker for the mutable forward iterator.*

- `Const\_rev\_iterator rbegin () const`

*Get the constant backward iterator for the last element in the set.*

- `Const\_rev\_iterator rend () const`

*Get the end marker for the constant backward iterator.*

- `Rev\_iterator rbegin ()`

*Get the mutable backward iterator for the last element of the set.*

- `Rev\_iterator rend ()`

*Get the end marker for the mutable backward iterator.*

### 15.37.1 Detailed Description

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GET_KEY>
```

```
class cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >
```

Internal: AVL set with internally managed nodes.

Use [Avl\\_set](#), [Avl\\_map](#), or [Avl\\_tree](#) in applications.

## Template Parameters

<i>ITEM_TYPE</i>	The type of the items to be stored in the set.
<i>COMPARE</i>	The relation to define the partial order, default is to use operator '<'.
<i>ALLOC</i>	The allocator to use for the nodes of the AVL set.
<i>GET_KEY</i>	Sort-key getter (must provide the <i>Key_type</i> and sort-key for an item (of <i>ITEM_TYPE</i> )).

Definition at line 133 of file [avl\\_set](#).

## 15.37.2 Member Enumeration Documentation

### 15.37.2.1 anonymous enum

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
anonymous enum
```

Return status constants.

These constants are compatible with the [L4](#) error codes, see [l4\\_error\\_code\\_t](#).

## Enumerator

E_noent	Item does not exist.
E_exist	Item exists already.
E_nomem	Memory allocation failed.
E_inval	Internal error.

Definition at line 144 of file [avl\\_set](#).

## 15.37.3 Constructor & Destructor Documentation

### 15.37.3.1 Base\_avl\_set() [1/2]

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Base_avl_set (
    Node_allocator const & alloc = Node_allocator() ) [inline], [explicit]
```

Create a AVL-tree based set.

## Parameters

<i>alloc</i>	<a href="#">Node</a> allocator.
--------------	---------------------------------

Create an empty set (AVL-tree based).

Definition at line 254 of file [avl\\_set](#).

## 15.37.3.2 Base\_avl\_set() [2/2]

```
template<typename Item , class Compare , template< typename A > class Alloc, typename KEY_TYPE >
cxx::Bits::Base_avl_set< Item, Compare, Alloc, KEY_TYPE >::Base_avl_set (
    Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > const & o ) [inline]
```

Create a copy of an AVL-tree based set.

## Parameters

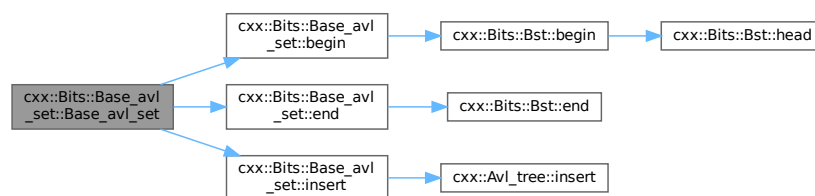
<i>o</i>	The set to copy.
----------	------------------

Creates a deep copy of the set with all its items.

Definition at line 413 of file [avl\\_set](#).

References [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >::begin\(\)](#), [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >::end\(\)](#) and [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >::insert\(\)](#).

Here is the call graph for this function:



## 15.37.4 Member Function Documentation

## 15.37.4.1 begin() [1/2]

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::begin ( ) [inline]
```

Get the mutable forward iterator for the first element of the set.

## Returns

The mutable forward iterator for the first element of the set.

Definition at line 367 of file [avl\\_set](#).

References [cxx::Bits::Bst< Node, Get\\_key, Compare >::begin\(\)](#).

Here is the call graph for this function:



### 15.37.4.2 begin() [2/2]

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Const_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::begin ( ) const
[inline]
```

Get the constant forward iterator for the first element in the set.

#### Returns

Constant forward iterator for the first element in the set.

Definition at line 356 of file [avl\\_set](#).

References [cxx::Bits::Bst< Node, Get\\_key, Compare >::begin\(\)](#).

Referenced by [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >::Base\\_avl\\_set\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.37.4.3 end() [1/2]

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::end ( ) [inline]
```

Get the end marker for the mutable forward iterator.

**Returns**

The end marker for mutable forward iterator.

Definition at line 372 of file [avl\\_set](#).

References [cxx::Bits::Bst< Node, Get\\_key, Compare >::end\(\)](#).

Here is the call graph for this function:

**15.37.4.4 end() [2/2]**

```

template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Const_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::end ( ) const
[inline]
  
```

Get the end marker for the constant forward iterator.

**Returns**

The end marker for the constant forward iterator.

Definition at line 361 of file [avl\\_set](#).

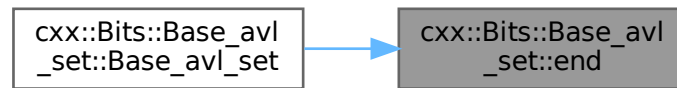
References [cxx::Bits::Bst< Node, Get\\_key, Compare >::end\(\)](#).

Referenced by [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >::Base\\_avl\\_set\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 15.37.4.5 erase()

```

template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
int cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::erase (
    Key_type const & item ) [inline]
  
```

Erase the item with the given key.

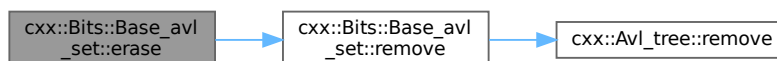
##### Parameters

<i>item</i>	The key of the item to remove.
-------------	--------------------------------

Definition at line 324 of file [avl\\_set](#).

References [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >::remove\(\)](#).

Here is the call graph for this function:



#### 15.37.4.6 find\_node()

```

template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Node cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::find_node (
    Key_type const & item ) const [inline]
  
```

Lookup a node equal to *item*.

## Parameters

<i>item</i>	The value to search for.
-------------	--------------------------

## Returns

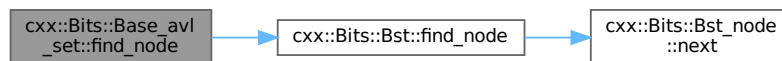
A smart pointer to the element found. If no element was found the smart pointer will be invalid.

Definition at line 335 of file [avl\\_set](#).

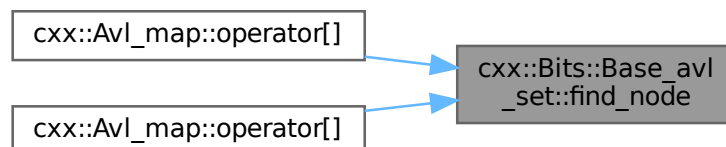
References [cxx::Bits::Bst< Node, Get\\_key, Compare >::find\\_node\(\)](#).

Referenced by [cxx::Avl\\_map< KEY\\_TYPE, DATA\\_TYPE, COMPARE, ALLOC >::operator\[\]\(\)](#), and [cxx::Avl\\_map< KEY\\_TYPE, DATA](#)

Here is the call graph for this function:



Here is the caller graph for this function:



## 15.37.4.7 insert()

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
```

```
Pair< typename Base_avl_set< Item, Compare, Alloc, KEY_TYPE >::Iterator, int > cxx::Bits::Base_avl_set<
Item, Compare, Alloc, KEY_TYPE >::insert (
    Item_type const & item )
```

Insert an item into the set.

## Parameters

<i>item</i>	The item to insert.
-------------	---------------------

**Returns**

A pair of iterator (*first*) and return value (*second*). *second* will be 0 if the element was inserted into the set and `-#E_exist` if the element was already in the set and the set was therefore not updated. In both cases, *first* contains an iterator that points to the element. *second* may also be `-#E_nomem` when memory for the node could not be allocated. *first* is then invalid.

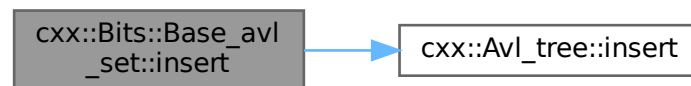
Insert a new item into the set, each item can only be once in the set.

Definition at line 423 of file [avl\\_set](#).

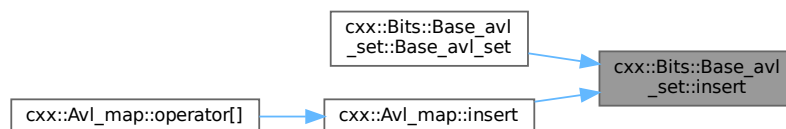
References [cxx::Pair< First, Second >::first](#), [cxx::Avl\\_tree< Node, Get\\_key, Compare >::insert\(\)](#), and [cxx::Pair< First, Second >::second](#).

Referenced by [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >::Base\\_avl\\_set\(\)](#), and [cxx::Avl\\_map< KEY\\_TYPE, DATA\\_TYPE, COMPARE, ALLOC >::insert\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.37.4.8 lower\_bound\_node()**

```

template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Node cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::lower_bound_node (
    Key_type const & key ) const [inline]
  
```

Find the first node greater or equal to *key*.

**Parameters**

<i>key</i>	Minimum key to look for.
------------	--------------------------



## 15.37 `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >` Class Template Reference 17

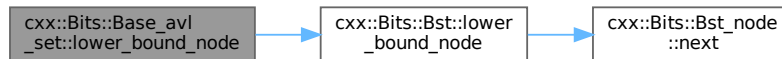
### Returns

Smart pointer to the first node greater or equal to `key`. Will be invalid if no such element was found.

Definition at line 346 of file `avl_set`.

References `cxx::Bits::Bst< Node, Get_key, Compare >::lower_bound_node()`.

Here is the call graph for this function:



### 15.37.4.9 `rbegin()` [1/2]

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename GET_KEY >
```

```
Rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rbegin ( ) [inline]
```

Get the mutable backward iterator for the last element of the set.

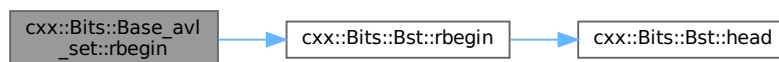
### Returns

The mutable backward iterator for the last element of the set.

Definition at line 389 of file `avl_set`.

References `cxx::Bits::Bst< Node, Get_key, Compare >::rbegin()`.

Here is the call graph for this function:



#### 15.37.4.10 `rbegin()` [2/2]

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Const_rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rbegin ( )
const [inline]
```

Get the constant backward iterator for the last element in the set.

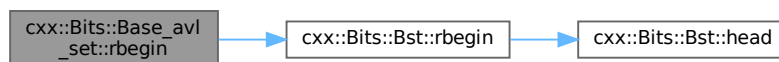
##### Returns

The constant backward iterator for the last element in the set.

Definition at line 378 of file [avl\\_set](#).

References [cxx::Bits::Bst< Node, Get\\_key, Compare >::rbegin\(\)](#).

Here is the call graph for this function:



#### 15.37.4.11 `remove()`

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
int cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::remove (
    Key_type const & item ) [inline]
```

Remove an item from the set.

##### Parameters

<i>item</i>	The item to remove.
-------------	---------------------

##### Return values

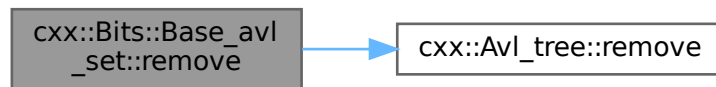
0	Success
-E_noent	Item does not exist

Definition at line 306 of file [avl\\_set](#).

References [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >::E\\_noent](#), and [cxx::Avl\\_tree< Node, Get\\_key,](#)

Referenced by [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >::erase\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 15.37.4.12 rend() [1/2]

```

template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rend ( ) [inline]
  
```

Get the end marker for the mutable backward iterator.

##### Returns

The end marker for mutable backward iterator.

Definition at line 394 of file [avl\\_set](#).

References [cxx::Bits::Bst< Node, Get\\_key, Compare >::rend\(\)](#).

Here is the call graph for this function:



**15.37.4.13** `rend()` [2/2]

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Const_rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rend ( )
const [inline]
```

Get the end marker for the constant backward iterator.

**Returns**

The end marker for the constant backward iterator.

Definition at line 383 of file [avl\\_set](#).

References [cxx::Bits::Bst< Node, Get\\_key, Compare >::rend\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

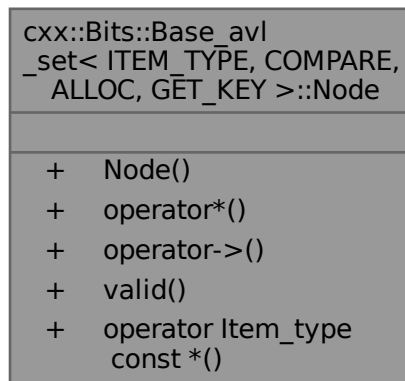
- [l4/cxx/avl\\_set](#)

## 15.38 `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node` Class Reference

A smart pointer to a tree item.

```
#include <avl_set>
```

Collaboration diagram for `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node`:



## Public Member Functions

- **Node ()**  
*Default construction for NIL pointer.*
- **Item\_type const & operator\* ()**  
*Dereference the pointer.*
- **Item\_type const \* operator-> ()**  
*Dereferenced member access.*
- **bool valid () const**  
*Validity check.*
- **operator Item\_type const \* ()**  
*Cast to a real item pointer.*

### 15.38.1 Detailed Description

```

template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
class cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node
  
```

A smart pointer to a tree item.

Definition at line 183 of file `avl_set`.

## 15.38.2 Member Function Documentation

### 15.38.2.1 operator\*()

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename  
GET_KEY >  
Item_type const & cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node::operator*  
( ) [inline]
```

Dereference the pointer.

#### Precondition

[Node](#) is valid.

Definition at line 200 of file [avl\\_set](#).

### 15.38.2.2 operator->()

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename  
GET_KEY >  
Item_type const * cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node::operator->  
( ) [inline]
```

Dereferenced member access.

#### Precondition

[Node](#) is valid.

Definition at line 206 of file [avl\\_set](#).

### 15.38.2.3 valid()

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename  
GET_KEY >  
bool cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node::valid ( ) const  
[inline]
```

Validity check.

#### Returns

false if the pointer is NIL, true if valid.

Definition at line 212 of file [avl\\_set](#).

The documentation for this class was generated from the following file:

- [I4/cxx/avl\\_set](#)



- Return the first element in the list.*
- void **clear** ()  
*Remove all elements from the list.*
- Iterator **begin** ()  
*Return an iterator to the beginning of the list.*
- Const\_iterator **begin** () const  
*Return a const iterator to the beginning of the list.*
- Const\_iterator **end** () const  
*Return a const iterator to the end of the list.*
- Iterator **end** ()  
*Return an iterator to the end of the list.*

### Static Public Member Functions

- static Const\_iterator **iter** (Const\_value\_type c)  
*Return a const iterator that begins at the given element.*

### Protected Attributes

- POLICY::Head\_type \_f  
*Pointer to front of the list.*

## 15.39.1 Detailed Description

**template<typename POLICY>**  
**class cxx::Bits::Basic\_list< POLICY >**

Internal: Common functions for all head-based list implementations.

Definition at line 50 of file [list\\_basics.h](#).

## 15.39.2 Member Function Documentation

### 15.39.2.1 clear()

```
template<typename POLICY >
void cxx::Bits::Basic_list< POLICY >::clear ( ) [inline]
```

Remove all elements from the list.

After the operation the state of the elements is undefined.

Definition at line 146 of file [list\\_basics.h](#).

References [cxx::Bits::Basic\\_list< POLICY >::\\_f](#).

### 15.39.2.2 iter()

```
template<typename POLICY >
static Const_iterator cxx::Bits::Basic_list< POLICY >::iter (
    Const_value_type c ) [inline], [static]
```

Return a const iterator that begins at the given element.



## Parameters

c	Element where the iterator should start.
---	--

### Precondition

The element `c` must already be in a list.

Definition at line 159 of file list\_basics.h.

The documentation for this class was generated from the following file:

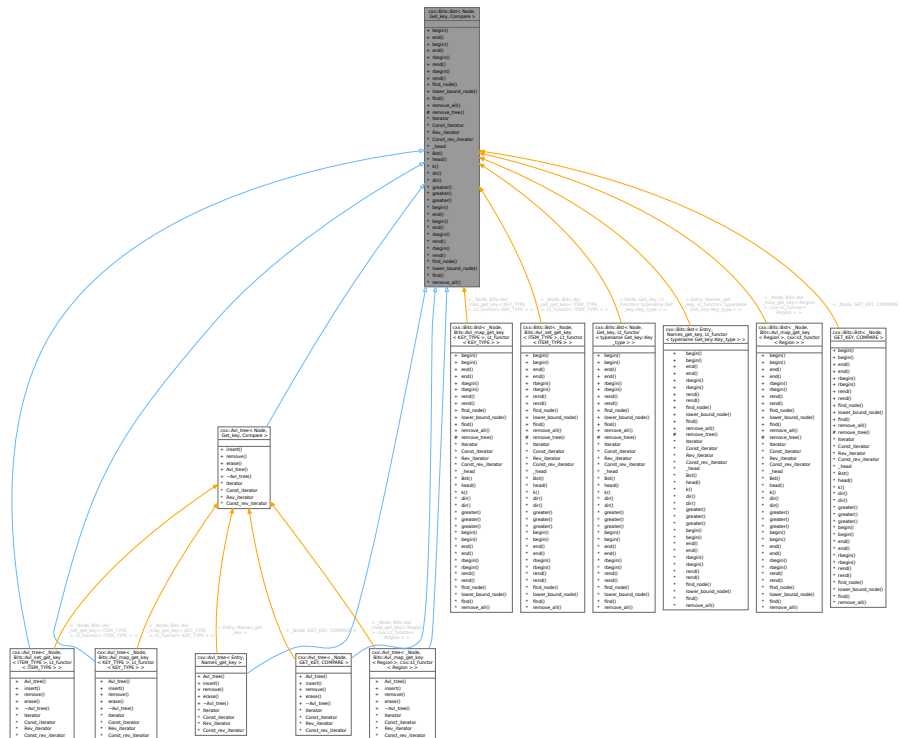
- `l4/cxx/bits/list_basics.h`

## 15.40 cxx::Bits::Bst< Node, Get\_key, Compare > Class Template Reference

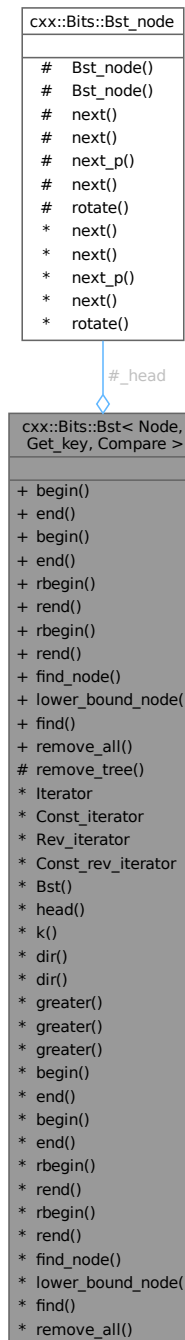
Basic binary search tree (BST).

```
#include <bst.h>
```

Inheritance diagram for `cxx::Bits::Bst< Node, Get_key, Compare >`:



Collaboration diagram for `cxx::Bits::Bst< Node, Get_key, Compare >`:



## Public Types

- `typedef Get_key::Key_type Key_type`  
*The type of key values used to generate the total order of the elements.*
- `typedef Type_traits< Key\_type >::Param_type Key_param_type`  
*The type for key parameters.*
- `typedef Fwd Fwd_iter_ops`

*Helper for building forward iterators for different wrapper classes.*

- typedef Rev **Rev\_iter\_ops**

*Helper for building reverse iterators for different wrapper classes.*

### Iterators

- typedef \_\_Bst\_iter< Node, Node, Fwd > **Iterator**  
*Forward iterator.*
- typedef \_\_Bst\_iter< Node, Node const, Fwd > **Const\_iterator**  
*Constant forward iterator.*
- typedef \_\_Bst\_iter< Node, Node, Rev > **Rev\_iterator**  
*Backward iterator.*
- typedef \_\_Bst\_iter< Node, Node const, Rev > **Const\_rev\_iterator**  
*Constant backward.*

### Public Member Functions

#### Get default iterators for the ordered tree.

- [Const\\_iterator begin](#) () const  
*Get the constant forward iterator for the first element in the set.*
- [Const\\_iterator end](#) () const  
*Get the end marker for the constant forward iterator.*
- [Iterator begin](#) ()  
*Get the mutable forward iterator for the first element of the set.*
- [Iterator end](#) ()  
*Get the end marker for the mutable forward iterator.*
- [Const\\_rev\\_iterator rbegin](#) () const  
*Get the constant backward iterator for the last element in the set.*
- [Const\\_rev\\_iterator rend](#) () const  
*Get the end marker for the constant backward iterator.*
- [Rev\\_iterator rbegin](#) ()  
*Get the mutable backward iterator for the last element of the set.*
- [Rev\\_iterator rend](#) ()  
*Get the end marker for the mutable backward iterator.*

#### Lookup functions.

- Node \* [find\\_node](#) (Key\_param\_type key) const  
*find the node with the given key.*
- Node \* [lower\\_bound\\_node](#) (Key\_param\_type key) const  
*Find the first node with a key not less than the given key.*
- [Const\\_iterator find](#) (Key\_param\_type key) const  
*find the node with the given key.*
- template<typename FUNC >  
void [remove\\_all](#) (FUNC &&callback)  
*Clear the tree.*

### Static Protected Member Functions

- template<typename FUNC >  
static void [remove\\_tree](#) (Bst\_node \*head, FUNC &&callback)  
*Remove all elements in the subtree of head.*

### Interior access for descendants.

As this class is an intended base class we provide protected access to our interior, use 'using' to make this private in concrete implementations.

- `Bst_node * _head`  
*The head pointer of the tree.*
- `Bst ()`  
*Create an empty tree.*
- `Node * head () const`  
*Access the head node as object of type Node.*
- `static Key_type k (Bst_node const *n)`  
*Get the key value of n.*
- `static Dir dir (Key_param_type l, Key_param_type r)`  
*Get the direction to go from l to search for r.*
- `static Dir dir (Key_param_type l, Bst_node const *r)`  
*Get the direction to go from l to search for r.*
- `static bool greater (Key_param_type l, Key_param_type r)`  
*Is l greater than r.*
- `static bool greater (Key_param_type l, Bst_node const *r)`  
*Is l greater than r.*
- `static bool greater (Bst_node const *l, Bst_node const *r)`  
*Is l greater than r.*

## 15.40.1 Detailed Description

```
template<typename Node, typename Get_key, typename Compare>
class cxx::Bits::Bst< Node, Get_key, Compare >
```

Basic binary search tree (BST).

This class is intended as a base class for concrete binary search trees, such as an AVL tree. This class already provides the basic lookup methods and iterator definitions for a BST.

Definition at line 42 of file [bst.h](#).

## 15.40.2 Member Function Documentation

### 15.40.2.1 begin() [1/2]

```
template<typename Node , typename Get_key , typename Compare >
Iterator cxx::Bits::Bst< Node, Get_key, Compare >::begin ( ) [inline]
```

Get the mutable forward iterator for the first element of the set.

**Returns**

The mutable forward iterator for the first element of the set.

Definition at line 194 of file [bst.h](#).

References [cxx::Bits::Bst< Node, Get\\_key, Compare >::head\(\)](#).

Here is the call graph for this function:

**15.40.2.2 begin() [2/2]**

```
template<typename Node , typename Get_key , typename Compare >
Const_iterator cxx::Bits::Bst< Node, Get_key, Compare >::begin ( ) const [inline]
```

Get the constant forward iterator for the first element in the set.

**Returns**

Constant forward iterator for the first element in the set.

Definition at line 183 of file [bst.h](#).

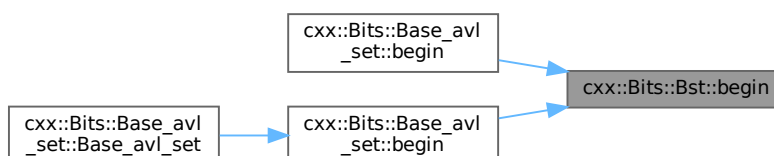
References [cxx::Bits::Bst< Node, Get\\_key, Compare >::head\(\)](#).

Referenced by [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >::begin\(\)](#), and [cxx::Bits::Base\\_avl\\_set< ITEM](#)

Here is the call graph for this function:



Here is the caller graph for this function:



**15.40.2.3** `dir()` [1/2]

```
template<typename Node , typename Get_key , typename Compare >
static Dir cxx::Bits::Bst< Node, Get_key, Compare >::dir (
    Key_param_type l,
    Bst_node const * r ) [inline], [static], [protected]
```

Get the direction to go from `l` to search for `r`.

**Parameters**

<code>l</code>	is the key to look for.
<code>r</code>	is the node at the current position.

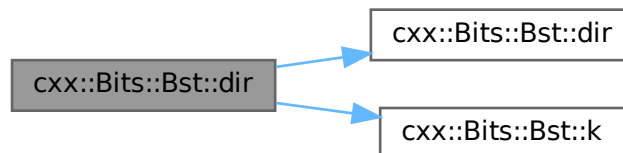
**Return values**

<code>Direction::L</code>	For left.
<code>Direction::R</code>	For right.
<code>Direction::N</code>	If <code>l</code> is equal to <code>r</code> .

Definition at line 139 of file `bst.h`.

References `cxx::Bits::Bst< Node, Get_key, Compare >::dir()`, and `cxx::Bits::Bst< Node, Get_key, Compare >::k()`.

Here is the call graph for this function:

**15.40.2.4** `dir()` [2/2]

```
template<typename Node , typename Get_key , typename Compare >
static Dir cxx::Bits::Bst< Node, Get_key, Compare >::dir (
    Key_param_type l,
    Key_param_type r ) [inline], [static], [protected]
```

Get the direction to go from `l` to search for `r`.

**Parameters**

<code>l</code>	is the key to look for.
<code>r</code>	is the key at the current position.

## Return values

<a href="#"><code>Direction::L</code></a>	for left
<a href="#"><code>Direction::R</code></a>	for right
<a href="#"><code>Direction::N</code></a>	if <code>l</code> is equal to <code>r</code> .

Definition at line 122 of file [bst.h](#).

References [`cxx::Bits::Direction::L`](#), and [`cxx::Bits::Direction::N`](#).

Referenced by [`cxx::Bits::Bst< Node, Get\_key, Compare >::dir\(\)`](#).

Here is the caller graph for this function:

15.40.2.5 `end()` [1/2]

```
template<typename Node , typename Get_key , typename Compare >
Iterator cxx::Bits::Bst< Node, Get_key, Compare >::end ( ) [inline]
```

Get the end marker for the mutable forward iterator.

## Returns

The end marker for mutable forward iterator.

Definition at line 199 of file [bst.h](#).

15.40.2.6 `end()` [2/2]

```
template<typename Node , typename Get_key , typename Compare >
Const_iterator cxx::Bits::Bst< Node, Get_key, Compare >::end ( ) const [inline]
```

Get the end marker for the constant forward iterator.

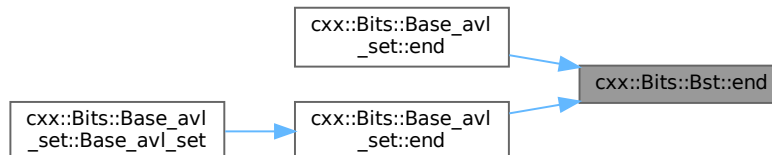
**Returns**

The end marker for the constant forward iterator.

Definition at line 188 of file [bst.h](#).

Referenced by [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >::end\(\)](#), and [cxx::Bits::Base\\_avl\\_set< ITEM](#)

Here is the caller graph for this function:

**15.40.2.7 find()**

```

template<typename Node , typename Get_key , class Compare >
Bst< Node, Get_key, Compare >::Const_iterator cxx::Bits::Bst< Node, Get_key, Compare >::find
(
    Key_param_type key ) const [inline]
  
```

find the node with the given *key*.

**Parameters**

<i>key</i>	The key value of the element to search.
------------	---

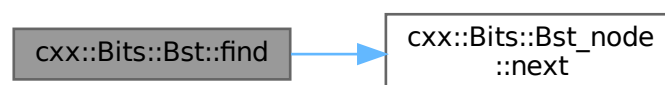
**Returns**

A valid iterator for the node with the given *key*, or an invalid iterator if *key* was not found.

Definition at line 316 of file [bst.h](#).

References [cxx::Bits::Bst\\_node::next\(\)](#).

Here is the call graph for this function:





## 15.40.2.8 find\_node()

```
template<typename Node , typename Get_key , class Compare >
Node * cxx::Bits::Bst< Node, Get_key, Compare >::find_node (
    Key_param_type key ) const [inline]
```

find the node with the given *key*.

## Parameters

<i>key</i>	The key value of the element to search.
------------	---

## Returns

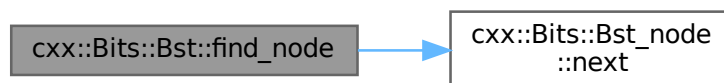
A pointer to the node with the given *key*, or NULL if *key* was not found.

Definition at line 280 of file [bst.h](#).

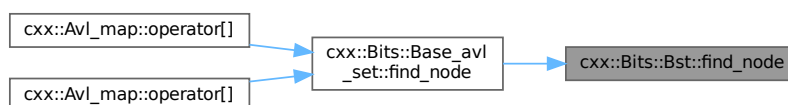
References [cxx::Bits::Bst\\_node::next\(\)](#).

Referenced by [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >::find\\_node\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 15.40.2.9 lower\_bound\_node()

```
template<typename Node , typename Get_key , class Compare >
Node * cxx::Bits::Bst< Node, Get_key, Compare >::lower_bound_node (
    Key_param_type key ) const [inline]
```

Find the first node with a key not less than the given *key*.

## Parameters

<i>key</i>	The key used for the search.
------------	------------------------------

## Returns

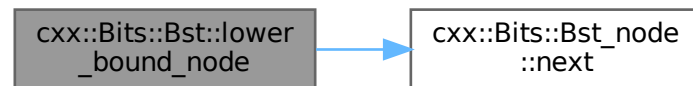
A pointer to the found node, or `NULL` if no node was found.

Definition at line 296 of file `bst.h`.

References `cxx::Bits::Bst_node::next()`.

Referenced by `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::lower_bound_node()`.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 15.40.2.10 `rbegin()` [1/2]

```

template<typename Node , typename Get_key , typename Compare >
Rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rbegin ( ) [inline]
  
```

Get the mutable backward iterator for the last element of the set.

**Returns**

The mutable backward iterator for the last element of the set.

Definition at line 216 of file `bst.h`.

References `cxx::Bits::Bst< Node, Get_key, Compare >::head()`.

Here is the call graph for this function:

**15.40.2.11 rbegin() [2/2]**

```
template<typename Node , typename Get_key , typename Compare >
Const_rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rbegin ( ) const [inline]
```

Get the constant backward iterator for the last element in the set.

**Returns**

The constant backward iterator for the last element in the set.

Definition at line 205 of file `bst.h`.

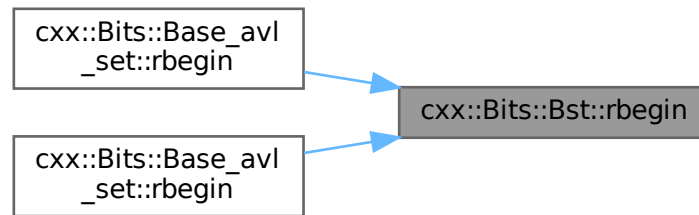
References `cxx::Bits::Bst< Node, Get_key, Compare >::head()`.

Referenced by `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rbegin()`, and `cxx::Bits::Base_avl_set< ITE`

Here is the call graph for this function:



Here is the caller graph for this function:



#### 15.40.2.12 remove\_all()

```

template<typename Node , typename Get_key , typename Compare >
template<typename FUNC >
void cxx::Bits::Bst< Node, Get_key, Compare >::remove_all (
    FUNC && callback ) [inline]
  
```

Clear the tree.

##### Parameters

<i>callback</i>	Optional function to be called on each removed element.
-----------------	---

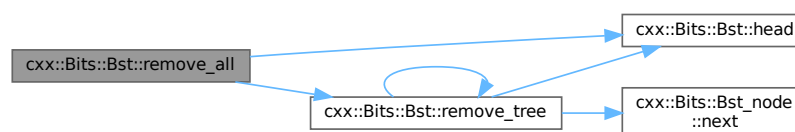
The callback may delete the elements. The function guarantees that the elements are no longer used after the callback has been called.

Definition at line 262 of file [bst.h](#).

References [cxx::Bits::Bst< Node, Get\\_key, Compare >::\\_head](#), [cxx::Bits::Bst< Node, Get\\_key, Compare >::head\(\)](#), and [cxx::Bits::Bst< Node, Get\\_key, Compare >::remove\\_tree\(\)](#).

Referenced by [cxx::Avl\\_tree< Node, Get\\_key, Compare >::~~Avl\\_tree\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.40.2.13 remove\_tree()

```

template<typename Node , typename Get_key , typename Compare >
template<typename FUNC >
static void cxx::Bits::Bst< Node, Get_key, Compare >::remove_tree (
    Bst_node * head,
    FUNC && callback ) [inline], [static], [protected]
  
```

Remove all elements in the subtree of head.

#### Parameters

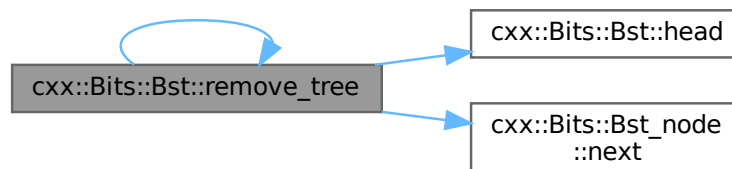
<i>head</i>	Head of the the subtree to remove
<i>callback</i>	Optional function called on each removed element.

Definition at line 162 of file [bst.h](#).

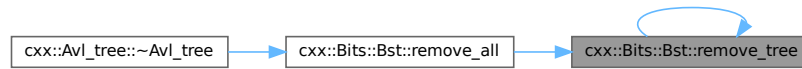
References [cxx::Bits::Bst< Node, Get\\_key, Compare >::head\(\)](#), [cxx::Bits::Direction::L](#), [cxx::Bits::Bst\\_node::next\(\)](#), [cxx::Bits::Direction::R](#), and [cxx::Bits::Bst< Node, Get\\_key, Compare >::remove\\_tree\(\)](#).

Referenced by [cxx::Bits::Bst< Node, Get\\_key, Compare >::remove\\_all\(\)](#), and [cxx::Bits::Bst< Node, Get\\_key, Compare >::remove\\_tr](#)

Here is the call graph for this function:



Here is the caller graph for this function:



#### 15.40.2.14 `rend()` [1/2]

```
template<typename Node , typename Get_key , typename Compare >
Rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rend ( ) [inline]
```

Get the end marker for the mutable backward iterator.

##### Returns

The end marker for mutable backward iterator.

Definition at line 221 of file [bst.h](#).

#### 15.40.2.15 `rend()` [2/2]

```
template<typename Node , typename Get_key , typename Compare >
Const_rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rend ( ) const [inline]
```

Get the end marker for the constant backward iterator.

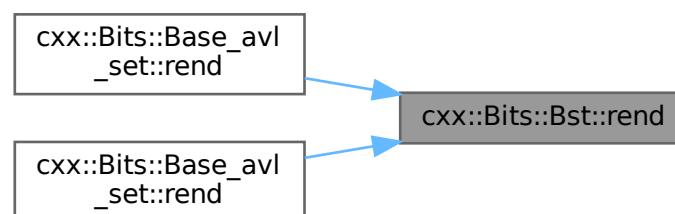
##### Returns

The end marker for the constant backward iterator.

Definition at line 210 of file [bst.h](#).

Referenced by `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rend()`, and `cxx::Bits::Base_avl_set< ITEM`

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

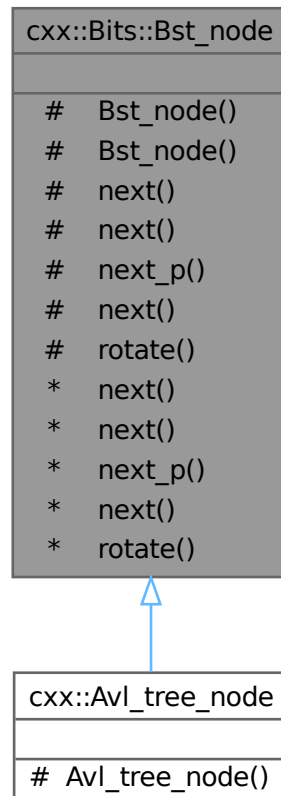
- [I4/cxx/bits/bst.h](#)

## 15.41 cxx::Bits::Bst\_node Class Reference

Basic type of a node in a binary search tree (BST).

```
#include <bst_base.h>
```

Inheritance diagram for cxx::Bits::Bst\_node:



Collaboration diagram for `cxx::Bits::Bst_node`:

cxx::Bits::Bst_node	
#	Bst_node()
#	Bst_node()
#	next()
#	next()
#	next_p()
#	next()
#	rotate()
*	next()
*	next()
*	next_p()
*	next()
*	rotate()

### Protected Member Functions

- **Bst\_node** ()  
*Create uninitialized node.*
- **Bst\_node** (bool)  
*Create initialized node.*

### Static Protected Member Functions

#### Access to BST linkage.

Provide access to the tree linkage to inherited classes. Inherited nodes, such as AVL nodes should make these methods private via 'using'

- static `Bst_node` \* **next** (`Bst_node` const \*p, `Direction` d)  
*Get next node in direction d.*
- static void **next** (`Bst_node` \*p, `Direction` d, `Bst_node` \*n)  
*Set next node of p in direction d to n.*
- static `Bst_node` \*\* **next\_p** (`Bst_node` \*p, `Direction` d)  
*Get pointer to link in direction d.*
- template<typename Node >  
static Node \* **next** (`Bst_node` const \*p, `Direction` d)  
*Get next node in direction d as type Node.*
- static void **rotate** (`Bst_node` \*\*t, `Direction` idir)  
*Rotate subtree t in the opposite direction of idir.*



### 15.41.1 Detailed Description

Basic type of a node in a binary search tree (BST).

Definition at line 81 of file [bst\\_base.h](#).

The documentation for this class was generated from the following file:

- [l4/cxx/bits/bst\\_base.h](#)

## 15.42 cxx::Bits::Direction Struct Reference

The direction to go in a binary search tree.

```
#include <bst_base.h>
```

Collaboration diagram for cxx::Bits::Direction:

cxx::Bits::Direction	
+	Direction()
+	Direction()
+	Direction()
+	operator!()
+	operator==(())
+	operator!==(())
+	operator==(())
+	operator!==(())
*	operator==(())
*	operator!==(())
*	operator==(())
*	operator!==(())

### Public Types

- enum [Direction\\_e](#) { [L](#) = 0 , [R](#) = 1 , [N](#) = 2 }

*The literal direction values.*

## Public Member Functions

- **Direction** ()=default  
*Uninitialized direction.*
- **Direction** ([Direction\\_e](#) d)  
*Convert a literal direction ([L](#), [R](#), [N](#)) to an object.*
- **Direction** (bool b)  
*Convert a boolean to a direction (false == [L](#), true == [R](#))*
- **Direction operator!** () const  
*Negate the direction.*

## Comparison operators (equality and inequality)

- bool **operator==** ([Direction\\_e](#) o) const  
*Compare for equality.*
- bool **operator!=** ([Direction\\_e](#) o) const  
*Compare for inequality.*
- bool **operator==** ([Direction](#) o) const  
*Compare for equality.*
- bool **operator!=** ([Direction](#) o) const  
*Compare for inequality.*

### 15.42.1 Detailed Description

The direction to go in a binary search tree.

Definition at line 39 of file [bst\\_base.h](#).

### 15.42.2 Member Enumeration Documentation

#### 15.42.2.1 Direction\_e

```
enum cxx::Bits::Direction::Direction\_e
```

The literal direction values.

Enumerator

L	Go to the left child.
R	Go to the right child.
N	Stop.

Definition at line 42 of file [bst\\_base.h](#).

### 15.42.3 Member Function Documentation

#### 15.42.3.1 operator"!"()

```
Direction cxx::Bits::Direction::operator! ( ) const [inline]
```

Negate the direction.

#### Note

This is only defined for a current value of [L](#) or [R](#)

Definition at line [63](#) of file [bst\\_base.h](#).

References [Direction\(\)](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

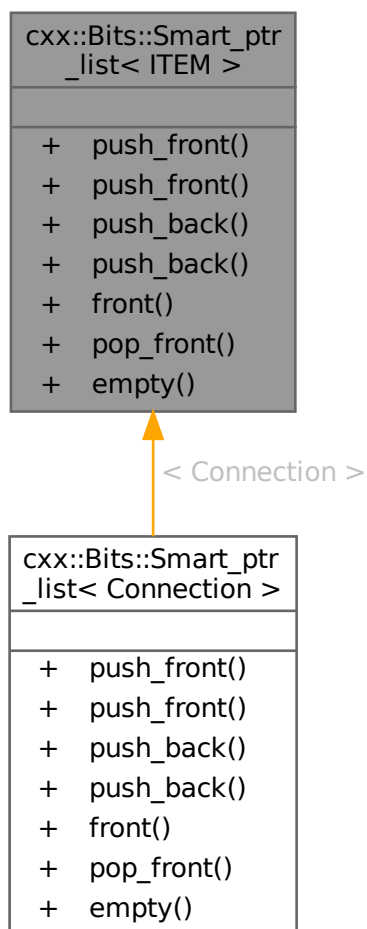
- [I4/cxx/bits/bst\\_base.h](#)

## 15.43 `cxx::Bits::Smart_ptr_list< ITEM >` Class Template Reference

[List](#) of smart-pointer-managed objects.

```
#include <smart_ptr_list.h>
```

Inheritance diagram for `cxx::Bits::Smart_ptr_list< ITEM >`:



Collaboration diagram for `cxx::Bits::Smart_ptr_list< ITEM >`:

<code>cxx::Bits::Smart_ptr_list&lt; ITEM &gt;</code>
<ul style="list-style-type: none"> <li>+ <code>push_front()</code></li> <li>+ <code>push_front()</code></li> <li>+ <code>push_back()</code></li> <li>+ <code>push_back()</code></li> <li>+ <code>front()</code></li> <li>+ <code>pop_front()</code></li> <li>+ <code>empty()</code></li> </ul>

### Public Member Functions

- void **push\_front** (Next\_type &&e)  
*Add an element to the front of the list.*
- void **push\_front** (Next\_type const &e)  
*Add an element to the front of the list.*
- void **push\_back** (Next\_type &&e)  
*Add an element at the end of the list.*
- void **push\_back** (Next\_type const &e)  
*Add an element at the end of the list.*
- Value\_type \* **front** () const  
*Return a pointer to the first element in the list.*
- Next\_type **pop\_front** ()  
*Remove the element in front of the list and return it.*
- bool **empty** () const  
*Check if the list is empty.*

### 15.43.1 Detailed Description

```
template<typename ITEM>
class cxx::Bits::Smart_ptr_list< ITEM >
```

[List](#) of smart-pointer-managed objects.

#### Template Parameters

<i>ITEM</i>	Type of the list items.
-------------	-------------------------

The list is implemented as a single-linked list connected via smart pointers, so that they are automatically cleaned up when they are removed from the list.

Definition at line 47 of file [smart\\_ptr\\_list.h](#).

## 15.43.2 Member Function Documentation

### 15.43.2.1 pop\_front()

```
template<typename ITEM >
Next_type cxx::Bits::Smart_ptr_list< ITEM >::pop_front ( ) [inline]
```

Remove the element in front of the list and return it.

#### Returns

The element that was previously in front of the list as a managed pointer or a nullptr-equivalent when the list was already empty.

Definition at line 150 of file [smart\\_ptr\\_list.h](#).

The documentation for this class was generated from the following file:

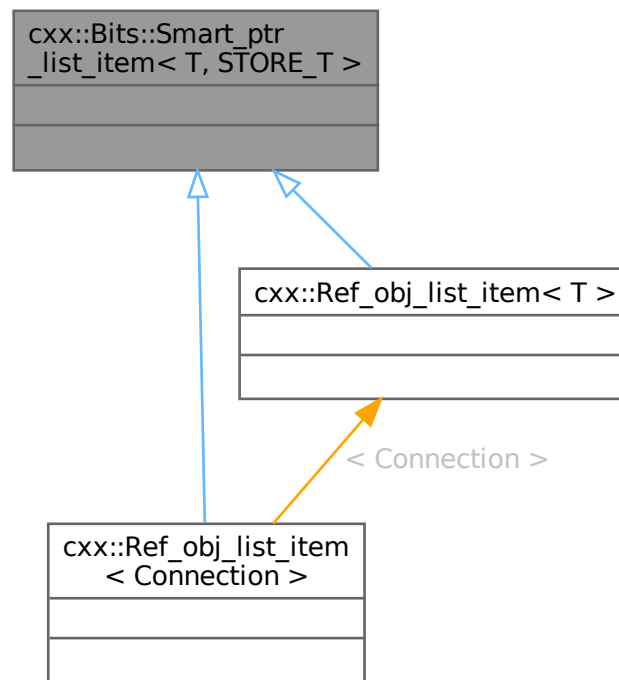
- l4/cxx/bits/[smart\\_ptr\\_list.h](#)

## 15.44 cxx::Bits::Smart\_ptr\_list\_item< T, STORE\_T > Class Template Reference

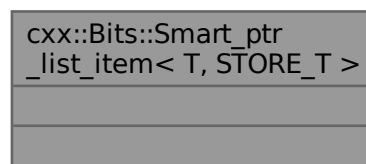
[List](#) item for an arbitrary item in a [Smart\\_ptr\\_list](#).

```
#include <smart_ptr_list.h>
```

Inheritance diagram for cxx::Bits::Smart\_ptr\_list\_item< T, STORE\_T >:



Collaboration diagram for cxx::Bits::Smart\_ptr\_list\_item< T, STORE\_T >:



### 15.44.1 Detailed Description

```

template<typename T, typename STORE_T>
class cxx::Bits::Smart_ptr_list_item< T, STORE_T >

```

List item for an arbitrary item in a [Smart\\_ptr\\_list](#).

## Template Parameters

<i>T</i>	Type of object to be stored in the list.
<i>STORE</i> <i>_T</i>	Storage type for pointer to next item. The class must implement a get() function that returns a pointer to the stored object and destroy the stored object when the item goes out of scope.

Definition at line 28 of file [smart\\_ptr\\_list.h](#).

The documentation for this class was generated from the following file:

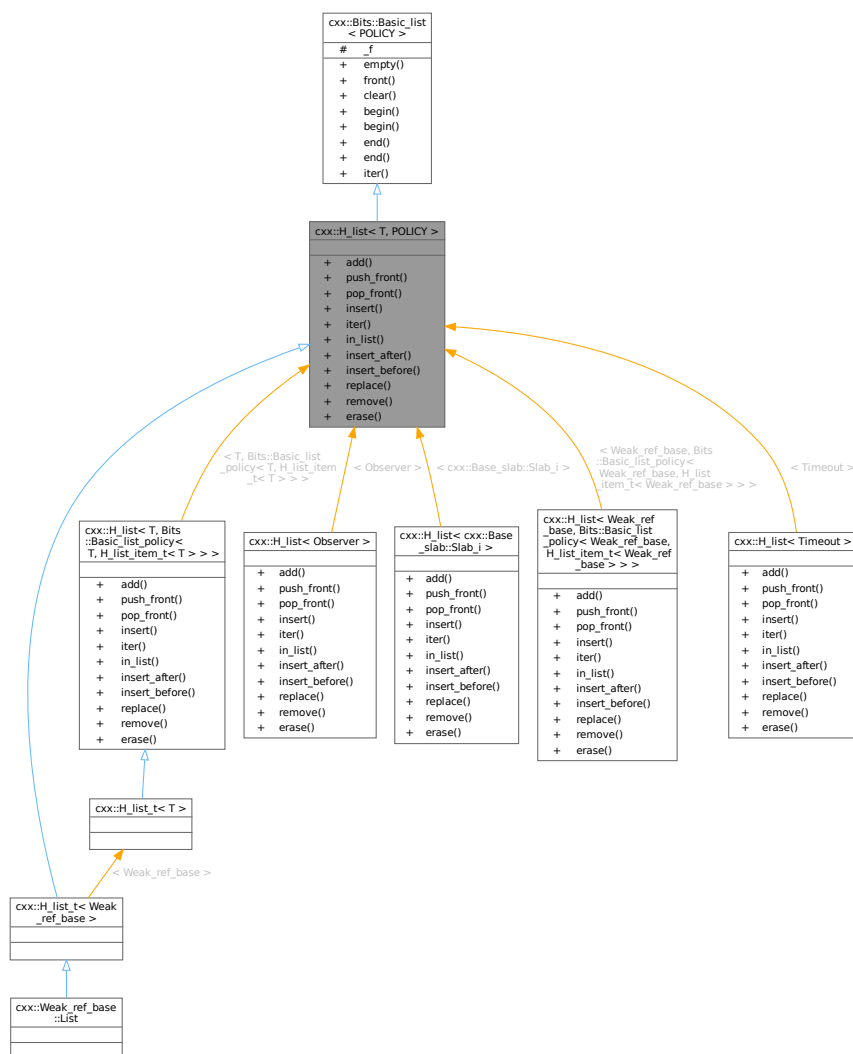
- [I4/cxx/bits/smart\\_ptr\\_list.h](#)

## 15.45 cxx::H\_list< T, POLICY > Class Template Reference

General double-linked list of unspecified [cxx::H\\_list\\_item](#) elements.

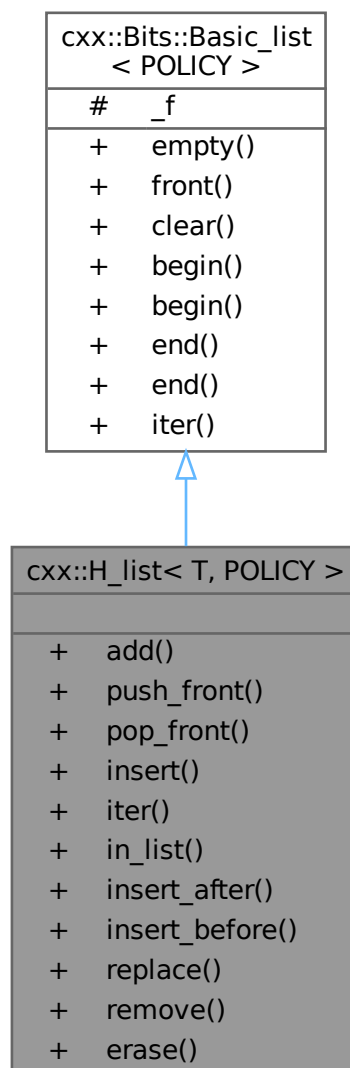
```
#include <hlist>
```

Inheritance diagram for `cxx::H_list< T, POLICY >`:





Collaboration diagram for cxx::H\_list< T, POLICY >:



## Public Member Functions

- void **add** (T \*e)  
*Add element to the front of the list.*
- void **push\_front** (T \*e)  
*Add element to the front of the list.*
- T \* **pop\_front** ()  
*Remove and return the head element of the list.*
- Iterator **insert** (T \*e, Iterator const &pred)  
*Insert an element at the iterator position.*

## Public Member Functions inherited from `cxx::Bits::Basic_list< POLICY >`

- `bool empty ()` `const`  
*Check if the list is empty.*
- `Value_type front ()` `const`  
*Return the first element in the list.*
- `void clear ()`  
*Remove all elements from the list.*
- `Iterator begin ()`  
*Return an iterator to the beginning of the list.*
- `Const_iterator begin ()` `const`  
*Return a const iterator to the beginning of the list.*
- `Const_iterator end ()` `const`  
*Return a const iterator to the end of the list.*
- `Iterator end ()`  
*Return an iterator to the end of the list.*

## Static Public Member Functions

- `static Iterator iter (T *c)`  
*Return an iterator for an arbitrary list element.*
- `static bool in_list (T const *e)`  
*Check if the given element is currently part of a list.*
- `static Iterator insert_after (T *e, Iterator const &pred)`  
*Insert an element after the iterator position.*
- `static void insert_before (T *e, Iterator const &succ)`  
*Insert an element before the iterator position.*
- `static void replace (T *p, T *e)`  
*Replace an element in a list with a new element.*
- `static void remove (T *e)`  
*Remove the given element from its list.*
- `static Iterator erase (Iterator const &e)`  
*Remove the element at the given iterator position.*

## Static Public Member Functions inherited from `cxx::Bits::Basic_list< POLICY >`

- `static Const_iterator iter (Const_value_type c)`  
*Return a const iterator that begins at the given element.*

## Additional Inherited Members

## Protected Attributes inherited from `cxx::Bits::Basic_list< POLICY >`

- `POLICY::Head_type _f`  
*Pointer to front of the list.*

### 15.45.1 Detailed Description

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
class cxx::H_list< T, POLICY >
```

General double-linked list of unspecified [cxx::H\\_list\\_item](#) elements.

Most of the time, you want to use [H\\_list\\_t](#).

Definition at line 80 of file [hlist](#).

### 15.45.2 Member Function Documentation

#### 15.45.2.1 erase()

```
template<typename T , typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static Iterator cxx::H\_list< T, POLICY >::erase (
    Iterator const & e ) [inline], [static]
```

Remove the element at the given iterator position.

##### Parameters

<i>e</i>	Iterator pointing to the element to be removed. Must not point to <a href="#">end()</a> .
----------	---

##### Returns

New iterator pointing to the element after the removed one.

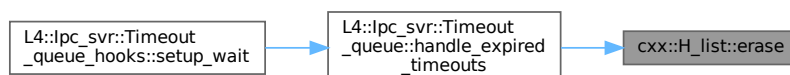
##### Note

The hlist implementation guarantees that the original iterator is still valid after the element has been removed. In fact, the iterator returned is the same as the one supplied in the *e* parameter.

Definition at line 247 of file [hlist](#).

Referenced by [L4::lpc\\_svr::Timeout\\_queue::handle\\_expired\\_timeouts\(\)](#).

Here is the caller graph for this function:



### 15.45.2.2 insert()

```
template<typename T , typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
Iterator cxx::H_list< T, POLICY >::insert (
    T * e,
    Iterator const & pred ) [inline]
```

Insert an element at the iterator position.

#### Parameters

<i>e</i>	New Element to be inserted
<i>pred</i>	Iterator pointing to the element after which the element will be inserted. If <a href="#">end()</a> is given, the element will be inserted at the beginning of the queue.

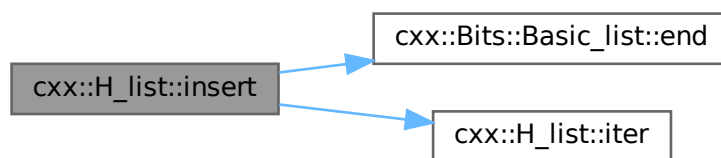
#### Returns

Iterator pointing to the newly inserted element.

Definition at line 144 of file [hlist](#).

References [cxx::Bits::Basic\\_list< POLICY >::\\_f](#), [cxx::Bits::Basic\\_list< POLICY >::end\(\)](#), and [cxx::H\\_list< T, POLICY >::iter\(\)](#).

Here is the call graph for this function:



### 15.45.2.3 insert\_after()

```
template<typename T , typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static Iterator cxx::H_list< T, POLICY >::insert_after (
    T * e,
    Iterator const & pred ) [inline], [static]
```

Insert an element after the iterator position.

#### Parameters

<i>e</i>	New element to be inserted.
<i>pred</i>	Iterator pointing to the element after which the element will be inserted. Must not be <a href="#">end()</a> .

**Returns**

Iterator pointing to the newly inserted element.

**Precondition**

The list must not be empty.

Definition at line 171 of file [hlist](#).

References [cxx::H\\_list< T, POLICY >::iter\(\)](#).

Here is the call graph for this function:

**15.45.2.4 insert\_before()**

```

template<typename T , typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static void cxx::H\_list< T, POLICY >::insert\_before (
    T * e,
    Iterator const & succ ) [inline], [static]
  
```

Insert an element before the iterator position.

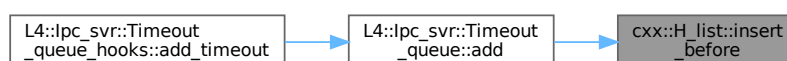
**Parameters**

<i>e</i>	New element to be inserted.
<i>succ</i>	Iterator pointing to the element before which the element will be inserted. Must not be <a href="#">end()</a> .

Definition at line 191 of file [hlist](#).

Referenced by [L4::lpc\\_svr::Timeout\\_queue::add\(\)](#).

Here is the caller graph for this function:



### 15.45.2.5 iter()

```
template<typename T , typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static Iterator cxx::H\_list< T, POLICY >::iter (
    T * c ) [inline], [static]
```

Return an iterator for an arbitrary list element.

#### Parameters

<b>c</b>	List element to start the iteration.
----------	--------------------------------------

#### Returns

A mutable forward iterator.

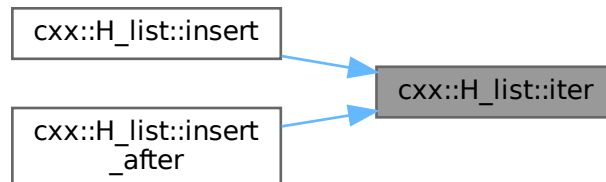
#### Precondition

The element must be in a list.

Definition at line 104 of file [hlist](#).

Referenced by [cxx::H\\_list< T, POLICY >::insert\(\)](#), and [cxx::H\\_list< T, POLICY >::insert\\_after\(\)](#).

Here is the caller graph for this function:



### 15.45.2.6 pop\_front()

```
template<typename T , typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
T * cxx::H\_list< T, POLICY >::pop_front ( ) [inline]
```

Remove and return the head element of the list.

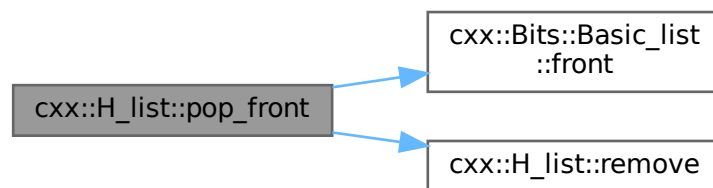
**Precondition**

The list must not be empty or the behaviour will be undefined.

Definition at line 127 of file [hlist](#).

References [cxx::Bits::Basic\\_list< POLICY >::front\(\)](#), and [cxx::H\\_list< T, POLICY >::remove\(\)](#).

Here is the call graph for this function:

**15.45.2.7 remove()**

```
template<typename T , typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static void cxx::H_list< T, POLICY >::remove (
    T * e ) [inline], [static]
```

Remove the given element from its list.

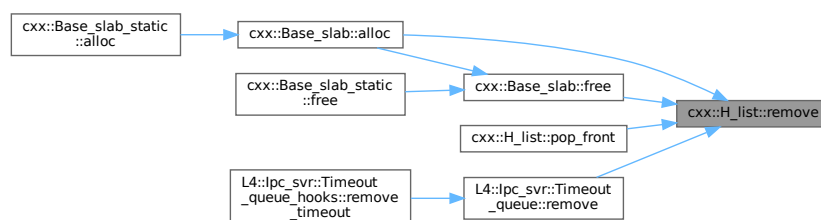
**Parameters**

<code>e</code>	Element to be removed. Must be in a list.
----------------	---

Definition at line 231 of file [hlist](#).

Referenced by [cxx::Base\\_slab< Obj\\_size, Slab\\_size, Max\\_free, Alloc >::alloc\(\)](#), [cxx::Base\\_slab< Obj\\_size, Slab\\_size, Max\\_free, Alloc >::free\(\)](#), [cxx::H\\_list< T, POLICY >::pop\\_front\(\)](#), and [L4::lpc\\_svr::Timeout\\_queue::remove\(\)](#).

Here is the caller graph for this function:



### 15.45.2.8 replace()

```
template<typename T , typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static void cxx::H_list< T, POLICY >::replace (
    T * p,
    T * e ) [inline], [static]
```

Replace an element in a list with a new element.

#### Parameters

<i>p</i>	Element in list to be replaced.
<i>e</i>	Replacement element, must not yet be in a list.

#### Precondition

*p* and *e* must not be NULL.

After the operation the *p* element is no longer in the list and may be reused.

Definition at line 215 of file [hlist](#).

The documentation for this class was generated from the following file:

- l4/cxx/hlist

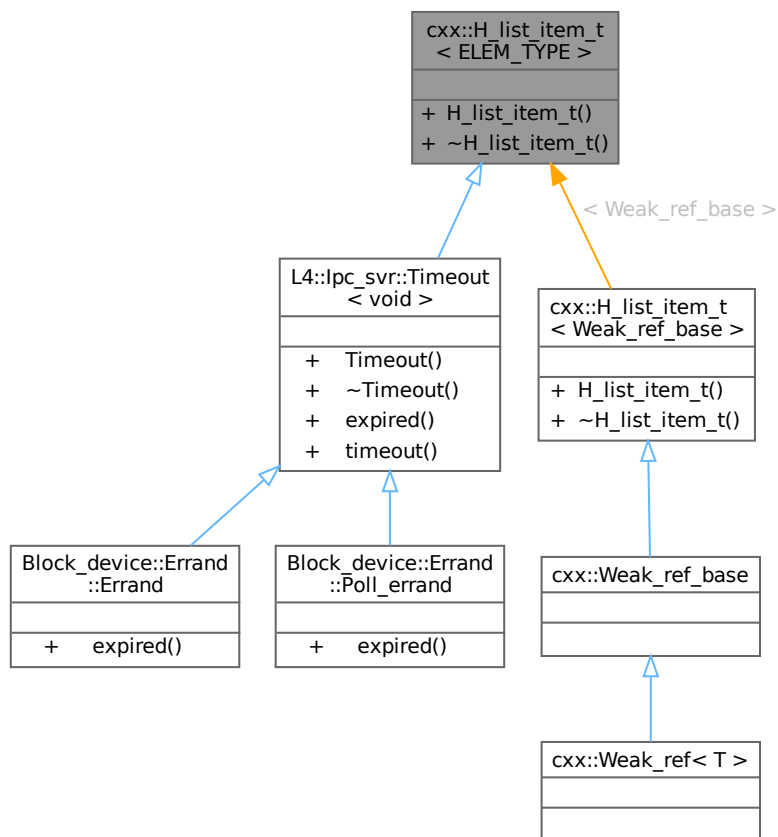
## 15.46 cxx::H\_list\_item\_t< ELEM\_TYPE > Class Template Reference

Basic element type for a double-linked [H\\_list](#).

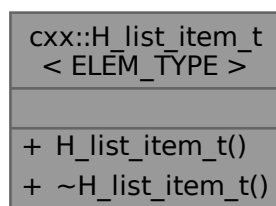
```
#include <hlist>
```



Inheritance diagram for `cxx::H_list_item_t< ELEM_TYPE >`:



Collaboration diagram for cxx::H\_list\_item\_t< ELEM\_TYPE >:



## Public Member Functions

- `H_list_item_t ()`  
*Constructor.*
- `~H_list_item_t () noexcept`  
*Destructor.*

### 15.46.1 Detailed Description

```
template<typename ELEM_TYPE>
class cxx::H_list_item_t< ELEM_TYPE >
```

Basic element type for a double-linked [H\\_list](#).

#### Template Parameters

<i>ELEM_TYPE</i>	Base class of the list element.
------------------	---------------------------------

Definition at line [33](#) of file [hlist](#).

### 15.46.2 Constructor & Destructor Documentation

#### 15.46.2.1 H\_list\_item\_t()

```
template<typename ELEM_TYPE >
cxx::H_list_item_t< ELEM_TYPE >::H_list_item_t ( ) [inline]
```

Constructor.

Creates an element that is not in any list.

Definition at line [41](#) of file [hlist](#).

#### 15.46.2.2 ~H\_list\_item\_t()

```
template<typename ELEM_TYPE >
cxx::H_list_item_t< ELEM_TYPE >::~~H_list_item_t ( ) [inline], [noexcept]
```

Destructor.

Automatically removes the element from any list it still might be enchainned in.

Definition at line [48](#) of file [hlist](#).

The documentation for this class was generated from the following file:

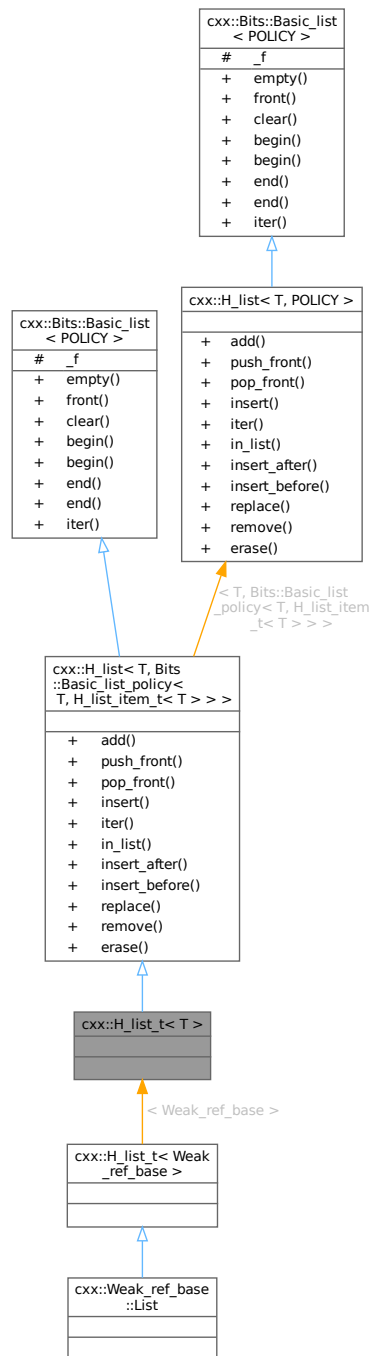
- [I4/cxx/hlist](#)

## 15.47 cxx::H\_list\_t< T > Struct Template Reference

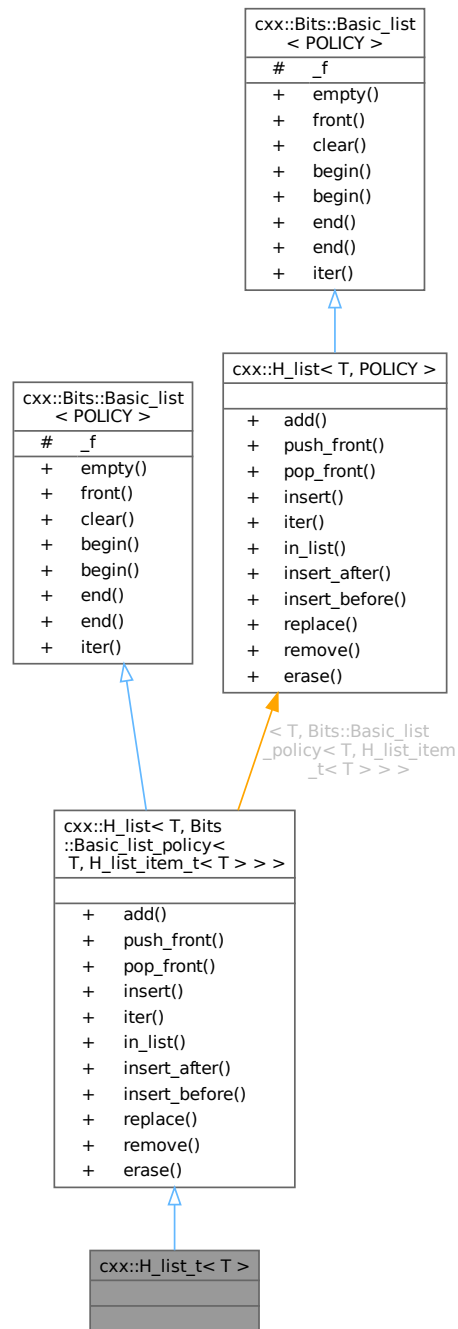
Double-linked list of typed [H\\_list\\_item\\_t](#) elements.

```
#include <hlist>
```

Inheritance diagram for cxx::H\_list\_t< T >:



Collaboration diagram for `cxx::H_list_t< T >`:



#### Additional Inherited Members

#### Public Member Functions inherited from

**`cxx::H_list< T, Bits::Basic_list_policy< T, H_list_item_t< T > > >`**

- `void add (T *e)`

- *Add element to the front of the list.*
- void **push\_front** (T \*e)  
*Add element to the front of the list.*
- T \* **pop\_front** ()  
*Remove and return the head element of the list.*
- Iterator **insert** (T \*e, Iterator const &pred)  
*Insert an element at the iterator position.*

### Public Member Functions inherited from `cxx::Bits::Basic_list< POLICY >`

- bool **empty** () const  
*Check if the list is empty.*
- Value\_type **front** () const  
*Return the first element in the list.*
- void **clear** ()  
*Remove all elements from the list.*
- Iterator **begin** ()  
*Return an iterator to the beginning of the list.*
- Const\_iterator **begin** () const  
*Return a const iterator to the beginning of the list.*
- Const\_iterator **end** () const  
*Return a const iterator to the end of the list.*
- Iterator **end** ()  
*Return an iterator to the end of the list.*

### Static Public Member Functions inherited from `cxx::H_list< T, Bits::Basic_list_policy< T, H_list_item_t< T > > >`

- static Iterator **iter** (T \*c)  
*Return an iterator for an arbitrary list element.*
- static bool **in\_list** (T const \*e)  
*Check if the given element is currently part of a list.*
- static Iterator **insert\_after** (T \*e, Iterator const &pred)  
*Insert an element after the iterator position.*
- static void **insert\_before** (T \*e, Iterator const &succ)  
*Insert an element before the iterator position.*
- static void **replace** (T \*p, T \*e)  
*Replace an element in a list with a new element.*
- static void **remove** (T \*e)  
*Remove the given element from its list.*
- static Iterator **erase** (Iterator const &e)  
*Remove the element at the given iterator position.*

### Static Public Member Functions inherited from `cxx::Bits::Basic_list< POLICY >`

- static Const\_iterator **iter** (Const\_value\_type c)  
*Return a const iterator that begins at the given element.*

## Protected Attributes inherited from [cxx::Bits::Basic\\_list< POLICY >](#)

- POLICY::Head\_type \_f  
*Pointer to front of the list.*

### 15.47.1 Detailed Description

```
template<typename T>
struct cxx::H_list_t< T >
```

Double-linked list of typed [H\\_list\\_item\\_t](#) elements.

#### Note

H\_lists are not self-cleaning. Elements that are still chained during destruction are not removed and will therefore be in an undefined state after the destruction.

Definition at line [259](#) of file [hlist](#).

The documentation for this struct was generated from the following file:

- [l4/cxx/hlist](#)

## 15.48 [cxx::List< D, Alloc >](#) Class Template Reference

Doubly linked list, with internal allocation.

```
#include <list>
```

Collaboration diagram for [cxx::List< D, Alloc >](#):

<a href="#">cxx::List&lt; D, Alloc &gt;</a>
<ul style="list-style-type: none"> <li>+ <a href="#">push_back()</a></li> <li>+ <a href="#">push_front()</a></li> <li>+ <a href="#">remove()</a></li> <li>+ <a href="#">size()</a></li> <li>+ <a href="#">operator[]()</a></li> <li>+ <a href="#">operator[]()</a></li> <li>+ <a href="#">items()</a></li> </ul>

## Data Structures

- class `Iter`  
*Iterator.*

## Public Member Functions

- void **push\_back** (D const &d) noexcept  
*Add element at the end of the list.*
- void **push\_front** (D const &d) noexcept  
*Add element at the beginning of the list.*
- void **remove** (`Iter` const &i) noexcept  
*Remove element pointed to by the iterator.*
- unsigned long **size** () const noexcept  
*Get the length of the list.*
- D const & **operator[]** (unsigned long idx) const noexcept  
*Random access.*
- D & **operator[]** (unsigned long idx) noexcept  
*Random access.*
- `Iter` **items** () noexcept  
*Get iterator for the list elements.*

### 15.48.1 Detailed Description

`template<typename D, template< typename A > class Alloc = New_allocator>`  
`class cxx::List< D, Alloc >`

Doubly linked list, with internal allocation.

Container for items of type D, implemented by a doubly linked list. Alloc defines the allocator policy.

Definition at line 334 of file `list`.

### 15.48.2 Member Function Documentation

#### 15.48.2.1 `operator[]()` [1/2]

```
template<typename D , template< typename A > class Alloc = New_allocator>
D const & cxx::List< D, Alloc >::operator[] (
    unsigned long idx ) const [inline], [noexcept]
```

Random access.

Complexity is O(n).

Definition at line 404 of file `list`.

### 15.48.2.2 operator[]() [2/2]

```
template<typename D , template< typename A > class Alloc = New_allocator>
D & cxx::List< D, Alloc >::operator[] (
    unsigned long idx ) [inline], [noexcept]
```

Random access.

Complexity is O(n).

Definition at line 408 of file [list](#).

The documentation for this class was generated from the following file:

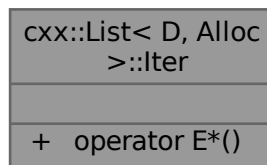
- l4/cxx/list

## 15.49 cxx::List< D, Alloc >::Iter Class Reference

Iterator.

```
#include <list>
```

Collaboration diagram for cxx::List< D, Alloc >::Iter:



### Public Member Functions

- **operator E\* ()** const noexcept  
*operator for testing validity (syntactically equal to pointers)*

### 15.49.1 Detailed Description

```
template<typename D, template< typename A > class Alloc = New_allocator>
class cxx::List< D, Alloc >::Iter
```

Iterator.

Forward and backward iterable.

Definition at line 354 of file [list](#).

The documentation for this class was generated from the following file:

- l4/cxx/list

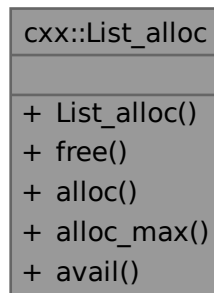


## 15.50 `cxx::List_alloc` Class Reference

Standard list-based allocator.

```
#include <list_alloc>
```

Collaboration diagram for `cxx::List_alloc`:



### Public Member Functions

- [List\\_alloc\(\)](#)  
*Initializes an empty list allocator.*
- void [free](#) (void \*block, unsigned long size, bool initial\_free=false)  
*Return a free memory block to the allocator.*
- void \* [alloc](#) (unsigned long size, unsigned long align, unsigned long lower=0, unsigned long upper=~0UL)  
*Allocate a memory block.*
- void \* [alloc\\_max](#) (unsigned long min, unsigned long \*max, unsigned long align, unsigned granularity, unsigned long lower=0, unsigned long upper=~0UL)  
*Allocate a memory block of  $min \leq size \leq max$ .*
- unsigned long [avail](#) ()  
*Get the amount of available memory.*

### 15.50.1 Detailed Description

Standard list-based allocator.

Definition at line 32 of file [list\\_alloc](#).

### 15.50.2 Constructor & Destructor Documentation

#### 15.50.2.1 List\_alloc()

```
cxx::List_alloc::List_alloc ( ) [inline]
```

Initializes an empty list allocator.

#### Note

To initialize the allocator with available memory use the [free\(\)](#) function.

Definition at line 57 of file [list\\_alloc](#).

### 15.50.3 Member Function Documentation

#### 15.50.3.1 alloc()

```
void * cxx::List_alloc::alloc (
    unsigned long size,
    unsigned long align,
    unsigned long lower = 0,
    unsigned long upper = ~0UL ) [inline]
```

Allocate a memory block.

##### Parameters

<i>size</i>	Size of the memory block.
<i>align</i>	Alignment constraint.
<i>lower</i>	Lower bound of the physical region the memory block should be allocated from.
<i>upper</i>	Upper bound of the physical region the memory block should be allocated from, value is inclusive.

##### Returns

Pointer to memory block

##### Precondition

$0 < \text{size} \leq \sim 0UL - 32$ .

Definition at line 400 of file [list\\_alloc](#).

#### 15.50.3.2 alloc\_max()

```
void * cxx::List_alloc::alloc_max (
    unsigned long min,
    unsigned long * max,
    unsigned long align,
    unsigned granularity,
    unsigned long lower = 0,
    unsigned long upper = ~0UL ) [inline]
```

Allocate a memory block of  $\text{min} \leq \text{size} \leq \text{max}$ .

##### Parameters

	<i>min</i>	Minimal size to allocate (in bytes).
<i>in, out</i>	<i>max</i>	Maximum size to allocate (in bytes). The actual allocated size is returned here.
	<i>align</i>	Alignment constraint.
	<i>granularity</i>	Granularity to use for the allocation (power of 2).
	<i>lower</i>	Lower bound of the physical region the memory block should be allocated from.
	<i>upper</i>	Upper bound of the physical region the memory block should be allocated from, value is inclusive.

**Returns**

Pointer to memory block

**Precondition**

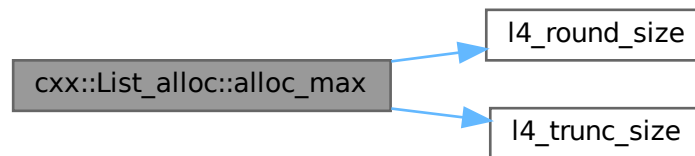
$0 < \text{min} \leq \sim 0\text{UL} - 32$ .

$0 < \text{max}$ .

Definition at line 280 of file [list\\_alloc](#).

References [l4\\_round\\_size\(\)](#), and [l4\\_trunc\\_size\(\)](#).

Here is the call graph for this function:

**15.50.3.3 avail()**

```
unsigned long cxx::List_alloc::avail ( ) [inline]
```

Get the amount of available memory.

**Returns**

Available memory in bytes

Definition at line 488 of file [list\\_alloc](#).

**15.50.3.4 free()**

```
void cxx::List_alloc::free (
    void * block,
    unsigned long size,
    bool initial_free = false ) [inline]
```

Return a free memory block to the allocator.

**Parameters**

<i>block</i>	Pointer to memory block.
<i>size</i>	Size of memory block.
<i>initial_free</i>	Set to true for putting fresh memory to the allocator. This will enforce alignment on that memory.

**Precondition**

`block` must not be NULL.

`2 * sizeof(void *) <= size <= ~0UL - 32.`

Definition at line 239 of file [list\\_alloc](#).

The documentation for this class was generated from the following file:

- `I4/cxx/list_alloc`

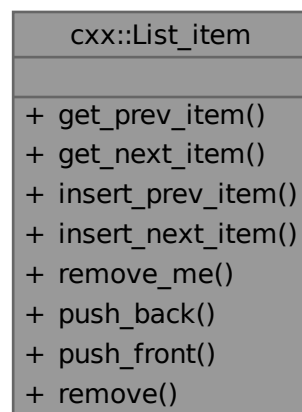
## 15.51 cxx::List\_item Class Reference

Basic list item.

```
#include <list>
```

Inherited by `cxx::T_list_item< T >`.

Collaboration diagram for `cxx::List_item`:

**Data Structures**

- class [Iter](#)  
*Iterator for a list of ListItem-s.*
- class [T\\_iter](#)  
*Iterator for derived classes from ListItem.*

## Public Member Functions

- [List\\_item](#) \* **get\_prev\_item** () const noexcept  
*Get previous item.*
- [List\\_item](#) \* **get\_next\_item** () const noexcept  
*Get next item.*
- void **insert\_prev\_item** ([List\\_item](#) \*p) noexcept  
*Insert item p before this item.*
- void **insert\_next\_item** ([List\\_item](#) \*p) noexcept  
*Insert item p after this item.*
- void **remove\_me** () noexcept  
*Remove this item from the list.*

## Static Public Member Functions

- template<typename C , typename N >  
static C \* **push\_back** (C \*head, N \*p) noexcept  
*Append item to a list.*
- template<typename C , typename N >  
static C \* **push\_front** (C \*head, N \*p) noexcept  
*Prepend item to a list.*
- template<typename C , typename N >  
static C \* **remove** (C \*head, N \*p) noexcept  
*Remove item from a list.*

### 15.51.1 Detailed Description

Basic list item.

Basic item that can be member of a doubly linked, cyclic list.

Definition at line 37 of file [list](#).

### 15.51.2 Member Function Documentation

#### 15.51.2.1 push\_back()

```
template<typename C , typename N >
C * cxx::List_item::push_back (
    C * head,
    N * p ) [inline], [static], [noexcept]
```

Append item to a list.

Convenience function for empty-head corner case.

#### Parameters

<i>head</i>	Pointer to the current list head.
<i>p</i>	Pointer to new item.

**Returns**

the pointer to the new head.

Definition at line 248 of file [list](#).

References [insert\\_prev\\_item\(\)](#).

Referenced by [cxx::List< D, Alloc >::push\\_back\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.51.2.2 push\_front()**

```

template<typename C , typename N >
C * cxx::List_item::push_front (
    C * head,
    N * p ) [inline], [static], [noexcept]
  
```

Prepend item to a list.

Convenience function for empty-head corner case.

**Parameters**

<i>head</i>	pointer to the current list head.
<i>p</i>	pointer to new item.

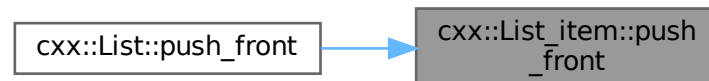
**Returns**

the pointer to the new head.

Definition at line 259 of file [list](#).

Referenced by [cxx::List< D, Alloc >::push\\_front\(\)](#).

Here is the caller graph for this function:

**15.51.2.3 remove()**

```

template<typename C , typename N >
C * cxx::List_item::remove (
    C * head,
    N * p ) [inline], [static], [noexcept]
  
```

Remove item from a list.

Convenience function for remove-head corner case.

**Parameters**

<i>head</i>	pointer to the current list head.
<i>p</i>	pointer to the item to remove.

**Returns**

the pointer to the new head.

Definition at line 269 of file [list](#).

Referenced by [cxx::List< D, Alloc >::remove\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

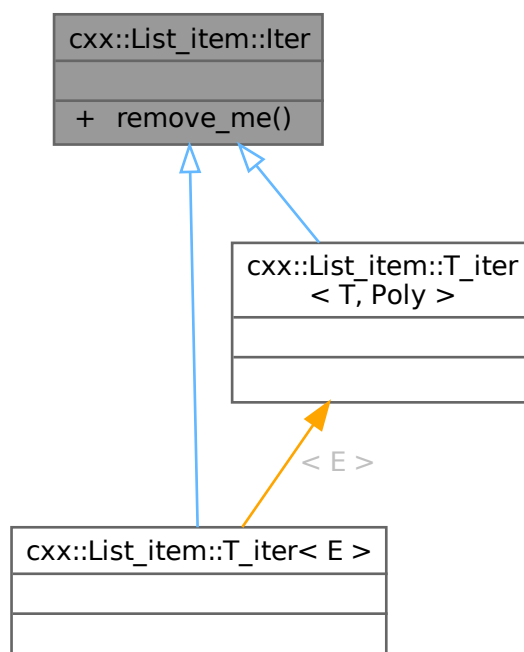
- l4/cxx/list

## 15.52 cxx::List\_item::Iter Class Reference

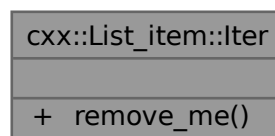
Iterator for a list of ListItem-s.

```
#include <list>
```

Inheritance diagram for cxx::List\_item::Iter:



Collaboration diagram for cxx::List\_item::Iter:





**Public Member Functions**

- [List\\_item](#) \* **remove\_me** () noexcept  
*Remove item pointed to by iterator, and return pointer to element.*

**15.52.1 Detailed Description**

Iterator for a list of ListItem-s.

The Iterator iterates till it finds the first element again.

Definition at line 45 of file [list](#).

The documentation for this class was generated from the following file:

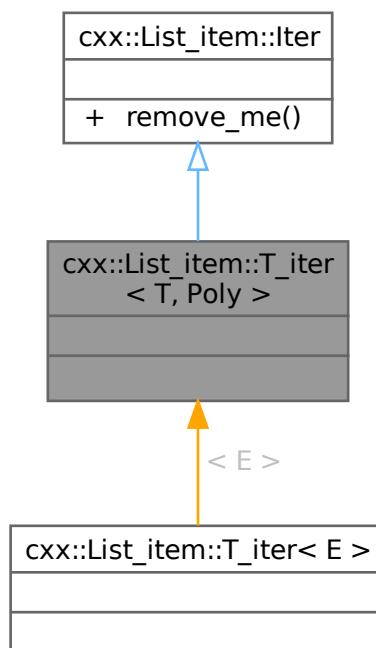
- l4/cxx/list

**15.53 cxx::List\_item::T\_iter< T, Poly > Class Template Reference**

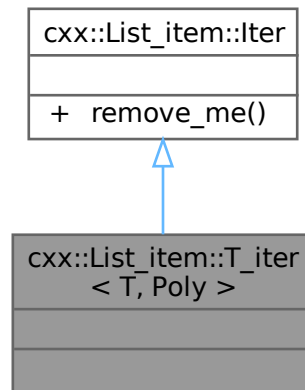
Iterator for derived classes from ListItem.

```
#include <list>
```

Inheritance diagram for cxx::List\_item::T\_iter< T, Poly >:



Collaboration diagram for `cxx::List_item::T_iter< T, Poly >`:



#### Additional Inherited Members

#### Public Member Functions inherited from `cxx::List_item::Iter`

- `List_item * remove_me ()` noexcept  
*Remove item pointed to by iterator, and return pointer to element.*

### 15.53.1 Detailed Description

```
template<typename T, bool Poly = false>
class cxx::List_item::T_iter< T, Poly >
```

Iterator for derived classes from `ListItem`.

Allows direct access to derived classes by `*` operator.

Example: `class Foo : public ListItem { public: typedef T_iter<Foo> Iter; ... };`

Definition at line 119 of file `list`.

The documentation for this class was generated from the following file:

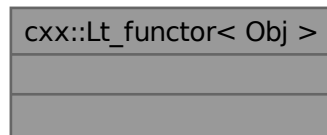
- `I4/cxx/list`

## 15.54 cxx::Lt\_functor< Obj > Struct Template Reference

Generic comparator class that defaults to the less-than operator.

```
#include <std_ops>
```

Collaboration diagram for cxx::Lt\_functor< Obj >:



### 15.54.1 Detailed Description

```
template<typename Obj>
struct cxx::Lt_functor< Obj >
```

Generic comparator class that defaults to the less-than operator.

Definition at line 29 of file [std\\_ops](#).

The documentation for this struct was generated from the following file:

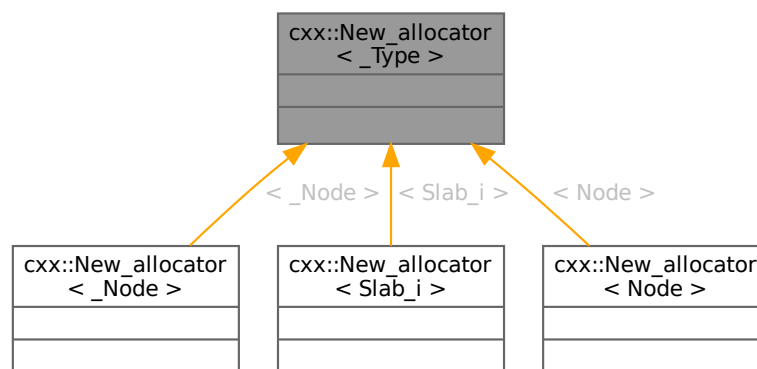
- I4/cxx/std\_ops

## 15.55 cxx::New\_allocator< \_Type > Class Template Reference

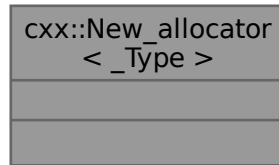
Standard allocator based on `operator new ()`.

```
#include <std_alloc>
```

Inheritance diagram for cxx::New\_allocator< \_Type >:



Collaboration diagram for `cxx::New_allocator<_Type>`:



### 15.55.1 Detailed Description

```
template<typename _Type>
class cxx::New_allocator<_Type>
```

Standard allocator based on `operator new ()`.

This allocator is the default allocator used for the *cxx Containers*, such as [cxx::Avl\\_set](#) and [cxx::Avl\\_map](#), to allocate the internal data structures.

Definition at line 67 of file [std\\_alloc](#).

The documentation for this class was generated from the following file:

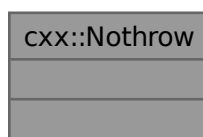
- `I4/cxx/std_alloc`

## 15.56 cxx::Nothrow Class Reference

Helper type to distinguish the `oeprator new` version that does not throw exceptions.

```
#include <std_alloc>
```

Collaboration diagram for `cxx::Nothrow`:



### 15.56.1 Detailed Description

Helper type to distinguish the `operator new` version that does not throw exceptions.

Definition at line 30 of file [std\\_alloc](#).

The documentation for this class was generated from the following file:

- `I4/cxx/std_alloc`

## 15.57 `cxx::Pair< First, Second >` Struct Template Reference

[Pair](#) of two values.

```
#include <pair>
```

Collaboration diagram for `cxx::Pair< First, Second >`:

<code>cxx::Pair&lt; First, Second &gt;</code>	
+	<code>first</code>
+	<code>second</code>
+	<code>Pair()</code>
+	<code>Pair()</code>
+	<code>Pair()</code>

### Public Types

- typedef First **First\_type**  
*Type of first value.*
- typedef Second **Second\_type**  
*Type of second value.*

### Public Member Functions

- `template<typename A1 , typename A2 >`  
[Pair](#) (A1 &&[first](#), A2 &&[second](#))  
*Create a pair from the two values.*
- `template<typename A1 >`  
[Pair](#) (A1 &&[first](#))  
*Create a pair, default constructing the second value.*
- **Pair** ()=default  
*Default construction.*

## Data Fields

- First **first**  
*First value.*
- Second **second**  
*Second value.*

### 15.57.1 Detailed Description

```
template<typename First, typename Second>
struct cxx::Pair< First, Second >
```

[Pair](#) of two values.

Standard container for a pair of values.

#### Parameters

<i>First</i>	Type of the first value.
<i>Second</i>	Type of the second value.

Definition at line [38](#) of file [pair](#).

### 15.57.2 Constructor & Destructor Documentation

#### 15.57.2.1 [Pair\(\)](#) [1/2]

```
template<typename First , typename Second >
template<typename A1 , typename A2 >
cxx::Pair< First, Second >::Pair (
    A1 && first,
    A2 && second ) [inline]
```

Create a pair from the two values.

#### Parameters

<i>first</i>	The first value.
<i>second</i>	The second value.

Definition at line [56](#) of file [pair](#).

#### 15.57.2.2 [Pair\(\)](#) [2/2]

```
template<typename First , typename Second >
template<typename A1 >
cxx::Pair< First, Second >::Pair (
    A1 && first ) [inline]
```

Create a pair, default constructing the second value.

## Parameters

<i>first</i>	The first value.
--------------	------------------

Definition at line 64 of file [pair](#).

The documentation for this struct was generated from the following file:

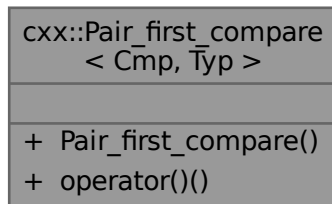
- [l4/cxx/pair](#)

## 15.58 cxx::Pair\_first\_compare< Cmp, Typ > Class Template Reference

Comparison functor for [Pair](#).

```
#include <pair>
```

Collaboration diagram for cxx::Pair\_first\_compare< Cmp, Typ >:



### Public Member Functions

- [Pair\\_first\\_compare](#) (Cmp const &cmp=Cmp())  
*Construction.*
- bool [operator\(\)](#) (Typ const &l, Typ const &r) const  
*Do the comaprison based on the first value.*

### 15.58.1 Detailed Description

```
template<typename Cmp, typename Typ>
class cxx::Pair_first_compare< Cmp, Typ >
```

Comparison functor for [Pair](#).



## Parameters

<i>Cmp</i>	Comparison functor for the first value of the pair.
<i>Typ</i>	The pair type.

This functor can be used to compare [Pair](#) values with respect to the first value.

Definition at line [85](#) of file [pair](#).

## 15.58.2 Constructor & Destructor Documentation

### 15.58.2.1 `Pair_first_compare()`

```
template<typename Cmp , typename Typ >
cxx::Pair_first_compare< Cmp, Typ >::Pair_first_compare (
    Cmp const & cmp = Cmp() ) [inline]
```

Construction.

## Parameters

<i>cmp</i>	The comparison functor used for the first value.
------------	--

Definition at line [95](#) of file [pair](#).

## 15.58.3 Member Function Documentation

### 15.58.3.1 `operator>()()`

```
template<typename Cmp , typename Typ >
bool cxx::Pair_first_compare< Cmp, Typ >::operator() (
    Typ const & l,
    Typ const & r ) const [inline]
```

Do the comaprison based on the first value.

## Parameters

<i>l</i>	The lefthand value.
<i>r</i>	The righthand value.

Definition at line [102](#) of file [pair](#).

The documentation for this class was generated from the following file:

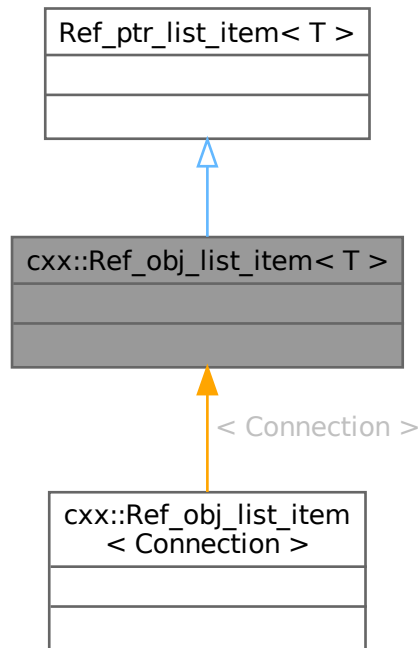
- [I4/cxx/pair](#)

## 15.59 cxx::Ref\_obj\_list\_item< T > Struct Template Reference

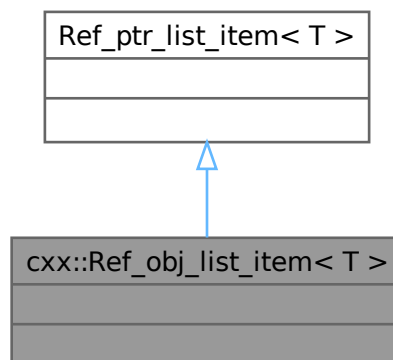
Item for list linked via [cxx::Ref\\_ptr](#) with default reference counting.

```
#include <ref_ptr_list>
```

Inheritance diagram for cxx::Ref\_obj\_list\_item< T >:



Collaboration diagram for cxx::Ref\_obj\_list\_item< T >:



### 15.59.1 Detailed Description

```
template<typename T>
struct cxx::Ref_obj_list_item< T >
```

Item for list linked via [cxx::Ref\\_ptr](#) with default reference counting.

Definition at line 27 of file [ref\\_ptr\\_list](#).

The documentation for this struct was generated from the following file:

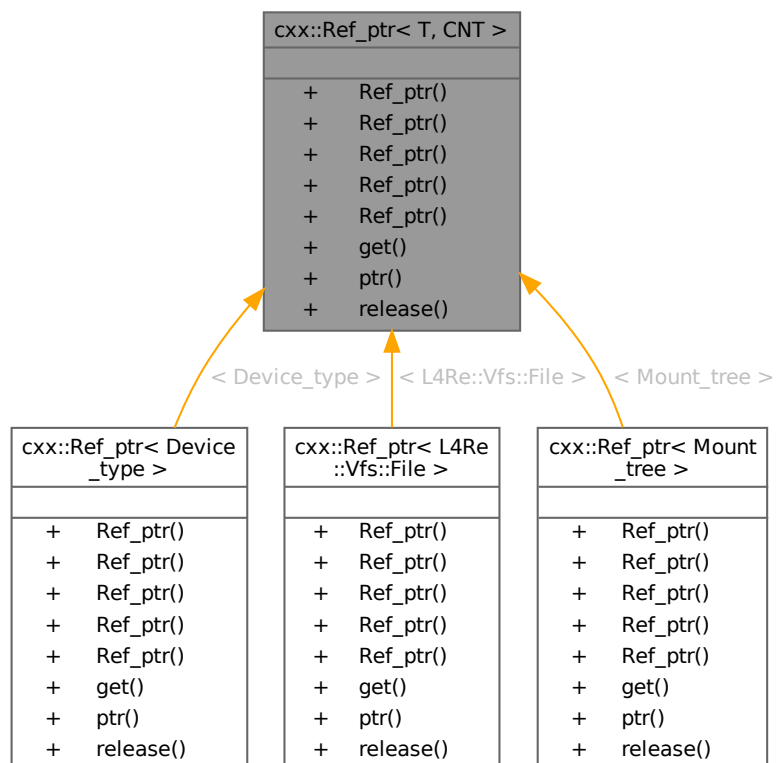
- [l4/cxx/ref\\_ptr\\_list](#)

## 15.60 cxx::Ref\_ptr< T, CNT > Class Template Reference

A reference-counting pointer with automatic cleanup.

```
#include <ref_ptr>
```

Inheritance diagram for cxx::Ref\_ptr< T, CNT >:



Collaboration diagram for `cxx::Ref_ptr< T, CNT >`:

<code>cxx::Ref_ptr&lt; T, CNT &gt;</code>	
+	<code>Ref_ptr()</code>
+	<code>Ref_ptr()</code>
+	<code>Ref_ptr()</code>
+	<code>Ref_ptr()</code>
+	<code>Ref_ptr()</code>
+	<code>get()</code>
+	<code>ptr()</code>
+	<code>release()</code>

## Public Member Functions

- **`Ref_ptr ()`** noexcept  
*Default constructor creates a pointer with no managed object.*
- **`Ref_ptr (Wp const &o)`** noexcept  
*Create a shared pointer from a weak pointer.*
- **`Ref_ptr (decltype(nullptr) n)`** noexcept  
*allow creation from `nullptr`*
- `template<typename X >`  
**`Ref_ptr (X *o)`** noexcept  
*Create a shared pointer from a raw pointer.*
- **`Ref_ptr (T *o, bool d)`** noexcept  
*Create a shared pointer from a raw pointer without creating a new reference.*
- `T * get ()` const noexcept  
*Return a raw pointer to the object this shared pointer points to.*
- `T * ptr ()` const noexcept  
*Return a raw pointer to the object this shared pointer points to.*
- `T * release ()` noexcept  
*Release the shared pointer without removing the reference.*

### 15.60.1 Detailed Description

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
class cxx::Ref_ptr< T, CNT >
```

A reference-counting pointer with automatic cleanup.

## Template Parameters

<i>T</i>	Type of object the pointer points to.
<i>CNT</i>	Type of management class that manages the life time of the object.

This pointer is similar to the standard C++-11 `shared_ptr` but it does the reference counting directly in the object being pointed to, so that no additional management structures need to be allocated from the heap.

Classes that use this pointer type must implement two functions:

```
int remove_ref()
```

is called when a reference is removed and must return 0 when there are no further references to the object.

```
void add_ref()
```

is called when another `ref_ptr` to the object is created.

`Ref_obj` provides a simple implementation of this interface from which classes may inherit.

## Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 81 of file [ref\\_ptr](#).

## 15.60.2 Constructor &amp; Destructor Documentation

15.60.2.1 `Ref_ptr()` [1/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
cxx::Ref_ptr< T, CNT >::Ref_ptr (
    Wp const & o ) [inline], [noexcept]
```

Create a shared pointer from a weak pointer.

Increases references.

Definition at line 99 of file [ref\\_ptr](#).

15.60.2.2 `Ref_ptr()` [2/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
template<typename X >
cxx::Ref_ptr< T, CNT >::Ref_ptr (
    X * o ) [inline], [explicit], [noexcept]
```

Create a shared pointer from a raw pointer.

In contrast to C++11 `shared_ptr` it is safe to use this constructor multiple times and have the same reference counter.

Definition at line 112 of file [ref\\_ptr](#).

15.60.2.3 `Ref_ptr()` [3/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
cxx::Ref_ptr< T, CNT >::Ref_ptr (
    T * o,
    bool d ) [inline], [noexcept]
```

Create a shared pointer from a raw pointer without creating a new reference.

**Parameters**

<i>o</i>	Pointer to the object.
<i>d</i>	Dummy parameter to select this constructor at compile time. The value may be true or false.

This is the counterpart to [release\(\)](#).

Definition at line [125](#) of file [ref\\_ptr](#).

### 15.60.3 Member Function Documentation

#### 15.60.3.1 `get()`

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T * cxx::Ref\_ptr< T, CNT >::get ( ) const [inline], [noexcept]
```

Return a raw pointer to the object this shared pointer points to.

This does not release the pointer or decrease the reference count.

Definition at line [132](#) of file [ref\\_ptr](#).

#### 15.60.3.2 `ptr()`

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T * cxx::Ref\_ptr< T, CNT >::ptr ( ) const [inline], [noexcept]
```

Return a raw pointer to the object this shared pointer points to.

This does not release the pointer or decrease the reference count.

Definition at line [138](#) of file [ref\\_ptr](#).

#### 15.60.3.3 `release()`

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T * cxx::Ref\_ptr< T, CNT >::release ( ) [inline], [noexcept]
```

Release the shared pointer without removing the reference.

**Returns**

A raw pointer to the managed object.

Definition at line [149](#) of file [ref\\_ptr](#).

The documentation for this class was generated from the following file:

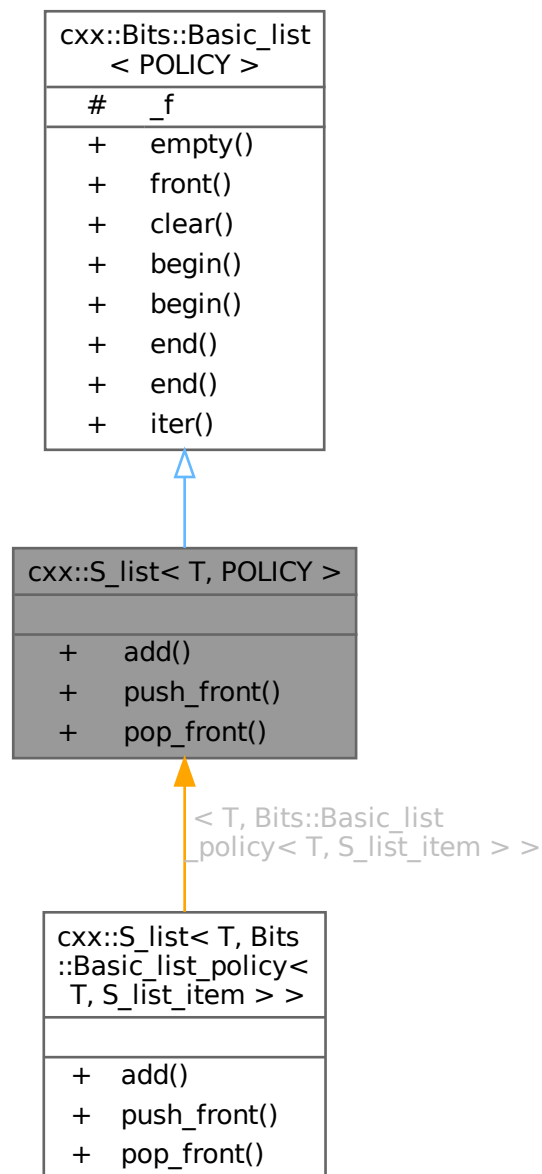
- `I4/cxx/ref_ptr`

## 15.61 cxx::S\_list< T, POLICY > Class Template Reference

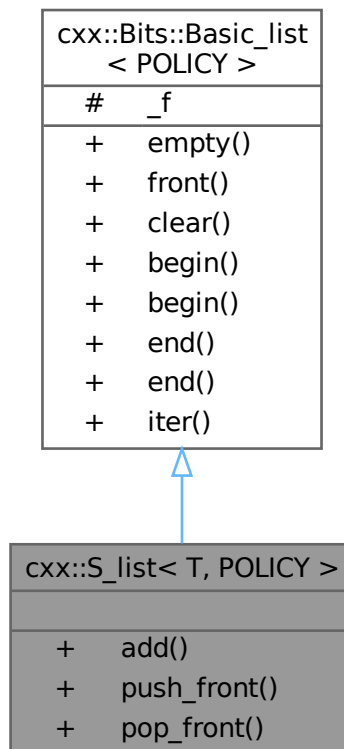
Simple single-linked list.

```
#include <slist>
```

Inheritance diagram for cxx::S\_list< T, POLICY >:



Collaboration diagram for `cxx::S_list< T, POLICY >`:



### Public Member Functions

- void **add** (T \*e)  
*Add an element to the front of the list.*
- void **push\_front** (T \*e)  
*Add an element to the front of the list.*
- T \* **pop\_front** ()  
*Remove and return the head element of the list.*

### Public Member Functions inherited from `cxx::Bits::Basic_list< POLICY >`

- bool **empty** () const  
*Check if the list is empty.*
- Value\_type **front** () const  
*Return the first element in the list.*
- void **clear** ()  
*Remove all elements from the list.*
- Iterator **begin** ()  
*Return an iterator to the beginning of the list.*
- Const\_iterator **begin** () const



*Return a const iterator to the beginning of the list.*

- Const\_iterator **end** () const

*Return a const iterator to the end of the list.*

- Iterator **end** ()

*Return an iterator to the end of the list.*

## Additional Inherited Members

## Static Public Member Functions inherited from `cxx::Bits::Basic_list< POLICY >`

- static Const\_iterator `iter` (Const\_value\_type c)

*Return a const iterator that begins at the given element.*

## Protected Attributes inherited from `cxx::Bits::Basic_list< POLICY >`

- POLICY::Head\_type \_f

*Pointer to front of the list.*

### 15.61.1 Detailed Description

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, S_list_item >>
class cxx::S_list< T, POLICY >
```

Simple single-linked list.

#### Template Parameters

<i>T</i>	Type of elements saved in the list. Must inherit from <code>cxx::S_list_item</code>
----------	---

Definition at line 51 of file `slist`.

### 15.61.2 Member Function Documentation

#### 15.61.2.1 `pop_front()`

```
template<typename T , typename POLICY = Bits::Basic_list_policy< T, S_list_item >>
T * cxx::S_list< T, POLICY >::pop_front ( ) [inline]
```

Remove and return the head element of the list.

**Precondition**

The list must not be empty or the behaviour will be undefined.

Definition at line 100 of file [slist](#).

References [cxx::Bits::Basic\\_list< POLICY >::\\_f](#), and [cxx::Bits::Basic\\_list< POLICY >::front\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

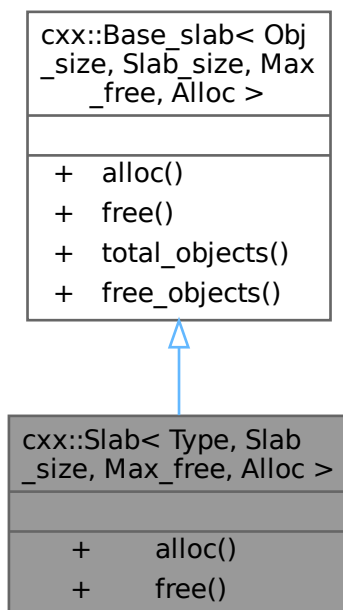
- `I4/cxx/slist`

## 15.62 `cxx::Slab< Type, Slab_size, Max_free, Alloc >` Class Template Reference

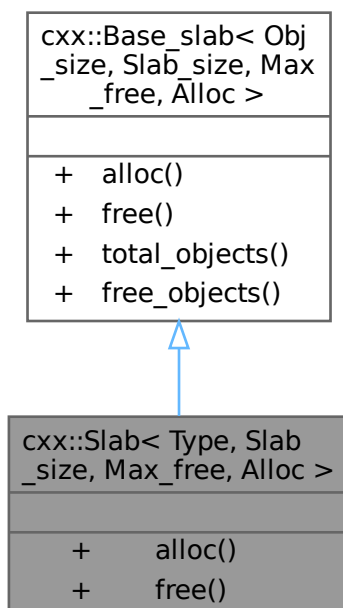
[Slab](#) allocator for object of type `Type`.

```
#include <slab_alloc>
```

Inheritance diagram for cxx::Slab< Type, Slab\_size, Max\_free, Alloc >:



Collaboration diagram for cxx::Slab< Type, Slab\_size, Max\_free, Alloc >:



## Public Member Functions

- `Type * alloc ()` noexcept  
*Allocate an object of type `Type`.*
- `void free (Type *o)` noexcept  
*Free the object addressed by `o`.*

## Public Member Functions inherited from

`cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >`

- `void * alloc ()` noexcept  
*Allocate a new object.*
- `void free (void *_o)` noexcept  
*Free the given object (`_o`).*
- `unsigned total\_objects ()` const noexcept  
*Get the total number of objects managed by the slab allocator.*
- `unsigned free\_objects ()` const noexcept  
*Get the number of objects which can be allocated before a new empty slab needs to be added to the slab allocator.*

## Additional Inherited Members

## Public Types inherited from `cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >`

- enum { `object\_size = Obj\_size` , `slab\_size = Slab\_size` , `objects\_per\_slab = (Slab\_size - sizeof(Slab\_head)) / object\_size` , `max\_free\_slabs = Max\_free` }
- typedef `Alloc< Slab\_i >` `Slab\_alloc`  
*Type of the backend allocator.*

## 15.62.1 Detailed Description

`template<typename Type, int Slab\_size = L4\_PAGESIZE, int Max\_free = 2, template< typename A > class Alloc = New\_allocator>`

`class cxx::Slab< Type, Slab\_size, Max\_free, Alloc >`

`Slab` allocator for object of type `Type`.

### Template Parameters

<i>Type</i>	The type of the objects to manage.
<i>Slab_size</i>	Size of a slab.
<i>Max_free</i>	The maximum number of free slabs.
<i>Alloc</i>	The allocator for the slabs.

Definition at line [346](#) of file `slab\_alloc`.

## 15.62.2 Member Function Documentation

### 15.62.2.1 `alloc()`

```
template<typename Type , int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A
> class Alloc = New_allocator>
Type * cxx::Slab< Type, Slab_size, Max_free, Alloc >::alloc ( ) [inline], [noexcept]
```

Allocate an object of type `Type`.

#### Returns

A pointer to the object just allocated, or 0 on failure.

#### Note

The user is responsible for initializing the object.

Definition at line 366 of file `slab_alloc`.

### 15.62.2.2 `free()`

```
template<typename Type , int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A
> class Alloc = New_allocator>
void cxx::Slab< Type, Slab_size, Max_free, Alloc >::free (
    Type * o ) [inline], [noexcept]
```

Free the object addressed by `o`.

#### Parameters

<code>o</code>	The pointer to the object to free.
----------------	------------------------------------

#### Precondition

The object must have been allocated with this allocator.

Definition at line 377 of file `slab_alloc`.

References `cxx::Slab< Type, Slab_size, Max_free, Alloc >::free()`.

Referenced by `cxx::Slab< Type, Slab_size, Max_free, Alloc >::free()`.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

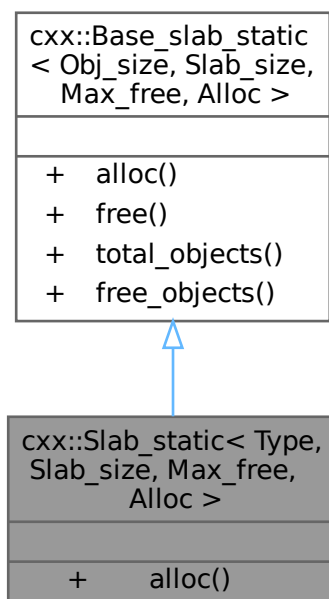
- `I4/cxx/slab_alloc`

## 15.63 `cxx::Slab_static< Type, Slab_size, Max_free, Alloc > Class` Template Reference

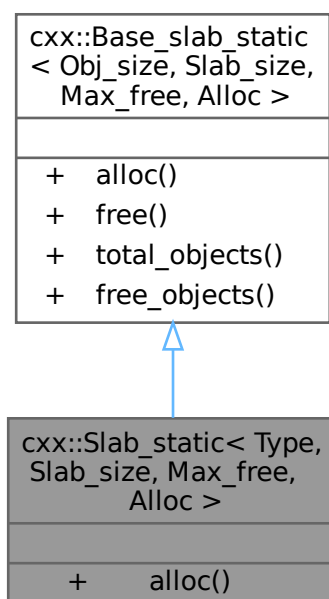
Merged slab allocator (allocators for objects of the same size are merged together).

```
#include <slab_alloc>
```

Inheritance diagram for cxx::Slab\_static< Type, Slab\_size, Max\_free, Alloc >:



Collaboration diagram for cxx::Slab\_static< Type, Slab\_size, Max\_free, Alloc >:



## Public Member Functions

- `Type * alloc () noexcept`  
*Allocate an object of type `Type`.*

## Public Member Functions inherited from

`cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >`

- `void * alloc () noexcept`  
*Allocate an object.*
- `void free (void *p) noexcept`  
*Free the given object (`p`).*
- `unsigned total_objects () const noexcept`  
*Get the total number of objects managed by the slab allocator.*
- `unsigned free_objects () const noexcept`  
*Get the number of free objects in the slab allocator.*

## Additional Inherited Members

## Public Types inherited from

`cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >`

- enum { `object_size` = `Obj_size` , `slab_size` = `Slab_size` , `objects_per_slab` = `_A::objects_per_slab` , `max_free_slabs` = `Max_free` }

## 15.63.1 Detailed Description

```
template<typename Type, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class
Alloc = New_allocator>
class cxx::Slab_static< Type, Slab_size, Max_free, Alloc >
```

Merged slab allocator (allocators for objects of the same size are merged together).

### Template Parameters

<i>Type</i>	The type of the objects to manage.
<i>Slab_size</i>	The size of a slab.
<i>Max_free</i>	The maximum number of free slabs.
<i>Alloc</i>	The allocator for the slabs.

This slab allocator class is useful for merging slab allocators with the same parameters (equal `sizeof(Type)`, `Slab_size`, `Max_free`, and `Alloc` parameters) together and share the overhead for the slab caches among all equal-sized objects.

Definition at line 476 of file `slab_alloc`.



## 15.63.2 Member Function Documentation

### 15.63.2.1 `alloc()`

```
template<typename Type , int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A  
> class Alloc = New_allocator>
```

```
Type * cxx::Slab_static< Type, Slab_size, Max_free, Alloc >::alloc ( ) [inline], [noexcept]
```

Allocate an object of type `Type`.

#### Returns

A pointer to the just allocated object, or 0 on failure.

#### Note

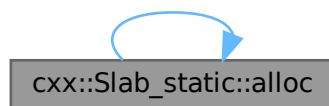
The object is not zeroed out by the slab allocator.

Definition at line 489 of file `slab_alloc`.

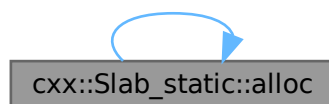
References `cxx::Slab_static< Type, Slab_size, Max_free, Alloc >::alloc()`.

Referenced by `cxx::Slab_static< Type, Slab_size, Max_free, Alloc >::alloc()`.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

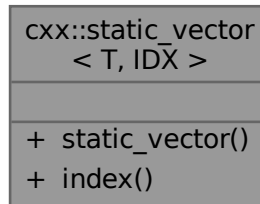
- `I4/cxx/slab_alloc`

## 15.64 `cxx::static_vector< T, IDX >` Class Template Reference

Simple encapsulation for a dynamically allocated array.

```
#include <static_vector>
```

Collaboration diagram for `cxx::static_vector< T, IDX >`:



### Public Member Functions

- `template<typename X, typename = enable_if_t<is_convertible<X, T>::value>>`  
**static\_vector** ([static\\_vector](#)< X, IDX > const &o)  
*Conversion from compatible arrays.*
- `index_type` **index** (value\_type const \*o) const  
*Get the index of the given element of the array.*

### 15.64.1 Detailed Description

```
template<typename T, typename IDX = unsigned>
class cxx::static_vector< T, IDX >
```

Simple encapsulation for a dynamically allocated array.

The main purpose of this class is to support C++11 range for for simple dynamically allocated array with static size.

Definition at line 16 of file [static\\_vector](#).

The documentation for this class was generated from the following file:

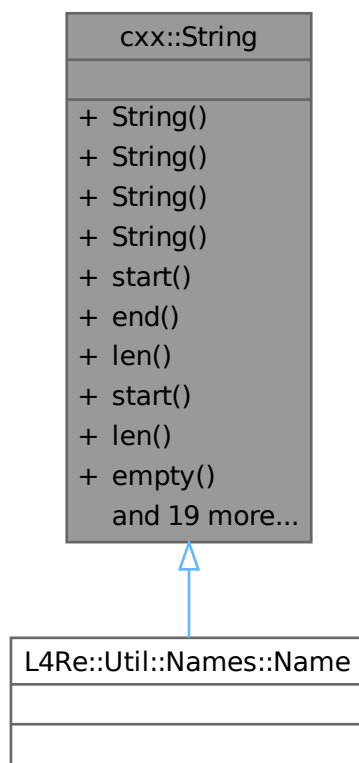
- `I4/cxx/static_vector`

## 15.65 cxx::String Class Reference

Allocation free string class with explicit length field.

```
#include <string>
```

Inheritance diagram for cxx::String:



Collaboration diagram for `cxx::String`:

<code>cxx::String</code>
<ul style="list-style-type: none"> <li>+ <code>String()</code></li> <li>+ <code>String()</code></li> <li>+ <code>String()</code></li> <li>+ <code>String()</code></li> <li>+ <code>start()</code></li> <li>+ <code>end()</code></li> <li>+ <code>len()</code></li> <li>+ <code>start()</code></li> <li>+ <code>len()</code></li> <li>+ <code>empty()</code></li> <li>and 19 more...</li> </ul>

## Public Types

- `typedef char const * Index`  
*Character index type.*

## Public Member Functions

- **`String`** (`char const *s`) `noexcept`  
*Initialize from a zero-terminated string.*
- **`String`** (`char const *s`, unsigned long `len`) `noexcept`  
*Initialize from a pointer to first character and a length.*
- **`String`** (`char const *s`, `char const *e`) `noexcept`  
*Initialize with start and end pointer.*
- **`String`** ()  
*Zero-initialize. Create an invalid string.*
- **`Index start`** () `const`  
*Pointer to first character.*
- **`Index end`** () `const`  
*Pointer to first byte behind the string.*
- `int len` () `const`  
*Length.*
- `void start` (`char const *s`)  
*Set start.*
- `void len` (unsigned long `len`)  
*Set length.*
- `bool empty` () `const`

- Check if the string has length zero.*

  - **String head** ([Index end](#)) const

*Return prefix up to index.*
- **String head** (unsigned long [end](#)) const

*Prefix of length [end](#).*
- **String substr** (unsigned long [idx](#), unsigned long [len](#)=~0UL) const

*Substring of length [len](#) starting at [idx](#).*
- **String substr** (char const \*[start](#), unsigned long [len](#)=0) const

*Substring of length [len](#) starting at [start](#).*
- template<typename F >  
char const \* **find\_match** (F &&match) const

*Find matching character. [match](#) should be a function such as [isspace](#).*
- char const \* **find** (char const \*c) const

*Find character. Return [end\(\)](#) if not found.*
- char const \* **find** (int c) const

*Find character. Return [end\(\)](#) if not found.*
- char const \* **rfind** (char const \*c) const

*Find right-most character. Return [end\(\)](#) if not found.*
- **Index starts\_with** (cxx::String const &c) const

*Check if [c](#) is a prefix of string.*
- char const \* **find** (int c, char const \*s) const

*Find character [c](#) starting at position [s](#). Return [end\(\)](#) if not found.*
- char const \* **find** (char const \*c, char const \*s) const

*Find character set at position.*
- char const & **operator[]** (unsigned long [idx](#)) const

*Get character at [idx](#).*
- char const & **operator[]** (int [idx](#)) const

*Get character at [idx](#).*
- char const & **operator[]** ([Index idx](#)) const

*Get character at [idx](#).*
- bool **eof** (char const \*s) const

*Check if pointer [s](#) points behind string.*
- template<typename INT >  
int **from\_dec** (INT \*v) const

*Convert decimal string to integer.*
- template<typename INT >  
int **from\_hex** (INT \*v) const

*Convert hex string to integer.*
- bool **operator==** ([String](#) const &o) const

*Equality.*
- bool **operator!=** ([String](#) const &o) const

*Inequality.*

### 15.65.1 Detailed Description

Allocation free string class with explicit length field.

This class is used to group characters of a string which belong to one syntactical token types number, identifier, string, whitespace or another single character.

Stings in this class can contain null bytes and may denote parts of other strings.

#### Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 41 of file [string](#).

## 15.65.2 Constructor & Destructor Documentation

### 15.65.2.1 String()

```
cxx::String::String (
    char const * s,
    char const * e ) [inline], [noexcept]
```

Initialize with start and end pointer.

#### Parameters

<i>s</i>	first character of the string
<i>e</i>	pointer to first byte behind the string

Definition at line 59 of file [string](#).

## 15.65.3 Member Function Documentation

### 15.65.3.1 find()

```
char const * cxx::String::find (
    char const * c,
    char const * s ) const [inline]
```

Find character set at position.

#### Parameters

<i>c</i>	zero-terminated string of characters to search for
<i>s</i>	start position of search in string

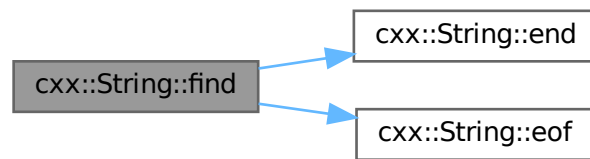
#### Return values

<a href="#">end()</a>	if no char in <i>c</i> is contained in string at or behind <i>s</i> .
<i>position</i>	in string of some character in <i>c</i> .

Definition at line 202 of file [string](#).

References [end\(\)](#), and [eof\(\)](#).

Here is the call graph for this function:



### 15.65.3.2 from\_dec()

```
template<typename INT >
int cxx::String::from_dec (
    INT * v ) const [inline]
```

Convert decimal string to integer.

#### Template Parameters

<i>INT</i>	result integer type
------------	---------------------

#### Parameters

out	v	conversion result
-----	---	-------------------

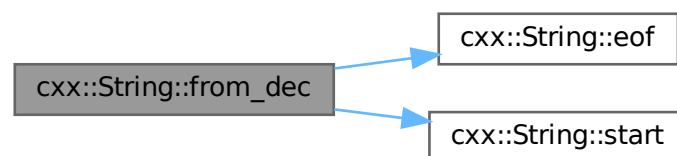
#### Returns

position of first character not converted.

Definition at line [239](#) of file [string](#).

References [eof\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



### 15.65.3.3 from\_hex()

```
template<typename INT >
int cxx::String::from_hex (
    INT * v ) const [inline]
```

Convert hex string to integer.

#### Template Parameters

<i>INT</i>	result integer type
------------	---------------------

#### Parameters

out	v	conversion result
-----	---	-------------------

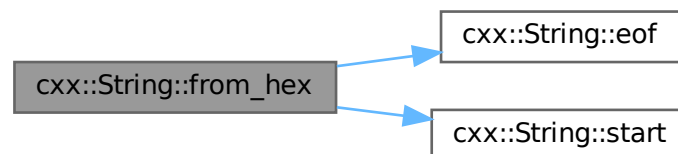
#### Return values

-1	if the maximal amount of digits fitting into <i>INT</i> have been read,
<i>position</i>	of first character not converted otherwise.

Definition at line 268 of file [string](#).

References [eof\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



### 15.65.3.4 starts\_with()

```
Index cxx::String::starts_with (
    cxx::String const & c ) const [inline]
```

Check if *c* is a prefix of string.



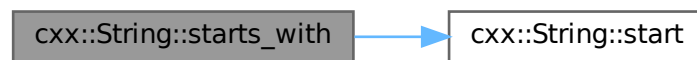
**Returns**

0 if `c` is not a prefix, if it is a prefix, return first position not in `c` (which might be `end()`).

Definition at line 166 of file `string`.

References `start()`.

Here is the call graph for this function:



The documentation for this class was generated from the following file:

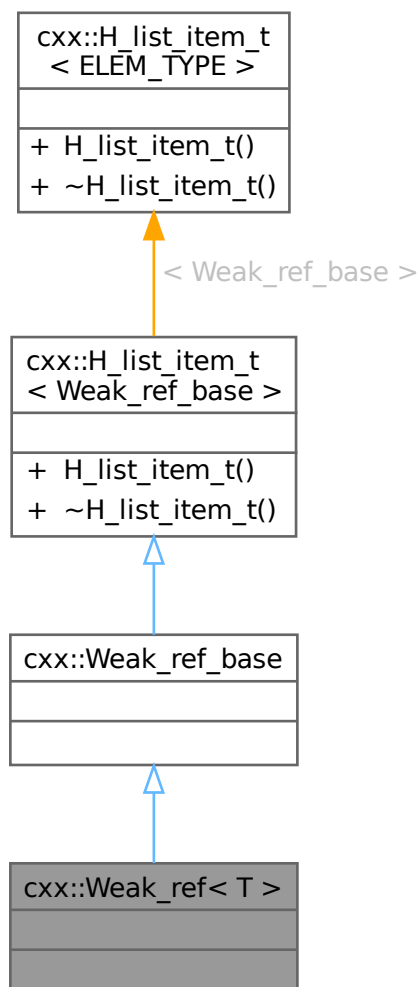
- `l4/cxx/string`

## 15.66 `cxx::Weak_ref< T >` Class Template Reference

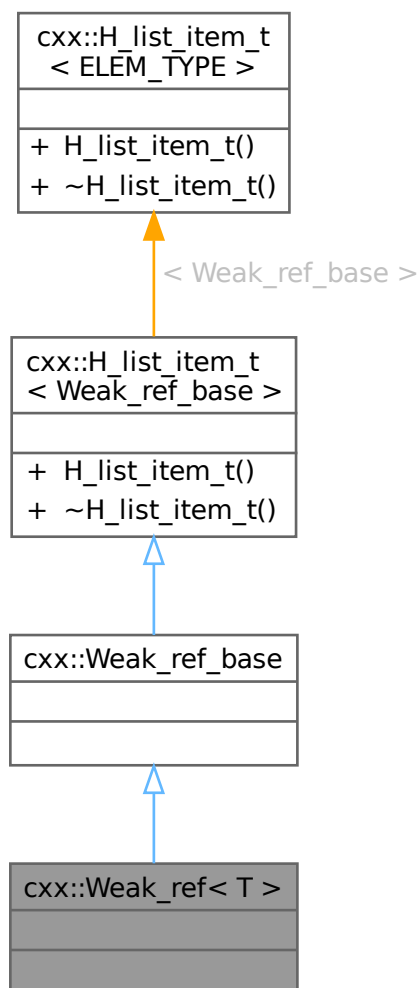
Typed weak reference to an object of type `T`.

```
#include <weak_ref>
```

Inheritance diagram for `cxx::Weak_ref< T >`:



Collaboration diagram for cxx::Weak\_ref< T >:



#### Additional Inherited Members

#### Public Member Functions inherited from `cxx::H_list_item_t< Weak_ref_base >`

- `H_list_item_t()`  
*Constructor.*
- `~H_list_item_t()` noexcept  
*Destructor.*

#### 15.66.1 Detailed Description

```
template<typename T>
class cxx::Weak_ref< T >
```

Typed weak reference to an object of type `T`.

## Template Parameters

<b><i>T</i></b>	The type of the referenced object.
-----------------	------------------------------------

A weak reference is a reference that is invalidated when the referenced object is about to be deleted. All weak references to an object are kept in a linked list (see [Weak\\_ref\\_base::List](#)) and all the weak references are iterated and reset by the [Weak\\_ref\\_base::List](#) destructor or `Weak_ref_base::List::reset()`.

The type `T` must provide two methods that handle the housekeeping of weak references: `remove_weak_ref(Weak_ref_base *)` and `add_weak_ref(Weak_ref_base *)`. These functions must handle the insertion and removal of the weak reference into the respective [Weak\\_ref\\_base::List](#) object. For convenience one can use the `cxx::Weak_ref_obj` as a base class that handles weak references for you.

For example:

```
class C : public cxx::Weak_ref_obj {};

int main()
{
    cxx::Weak_ref<C> r; // r is nullptr
    {
        C c;
        r = &c; // now r points to c
    } // c is destructed, which implies resetting all weak references to c
    // now r is nullptr
    return 0;
}
```

## Note

Weak references have no effect on the lifetime of the referenced object. Hence, a referenced object is *not* deleted when all weak references for it are gone. If automatic deletion is needed, see [cxx::Ref\\_ptr](#).

Definition at line 96 of file [weak\\_ref](#).

The documentation for this class was generated from the following file:

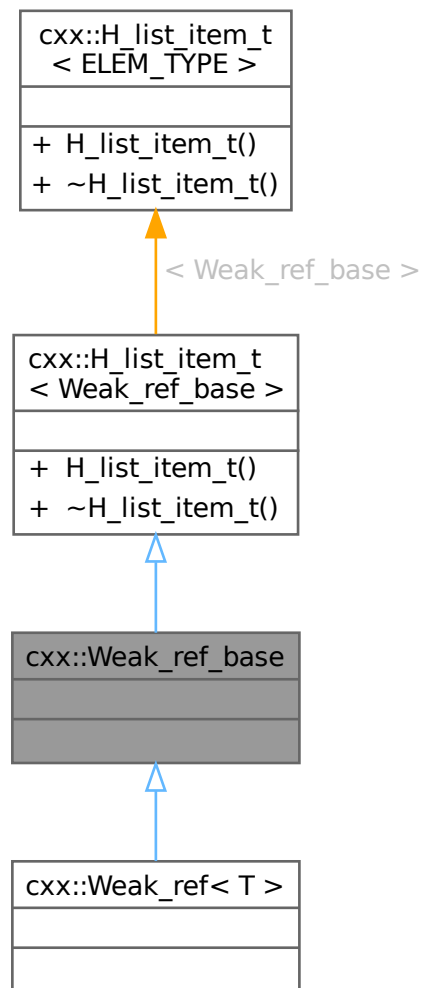
- `I4/cxx/weak_ref`

## 15.67 cxx::Weak\_ref\_base Class Reference

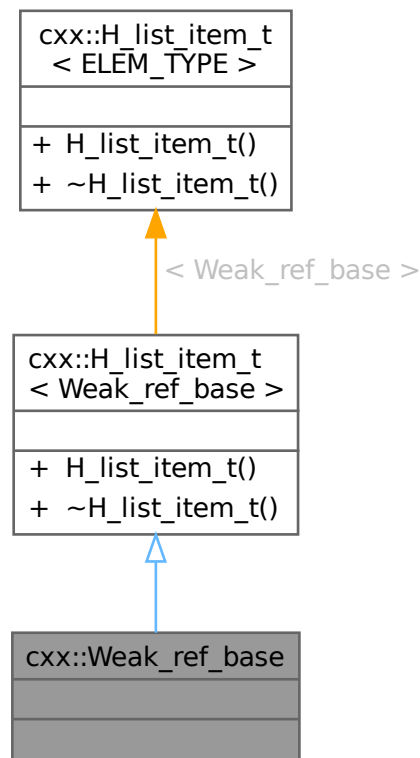
Generic (base) weak reference to some object.

```
#include <weak_ref>
```

Inheritance diagram for cxx::Weak\_ref\_base:



Collaboration diagram for `cxx::Weak_ref_base`:



## Data Structures

- struct [List](#)

*The list type for keeping all weak references to an object.*

## Additional Inherited Members

### Public Member Functions inherited from `cxx::H_list_item_t< Weak_ref_base >`

- [H\\_list\\_item\\_t\(\)](#)  
*Constructor.*
- [~H\\_list\\_item\\_t\(\)](#) noexcept  
*Destructor.*

### 15.67.1 Detailed Description

Generic (base) weak reference to some object.

A weak reference is a reference that gets reset to NULL when the object shall be deleted. All weak references to the same object are kept in a linked list of weak references.

For typed weak references see [cxx::Weak\\_ref](#).

Definition at line 25 of file [weak\\_ref](#).

The documentation for this class was generated from the following file:

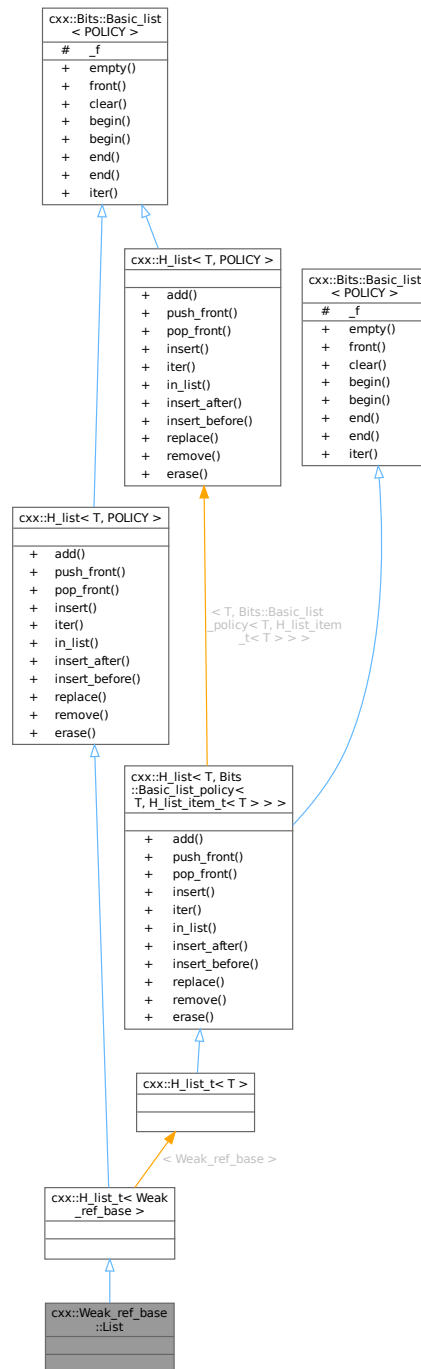
- `I4/cxx/weak_ref`

## 15.68 `cxx::Weak_ref_base::List` Struct Reference

The list type for keeping all weak references to an object.

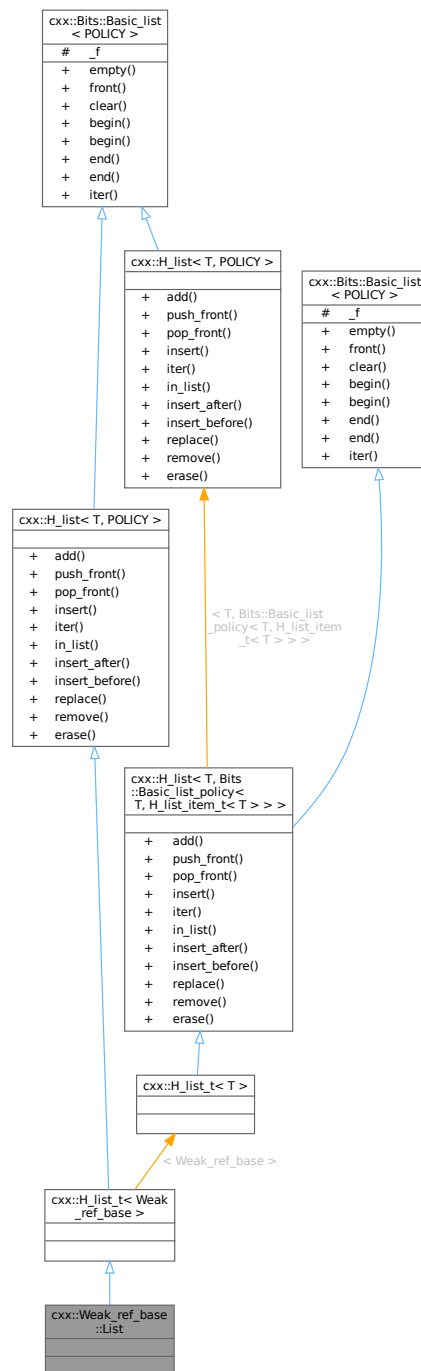
```
#include <weak_ref>
```

Inheritance diagram for `cxx::Weak_ref_base::List`:





Collaboration diagram for cxx::Weak\_ref\_base::List:



### Additional Inherited Members

### Public Member Functions inherited from **cxx::H\_list< T, POLICY >**

- void **add** (T \*e)  
Add element to the front of the list.

- void **push\_front** (T \*e)  
*Add element to the front of the list.*
- T \* **pop\_front** ()  
*Remove and return the head element of the list.*
- Iterator **insert** (T \*e, Iterator const &pred)  
*Insert an element at the iterator position.*

### Public Member Functions inherited from `cxx::Bits::Basic_list< POLICY >`

- bool **empty** () const  
*Check if the list is empty.*
- Value\_type **front** () const  
*Return the first element in the list.*
- void **clear** ()  
*Remove all elements from the list.*
- Iterator **begin** ()  
*Return an iterator to the beginning of the list.*
- Const\_iterator **begin** () const  
*Return a const iterator to the beginning of the list.*
- Const\_iterator **end** () const  
*Return a const iterator to the end of the list.*
- Iterator **end** ()  
*Return an iterator to the end of the list.*

### Static Public Member Functions inherited from `cxx::H_list< T, POLICY >`

- static Iterator **iter** (T \*c)  
*Return an iterator for an arbitrary list element.*
- static bool **in\_list** (T const \*e)  
*Check if the given element is currently part of a list.*
- static Iterator **insert\_after** (T \*e, Iterator const &pred)  
*Insert an element after the iterator position.*
- static void **insert\_before** (T \*e, Iterator const &succ)  
*Insert an element before the iterator position.*
- static void **replace** (T \*p, T \*e)  
*Replace an element in a list with a new element.*
- static void **remove** (T \*e)  
*Remove the given element from its list.*
- static Iterator **erase** (Iterator const &e)  
*Remove the element at the given iterator position.*

### Static Public Member Functions inherited from `cxx::Bits::Basic_list< POLICY >`

- static Const\_iterator **iter** (Const\_value\_type c)  
*Return a const iterator that begins at the given element.*

**Protected Attributes inherited from [cxx::Bits::Basic\\_list< POLICY >](#)**

- `POLICY::Head_type _f`  
*Pointer to front of the list.*

**15.68.1 Detailed Description**

The list type for keeping all weak references to an object.

On destrunction of a list, all weak references to the respective object are set to `nullptr`.

Definition at line 39 of file [weak\\_ref](#).

The documentation for this struct was generated from the following file:

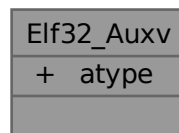
- `I4/cxx/weak_ref`

**15.69 Elf32\_Auxv Struct Reference**

Auxiliary vector (32-bit).

```
#include <elf.h>
```

Collaboration diagram for Elf32\_Auxv:

**Data Fields**

- [Elf32\\_Word atype](#)

**15.69.1 Detailed Description**

Auxiliary vector (32-bit).

Definition at line 967 of file [elf.h](#).

## 15.69.2 Field Documentation

### 15.69.2.1 atype

[Elf32\\_Word](#) `Elf32_Auxv::atype`

See also

[Elf\\_ATs](#)

Definition at line [969](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

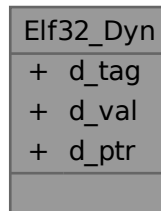
- [l4/util/elf.h](#)

## 15.70 Elf32\_Dyn Struct Reference

ELF32 dynamic entry.

```
#include <elf.h>
```

Collaboration diagram for Elf32\_Dyn:



### Data Fields

- [Elf32\\_Sword](#) `d_tag`

### 15.70.1 Detailed Description

ELF32 dynamic entry.

Definition at line [518](#) of file [elf.h](#).

## 15.70.2 Field Documentation

### 15.70.2.1 d\_tag

[Elf32\\_Sword](#) `Elf32_Dyn::d_tag`

See also

[Elf\\_DTs](#)

Definition at line 520 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

## 15.71 Elf32\_Ehdr Struct Reference

ELF32 header.

```
#include <elf.h>
```

Collaboration diagram for Elf32\_Ehdr:

Elf32_Ehdr
+ e_ident
+ e_type
+ e_machine
+ e_version
+ e_entry
+ e_phoff
+ e_shoff
+ e_flags
+ e_ehsize
+ e_phentsize
+ e_phnum
+ e_shentsize
+ e_shnum
+ e_shstrndx

## Data Fields

- unsigned char **e\_ident** [[EI\\_NIDENT](#)]  
*see [Elf\\_EI](#)s*
- [Elf32\\_Half](#) **e\_type**  
*type of ELF file*
- [Elf32\\_Half](#) **e\_machine**  
*required architecture*
- [Elf32\\_Word](#) **e\_version**  
*file version*
- [Elf32\\_Addr](#) **e\_entry**  
*initial program counter*
- [Elf32\\_Off](#) **e\_phoff**  
*offset of program header table*
- [Elf32\\_Off](#) **e\_shoff**  
*offset of file header table*
- [Elf32\\_Word](#) **e\_flags**  
*processor-specific flags*
- [Elf32\\_Half](#) **e\_ehsize**  
*size of ELF header*
- [Elf32\\_Half](#) **e\_phentsize**  
*size of program header entry*
- [Elf32\\_Half](#) **e\_phnum**  
*number of entries in program header table*
- [Elf32\\_Half](#) **e\_shentsize**  
*size of section header entry*
- [Elf32\\_Half](#) **e\_shnum**  
*number of entries in section header table*
- [Elf32\\_Half](#) **e\_shstrndx**  
*section header table index of strtab*

### 15.71.1 Detailed Description

ELF32 header.

Definition at line [129](#) of file [elf.h](#).

### 15.71.2 Field Documentation

#### 15.71.2.1 e\_flags

[Elf32\\_Word](#) [Elf32\\_Ehdr::e\\_flags](#)

processor-specific flags

See also

[Elf\\_EF\\_ARM\\_s](#)

Definition at line [138](#) of file [elf.h](#).

### 15.71.2.2 e\_machine

`Elf32_Half Elf32_Ehdr::e_machine`

required architecture

See also

[Elf\\_EMs](#)

Definition at line 133 of file [elf.h](#).

### 15.71.2.3 e\_type

`Elf32_Half Elf32_Ehdr::e_type`

type of ELF file

See also

[Elf\\_ETs](#)

Definition at line 132 of file [elf.h](#).

### 15.71.2.4 e\_version

`Elf32_Word Elf32_Ehdr::e_version`

file version

See also

[Elf\\_EVs](#)

Definition at line 134 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

## 15.72 Elf32\_Phdr Struct Reference

ELF32 program header.

```
#include <elf.h>
```

Collaboration diagram for Elf32\_Phdr:

Elf32_Phdr
+ p_type
+ p_offset
+ p_vaddr
+ p_paddr
+ p_filesz
+ p_memsz
+ p_flags
+ p_align

### Data Fields

- [Elf32\\_Word p\\_type](#)  
*type of program section*
- [Elf32\\_Off p\\_offset](#)  
*file offset of program section*
- [Elf32\\_Addr p\\_vaddr](#)  
*memory address of prog section*
- [Elf32\\_Addr p\\_paddr](#)  
*physical address (ignored)*
- [Elf32\\_Word p\\_filesz](#)  
*file size of program section*
- [Elf32\\_Word p\\_memsz](#)  
*memory size of program section*
- [Elf32\\_Word p\\_flags](#)  
*flags*
- [Elf32\\_Word p\\_align](#)  
*alignment of section*

### 15.72.1 Detailed Description

ELF32 program header.

Definition at line 429 of file [elf.h](#).



## 15.72.2 Field Documentation

### 15.72.2.1 p\_flags

`Elf32_Word Elf32_Phdr::p_flags`

flags

See also

[Elf\\_PFs](#)

Definition at line [437](#) of file [elf.h](#).

### 15.72.2.2 p\_type

`Elf32_Word Elf32_Phdr::p_type`

type of program section

See also

[Elf\\_PTs](#)

Definition at line [431](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

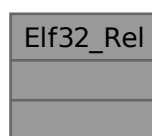
- [l4/util/elf.h](#)

## 15.73 Elf32\_Rel Struct Reference

ELF32 relocation entry w/o addend.

```
#include <elf.h>
```

Collaboration diagram for Elf32\_Rel:



### 15.73.1 Detailed Description

ELF32 relocation entry w/o addend.

Definition at line 636 of file [elf.h](#).

The documentation for this struct was generated from the following file:

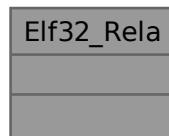
- [l4/util/elf.h](#)

## 15.74 Elf32\_Rela Struct Reference

ELF32 relocation entry w/ addend.

```
#include <elf.h>
```

Collaboration diagram for Elf32\_Rela:



### 15.74.1 Detailed Description

ELF32 relocation entry w/ addend.

Definition at line 643 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

## 15.75 Elf32\_Shdr Struct Reference

ELF32 section header.

```
#include <elf.h>
```

Collaboration diagram for Elf32\_Shdr:

Elf32_Shdr
+ sh_name
+ sh_type
+ sh_flags
+ sh_addr
+ sh_offset
+ sh_size
+ sh_link
+ sh_info
+ sh_addralign
+ sh_entsize

### Data Fields

- [Elf32\\_Word](#) **sh\_name**  
*name of sect (idx into strtab)*
- [Elf32\\_Word](#) **sh\_type**  
*section's type*
- [Elf32\\_Word](#) **sh\_flags**  
*section's flags*
- [Elf32\\_Addr](#) **sh\_addr**  
*memory address of section*
- [Elf32\\_Off](#) **sh\_offset**  
*file offset of section*
- [Elf32\\_Word](#) **sh\_size**  
*file size of section*
- [Elf32\\_Word](#) **sh\_link**  
*idx to associated header section*
- [Elf32\\_Word](#) **sh\_info**  
*extra info of header section*
- [Elf32\\_Word](#) **sh\_addralign**  
*address alignment constraints*
- [Elf32\\_Word](#) **sh\_entsize**  
*size of entry if sect is table*

### 15.75.1 Detailed Description

ELF32 section header.

Definition at line 351 of file [elf.h](#).

### 15.75.2 Field Documentation

#### 15.75.2.1 sh\_flags

[Elf32\\_Word](#) `Elf32_Shdr::sh_flags`

section's flags

See also

[Elf\\_SHFs](#)

Definition at line 355 of file [elf.h](#).

#### 15.75.2.2 sh\_type

[Elf32\\_Word](#) `Elf32_Shdr::sh_type`

section's type

See also

[Elf\\_SHTs](#)

Definition at line 354 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

## 15.76 Elf32\_Sym Struct Reference

ELF32 symbol table entry.

```
#include <elf.h>
```

Collaboration diagram for Elf32\_Sym:

Elf32_Sym
+ st_name
+ st_value
+ st_size
+ st_info
+ st_other
+ st_shndx

## Data Fields

- [Elf32\\_Word](#) **st\_name**  
*name of symbol (idx symstrtab)*
- [Elf32\\_Addr](#) **st\_value**  
*value of associated symbol*
- [Elf32\\_Word](#) **st\_size**  
*size of associated symbol*
- unsigned char **st\_info**  
*type and binding info*
- unsigned char **st\_other**  
*undefined*
- [Elf32\\_Half](#) **st\_shndx**  
*associated section header*

### 15.76.1 Detailed Description

ELF32 symbol table entry.

Definition at line 876 of file [elf.h](#).

The documentation for this struct was generated from the following file:

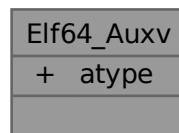
- [l4/util/elf.h](#)

## 15.77 Elf64\_Auxv Struct Reference

Auxiliary vector (64-bit).

```
#include <elf.h>
```

Collaboration diagram for Elf64\_Auxv:



## Data Fields

- [Elf64\\_Word](#) **atype**

### 15.77.1 Detailed Description

Auxiliary vector (64-bit).

Definition at line 974 of file [elf.h](#).

### 15.77.2 Field Documentation

#### 15.77.2.1 atype

[Elf64\\_Word](#) `Elf64_Auxv::atype`

See also

[Elf\\_ATs](#)

Definition at line 976 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

## 15.78 Elf64\_Dyn Struct Reference

ELF64 dynamic entry.

```
#include <elf.h>
```

Collaboration diagram for Elf64\_Dyn:



#### Data Fields

- [Elf64\\_Sxword](#) `d_tag`

### 15.78.1 Detailed Description

ELF64 dynamic entry.

Definition at line 529 of file [elf.h](#).

### 15.78.2 Field Documentation

#### 15.78.2.1 d\_tag

[Elf64\\_Sxword](#) Elf64\_Dyn::d\_tag

See also

[Elf\\_DTs](#)

Definition at line 531 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

## 15.79 Elf64\_Ehdr Struct Reference

ELF64 header.

```
#include <elf.h>
```

Collaboration diagram for Elf64\_Ehdr:

Elf64_Ehdr
+ e_ident
+ e_type
+ e_machine
+ e_version
+ e_entry
+ e_phoff
+ e_shoff
+ e_flags
+ e_ehsize
+ e_phentsize
+ e_phnum
+ e_shentsize
+ e_shnum
+ e_shstrndx

## Data Fields

- unsigned char **e\_ident** [[EI\\_NIDENT](#)]  
*see [Elf\\_EI](#)s*
- [Elf64\\_Half](#) **e\_type**  
*type of ELF file*
- [Elf64\\_Half](#) **e\_machine**  
*required architecture*
- [Elf64\\_Word](#) **e\_version**  
*file version*
- [Elf64\\_Addr](#) **e\_entry**  
*initial program counter*
- [Elf64\\_Off](#) **e\_phoff**  
*offset of program header table*
- [Elf64\\_Off](#) **e\_shoff**  
*offset of file header table*
- [Elf64\\_Word](#) **e\_flags**  
*processor-specific flags*
- [Elf64\\_Half](#) **e\_ehsize**  
*size of ELF header*
- [Elf64\\_Half](#) **e\_phentsize**  
*size of program header entry*
- [Elf64\\_Half](#) **e\_phnum**  
*number of entries in program header table*
- [Elf64\\_Half](#) **e\_shentsize**  
*size of section header entry*
- [Elf64\\_Half](#) **e\_shnum**  
*number of entries in section header table*
- [Elf64\\_Half](#) **e\_shstrndx**  
*section header table index of strtab*

## 15.79.1 Detailed Description

ELF64 header.

Definition at line [150](#) of file [elf.h](#).

## 15.79.2 Field Documentation

### 15.79.2.1 e\_flags

[Elf64\\_Word](#) [Elf64\\_Ehdr::e\\_flags](#)

processor-specific flags

See also

[Elf\\_EF\\_ARM\\_s](#)

Definition at line [159](#) of file [elf.h](#).



### 15.79.2.2 e\_machine

`Elf64_Half Elf64_Ehdr::e_machine`

required architecture

See also

[Elf\\_EMs](#)

Definition at line 154 of file [elf.h](#).

### 15.79.2.3 e\_type

`Elf64_Half Elf64_Ehdr::e_type`

type of ELF file

See also

[Elf\\_ETs](#)

Definition at line 153 of file [elf.h](#).

### 15.79.2.4 e\_version

`Elf64_Word Elf64_Ehdr::e_version`

file version

See also

[Elf\\_EVs](#)

Definition at line 155 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

## 15.80 Elf64\_Phdr Struct Reference

ELF64 program header.

```
#include <elf.h>
```

Collaboration diagram for Elf64\_Phdr:

Elf64_Phdr
+ p_type
+ p_flags
+ p_offset
+ p_vaddr
+ p_paddr
+ p_filesz
+ p_memsz
+ p_align

### Data Fields

- [Elf64\\_Word p\\_type](#)  
*type of program section*
- [Elf64\\_Word p\\_flags](#)  
*flags*
- [Elf64\\_Off p\\_offset](#)  
*file offset of program section*
- [Elf64\\_Addr p\\_vaddr](#)  
*memory address of prog section*
- [Elf64\\_Addr p\\_paddr](#)  
*physical address (ignored)*
- [Elf64\\_Xword p\\_filesz](#)  
*file size of program section*
- [Elf64\\_Xword p\\_memsz](#)  
*memory size of program section*
- [Elf64\\_Xword p\\_align](#)  
*alignment of section*

### 15.80.1 Detailed Description

ELF64 program header.

Definition at line 442 of file [elf.h](#).

## 15.80.2 Field Documentation

### 15.80.2.1 p\_flags

`Elf64_Word Elf64_Phdr::p_flags`

flags

See also

[Elf\\_PFs](#)

Definition at line [445](#) of file [elf.h](#).

### 15.80.2.2 p\_type

`Elf64_Word Elf64_Phdr::p_type`

type of program section

See also

[Elf\\_PTs](#)

Definition at line [444](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

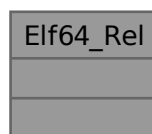
- [l4/util/elf.h](#)

## 15.81 Elf64\_Rel Struct Reference

ELF64 relocation entry w/o addend.

```
#include <elf.h>
```

Collaboration diagram for Elf64\_Rel:



### 15.81.1 Detailed Description

ELF64 relocation entry w/o addend.

Definition at line [651](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

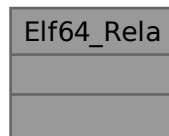
- [l4/util/elf.h](#)

## 15.82 Elf64\_Rela Struct Reference

ELF64 relocation entry w/ addend.

```
#include <elf.h>
```

Collaboration diagram for Elf64\_Rela:



### 15.82.1 Detailed Description

ELF64 relocation entry w/ addend.

Definition at line [658](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

## 15.83 Elf64\_Shdr Struct Reference

ELF64 section header.

```
#include <elf.h>
```

Collaboration diagram for Elf64\_Shdr:

Elf64_Shdr
+ sh_name
+ sh_type
+ sh_flags
+ sh_addr
+ sh_offset
+ sh_size
+ sh_link
+ sh_info
+ sh_addralign
+ sh_entsize

### Data Fields

- [Elf64\\_Word](#) **sh\_name**  
*name of sect (idx into strtab)*
- [Elf64\\_Word](#) **sh\_type**  
*section's type*
- [Elf64\\_Xword](#) **sh\_flags**  
*section's flags*
- [Elf64\\_Addr](#) **sh\_addr**  
*memory address of section*
- [Elf64\\_Off](#) **sh\_offset**  
*file offset of section*
- [Elf64\\_Xword](#) **sh\_size**  
*file size of section*
- [Elf64\\_Word](#) **sh\_link**  
*idx to associated header section*
- [Elf64\\_Word](#) **sh\_info**  
*extra info of header section*
- [Elf64\\_Xword](#) **sh\_addralign**  
*address alignment constraints*
- [Elf64\\_Xword](#) **sh\_entsize**  
*size of entry if sect is table*

### 15.83.1 Detailed Description

ELF64 section header.

Definition at line 366 of file [elf.h](#).

### 15.83.2 Field Documentation

#### 15.83.2.1 sh\_flags

[Elf64\\_Xword](#) Elf64\_Shdr::sh\_flags

section's flags

See also

[Elf\\_SHFs](#)

Definition at line 370 of file [elf.h](#).

#### 15.83.2.2 sh\_type

[Elf64\\_Word](#) Elf64\_Shdr::sh\_type

section's type

See also

[Elf\\_SHTs](#)

Definition at line 369 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

## 15.84 Elf64\_Sym Struct Reference

ELF64 symbol table entry.

```
#include <elf.h>
```

Collaboration diagram for Elf64\_Sym:

Elf64_Sym
+ st_name
+ st_info
+ st_other
+ st_shndx
+ st_value
+ st_size

## Data Fields

- [Elf64\\_Word](#) **st\_name**  
*name of symbol (idx symstrtab)*
- unsigned char **st\_info**  
*type and binding info*
- unsigned char **st\_other**  
*undefined*
- [Elf64\\_Half](#) **st\_shndx**  
*associated section header*
- [Elf64\\_Addr](#) **st\_value**  
*value of associated symbol*
- [Elf64\\_Xword](#) **st\_size**  
*size of associated symbol*

### 15.84.1 Detailed Description

ELF64 symbol table entry.

Definition at line 887 of file [elf.h](#).

The documentation for this struct was generated from the following file:

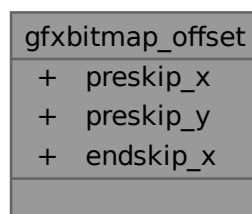
- [l4/util/elf.h](#)

## 15.85 gfxbitmap\_offset Struct Reference

offsets in pmap[] and bmap[]

```
#include <bitmap.h>
```

Collaboration diagram for gfxbitmap\_offset:



## Data Fields

- [l4\\_uint32\\_t](#) **preskip\_x**  
*skip pixels at beginning of line*
- [l4\\_uint32\\_t](#) **preskip\_y**  
*skip lines*
- [l4\\_uint32\\_t](#) **endskip\_x**  
*skip pixels at end of line*

### 15.85.1 Detailed Description

offsets in `pmap[]` and `bmap[]`

Definition at line 67 of file [bitmap.h](#).

The documentation for this struct was generated from the following file:

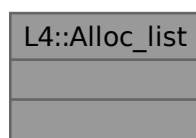
- [l4/libgfxbitmap/bitmap.h](#)

## 15.86 L4::Alloc\_list Class Reference

A simple list-based allocator.

```
#include <alloc.h>
```

Collaboration diagram for L4::Alloc\_list:



### 15.86.1 Detailed Description

A simple list-based allocator.

Definition at line 31 of file [alloc.h](#).

The documentation for this class was generated from the following file:

- [l4/cxx/alloc.h](#)



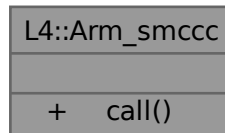
## 15.87 L4::Arm\_smccc Class Reference

Wrapper for function calls that follow the ARM SMC/HVC calling convention.

```
#include <arm_smccc>
```

Inherits L4::Kobject\_0t< Derived, PROTO, S\_DEMAND >.

Collaboration diagram for L4::Arm\_smccc:



### Public Member Functions

- [l4\\_msgtag\\_t](#) [call](#) ([l4\\_umword\\_t](#) func, [l4\\_umword\\_t](#) in0, [l4\\_umword\\_t](#) in1, [l4\\_umword\\_t](#) in2, [l4\\_umword\\_t](#) in3, [l4\\_umword\\_t](#) in4, [l4\\_umword\\_t](#) in5, [l4\\_umword\\_t](#) \*out0, [l4\\_umword\\_t](#) \*out1, [l4\\_umword\\_t](#) \*out2, [l4\\_umword\\_t](#) \*out3, [l4\\_umword\\_t](#) client\_id)

*ARM SMC/HVC function call.*

### 15.87.1 Detailed Description

Wrapper for function calls that follow the ARM SMC/HVC calling convention.

See [l4\\_arm\\_smccc\\_call\(\)](#) for the corresponding C interface.

Definition at line 24 of file [arm\\_smccc](#).

### 15.87.2 Member Function Documentation

#### 15.87.2.1 call()

```

l4_msgtag_t L4::Arm_smccc::call (
    l4_umword_t func,
    l4_umword_t in0,
    l4_umword_t in1,
    l4_umword_t in2,
    l4_umword_t in3,
    l4_umword_t in4,
    l4_umword_t in5,
    l4_umword_t * out0,
    l4_umword_t * out1,
    l4_umword_t * out2,
    l4_umword_t * out3,
    l4_umword_t client_id )
  
```

ARM SMC/HVC function call.

The input parameters consist of a function identifier, 6 arguments and a client id. Results are returned in 4 output parameters.

## Parameters

	<i>func</i>	Function identifier. <ul style="list-style-type: none"> <li>• Bit 31 has to be set: This marks the call as <i>Fast Call</i>. <i>Yielding Calls</i> (bit 31 unset) are rejected by the kernel.</li> <li>• Bit 30 defines the calling convention:</li> <li>• Bit 30 == 1: 64-bit calling convention.</li> <li>• Bit 30 == 0: 32-bit calling convention.</li> <li>• Bits 24..29 determine the service call ID. Only service IDs <math>\geq 0x30000000</math> (<i>Trusted Application Calls</i> and <i>Trusted OS Calls</i>) are allowed.</li> </ul>
in	<i>in0</i>	First input parameter.
in	<i>in1</i>	Second input parameter.
in	<i>in2</i>	Third input parameter.
in	<i>in3</i>	Fourth input parameter.
in	<i>in4</i>	Fifth input parameter.
in	<i>in5</i>	Sixth input parameter.
out	<i>out0</i>	First output parameter.
out	<i>out1</i>	Second output parameter.
out	<i>out2</i>	Third output parameter.
out	<i>out3</i>	Fourth output parameter.
in	<i>client_id</i>	Client ID. According to the specification, this value might be ignored by certain functions.

## Return values

-L4_ENOSYS	Either bit 31 of the function call not set or service ID $< 0x30000000$ .
-L4_EINVAL	Invalid number of parameters.
$< 0$	Other L4 error.
0	Success.

The documentation for this class was generated from the following file:

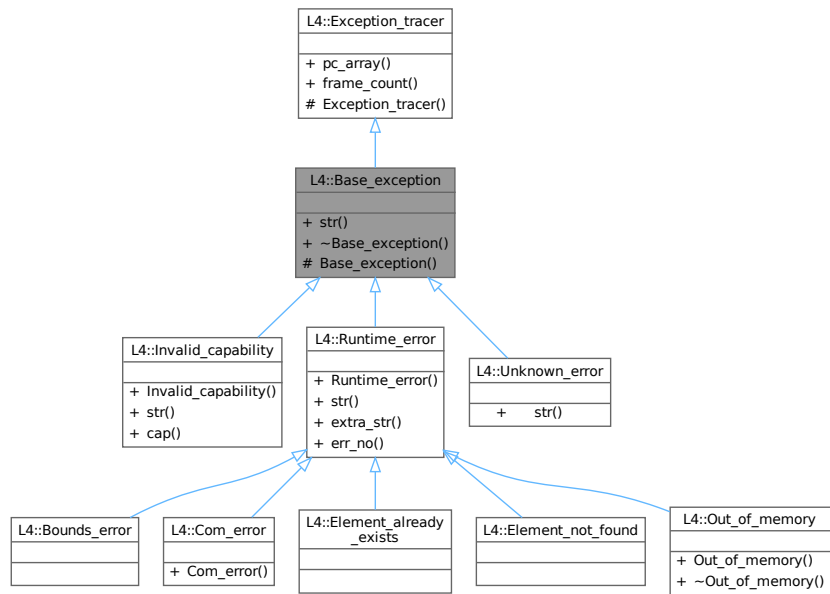
- l4/sys/arm\_smccc

## 15.88 L4::Base\_exception Class Reference

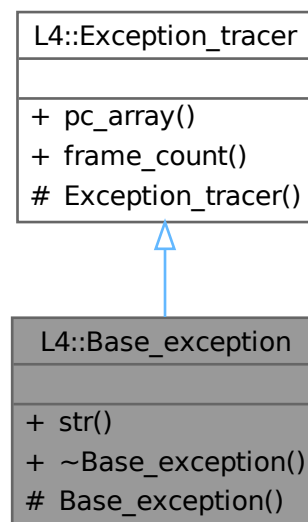
Base class for all exceptions, thrown by the L4Re framework.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Base\_exception:



Collaboration diagram for L4::Base\_exception:



## Public Member Functions

- virtual char const \* **str** () const noexcept=0

*Return a human readable string for the exception.*

- virtual `~Base_exception ()` noexcept

*Destruction.*

### Public Member Functions inherited from [L4::Exception\\_tracer](#)

- void const \*const \* `pc_array ()` const noexcept

*Get the array containing the call trace.*

- int `frame_count ()` const noexcept

*Get the number of entries that are valid in the call trace.*

### Protected Member Functions

- `Base_exception ()` noexcept

*Create a base exception.*

### Protected Member Functions inherited from [L4::Exception\\_tracer](#)

- `Exception_tracer ()` noexcept

*Create a back trace.*

## 15.88.1 Detailed Description

Base class for all exceptions, thrown by the [L4Re](#) framework.

This is the abstract base of all exceptions thrown within the [L4Re](#) framework. It is basically also a good idea to use it as base of all user defined exceptions.

Definition at line [116](#) of file [exceptions](#).

The documentation for this class was generated from the following file:

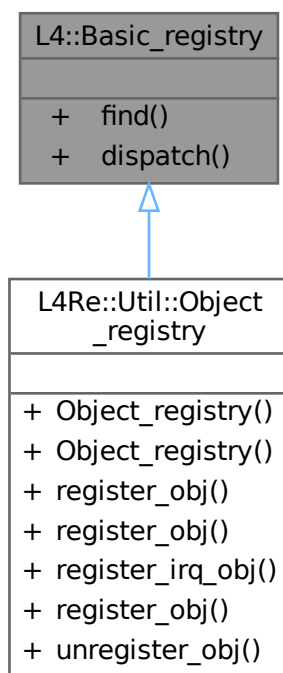
- [l4/cxx/exceptions](#)

## 15.89 L4::Basic\_registry Class Reference

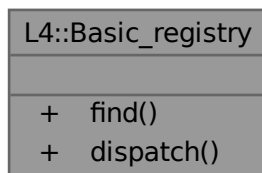
This registry returns the corresponding server object based on the label of an [lpc\\_gate](#).

```
#include <ipc_epiface>
```

Inheritance diagram for L4::Basic\_registry:



Collaboration diagram for L4::Basic\_registry:



## Static Public Member Functions

- static [Value](#) \* [find](#) ([l4\\_umword\\_t](#) label)  
*Get the server object for an [lpc\\_gate](#) label.*
- static [l4\\_msgtag\\_t](#) [dispatch](#) ([l4\\_msgtag\\_t](#) tag, [l4\\_umword\\_t](#) label, [l4\\_utcb\\_t](#) \*utcb)  
*The dispatch function called by the server loop.*

### 15.89.1 Detailed Description

This registry returns the corresponding server object based on the label of an [lpc\\_gate](#).

Definition at line [540](#) of file [ipc\\_epiface](#).

### 15.89.2 Member Function Documentation

#### 15.89.2.1 [dispatch\(\)](#)

```
static l4\_msgtag\_t L4::Basic_registry::dispatch (
    l4\_msgtag\_t tag,
    l4\_umword\_t label,
    l4\_utcb\_t * utcb ) [inline], [static]
```

The dispatch function called by the server loop.

This function forwards the message to the server object identified by the given *label*.

#### Parameters

<i>tag</i>	The message tag used for the invocation.
<i>label</i>	The label used to find the object including the rights bits of the invoked capability.
<i>utcb</i>	The UTCB used for the invocation.

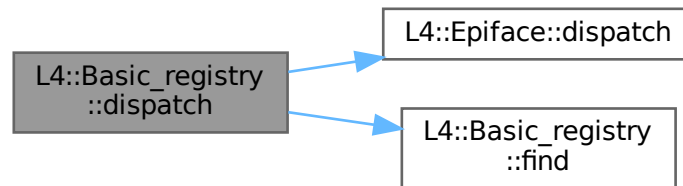
#### Returns

The return code from the object's dispatch function or `-L4_ENOENT` if the object does not exist.

Definition at line [565](#) of file [ipc\\_epiface](#).

References [L4::Epiface::dispatch\(\)](#), and [find\(\)](#).

Here is the call graph for this function:



### 15.89.2.2 find()

```
static Value * L4::Basic_registry::find (
    l4_umword_t label ) [inline], [static]
```

Get the server object for an `lpc_gate` label.

#### Parameters

<i>label</i>	The label usually stored in an <code>lpc_gate</code> .
--------------	--

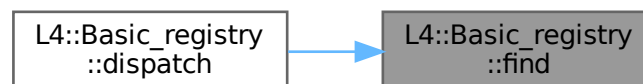
#### Returns

A pointer to the `Epiface` identified by the given label.

Definition at line 549 of file `ipc_epiface`.

Referenced by `dispatch()`.

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

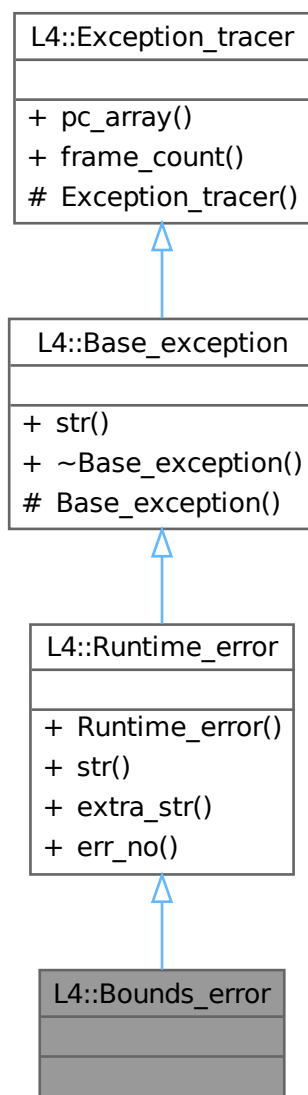
- `l4/sys/cxx/ipc_epiface`

## 15.90 L4::Bounds\_error Class Reference

Access out of bounds.

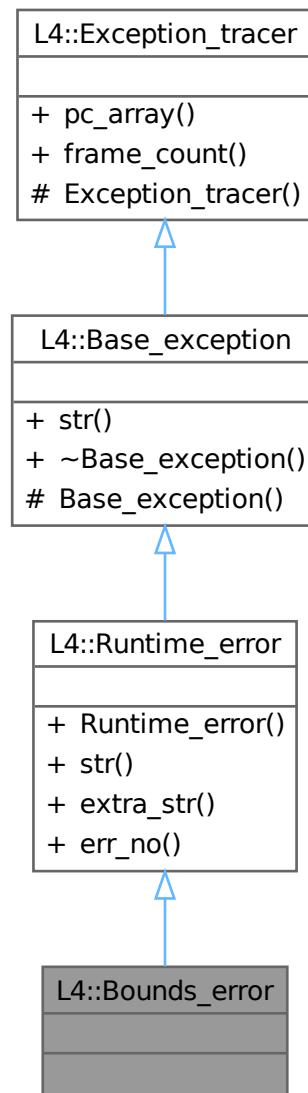
```
#include <exceptions>
```

Inheritance diagram for L4::Bounds\_error:





Collaboration diagram for L4::Bounds\_error:



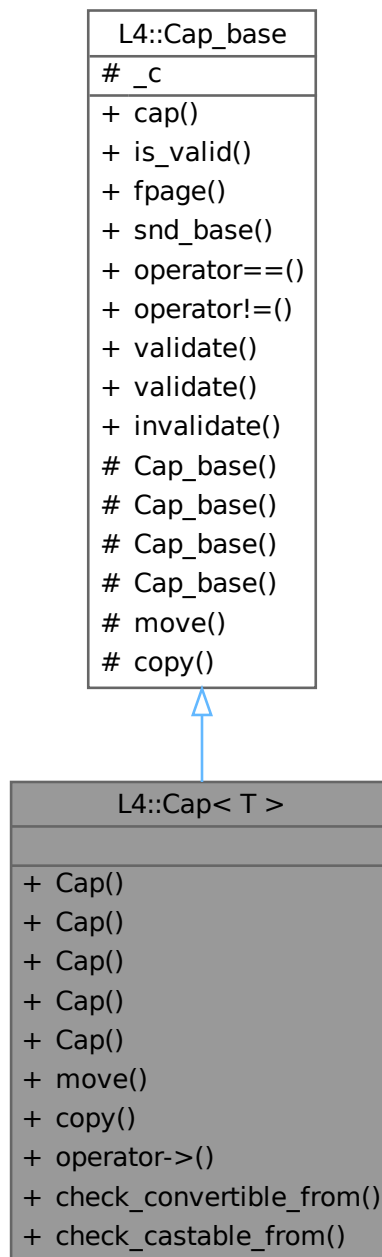
### Additional Inherited Members

### Public Member Functions inherited from [L4::Runtime\\_error](#)

- [Runtime\\_error](#) (long [err\\_no](#), char const \*extra=0) noexcept  
Create a new [Runtime\\_error](#).
- char const \* [str](#) () const noexcept override  
Return a human readable string for the exception.
- char const \* [extra\\_str](#) () const  
Get the description text for this runtime error.
- long [err\\_no](#) () const noexcept  
Get the error value for this runtime error.



Collaboration diagram for L4::Cap< T >:



## Public Member Functions

- `template<typename O >`  
`Cap (Cap< O > const &o) noexcept`  
*Create a copy from o, supporting implicit type casting.*
- `Cap (Cap_type cap) noexcept`  
*Constructor to create an invalid capability selector.*

- [Cap](#) ([l4\\_default\\_caps\\_t cap](#)) noexcept  
*Initialize capability with one of the default capability selectors.*
- [Cap](#) ([l4\\_cap\\_idx\\_t idx=L4\\_INVALID\\_CAP](#)) noexcept  
*Initialize capability, defaults to the invalid capability selector.*
- [Cap](#) ([No\\_init\\_type](#)) noexcept  
*Create an uninitialized cap selector.*
- [Cap move](#) ([Cap](#) const &src) const  
*Move a capability to this cap slot.*
- [Cap copy](#) ([Cap](#) const &src) const  
*Copy a capability to this cap slot.*
- [T \\* operator->](#) () const noexcept  
*Member access of a T.*

## Public Member Functions inherited from [L4::Cap\\_base](#)

- [l4\\_cap\\_idx\\_t cap](#) () const noexcept  
*Return capability selector.*
- bool [is\\_valid](#) () const noexcept  
*Test whether the capability is a valid capability index (i.e., not L4\_INVALID\_CAP).*
- [l4\\_fpage\\_t fpage](#) (unsigned rights=[L4\\_CAP\\_FPAGE\\_RWS](#)) const noexcept  
*Return flex-page for the capability.*
- [l4\\_umword\\_t snd\\_base](#) (unsigned grant=[L4\\_MAP\\_ITEM\\_MAP](#), [l4\\_cap\\_idx\\_t base=L4\\_INVALID\\_CAP](#)) const noexcept  
*Return send base.*
- bool [operator==](#) ([Cap\\_base](#) const &o) const noexcept  
*Test if two capabilities are equal.*
- bool [operator!=](#) ([Cap\\_base](#) const &o) const noexcept  
*Test if two capabilities are not equal.*
- [l4\\_msgtag\\_t validate](#) ([l4\\_utcb\\_t \\*u=l4\\_utcb\(\)](#)) const noexcept  
*Check whether a capability is present (refers to an object).*
- [l4\\_msgtag\\_t validate](#) ([Cap< Task > task](#), [l4\\_utcb\\_t \\*u=l4\\_utcb\(\)](#)) const noexcept  
*Check whether a capability is present (refers to an object).*
- void [invalidate](#) () noexcept  
*Set this capability to invalid (L4\_INVALID\_CAP).*

## Static Public Member Functions

- template<typename From >  
static void [check\\_convertible\\_from](#) () noexcept  
*Perform the type conversion that needs to compile in order for a capability of type From to be convertible to one of type T.*
- template<typename From >  
static void [check\\_castable\\_from](#) () noexcept  
*Perform the type conversion that needs to compile in order for a capability of type From to be castable (via the correct cap\_cast) to one of type T.*

## Friends

- class [L4::Kobject](#)

## Additional Inherited Members

### Public Types inherited from L4::Cap\_base

- enum [No\\_init\\_type](#) { [No\\_init](#) }  
*Special value for uninitialized capability objects.*
- enum [Cap\\_type](#) { [Invalid](#) = L4\_INVALID\_CAP }  
*Invalid capability type.*

### Protected Member Functions inherited from L4::Cap\_base

- [Cap\\_base](#) ([l4\\_cap\\_idx\\_t](#) c) noexcept  
*Generate a capability from its C representation.*
- [Cap\\_base](#) ([Cap\\_type](#) cap) noexcept  
*Constructor to create an invalid capability.*
- [Cap\\_base](#) ([l4\\_default\\_caps\\_t](#) cap) noexcept  
*Initialize capability with one of the default capabilities.*
- [Cap\\_base](#) () noexcept  
*Create an uninitialized instance.*
- void [move](#) ([Cap\\_base](#) const &src) const  
*Replace this capability with the contents of `src`.*
- void [copy](#) ([Cap\\_base](#) const &src) const  
*Copy a capability.*

### Protected Attributes inherited from L4::Cap\_base

- [l4\\_cap\\_idx\\_t\\_c](#)  
*The C representation of a capability selector.*

## 15.91.1 Detailed Description

```
template<typename T>
class L4::Cap< T >
```

C++ interface for capabilities.

#### Template Parameters

<a href="#">T</a>	Type of the object the capability points to.
-------------------	--

The C++ version of a capability is comparable to a pointer, in fact it is a kind of smart pointer for our kernel objects and the objects derived from the kernel objects ([L4::Kobject](#)).

Add

```
#include <l4/sys/capability>
```

to your code to use the capability interface.

#### Examples

[examples/clntsrv/client.cc](#), [examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#), [examples/libs/l4re/streammap/client.cc](#), and [examples/sys/migrate/thread\\_migrate](#)

Definition at line 218 of file [capability.h](#).

## 15.91.2 Constructor & Destructor Documentation

### 15.91.2.1 Cap() [1/4]

```
template<typename T >
template<typename O >
L4::Cap< T >::Cap (
    Cap< O > const & o ) [inline], [noexcept]
```

Create a copy from `o`, supporting implicit type casting.

#### Parameters

<code>o</code>	The source selector that shall be copied (and casted).
----------------	--

Definition at line 270 of file [capability.h](#).

### 15.91.2.2 Cap() [2/4]

```
template<typename T >
L4::Cap< T >::Cap (
    Cap_type cap ) [inline], [noexcept]
```

Constructor to create an invalid capability selector.

#### Parameters

<code>cap</code>	Capability selector.
------------------	----------------------

Definition at line 277 of file [capability.h](#).

### 15.91.2.3 Cap() [3/4]

```
template<typename T >
L4::Cap< T >::Cap (
    l4_default_caps_t cap ) [inline], [noexcept]
```

Initialize capability with one of the default capability selectors.

## Parameters

<i>cap</i>	Capability selector.
------------	----------------------

Definition at line 283 of file [capability.h](#).

**15.91.2.4 Cap()** [4/4]

```
template<typename T >
L4::Cap< T >::Cap (
    l4_cap_idx_t idx = L4_INVALID_CAP ) [inline], [explicit], [noexcept]
```

Initialize capability, defaults to the invalid capability selector.

## Parameters

<i>idx</i>	Capability selector.
------------	----------------------

Definition at line 289 of file [capability.h](#).

**15.91.3 Member Function Documentation****15.91.3.1 check\_castable\_from()**

```
template<typename T >
template<typename From >
static void L4::Cap< T >::check_castable_from ( ) [inline], [static], [noexcept]
```

Perform the type conversion that needs to compile in order for a capability of type From to be castable (via the correct `cap_cast`) to one of type T.

## Template Parameters

<i>From</i>	Type to convert from
-------------	----------------------

Definition at line 259 of file [capability.h](#).

**15.91.3.2 check\_convertible\_from()**

```
template<typename T >
template<typename From >
static void L4::Cap< T >::check_convertible_from ( ) [inline], [static], [noexcept]
```

Perform the type conversion that needs to compile in order for a capability of type From to be convertible to one of type T.

## Template Parameters

<i>From</i>	Type to convert from
-------------	----------------------

Definition at line 246 of file [capability.h](#).

**15.91.3.3 copy()**

```
template<typename T >
Cap L4::Cap< T >::copy (
    Cap< T > const & src ) const [inline]
```

Copy a capability to this cap slot.

## Parameters

<i>src</i>	the source capability slot.
------------	-----------------------------

Definition at line 312 of file [capability.h](#).

**15.91.3.4 move()**

```
template<typename T >
Cap L4::Cap< T >::move (
    Cap< T > const & src ) const [inline]
```

Move a capability to this cap slot.

## Parameters

<i>src</i>	the source capability slot.
------------	-----------------------------

After this operation the source slot is no longer valid.

Definition at line 302 of file [capability.h](#).

The documentation for this class was generated from the following file:

- l4/sys/cxx/capability.h

**15.92 L4::Cap\_base Class Reference**

Base class for all kinds of capabilities.

```
#include <l4/sys/capability>
```



[illegible]

L4::Cap_base
# _c
+ cap()
+ is_valid()
+ fpage()
+ snd_base()
+ operator==( )
+ operator!=( )
+ validate()
+ validate()
+ invalidate()
# Cap_base()
# Cap_base()
# Cap_base()
# Cap_base()
# move()
# copy()

- enum **No\_init\_type** { **No\_init** }  
*Special value for uninitialized capability objects.*
- enum **Cap\_type** { **Invalid** = L4\_INVALID\_CAP }  
*Invalid capability type.*

## Public Member Functions

- [l4\\_cap\\_idx\\_t](#) [cap](#) () const noexcept  
*Return capability selector.*
- bool [is\\_valid](#) () const noexcept  
*Test whether the capability is a valid capability index (i.e., not L4\_INVALID\_CAP).*
- [l4\\_fpage\\_t](#) [fpage](#) (unsigned rights=[L4\\_CAP\\_FPAGE\\_RWS](#)) const noexcept  
*Return flex-page for the capability.*
- [l4\\_umword\\_t](#) [snd\\_base](#) (unsigned grant=[L4\\_MAP\\_ITEM\\_MAP](#), [l4\\_cap\\_idx\\_t](#) base=[L4\\_INVALID\\_CAP](#)) const noexcept  
*Return send base.*
- bool **operator==** ([Cap\\_base](#) const &o) const noexcept  
*Test if two capabilities are equal.*
- bool **operator!=** ([Cap\\_base](#) const &o) const noexcept  
*Test if two capabilities are not equal.*
- [l4\\_msgtag\\_t](#) [validate](#) ([l4\\_utcb\\_t](#) \*u=[l4\\_utcb](#)()) const noexcept  
*Check whether a capability is present (refers to an object).*
- [l4\\_msgtag\\_t](#) [validate](#) ([Cap](#)< [Task](#) > task, [l4\\_utcb\\_t](#) \*u=[l4\\_utcb](#)()) const noexcept  
*Check whether a capability is present (refers to an object).*
- void **invalidate** () noexcept  
*Set this capability to invalid (L4\_INVALID\_CAP).*

## Protected Member Functions

- [Cap\\_base](#) ([l4\\_cap\\_idx\\_t](#) c) noexcept  
*Generate a capability from its C representation.*
- **Cap\_base** ([Cap\\_type](#) cap) noexcept  
*Constructor to create an invalid capability.*
- [Cap\\_base](#) ([l4\\_default\\_caps\\_t](#) cap) noexcept  
*Initialize capability with one of the default capabilities.*
- **Cap\_base** () noexcept  
*Create an uninitialized instance.*
- void [move](#) ([Cap\\_base](#) const &src) const  
*Replace this capability with the contents of src.*
- void [copy](#) ([Cap\\_base](#) const &src) const  
*Copy a capability.*

## Protected Attributes

- [l4\\_cap\\_idx\\_t](#) [\\_c](#)  
*The C representation of a capability selector.*

### 15.92.1 Detailed Description

Base class for all kinds of capabilities.

#### Attention

This class is not for direct use, use [L4::Cap](#) instead.

This class contains all the things that are independent of the type of the object referred by the capability.

#### See also

[L4::Cap](#) for typed capabilities.

Definition at line 25 of file [capability.h](#).

## 15.92.2 Member Enumeration Documentation

### 15.92.2.1 Cap\_type

enum [L4::Cap\\_base::Cap\\_type](#)

Invalid capability type.

Enumerator

Invalid	Invalid capability selector.
---------	------------------------------

Definition at line 40 of file [capability.h](#).

### 15.92.2.2 No\_init\_type

enum [L4::Cap\\_base::No\\_init\\_type](#)

Special value for uninitialized capability objects.

Enumerator

No_init	Special value for constructing uninitialized <a href="#">Cap</a> objects.
---------	---

Definition at line 29 of file [capability.h](#).

## 15.92.3 Constructor & Destructor Documentation

### 15.92.3.1 Cap\_base() [1/2]

```
L4::Cap_base::Cap_base (
    l4\_cap\_idx\_t c ) [inline], [explicit], [protected], [noexcept]
```

Generate a capability from its C representation.

Parameters

c	The C capability
---	------------------

Definition at line 144 of file [capability.h](#).

### 15.92.3.2 Cap\_base() [2/2]

```
L4::Cap_base::Cap_base (
    l4\_default\_caps\_t cap ) [inline], [explicit], [protected], [noexcept]
```

Initialize capability with one of the default capabilities.

## Parameters

<i>cap</i>	Capability.
------------	-------------

Definition at line 155 of file [capability.h](#).

## 15.92.4 Member Function Documentation

### 15.92.4.1 `cap()`

```
l4_cap_idx_t L4::Cap_base::cap ( ) const [inline], [noexcept]
```

Return capability selector.

## Returns

Capability selector.

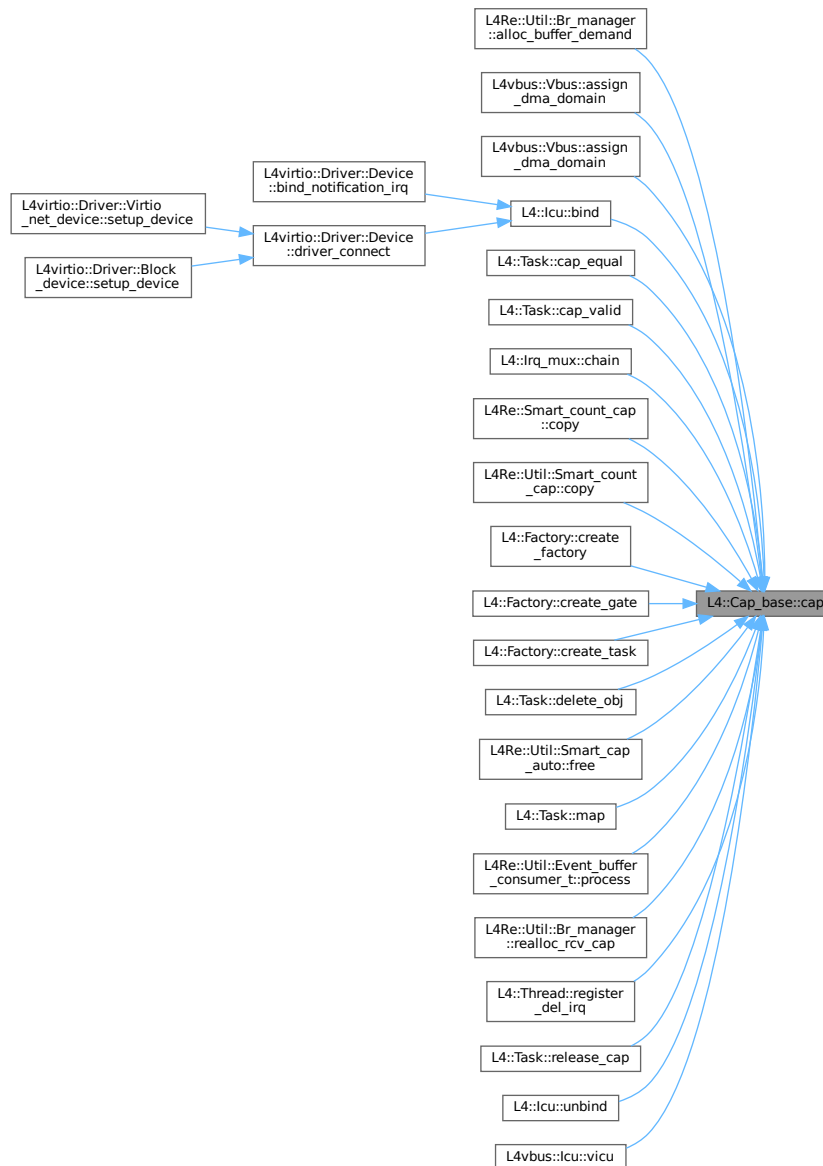
## Examples

[examples/libs/l4re/streammap/client.cc](#).

Definition at line 49 of file [capability.h](#).

Referenced by [L4Re::Util::Br\\_manager::alloc\\_buffer\\_demand\(\)](#), [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#), [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#), [L4::lcu::bind\(\)](#), [L4::Task::cap\\_equal\(\)](#), [L4::Task::cap\\_valid\(\)](#), [L4::lrc\\_mux::chain\(\)](#), [L4Re::Smart\\_count\\_cap< Unmap\\_flags >::copy\(\)](#), [L4Re::Util::Smart\\_count\\_cap< Unmap\\_flags >::copy\(\)](#), [L4::Factory::create\\_factory\(\)](#), [L4::Factory::create\\_gate\(\)](#), [L4::Factory::create\\_task\(\)](#), [L4::Task::delete\\_obj\(\)](#), [L4Re::Util::Smart\\_cap\\_auto< Unmap\\_flags >::free\(\)](#), [L4::Task::map\(\)](#), [L4Re::Util::Event\\_buffer\\_consumer\\_t< PAYLOAD >::process\(\)](#), [L4Re::Util::Br\\_manager::realloc\\_rcv\\_cap\(\)](#), [L4::Thread::register\\_del\\_irq\(\)](#), [L4::Task::release\\_cap\(\)](#), [L4::lcu::unbind\(\)](#), and [L4vbus::lcu::vicu\(\)](#).

Here is the caller graph for this function:



### 15.92.4.2 copy()

```
void L4::Cap_base::copy (
    Cap_base const & src ) const [inline], [protected]
```

Copy a capability.

#### Parameters

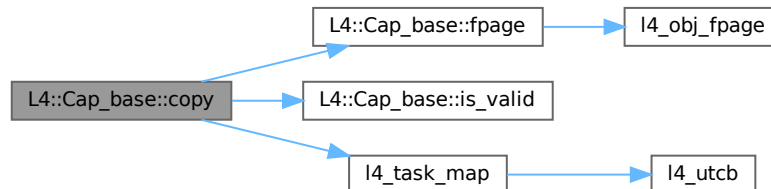
src	the source capability.
-----	------------------------

After this operation this capability refers to the same object as `src`.

Definition at line 187 of file [capability.h](#).

References [fpage\(\)](#), [is\\_valid\(\)](#), [L4\\_BASE\\_TASK\\_CAP](#), [L4\\_CAP\\_FPAGE\\_RWSD](#), [L4\\_FPAGE\\_C\\_OBJ\\_RIGHTS](#), and [l4\\_task\\_map\(\)](#).

Here is the call graph for this function:



#### 15.92.4.3 fpage()

```
l4_fpage_t L4::Cap_base::fpage (
    unsigned rights = L4_CAP_FPAGE_RWS ) const [inline], [noexcept]
```

Return flex-page for the capability.

##### Parameters

<i>rights</i>	Rights, defaults to 'rws'
---------------	---------------------------

##### Returns

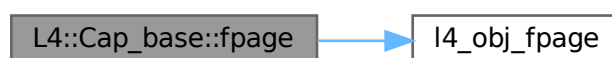
flex-page

Definition at line 69 of file [capability.h](#).

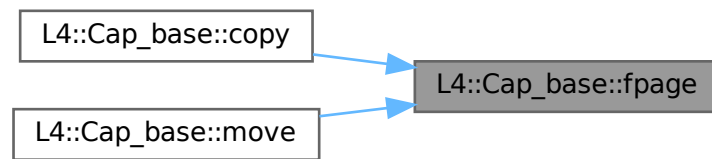
References [l4\\_obj\\_fpage\(\)](#).

Referenced by [copy\(\)](#), and [move\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 15.92.4.4 is\_valid()

```
bool L4::Cap_base::is_valid ( ) const [inline], [noexcept]
```

Test whether the capability is a valid capability index (i.e., not `L4_INVALID_CAP`).

##### Returns

True if capability is not invalid, false if invalid

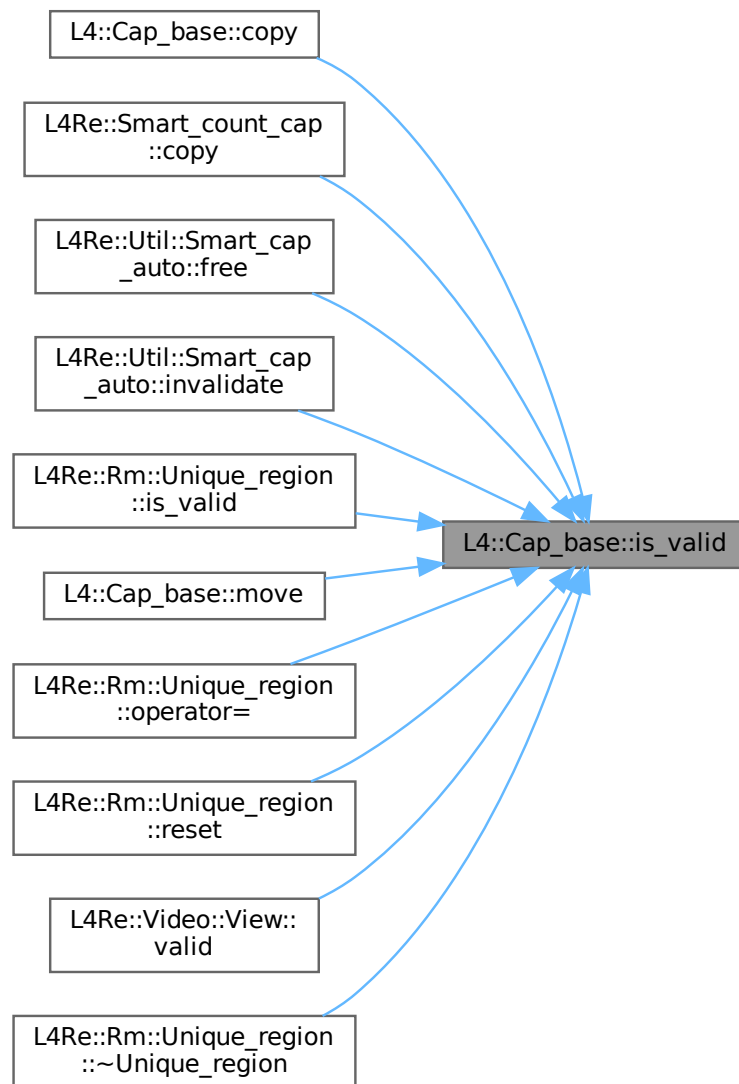
##### Examples

[examples/clntsrv/client.cc](#), [examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#),  
and [examples/libs/l4re/streammap/client.cc](#).

Definition at line 57 of file [capability.h](#).

Referenced by [copy\(\)](#), [L4Re::Smart\\_count\\_cap< Unmap\\_flags >::copy\(\)](#), [L4Re::Util::Smart\\_cap\\_auto< Unmap\\_flags >::free\(\)](#), [L4Re::Util::Smart\\_cap\\_auto< Unmap\\_flags >::invalidate\(\)](#), [L4Re::Rm::Unique\\_region< T >::is\\_valid\(\)](#), [move\(\)](#), [L4Re::Rm::Unique\\_region< T >::operator=\(\)](#), [L4Re::Rm::Unique\\_region< T >::reset\(\)](#), [L4Re::Video::View::valid\(\)](#), and [L4Re::Rm::Unique\\_region< T >::~~Unique\\_region\(\)](#).

Here is the caller graph for this function:



#### 15.92.4.5 move()

```
void L4::Cap_base::move (
    Cap_base const & src ) const [inline], [protected]
```

Replace this capability with the contents of `src`.

##### Parameters

<code>src</code>	the source capability.
------------------	------------------------

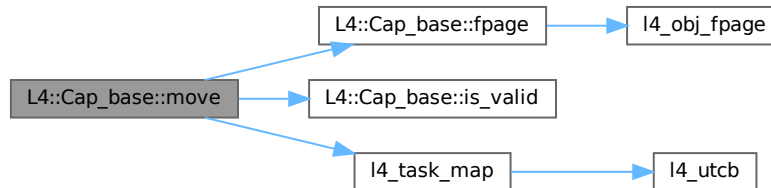


After the operation this capability refers to the object formerly referred to by the source capability `src`, and the source capability no longer refers to an object.

Definition at line 171 of file [capability.h](#).

References [fpage\(\)](#), [is\\_valid\(\)](#), [L4\\_BASE\\_TASK\\_CAP](#), [L4\\_CAP\\_FPAGE\\_RWSD](#), [L4\\_FPAGE\\_C\\_OBJ\\_RIGHTS](#), [L4\\_MAP\\_ITEM\\_GRANT](#), and [l4\\_task\\_map\(\)](#).

Here is the call graph for this function:



#### 15.92.4.6 `snd_base()`

```

l4_umword_t L4::Cap_base::snd_base (
    unsigned grant = L4_MAP_ITEM_MAP,
    l4_cap_idx_t base = L4_INVALID_CAP ) const [inline], [noexcept]
  
```

Return send base.

##### Parameters

<i>grant</i>	Indicates if object shall be granted. Allowed values: <a href="#">L4_MAP_ITEM_MAP</a> , <a href="#">L4_MAP_ITEM_GRANT</a> .
<i>base</i>	Base capability (first in a bundle of aligned capabilities)

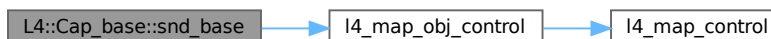
##### Returns

Map object.

Definition at line 81 of file [capability.h](#).

References [L4\\_INVALID\\_CAP](#), and [l4\\_map\\_obj\\_control\(\)](#).

Here is the call graph for this function:



**15.92.4.7 validate()** [1/2]

```
l4_msgtag_t L4::Cap_base::validate (
    Cap< Task > task,
    l4_utcb_t * u = l4_utcb() ) const [inline], [noexcept]
```

Check whether a capability is present (refers to an object).

**Parameters**

<i>task</i>	<a href="#">Task</a> to check the capability in.
<i>u</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

**Return values**

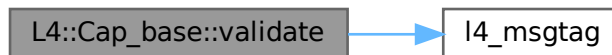
<i>l4_msgtag_t::label()</i> > 0	Capability is present (refers to an object).
<i>l4_msgtag_t::label()</i> == 0	No capability present (void object or invalid capability slot).

A capability is considered present when it refers to an existing kernel object.

Definition at line 84 of file [capability](#).

References [l4\\_msgtag\(\)](#).

Here is the call graph for this function:

**15.92.4.8 validate()** [2/2]

```
l4_msgtag_t L4::Cap_base::validate (
    l4_utcb_t * u = l4_utcb() ) const [inline], [noexcept]
```

Check whether a capability is present (refers to an object).

**Parameters**

<i>u</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
----------	--

**Return values**

<i>l4_msgtag_t::label()</i> > 0	Capability is present (refers to an object).
---------------------------------	--

## Return values

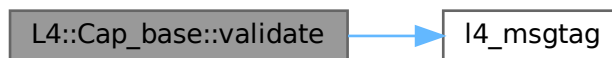
<code>l4_msgtag_t::label() == 0</code>	No capability present (void object or invalid capability slot).
--	---

A capability is considered present when it refers to an existing kernel object.

Definition at line 91 of file [capability](#).

References [L4\\_BASE\\_TASK\\_CAP](#), and [l4\\_msgtag\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

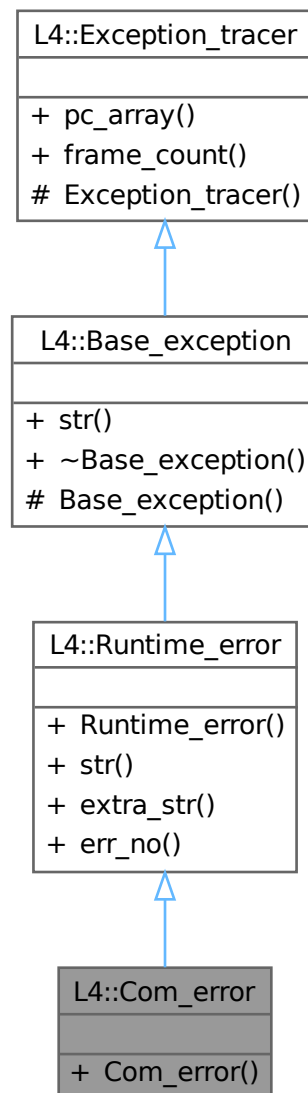
- `l4/sys/cxx/capability.h`
- `l4/sys/capability`

## 15.93 L4::Com\_error Class Reference

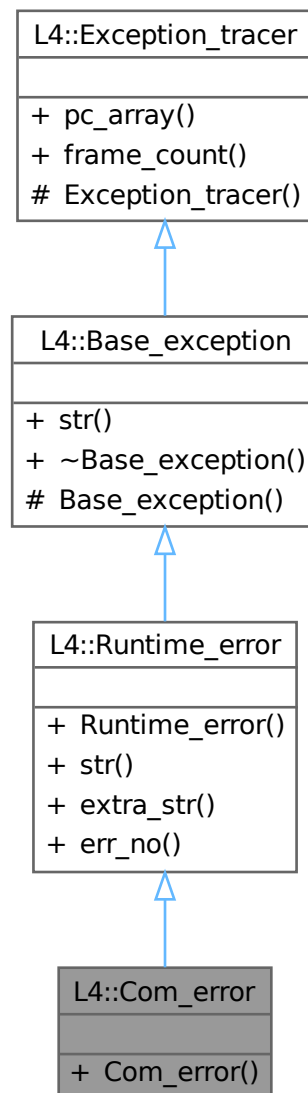
Error conditions during IPC.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Com\_error:



Collaboration diagram for L4::Com\_error:



### Public Member Functions

- [Com\\_error](#) (long err) noexcept  
Create a [Com\\_error](#) for the given [L4](#) IPC error code.

### Public Member Functions inherited from [L4::Runtime\\_error](#)

- [Runtime\\_error](#) (long err\_no, char const \*extra=0) noexcept  
Create a new [Runtime\\_error](#).
- char const \* [str](#) () const noexcept override

*Return a human readable string for the exception.*

- `char const * extra\_str () const`

*Get the description text for this runtime error.*

- `long err\_no () const noexcept`

*Get the error value for this runtime error.*

## Public Member Functions inherited from [L4::Base\\_exception](#)

- `virtual ~Base_exception () noexcept`

*Destruction.*

## Public Member Functions inherited from [L4::Exception\\_tracer](#)

- `void const *const * pc_array () const noexcept`

*Get the array containing the call trace.*

- `int frame_count () const noexcept`

*Get the number of entries that are valid in the call trace.*

## Additional Inherited Members

## Protected Member Functions inherited from [L4::Base\\_exception](#)

- `Base_exception () noexcept`

*Create a base exception.*

## Protected Member Functions inherited from [L4::Exception\\_tracer](#)

- `Exception_tracer () noexcept`

*Create a back trace.*

## 15.93.1 Detailed Description

Error conditions during IPC.

This exception encapsulates all IPC error conditions of [L4](#) IPC.

Definition at line [274](#) of file [exceptions](#).

## 15.93.2 Constructor & Destructor Documentation

### 15.93.2.1 `Com_error()`

```
L4::Com_error::Com_error (
    long err ) [inline], [explicit], [noexcept]
```

Create a [Com\\_error](#) for the given [L4](#) IPC error code.

## Parameters

<i>err</i>	The <a href="#">L4</a> IPC error code (l4_ipc... return value).
------------	---

Definition at line 281 of file [exceptions](#).

The documentation for this class was generated from the following file:

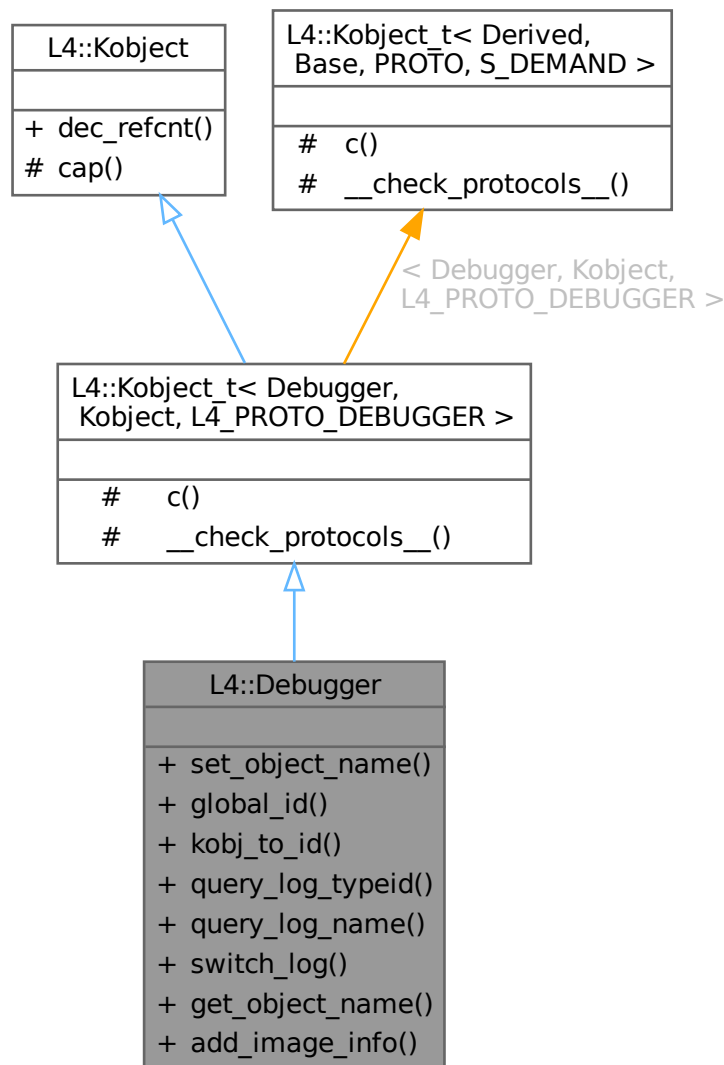
- [l4/cxx/exceptions](#)

## 15.94 L4::Debugger Class Reference

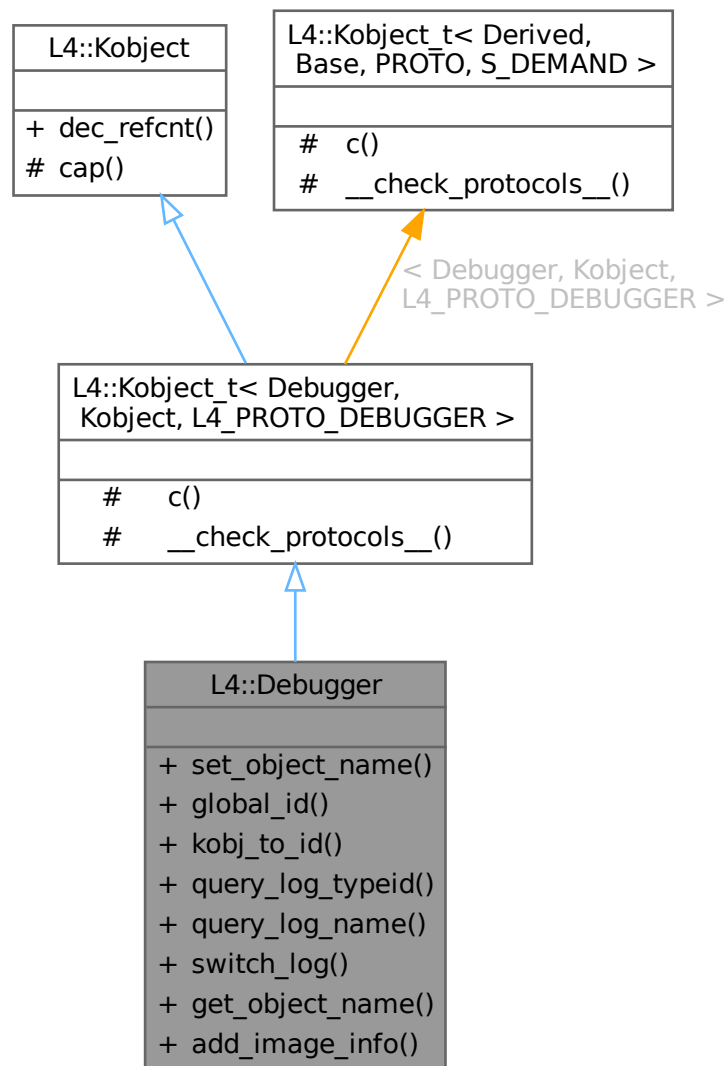
C++ kernel debugger API.

```
#include <debugger>
```

Inheritance diagram for L4::Debugger:



Collaboration diagram for L4::Debugger:



## Public Member Functions

- `l4_msgtag_t set_object_name` (const char \*name, `l4_utcb_t` \*utcb=`l4_utcb()`) noexcept  
*Set the name of a kernel object.*
- unsigned long `global_id` (`l4_utcb_t` \*utcb=`l4_utcb()`) noexcept  
*Get the globally unique ID of the object behind a capability.*
- unsigned long `kobj_to_id` (`l4_addr_t` kobjp, `l4_utcb_t` \*utcb=`l4_utcb()`) noexcept  
*Get the globally unique ID of the object behind the kobject pointer.*
- long `query_log_typeid` (const char \*name, unsigned idx, `l4_utcb_t` \*utcb=`l4_utcb()`) noexcept  
*Query the log-id for a log type.*
- long `query_log_name` (unsigned idx, char \*name, unsigned namelen, char \*shortname, unsigned short-namelen, `l4_utcb_t` \*utcb=`l4_utcb()`) noexcept



*Query the name of a log type given the ID.*

- [l4\\_msgtag\\_t switch\\_log](#) (const char \*name, unsigned on\_off, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept

*Set or unset log.*

- [l4\\_msgtag\\_t get\\_object\\_name](#) (unsigned id, char \*name, unsigned size, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept

*Get name of object with Id *i* d.*

- [l4\\_msgtag\\_t add\\_image\\_info](#) ([l4\\_addr\\_t](#) base, const char \*name, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept

*Add loaded image information for a task.*

## Public Member Functions inherited from [L4::Kobject](#)

- [l4\\_msgtag\\_t dec\\_refcnt](#) ([l4\\_mword\\_t](#) diff, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#))

*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

## Protected Types inherited from

[L4::Kobject\\_t](#)< [Debugger](#), [Kobject](#), [L4\\_PROTO\\_DEBUGGER](#) >

- typedef [Debugger](#) **Class**

*The target interface type (inheriting from [Kobject\\_t](#))*

- typedef Typeid::Iface< [PROTO](#), [Debugger](#) > **\_\_lface**

*The interface description for the derived class.*

- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_lface** >, typename Base::\_\_lface\_list > **\_\_lface\_list**

*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Debugger](#), [Kobject](#), [L4\\_PROTO\\_DEBUGGER](#) >

- [L4::Cap](#)< **Class** > **c** () const noexcept

*Get the capability to ourselves.*

## Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t cap](#) () const noexcept

*Return capability selector.*

## Static Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Debugger](#), [Kobject](#), [L4\\_PROTO\\_DEBUGGER](#) >

- static void **\_\_check\_protocols** () noexcept

*Helper to check for protocol conflicts.*

## 15.94.1 Detailed Description

C++ kernel debugger API.

### Attention

This API is subject to change! Do not rely on it in production code.

This API is to be used for debugging exclusively.

This is the API for accessing kernel-debugger functionality from user-level programs. Specifically, it provides functionality to enrich the kernel debugger with insights into the program. The purpose is to facilitate debugging with the kernel debugger. For instance, a developer might choose to name the threads of her program so that she can find them in the kernel debugger thread list.

This API interacts with a kernel object that interfaces with the kernel debugger, the `jdb-kernel` object. The `jdb-kernel` object is fix and only available when the kernel debugger is built into the microkernel. The developer needs to pass the capability through to her program.

### Include File

```
#include <l4/sys/debugger>
```

Definition at line 53 of file [debugger](#).

## 15.94.2 Member Function Documentation

### 15.94.2.1 `add_image_info()`

```
l4_msgtag_t L4::Debugger::add_image_info (
    l4_addr_t base,
    const char * name,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

Add loaded image information for a task.

### Parameters

<i>base</i>	Image load base address
<i>name</i>	Image name
<i>utcb</i>	The UTCB to use for the operation.

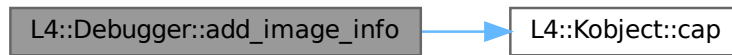
### Returns

System call return tag.

Definition at line 172 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 15.94.2.2 get\_object\_name()

```

l4_msgtag_t L4::Debugger::get_object_name (
    unsigned id,
    char * name,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Get name of object with Id `id`.

#### Parameters

	<i>id</i>	Id of the object whose name is asked.
out	<i>name</i>	Buffer to copy the name into. The buffer must be allocated by the caller.
	<i>size</i>	Length of the <code>name</code> buffer.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

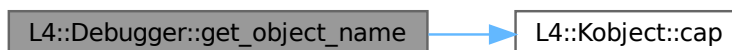
#### Returns

Syscall return tag

Definition at line [159](#) of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 15.94.2.3 global\_id()

```

unsigned long L4::Debugger::global_id (
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Get the globally unique ID of the object behind a capability.

## Parameters

<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .
-------------	--

## Return values

$\sim 0UL$	The capability is invalid.
$\geq 0$	The global debugger id.

Definition at line 82 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



#### 15.94.2.4 kobj\_to\_id()

```

unsigned long L4::Debugger::kobj_to_id (
    l4_addr_t kobjp,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Get the globally unique ID of the object behind the kobject pointer.

## Parameters

<i>kobjp</i>	<a href="#">Kobject</a> pointer
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

## Return values

$\sim 0UL$	The capability or the <a href="#">Kobject</a> pointer are invalid.
$\geq 0$	The globally unique id.

Definition at line 94 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 15.94.2.5 query\_log\_name()

```

long L4::Debugger::query_log_name (
    unsigned idx,
    char * name,
    unsigned namelen,
    char * shortname,
    unsigned shortnamelen,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Query the name of a log type given the ID.

#### Parameters

	<i>idx</i>	ID to query.
out	<i>name</i>	Buffer to copy name to. The buffer must be allocated by the caller.
	<i>namelen</i>	Buffer length of name.
out	<i>shortname</i>	Buffer to copy <i>shortname</i> to. The buffer must be allocated by the caller.
	<i>shortnamelen</i>	Buffer length of <i>shortname</i> .
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

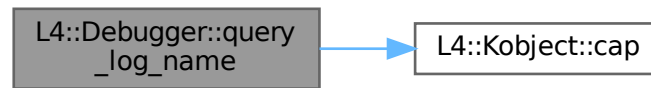
#### Return values

0	Success
<0	Error

Definition at line 127 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 15.94.2.6 query\_log\_typeid()

```

long L4::Debugger::query_log_typeid (
    const char * name,
    unsigned idx,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Query the log-id for a log type.

#### Parameters

<i>name</i>	Name to query for.
<i>idx</i>	Idx to start searching, start with 0
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

#### Return values

$\geq 0$	Id
$< 0$	Error

Definition at line [108](#) of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 15.94.2.7 set\_object\_name()

```
l4_msgtag_t L4::Debugger::set_object_name (
    const char * name,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

Set the name of a kernel object.

#### Parameters

<i>name</i>	Name
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

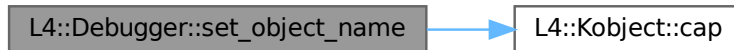
#### Returns

System call return tag.

Definition at line 70 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 15.94.2.8 switch\_log()

```
l4_msgtag_t L4::Debugger::switch_log (
    const char * name,
    unsigned on_off,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

Set or unset log.

#### Parameters

<i>name</i>	Name of the log type.
<i>on_off</i>	1: turn log on, 0: turn log off
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

#### Returns

Syscall return tag

Definition at line 144 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- [l4/sys/debugger](#)

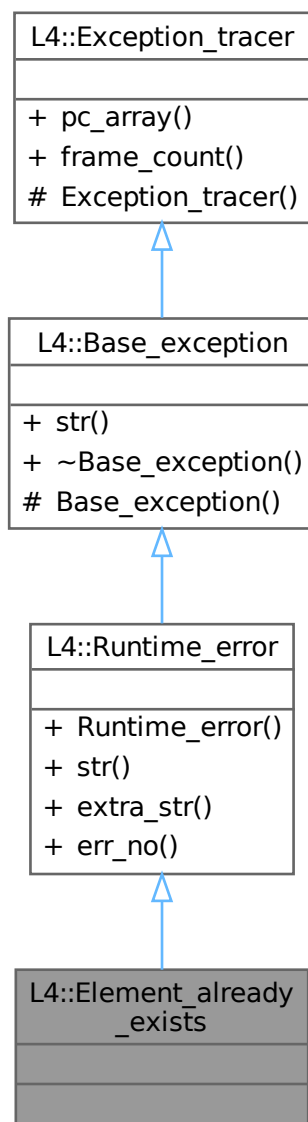
## 15.95 L4::Element\_already\_exists Class Reference

[Exception](#) for duplicate element insertions.

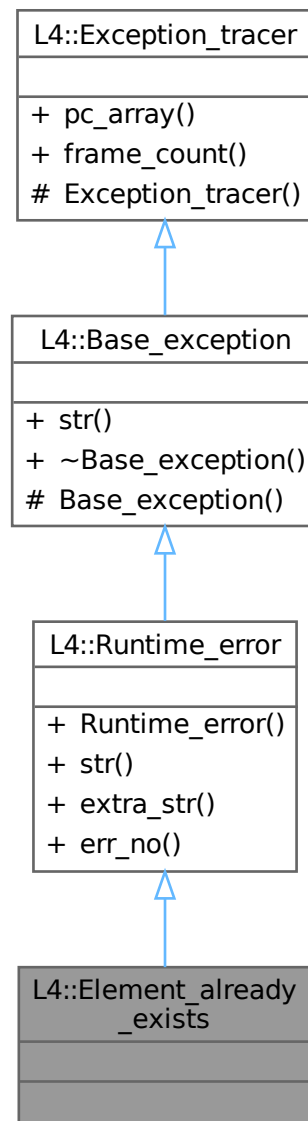
```
#include <l4/cxx/exceptions>
```



Inheritance diagram for L4::Element\_already\_exists:



Collaboration diagram for L4::Element\_already\_exists:



#### Additional Inherited Members

#### Public Member Functions inherited from [L4::Runtime\\_error](#)

- [Runtime\\_error](#) (long [err\\_no](#), char const \*extra=0) noexcept  
Create a new [Runtime\\_error](#).
- char const \* [str](#) () const noexcept override  
Return a human readable string for the exception.
- char const \* [extra\\_str](#) () const  
Get the description text for this runtime error.
- long [err\\_no](#) () const noexcept  
Get the error value for this runtime error.

## Public Member Functions inherited from L4::Base\_exception

- virtual **~Base\_exception** () noexcept  
*Destruction.*

## Public Member Functions inherited from L4::Exception\_tracer

- void const \*const \* **pc\_array** () const noexcept  
*Get the array containing the call trace.*
- int **frame\_count** () const noexcept  
*Get the number of entries that are valid in the call trace.*

## Protected Member Functions inherited from L4::Base\_exception

- **Base\_exception** () noexcept  
*Create a base exception.*

## Protected Member Functions inherited from L4::Exception\_tracer

- **Exception\_tracer** () noexcept  
*Create a back trace.*

### 15.95.1 Detailed Description

[Exception](#) for duplicate element insertions.

Definition at line 203 of file [exceptions](#).

The documentation for this class was generated from the following file:

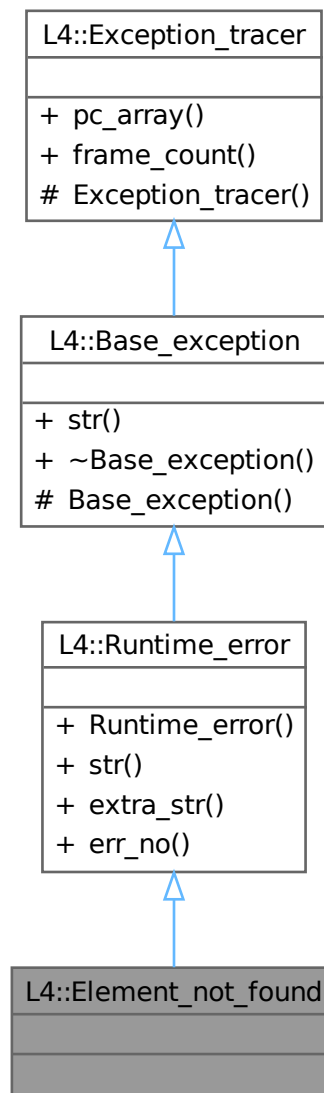
- l4/cxx/[exceptions](#)

## 15.96 L4::Element\_not\_found Class Reference

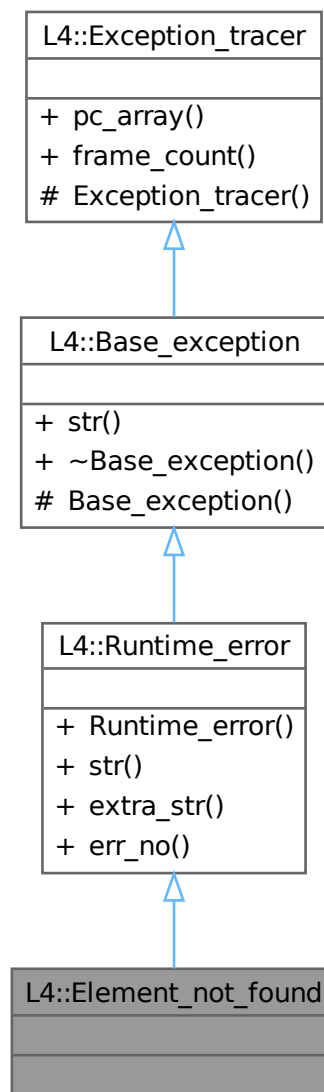
[Exception](#) for a failed lookup (element not found).

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Element\_not\_found:



Collaboration diagram for L4::Element\_not\_found:



### Additional Inherited Members

### Public Member Functions inherited from [L4::Runtime\\_error](#)

- [Runtime\\_error](#) (long [err\\_no](#), char const \*extra=0) noexcept  
*Create a new [Runtime\\_error](#).*
- char const \* [str](#) () const noexcept override  
*Return a human readable string for the exception.*
- char const \* [extra\\_str](#) () const  
*Get the description text for this runtime error.*
- long [err\\_no](#) () const noexcept  
*Get the error value for this runtime error.*

## Public Member Functions inherited from [L4::Base\\_exception](#)

- virtual `~Base_exception ()` noexcept  
*Destruction.*

## Public Member Functions inherited from [L4::Exception\\_tracer](#)

- void const \*const \* **pc\_array** () const noexcept  
*Get the array containing the call trace.*
- int **frame\_count** () const noexcept  
*Get the number of entries that are valid in the call trace.*

## Protected Member Functions inherited from [L4::Base\\_exception](#)

- **Base\_exception** () noexcept  
*Create a base exception.*

## Protected Member Functions inherited from [L4::Exception\\_tracer](#)

- **Exception\_tracer** () noexcept  
*Create a back trace.*

### 15.96.1 Detailed Description

[Exception](#) for a failed lookup (element not found).

Definition at line 231 of file [exceptions](#).

The documentation for this class was generated from the following file:

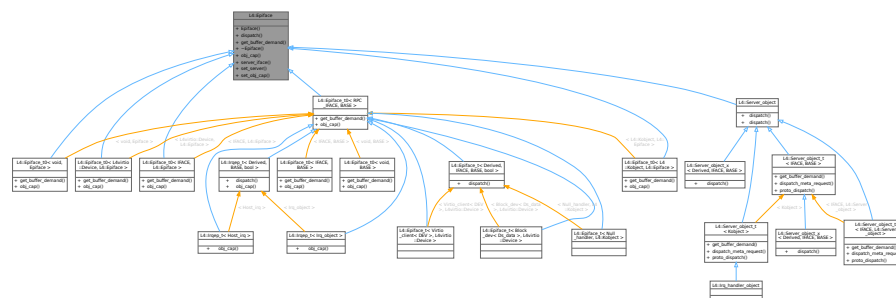
- [l4/cxx/exceptions](#)

## 15.97 L4::Epiface Struct Reference

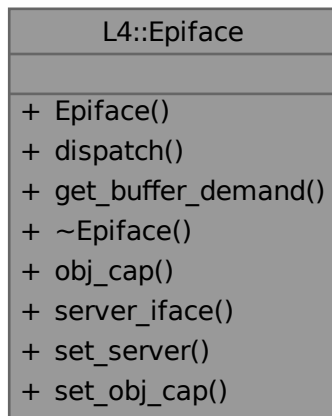
Base class for interface implementations.

```
#include <ipc_epiface>
```

Inheritance diagram for L4::Epiface:



Collaboration diagram for L4::Epiface:



## Public Types

- typedef [lpc\\_svr::Server\\_iface](#) **Server\_iface**  
*Type for abstract server interface.*
- typedef [lpc\\_svr::Server\\_iface::Demand](#) **Demand**  
*Type for server-side receive buffer demand.*

## Public Member Functions

- **Epiface ()**  
*Make a server object.*
- virtual [l4\\_msgtag\\_t](#) **dispatch** ([l4\\_msgtag\\_t](#) tag, unsigned rights, [l4\\_utcb\\_t](#) \*utcb)=0  
*The abstract handler for client requests to the object.*
- virtual [Demand](#) **get\_buffer\_demand** () const =0  
*Get the server-side receive buffer demand for this object.*
- virtual **~Epiface** ()=0  
*Destroy the object.*
- Stored\_cap **obj\_cap** () const  
*Get the capability to the kernel object belonging to this object.*
- [Server\\_iface](#) \* **server\_iface** () const  
*Get pointer to server interface at which the object is currently registered.*
- int **set\_server** ([Server\\_iface](#) \*srv, [Cap](#)< void > cap, bool managed=false)  
*Set server registration info for the object.*
- void **set\_obj\_cap** ([Cap](#)< void > const &cap)  
*Deprecated server registration function.*

### 15.97.1 Detailed Description

Base class for interface implementations.

An [Epiface](#) is the base interface of objects registered in the server loop. Incoming IPC gets dispatched to the appropriate [Epiface](#) object where the call is then handled appropriately.

#### Note

[Server](#) loops are allowed to internally keep raw pointers to [Epiface](#) objects for dispatching calls. Instances must therefore never be copied or moved.

Definition at line 156 of file [ipc\\_epiface](#).

### 15.97.2 Member Function Documentation

#### 15.97.2.1 dispatch()

```
virtual l4_msgtag_t L4::Epiface::dispatch (
    l4_msgtag_t tag,
    unsigned rights,
    l4_utcb_t * utcb ) [pure virtual]
```

The abstract handler for client requests to the object.

#### Parameters

<i>tag</i>	The message tag for this invocation.
<i>rights</i>	The rights bits in the invoked capability.
<i>utcb</i>	The UTCB used for the invocation.

#### Return values

<code>-L4_ENOREPLY</code>	No reply message is send.
<code>&lt;0</code>	Error, reply with error code.
<code>&gt;=0</code>	Success, reply with return value.

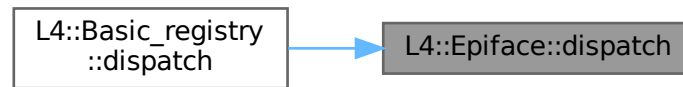
This function must be implemented by application specific server objects.

Implemented in [L4::Epiface\\_t< Block\\_dev< Ds\\_data >, L4virtio::Device >, L4::Epiface\\_t< Null\\_handler, L4::Kobject >, L4::Epiface\\_t< Virtio\\_client< DEV >, L4virtio::Device >, L4::Epiface\\_t< Derived, IFACE, BASE, bool >, L4::Server\\_object, L4::lrqep\\_t< Host\\_irq >, L4::lrqep\\_t< Irq\\_object >, and L4::lrqep\\_t< Derived, BASE, bool >](#).

Referenced by [L4::Basic\\_registry::dispatch\(\)](#).



Here is the caller graph for this function:



### 15.97.2.2 get\_buffer\_demand()

```
virtual Demand L4::Epiface::get_buffer_demand ( ) const [pure virtual]
```

Get the server-side receive buffer demand for this object.

#### Note

This function is usually not implemented directly, but by using [Server\\_object\\_t](#) template with an IPC interface definition.

#### Returns

The needed server-side receive buffers for this object

Implemented in [L4::Epiface\\_t0< IFACE, L4::Epiface >](#), [L4::Epiface\\_t0< L4::Kobject, L4::Epiface >](#), [L4::Epiface\\_t0< L4::virtio::Device, L4::Epiface\\_t0< void, Epiface >, L4::Epiface\\_t0< RPC\\_IFACE, BASE >, L4::Server\\_object\\_t< IFACE, BASE >, L4::Server\\_object\\_t< IFACE, L4::Server\\_object >, and L4::Server\\_object\\_t< Kobject >](#).

### 15.97.2.3 obj\_cap()

```
Stored_cap L4::Epiface::obj_cap ( ) const [inline]
```

Get the capability to the kernel object belonging to this object.

#### Returns

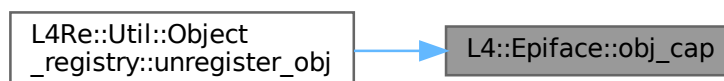
Capability for the kernel object behind the server.

This is usually either an [lpc\\_gate](#) or an [lirq](#).

Definition at line 217 of file [ipc\\_epiface](#).

Referenced by [L4Re::Util::Object\\_registry::unregister\\_obj\(\)](#).

Here is the caller graph for this function:



### 15.97.2.4 server\_iface()

```
Server_iface * L4::Epiface::server_iface ( ) const [inline]
```

Get pointer to server interface at which the object is currently registered.

#### Returns

Pointer to the server at which the object is currently registered, NULL if the object is not registered at any server.

Definition at line 224 of file [ipc\\_epiface](#).

### 15.97.2.5 set\_server()

```
int L4::Epiface::set_server (
    Server_iface * srv,
    Cap< void > cap,
    bool managed = false ) [inline]
```

Set server registration info for the object.

#### Parameters

<i>srv</i>	The server to register at
<i>cap</i>	The capability that connects the object.
<i>managed</i>	Mark the capability as managed or unmanaged. Typical server implementations use this flag to remember whether the capability was internally allocated or not.

#### Returns

0 on success, -L4\_EINVAL if the srv and cap are not consistent.

Definition at line 235 of file [ipc\\_epiface](#).

References [L4\\_EINVAL](#).

Referenced by [L4Re::Util::Object\\_registry::unregister\\_obj\(\)](#).

Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

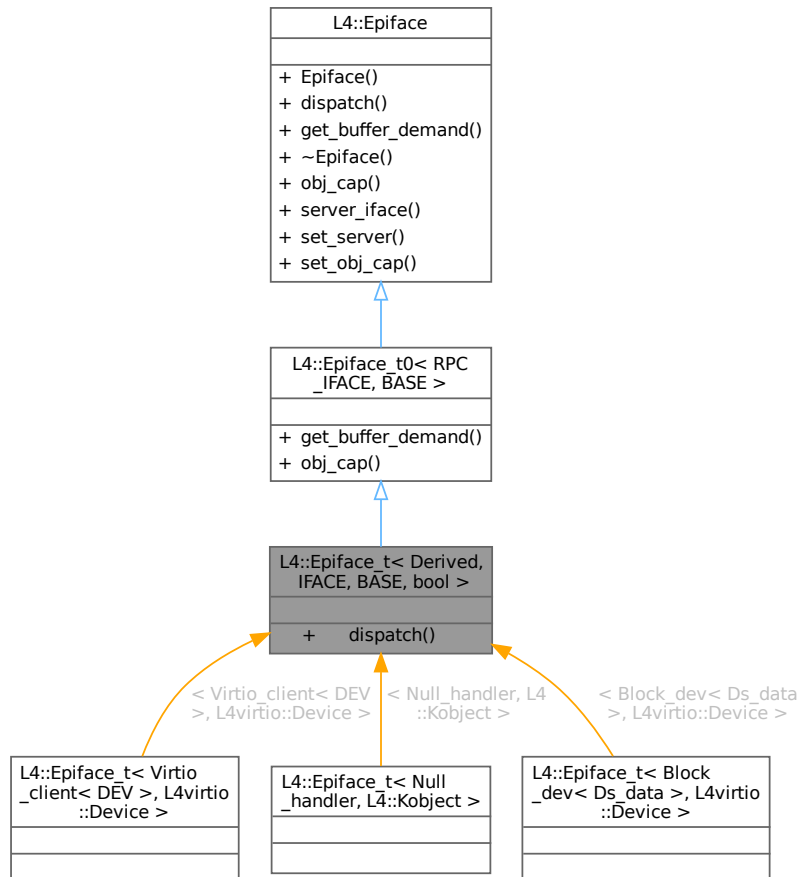
- [l4/sys/cxx/ipc\\_epiface](#)

## 15.98 L4::Epiface\_t< Derived, IFACE, BASE, bool > Struct Template Reference

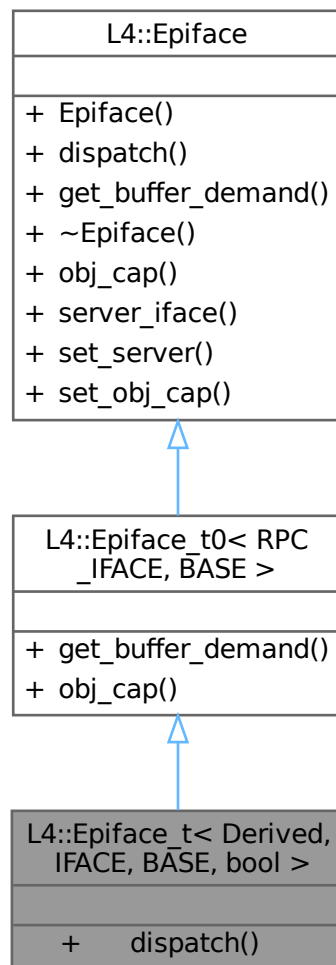
[Epiface](#) implementation for Kobject-based interface implementations.

```
#include <ipc_epiface>
```

Inheritance diagram for L4::Epiface\_t< Derived, IFACE, BASE, bool >:



Collaboration diagram for L4::Epiface\_t< Derived, IFACE, BASE, bool >:



### Public Member Functions

- [l4\\_msgtag\\_t dispatch](#) ([l4\\_msgtag\\_t](#) tag, unsigned rights, [l4\\_utcb\\_t](#) \*utcb) final  
The abstract handler for client requests to the object.

### Public Member Functions inherited from [L4::Epiface\\_t0< RPC\\_IFACE, BASE >](#)

- [Type\\_info::Demand](#) [get\\_buffer\\_demand](#) () const  
Get the server-side buffer demand based in IFACE.
- [Cap< RPC\\_IFACE >](#) [obj\\_cap](#) () const  
Get the (typed) capability to this object.

## Public Member Functions inherited from L4::Epiface

- **Epiface** ()  
*Make a server object.*
- virtual **~Epiface** ()=0  
*Destroy the object.*
- Stored\_cap **obj\_cap** () const  
*Get the capability to the kernel object belonging to this object.*
- **Server\_iface** \* **server\_iface** () const  
*Get pointer to server interface at which the object is currently registered.*
- int **set\_server** (**Server\_iface** \*srv, **Cap**< void > cap, bool managed=false)  
*Set server registration info for the object.*
- void **set\_obj\_cap** (**Cap**< void > const &cap)  
*Deprecated server registration function.*

## Additional Inherited Members

## Public Types inherited from L4::Epiface\_t0< RPC\_IFACE, BASE >

- typedef **RPC\_IFACE** **Interface**  
*Data type of the IPC interface definition.*

## Public Types inherited from L4::Epiface

- typedef **lpc\_svr::Server\_iface** **Server\_iface**  
*Type for abstract server interface.*
- typedef **lpc\_svr::Server\_iface::Demand** **Demand**  
*Type for server-side receive buffer demand.*

## 15.98.1 Detailed Description

```
template<typename Derived, typename IFACE, typename BASE = L4::Epiface, bool = cxx::is_↔
polymorphic<BASE>::value>
struct L4::Epiface_t< Derived, IFACE, BASE, bool >
```

**Epiface** implementation for Kobject-based interface implementations.

### Template Parameters

<i>Derived</i>	Class providing the interface implementations.
<i>BASE</i>	<b>Epiface</b> base class.

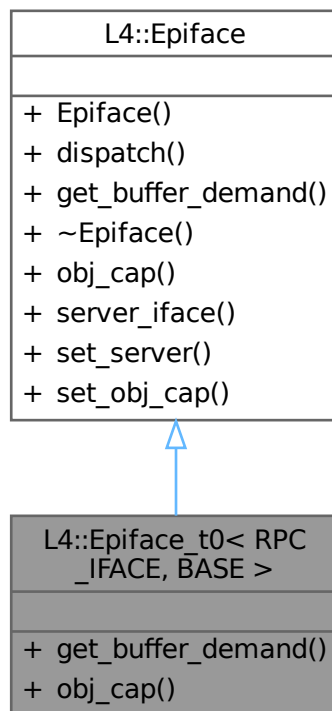
### Examples

[examples/clntsrv/server.cc](#).

Definition at line 514 of file [ipc\\_epiface](#).



Collaboration diagram for L4::Epiface\_t0< RPC\_IFACE, BASE >:



## Public Types

- typedef `RPC_IFACE` **Interface**  
*Data type of the IPC interface definition.*

## Public Types inherited from **L4::Epiface**

- typedef `lpc_svr::Server_iface` **Server\_iface**  
*Type for abstract server interface.*
- typedef `lpc_svr::Server_iface::Demand` **Demand**  
*Type for server-side receive buffer demand.*

## Public Member Functions

- `Type_info::Demand` **get\_buffer\_demand** () const  
*Get the server-side buffer demand based in IFACE.*
- `Cap< RPC_IFACE >` **obj\_cap** () const  
*Get the (typed) capability to this object.*

## Public Member Functions inherited from [L4::Epiface](#)

- **Epiface** ()  
*Make a server object.*
- virtual [l4\\_msgtag\\_t](#) **dispatch** ([l4\\_msgtag\\_t](#) tag, unsigned rights, [l4\\_utcb\\_t](#) \*utcb)=0  
*The abstract handler for client requests to the object.*
- virtual **~Epiface** ()=0  
*Destroy the object.*
- Stored\_cap [obj\\_cap](#) () const  
*Get the capability to the kernel object belonging to this object.*
- [Server\\_iface](#) \* [server\\_iface](#) () const  
*Get pointer to server interface at which the object is currently registered.*
- int **set\_server** ([Server\\_iface](#) \*srv, [Cap](#)< void > cap, bool managed=false)  
*Set server registration info for the object.*
- void **set\_obj\_cap** ([Cap](#)< void > const &cap)  
*Deprecated server registration function.*

### 15.99.1 Detailed Description

```
template<typename RPC_IFACE, typename BASE = Epiface>
struct L4::Epiface_t0< RPC_IFACE, BASE >
```

[Epiface](#) mixin for generic Kobject-based interfaces.

#### Template Parameters

<i>RPC_IFACE</i>	Data type of the IPC interface definition.
<i>BASE</i>	Base <a href="#">Epiface</a> class.

Definition at line 267 of file [ipc\\_epiface](#).

### 15.99.2 Member Function Documentation

#### 15.99.2.1 [obj\\_cap\(\)](#)

```
template<typename RPC_IFACE , typename BASE = Epiface>
Cap< RPC_IFACE > L4::Epiface\_t0< RPC_IFACE, BASE >::obj_cap ( ) const [inline]
```

Get the (typed) capability to this object.

#### Returns

Capability for the kernel object behind the server.

Definition at line 280 of file [ipc\\_epiface](#).

The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc\\_epiface](#)



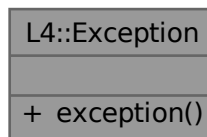
## 15.100 L4::Exception Class Reference

[Exception](#) interface.

```
#include <exception>
```

Inherits L4::Kobject\_0t< Derived, PROTO, S\_DEMAND >.

Collaboration diagram for L4::Exception:



### Public Member Functions

- [l4\\_msgtag\\_t](#) [exception](#) ([L4::lpc::In\\_out](#)< [l4\\_exc\\_regs\\_t](#) \* > regs, [L4::lpc::Rcv\\_fpage](#) rwin, [L4::lpc::Opt](#)< [L4::lpc::Snd\\_fpage](#) & > fp)  
*Exception call.*

### 15.100.1 Detailed Description

[Exception](#) interface.

This class defines the interface for handling exception IPC. When an exception occurs during program execution, for example due to a division by zero, the kernel will synthesise an exception IPC and send it to the thread's exception handler, who can then handle it.

The exception handler is set with the [L4::Thread::control](#) interface.

Definition at line 42 of file [exception](#).

### 15.100.2 Member Function Documentation

#### 15.100.2.1 exception()

```
l4\_msgtag\_t L4::Exception::exception (
    L4::lpc::In\_out< l4\_exc\_regs\_t * > regs,
    L4::lpc::Rcv\_fpage rwin,
    L4::lpc::Opt< L4::lpc::Snd\_fpage & > fp )
```

[Exception](#) call.

## Parameters

	<i>regs</i>	Register state of the faulting thread.
	<i>rwin</i>	Receive window in the address space.
out	<i>fp</i>	Optional flex-page to resolve the exception.

## Returns

Message tag containing error code.

The documentation for this class was generated from the following file:

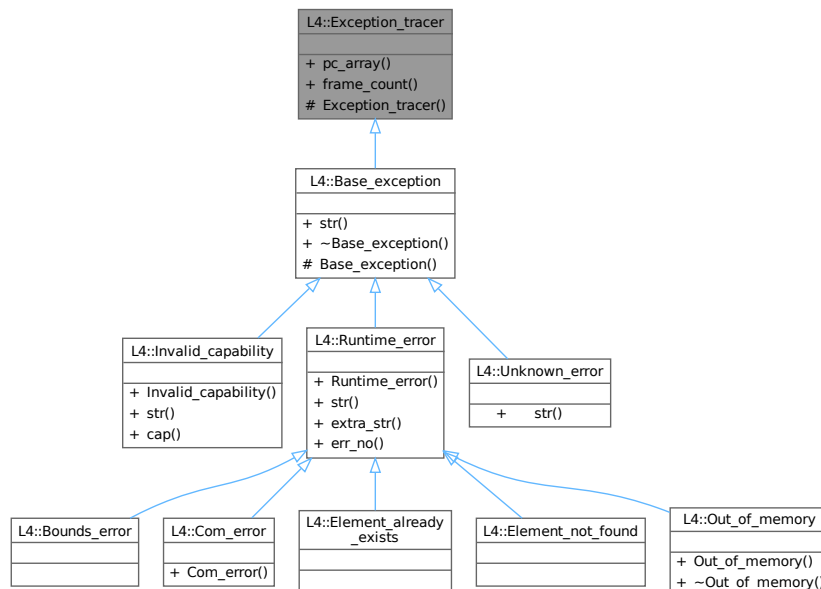
- [l4/sys/exception](#)

## 15.101 L4::Exception\_tracer Class Reference

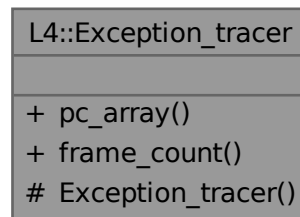
Back-trace support for exceptions.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Exception\_tracer:



Collaboration diagram for L4::Exception\_tracer:



### Public Member Functions

- void const \*const \* **pc\_array** () const noexcept  
*Get the array containing the call trace.*
- int **frame\_count** () const noexcept  
*Get the number of entries that are valid in the call trace.*

### Protected Member Functions

- **Exception\_tracer** () noexcept  
*Create a back trace.*

## 15.101.1 Detailed Description

Back-trace support for exceptions.

This class holds an array of at most [L4\\_CXX\\_EXCEPTION\\_BACKTRACE](#) instruction pointers containing the call trace at the instant when an exception was thrown.

Definition at line 62 of file [exceptions](#).

The documentation for this class was generated from the following file:

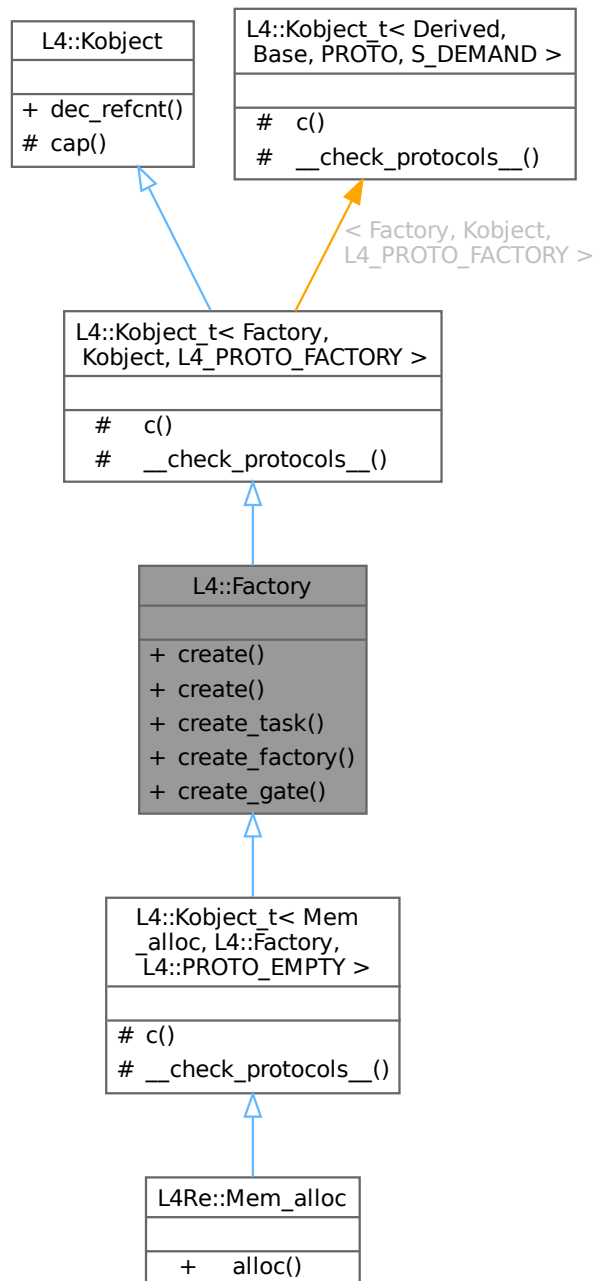
- [l4/cxx/exceptions](#)

## 15.102 L4::Factory Class Reference

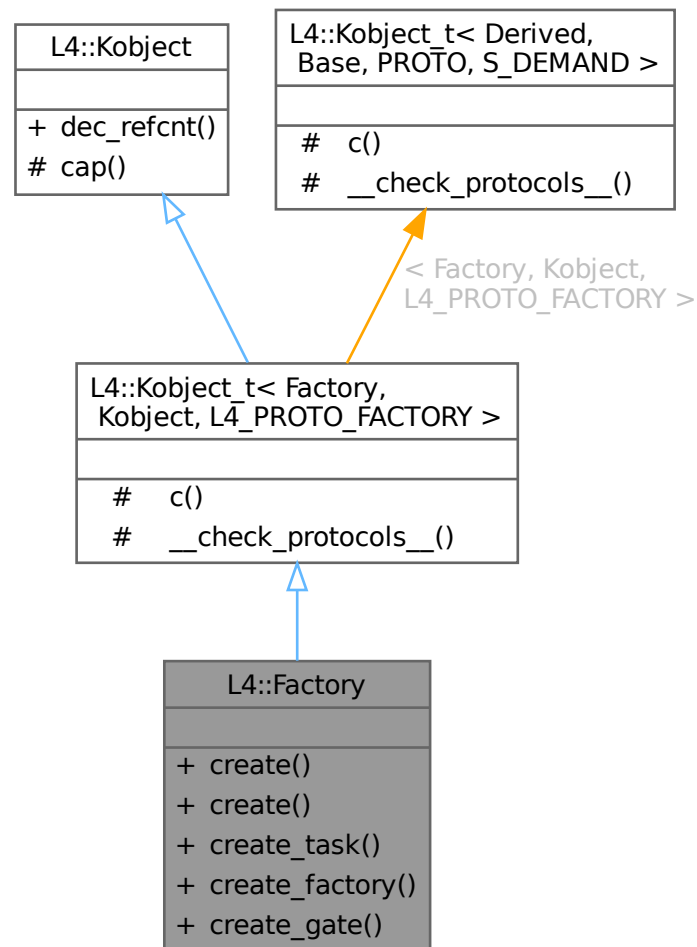
C++ Factory interface, see [Factory](#) for the C interface.

```
#include <factory>
```

Inheritance diagram for L4::Factory:



Collaboration diagram for L4::Factory:



## Data Structures

- struct [Lstr](#)  
*Special type to add a pascal string into the factory create stream.*
- struct [Nil](#)  
*Special type to add a void argument into the factory create stream.*
- class [S](#)  
*Stream class for the [create\(\)](#) argument stream.*

## Public Member Functions

- [S create](#) ([Cap](#)< void > target, long obj, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Generic create call to the factory.*
- template<typename OBJ >  
[S create](#) ([Cap](#)< OBJ > target, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept

*Create call for typed capabilities.*

- `l4_msgtag_t create_task (Cap< Task > const &target_cap, l4_fpage_t *utcb_area, l4_utcb_t *utcb=l4_utcb()) noexcept`

*Create a new task.*

- `l4_msgtag_t create_factory (Cap< Factory > const &target_cap, unsigned long limit, l4_utcb_t *utcb=l4_utcb()) noexcept`

*Create a new factory.*

- `l4_msgtag_t create_gate (Cap< void > const &target_cap, Cap< Thread > const &thread_cap, l4_umword_t label, l4_utcb_t *utcb=l4_utcb()) noexcept`

*Create a new IPC gate.*

## Public Member Functions inherited from `L4::Kobject`

- `l4_msgtag_t dec_refcnt (l4_mword_t diff, l4_utcb_t *utcb=l4_utcb())`

*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

## Protected Types inherited from `L4::Kobject_t< Factory, Kobject, L4_PROTO_FACTORY >`

- typedef `Factory Class`

*The target interface type (inheriting from `Kobject_t`)*

- typedef `Typeid::Iface< PROTO, Factory > __Iface`

*The interface description for the derived class.*

- typedef `Typeid::Merge_list< Typeid::Iface_list< __Iface >, typename Base::__Iface_list > __Iface_list`

*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from `L4::Kobject_t< Factory, Kobject, L4_PROTO_FACTORY >`

- `L4::Cap< Class > c () const noexcept`

*Get the capability to ourselves.*

## Protected Member Functions inherited from `L4::Kobject`

- `l4_cap_idx_t cap () const noexcept`

*Return capability selector.*

## Static Protected Member Functions inherited from `L4::Kobject_t< Factory, Kobject, L4_PROTO_FACTORY >`

- static void `__check_protocols__ () noexcept`

*Helper to check for protocol conflicts.*

### 15.102.1 Detailed Description

C++ Factory interface, see [Factory](#) for the C interface.

Factories provide an interface to create objects which are accessed via capabilities.

For additional information about which objects can be created via this interface, see server-specific information in [Kernel Factory](#) and [L4Re Servers](#).

#### Include File

```
#include <l4/sys/factory>
```

For the C interface refer to [Factory](#).

Definition at line 48 of file [factory](#).

### 15.102.2 Member Function Documentation

#### 15.102.2.1 create() [1/2]

```
template<typename OBJ >
S L4::Factory::create (
    Cap< OBJ > target,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

Create call for typed capabilities.

#### Template Parameters

<i>OBJ</i>	Capability type of the object to be created.
------------	--

#### Parameters

out	<i>target</i>	Capability of type OBJ.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

#### Returns

A create stream that allows additional arguments to be passed to the [create\(\)](#) call via the left-shift (<<) operator.

This method does not directly invoke the factory. The factory is invoked when the create stream returned by this method is converted to an [l4\\_msgtag\\_t](#), or otherwise when the stream goes out of scope.

#### Note

Refer to [S::operator l4\\_msgtag\\_t \(\)](#) for description of error codes in the returned create stream.

The create stream uses the UTCB to store parameters for the service call. During the lifetime of a create stream or, until it is converted to a [l4\\_msgtag\\_t](#), other UTCB-using operations must not be used.

**Usage:**

```
L4::Cap<L4Re::Dataspace> ds = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
factory->create(ds) << l4_mword_t(size_in_bytes);
```

Definition at line 308 of file [factory](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**15.102.2.2 create() [2/2]**

```
S L4::Factory::create (
    Cap< void > target,
    long obj,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

Generic create call to the factory.

**Parameters**

out	<i>target</i>	Capability selector for the new object. The caller must allocate the capability slot. The kernel stores the new objects's capability into this slot.
	<i>obj</i>	The protocol ID that specifies which kind of object shall be created.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

**Returns**

A create stream that allows additional arguments to be passed to the [create\(\)](#) call via the left-shift (<<) operator.

This method does not directly invoke the factory. The factory is invoked when the create stream returned by this method is converted to an [l4\\_msgtag\\_t](#), or otherwise when the stream goes out of scope.

**Note**

Refer to [S::operator l4\\_msgtag\\_t \(\)](#) for description of error codes in the returned create stream.

The create stream uses the UTCB to store parameters for the service call. During the lifetime of a create stream or, until it is converted to a [l4\\_msgtag\\_t](#), other UTCB-using operations must not be used.



See also

[create\(Cap<OBJ>, l4\\_utcb\\_t \\*\)](#)

Definition at line 274 of file [factory](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 15.102.2.3 create\_factory()

```

l4_msgtag_t L4::Factory::create_factory (
    Cap< Factory > const & target_cap,
    unsigned long limit,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Create a new factory.

#### Parameters

out	<i>target_cap</i>	The kernel stores the new factory's capability into this slot.
	<i>limit</i>	Limit for the new factory in bytes.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

#### Returns

Syscall return tag

#### Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	The factory instance requires <a href="#">L4_CAP_FPAGE_S</a> rights on the invoked capability and <a href="#">L4_CAP_FPAGE_S</a> is not present.
<i>&lt;0</i>	Error code.

**Note**

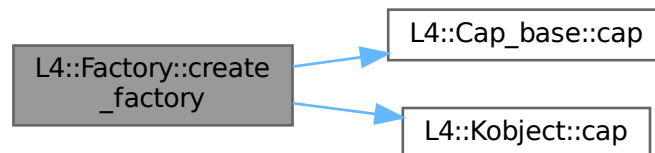
In addition to memory needed for internal data structures, the `limit` (quota) of the new factory is counted towards the quota of the creating factory. The `limit` must be within  $1 \leq \text{limit} \leq 2^8 * \text{sizeof}(\text{l4\_umword\_t}) - 1) - 2$  otherwise the behavior is undefined.

This method is only guaranteed to work with the [Kernel Factory](#). For other services, use the generic [create\(\)](#) method and consult the service documentation for information on the arguments that need to be passed to the create stream.

Definition at line 381 of file [factory](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**15.102.2.4 create\_gate()**

```

l4_msgtag_t L4::Factory::create_gate (
    Cap< void > const & target_cap,
    Cap< Thread > const & thread_cap,
    l4_umword_t label,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Create a new IPC gate.

**Parameters**

out	<i>target_cap</i>	The kernel stores the new IPC gate's capability into this slot.
	<i>thread_cap</i>	Optional capability selector of a thread to bind the gate to. Use <a href="#">L4_INVALID_CAP</a> to create an unbound IPC gate.
	<i>label</i>	Optional label of the gate (precisely used if <i>thread_cap</i> is valid). If <i>thread_cap</i> is valid, <i>label</i> must be present.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

**Returns**

Syscall return tag containing one of the following return codes.

## Return values

<code>L4_EOK</code>	No error occurred.
<code>-L4_ENOMEM</code>	Out-of-memory during allocation of the <code>lpc_gate</code> object.
<code>-L4_EINVAL</code>	<code>thread_cap</code> is void or points to something that is not a thread.
<code>-L4_EPERM</code>	The factory instance requires <code>L4_CAP_FPAGE_S</code> rights on the invoked capability or <code>thread_cap</code> and <code>L4_CAP_FPAGE_S</code> is not present.

An unbound IPC gate can be bound to a thread using `L4::lpc_gate::bind_thread()`.

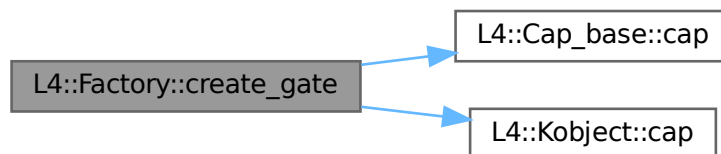
## See also

[L4::lpc\\_gate](#)

Definition at line 414 of file [factory](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



## 15.102.2.5 create\_task()

```

14_msgtag_t L4::Factory::create_task (
    Cap< Task > const & target_cap,
    14_fpage_t * utcb_area,
    14_utcb_t * utcb = 14_utcb() ) [inline], [noexcept]
  
```

Create a new task.

## Parameters

out	<code>target_cap</code>	The kernel stores the new task's capability into this slot.
in, out	<code>utcb_area</code>	Flexpage that describes an area in the address space of the new task, where the kernel should map the kernel-allocated kernel-user memory to. The kernel uses the kernel-user memory to store UTCBs and vCPU state-save-areas of the new task.

On systems without MMU, the flexpage is adjusted to reflect the actually allocated physical address.

## Parameters

<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .
-------------	--

## Returns

Syscall return tag

## Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	The factory instance requires <a href="#">L4_CAP_FPAGE_S</a> rights on the invoked capability and <a href="#">L4_CAP_FPAGE_S</a> is not present.
<i>&lt; 0</i>	Error code.

## Note

The size of the UTCB area specifies indirectly the number of UTCBs available for this task. Refer to [L4::Task::add\\_ku\\_mem](#) for adding more of this type of memory.

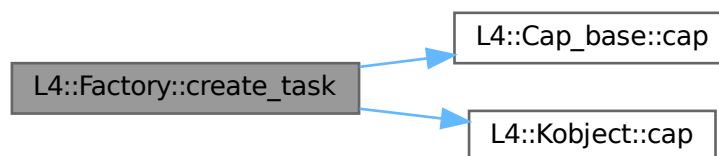
## See also

[L4::Task](#)

Definition at line 348 of file [factory](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- [l4/sys/factory](#)

## 15.103 L4::Factory::Lstr Struct Reference

Special type to add a pascal string into the factory create stream.

```
#include <factory>
```

Collaboration diagram for L4::Factory::Lstr:

L4::Factory::Lstr	
+	s
+	len
+	Lstr()

### Public Member Functions

- [Lstr](#) (char const \*[s](#), unsigned [len](#)) noexcept

### Data Fields

- char const \* **s**  
*The character buffer.*
- unsigned **len**  
*The number of characters in the buffer.*

### 15.103.1 Detailed Description

Special type to add a pascal string into the factory create stream.

This encapsulates a string that has an explicit length.

Definition at line [64](#) of file [factory](#).

### 15.103.2 Constructor & Destructor Documentation

#### 15.103.2.1 Lstr()

```
L4::Factory::Lstr::Lstr (  
    char const * s,  
    unsigned len ) [inline], [noexcept]
```

## Parameters

<i>s</i>	Pointer to the c-style string.
<i>len</i>	Length in number of characters of the string <i>s</i> .

Definition at line 80 of file [factory](#).

The documentation for this struct was generated from the following file:

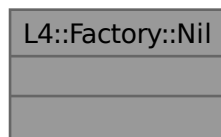
- [l4/sys/factory](#)

## 15.104 L4::Factory::Nil Struct Reference

Special type to add a void argument into the factory create stream.

```
#include <factory>
```

Collaboration diagram for L4::Factory::Nil:



### 15.104.1 Detailed Description

Special type to add a void argument into the factory create stream.

Definition at line 57 of file [factory](#).

The documentation for this struct was generated from the following file:

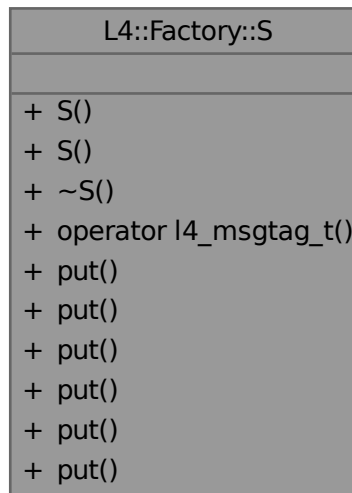
- [l4/sys/factory](#)

## 15.105 L4::Factory::S Class Reference

Stream class for the [create\(\)](#) argument stream.

```
#include <factory>
```

Collaboration diagram for L4::Factory::S:



### Public Member Functions

- **S** (**S** &&o) noexcept  
*Move ...*
- **S** (**l4\_cap\_idx\_t** f, long obj, **L4::Cap**< void > target, **l4\_utcb\_t** \*utcb) noexcept  
*Create a stream for a specific [create\(\)](#) call.*
- **~S** () noexcept  
*Commit the operation in the destructor to have a cool syntax for [create\(\)](#).*
- **operator l4\_msgtag\_t** () noexcept  
*Explicitly commits the operation and returns the result.*
- void **put** (**l4\_mword\_t** i) noexcept  
*Put a single **l4\_mword\_t** as next argument.*
- void **put** (**l4\_umword\_t** i) noexcept  
*Put a single **l4\_umword\_t** as next argument.*
- void **put** (char const \*s) &noexcept  
*Add a zero-terminated string as next argument.*
- void **put** (**Lstr** const &s) &noexcept  
*Add a pascal string as next argument.*
- void **put** (**Nil**) &noexcept  
*Add an empty argument.*
- void **put** (**l4\_fpage\_t** d) &noexcept  
*Add a flex page as next argument.*

### 15.105.1 Detailed Description

Stream class for the [create\(\)](#) argument stream.

This stream allows a variable number of arguments to be added to a [create\(\)](#) call.

Definition at line 89 of file [factory](#).

### 15.105.2 Constructor & Destructor Documentation

#### 15.105.2.1 S() [1/2]

```
L4::Factory::S::S (
    S && o ) [inline], [noexcept]
```

Move ...

##### Parameters

<i>o</i>	Instance of <a href="#">S</a> to move.
----------	--

Definition at line 108 of file [factory](#).

References [l4\\_msgtag\\_t::raw](#).

#### 15.105.2.2 S() [2/2]

```
L4::Factory::S::S (
    l4_cap_idx_t f,
    long obj,
    L4::Cap< void > target,
    l4_utcb_t * utcb ) [inline], [noexcept]
```

Create a stream for a specific [create\(\)](#) call.

##### Parameters

	<i>f</i>	The capability for the factory object ( <a href="#">L4::Factory</a> ).
	<i>obj</i>	The protocol ID to describe the type of the object that shall be created.
out	<i>target</i>	The capability selector for the new object. The caller must allocate the capability slot. The kernel stores the new object's capability into this slot.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

Definition at line 132 of file [factory](#).



### 15.105.3 Member Function Documentation

#### 15.105.3.1 operator l4\_msgtag\_t()

```
L4::Factory::S::operator l4_msgtag_t ( ) [inline], [noexcept]
```

Explicitly commits the operation and returns the result.

##### Returns

The result of the [create\(\)](#) operation.

##### Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	The factory instance requires <a href="#">L4_CAP_FPAGE_S</a> rights on the invoked capability and <a href="#">L4_CAP_FPAGE_S</a> is not present.
<i>&lt;0</i>	Error code.

Definition at line [158](#) of file [factory](#).

References [l4\\_msgtag\\_t::raw](#).

#### 15.105.3.2 put() [1/5]

```
void L4::Factory::S::put (
    char const * s ) & [inline], [noexcept]
```

Add a zero-terminated string as next argument.

##### Parameters

<i>s</i>	The string to add as next argument.
----------	-------------------------------------

The string will be added with the zero-terminator.

Definition at line [192](#) of file [factory](#).

#### 15.105.3.3 put() [2/5]

```
void L4::Factory::S::put (
    l4_fpage_t d ) & [inline], [noexcept]
```

Add a flex page as next argument.

##### Parameters

<i>d</i>	The flex page to add (there will be no map operation).
----------	--

Definition at line [224](#) of file [factory](#).

#### 15.105.3.4 put() [3/5]

```
void L4::Factory::S::put (
    l4_mword_t i ) [inline], [noexcept]
```

Put a single l4\_mword\_t as next argument.

##### Parameters

<i>i</i>	The value to add as next argument.
----------	------------------------------------

Definition at line [170](#) of file [factory](#).

#### 15.105.3.5 put() [4/5]

```
void L4::Factory::S::put (
    l4_umword_t i ) [inline], [noexcept]
```

Put a single l4\_umword\_t as next argument.

##### Parameters

<i>i</i>	The value to add as next argument.
----------	------------------------------------

Definition at line [180](#) of file [factory](#).

#### 15.105.3.6 put() [5/5]

```
void L4::Factory::S::put (
    Lstr const & s ) & [inline], [noexcept]
```

Add a pascal string as next argument.

##### Parameters

<i>s</i>	The string to add as next argument.
----------	-------------------------------------

The string will be added with the exact length given. It is the responsibility of the caller to make sure that the string is zero-terminated when that is required by the server.

Definition at line [206](#) of file [factory](#).

The documentation for this class was generated from the following file:

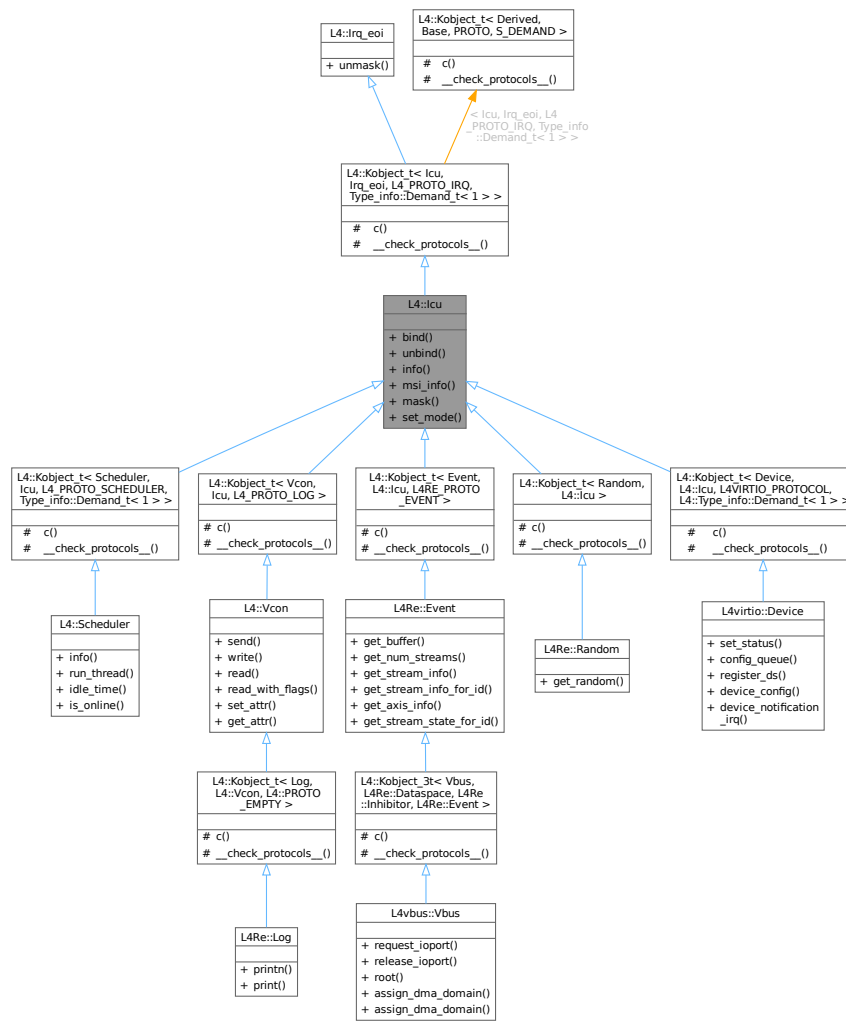
- l4/sys/[factory](#)

## 15.106 L4::Icu Class Reference

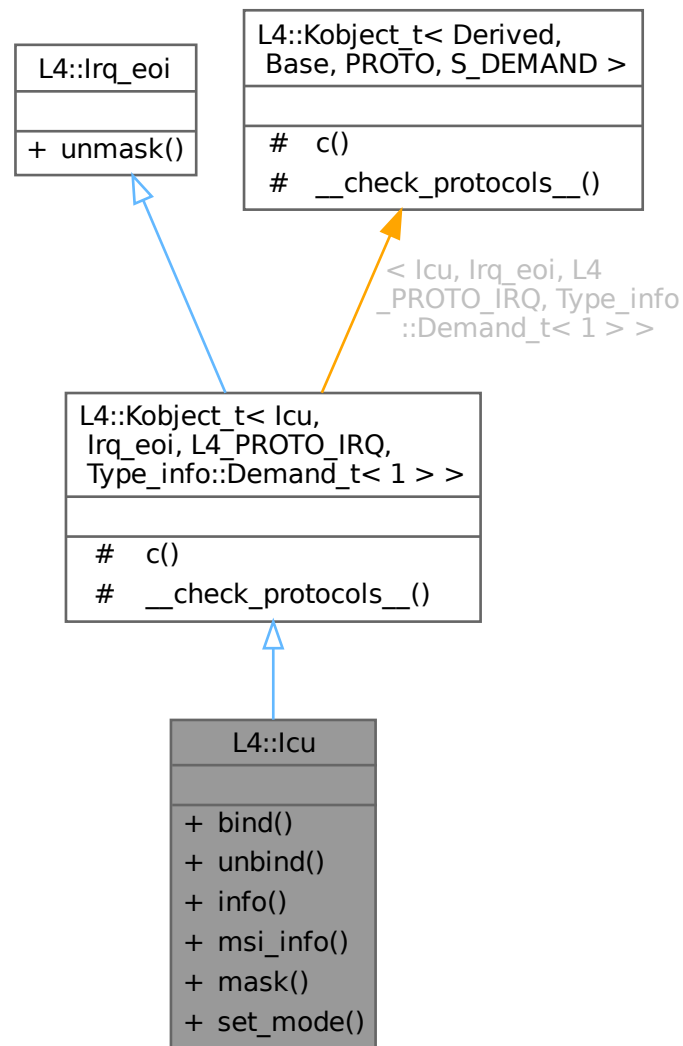
C++ [Icu](#) interface, see [Interrupt controller](#) for the C interface.

```
#include <irq>
```

Inheritance diagram for L4::Icu:



Collaboration diagram for L4::Icu:



## Data Structures

- class [Info](#)

*This class encapsulates information about an ICU.*

## Public Member Functions

- [l4\\_msgtag\\_t bind](#) (unsigned irqnum, [L4::Cap< Triggerable >](#) irq, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- [l4\\_msgtag\\_t unbind](#) (unsigned irqnum, [L4::Cap< Triggerable >](#) irq, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Remove binding of an interrupt line from the interrupt controller object.*
- [l4\\_msgtag\\_t info](#) ([l4\\_icu\\_info\\_t](#) \*info, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept

*Get information about the ICU features.*

- [l4\\_msgtag\\_t msi\\_info](#) ([l4\\_umword\\_t](#) irqnum, [l4\\_uint64\\_t](#) source, [l4\\_icu\\_msi\\_info\\_t](#) \*msi\_info)

*Get MSI info about IRQ.*

- [l4\\_msgtag\\_t mask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=L4\_IPC\_NEVER, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\\_t](#)()) noexcept

*Mask an IRQ line.*

- [l4\\_msgtag\\_t set\\_mode](#) (unsigned irqnum, [l4\\_umword\\_t](#) mode, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\\_t](#)()) noexcept

*Set interrupt mode.*

## Public Member Functions inherited from [L4::Irq\\_eoi](#)

- [l4\\_msgtag\\_t unmask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=L4\_IPC\_NEVER, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\\_t](#)()) noexcept

*Unmask the given interrupt line.*

## Additional Inherited Members

## Protected Types inherited from

[L4::Kobject\\_t](#)< [Icu](#), [Irq\\_eoi](#), [L4\\_PROTO\\_IRQ](#), [Type\\_info::Demand\\_t](#)< 1 > >

- typedef [Icu](#) **Class**

*The target interface type (inheriting from [Kobject\\_t](#))*

- typedef [Typeid::Iface](#)< [PROTO](#), [Icu](#) > **\_\_Iface**

*The interface description for the derived class.*

- typedef [Typeid::Merge\\_list](#)< [Typeid::Iface\\_list](#)< **\_\_Iface** >, typename [Base::\\_\\_Iface\\_list](#) > **\_\_Iface\_list**

*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Icu](#), [Irq\\_eoi](#), [L4\\_PROTO\\_IRQ](#), [Type\\_info::Demand\\_t](#)< 1 > >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept

*Get the capability to ourselves.*

## Static Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Icu](#), [Irq\\_eoi](#), [L4\\_PROTO\\_IRQ](#), [Type\\_info::Demand\\_t](#)< 1 > >

- static void **\_\_check\_protocols**\_\_ () noexcept

*Helper to check for protocol conflicts.*

### 15.106.1 Detailed Description

C++ [Icu](#) interface, see [Interrupt controller](#) for the C interface.

#### Note

"ICU" is short for "interrupt control unit".

This class defines the interface for interrupt controllers. It defines functions for binding [L4::Irq](#) objects to interrupt lines and other interrupt sources, as well as functions for masking and unmasking of interrupts.

To setup an interrupt line the following steps are required:

1. [set\\_mode\(\)](#) (optional if interrupt has a default mode)
2. [L4::Rcv\\_endpoint::bind\\_thread\(\)](#) to attach the [L4::Irq](#) object to a thread
3. [bind\(\)](#)
4. [unmask\(\)](#) to receive the first interrupt

For certain interrupt sources only some of these steps are necessary and supported, see [L4::Scheduler](#) and [L4::Vcon](#).

At most one [L4::Irq](#) object can be bound to a certain interrupt source and a certain [L4::Irq](#) object can be bound to at most one interrupt source.

#### Include File

```
#include <l4/sys/icu>
```

Definition at line 257 of file [irq](#).

### 15.106.2 Member Function Documentation

#### 15.106.2.1 bind()

```
l4_msgtag_t L4::Icu::bind (
    unsigned irqnum,
    L4::Cap< Triggerable > irq,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

Bind an interrupt line of an interrupt controller to an interrupt object.

#### Parameters

<i>irqnum</i>	IRQ line at the ICU.
<i>irq</i>	IRQ object for the given IRQ line to bind to this ICU.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

## Returns

Syscall return tag. The caller should check the return value using [l4\\_error\(\)](#) to check for errors and to identify the correct method for unmasking the interrupt. Return values < 0 indicate an error. A return value of 0 means a direct unmask via the IRQ object using [L4::Irq::unmask](#). A return value of 1 means that the interrupt has to be unmasked via the ICU using [L4::Icu::unmask](#).

## Return values

-L4_EINVAL	<code>irq</code> is bound to an interrupt source.
-L4_EPERM	The ICU instance requires <a href="#">L4_CAP_FPAGE_W</a> on <code>irq</code> and <a href="#">L4_CAP_FPAGE_W</a> was not present.

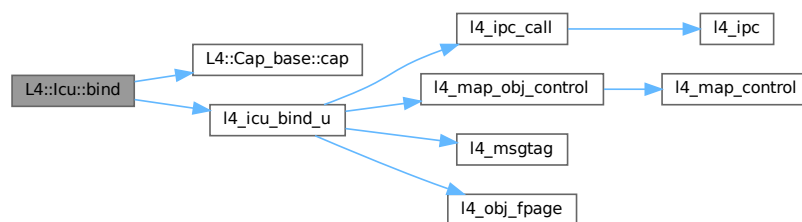
In case the `irq` is already bound to an interrupt source, it is unbound first. In case the `irq` is bound and the interrupt source is bound to a different [L4::Irq](#) object, only the unbinding happens. An [L4::Irq](#) object that is bound to an interrupt source will get unbound if the [L4::Irq](#) object is deleted.

Definition at line 316 of file [irq](#).

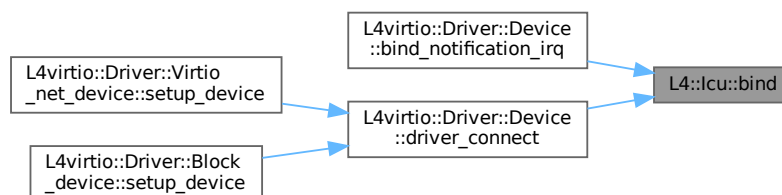
References [L4::Cap\\_base::cap\(\)](#), and [l4\\_icu\\_bind\\_u\(\)](#).

Referenced by [L4virtio::Driver::Device::bind\\_notification\\_irq\(\)](#), and [L4virtio::Driver::Device::driver\\_connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 15.106.2.2 info()

```

l4_msgtag_t L4::Icu::info (
    l4_icu_info_t * info,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Get information about the ICU features.

## Parameters

out	info	<a href="#">Info</a> structure to be filled with information.
	utcb	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

## Returns

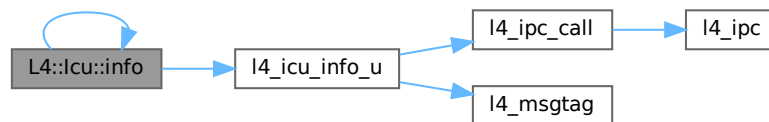
Syscall return tag

Definition at line 351 of file [irq](#).

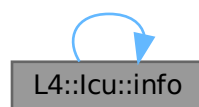
References [info\(\)](#), and [l4\\_icu\\_info\\_u\(\)](#).

Referenced by [info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.106.2.3 mask()

```

l4_msgtag_t L4::Icu::mask (
    unsigned irqnum,
    l4_umword_t * label = 0,
    l4_timeout_t to = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Mask an IRQ line.



## Parameters

<i>irqnum</i>	IRQ line at the ICU.
<i>label</i>	If NULL, this function is a send-only message to the ICU. If not NULL, this function will enter an open wait after sending the mask message and the received label is returned here.
<i>to</i>	The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non-NULL <i>label</i> only.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

## Returns

Syscall return tag. If *label* is NULL, this function performs an IPC send-only operation and there is no return value except [L4\\_MSGTAG\\_ERROR](#) indicating success or failure of the send operation. In this case use [l4\\_ipc\\_error\(\)](#) to check for errors and **do not** use [l4\\_error\(\)](#).

Definition at line 399 of file [irq](#).

References [l4\\_icu\\_mask\\_u\(\)](#).

Here is the call graph for this function:



## 15.106.2.4 msi\_info()

```

l4_msgtag_t L4::Icu::msi_info (
    l4_umword_t irqnum,
    l4_uint64_t source,
    l4_icu_msi_info_t * msi_info )
  
```

Get MSI info about IRQ.

## Parameters

	<i>irqnum</i>	IRQ line at the ICU.
	<i>source</i>	Platform dependent requester ID for MSIs. On IA32 we use a 20bit source filter value as described in the Intel IRQ remapping specification.
out	<i>msi_info</i>	A <a href="#">l4_icu_msi_info_t</a> structure receiving the address and the data value to trigger this MSI.

## Returns

Syscall return tag

### 15.106.2.5 set\_mode()

```
l4_msgtag_t L4::Icu::set_mode (
    unsigned irqnum,
    l4_umword_t mode,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

Set interrupt mode.

#### Parameters

<i>irqnum</i>	IRQ line at the ICU.
<i>mode</i>	Mode, see <a href="#">L4_irq_mode</a> .
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

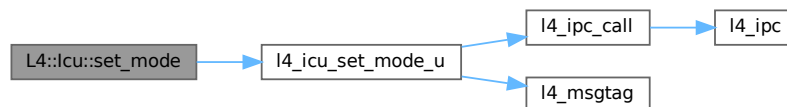
#### Returns

Syscall return tag

Definition at line 427 of file [irq](#).

References [l4\\_icu\\_set\\_mode\\_u\(\)](#).

Here is the call graph for this function:



### 15.106.2.6 unbind()

```
l4_msgtag_t L4::Icu::unbind (
    unsigned irqnum,
    L4::Cap< Triggerable > irq,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

Remove binding of an interrupt line from the interrupt controller object.

#### Parameters

<i>irqnum</i>	IRQ line at the ICU.
<i>irq</i>	IRQ object to remove from the ICU.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

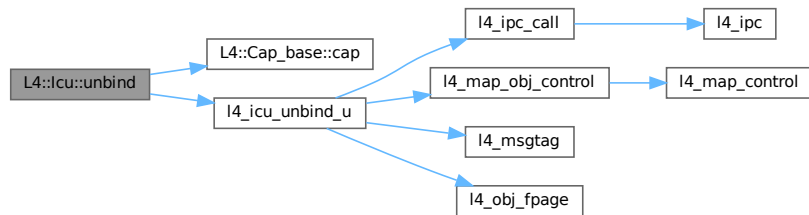
## Returns

Syscall return tag

Definition at line 334 of file [irq](#).

References [L4::Cap\\_base::cap\(\)](#), and [l4\\_icu\\_unbind\\_u\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

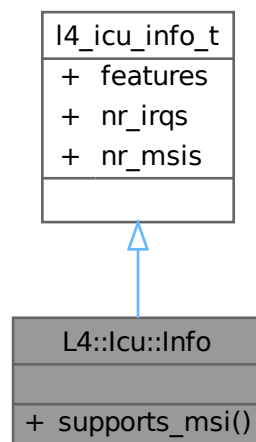
- [l4/sys/irq](#)

## 15.107 L4::lcu::Info Class Reference

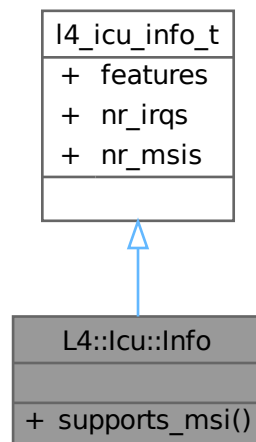
This class encapsulates information about an ICU.

```
#include <irq>
```

Inheritance diagram for L4::lcu::Info:



Collaboration diagram for L4::Icu::Info:



### Public Member Functions

- bool **supports\_msi** () const noexcept  
*True, if the ICU has support for MSIs.*

### Additional Inherited Members

#### Data Fields inherited from [l4\\_icu\\_info\\_t](#)

- unsigned [features](#)  
*Feature flags.*
- unsigned **nr\_irqs**  
*The number of IRQ lines supported by the ICU,.*
- unsigned **nr\_msis**  
*The number of MSI vectors supported by the ICU,.*

## 15.107.1 Detailed Description

This class encapsulates information about an ICU.

Definition at line [284](#) of file [irq](#).

The documentation for this class was generated from the following file:

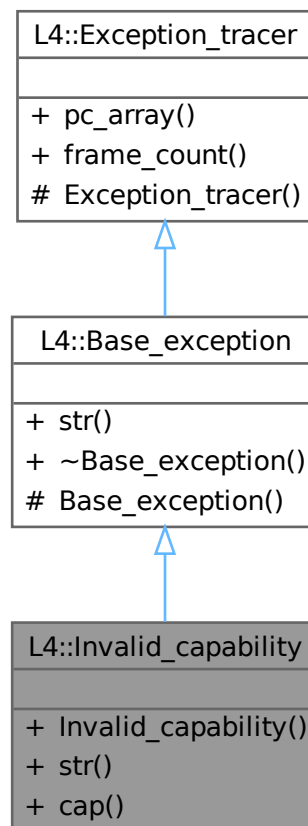
- [l4/sys/irq](#)

## 15.108 L4::Invalid\_capability Class Reference

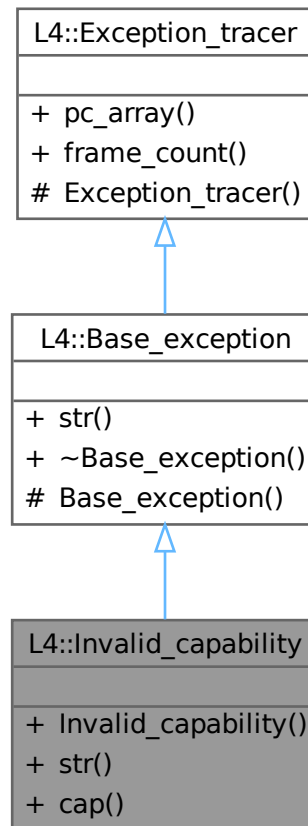
Indicates that an invalid object was invoked.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Invalid\_capability:



Collaboration diagram for L4::Invalid\_capability:



### Public Member Functions

- `Invalid_capability (Cap< void > const &o) noexcept`  
*Create an Invalid\_object exception for the Object o.*
- `char const * str () const noexcept` override  
*Return a human readable string for the exception.*
- `Cap< void > const & cap () const noexcept`  
*Get the object that caused the error.*

### Public Member Functions inherited from [L4::Base\\_exception](#)

- `virtual ~Base_exception () noexcept`  
*Destruction.*

### Public Member Functions inherited from [L4::Exception\\_tracer](#)

- `void const *const * pc_array () const noexcept`  
*Get the array containing the call trace.*
- `int frame_count () const noexcept`  
*Get the number of entries that are valid in the call trace.*

## Additional Inherited Members

## Protected Member Functions inherited from [L4::Base\\_exception](#)

- **Base\_exception** () noexcept  
*Create a base exception.*

## Protected Member Functions inherited from [L4::Exception\\_tracer](#)

- **Exception\_tracer** () noexcept  
*Create a back trace.*

## 15.108.1 Detailed Description

Indicates that an invalid object was invoked.

An Object is invalid if it has L4\_INVALID\_ID as server [L4](#) UID, or if the server does not know the object ID.

Definition at line [245](#) of file [exceptions](#).

## 15.108.2 Constructor & Destructor Documentation

### 15.108.2.1 Invalid\_capability()

```
L4::Invalid_capability::Invalid_capability (
    Cap< void > const & o ) [inline], [explicit], [noexcept]
```

Create an Invalid\_object exception for the Object o.

#### Parameters

<i>o</i>	The object that caused the server side error.
----------	---

Definition at line [255](#) of file [exceptions](#).

## 15.108.3 Member Function Documentation

### 15.108.3.1 cap()

```
Cap< void > const & L4::Invalid_capability::cap ( ) const [inline], [noexcept]
```

Get the object that caused the error.

#### Returns

The object that caused the error on invocation.

Definition at line [264](#) of file [exceptions](#).

The documentation for this class was generated from the following file:

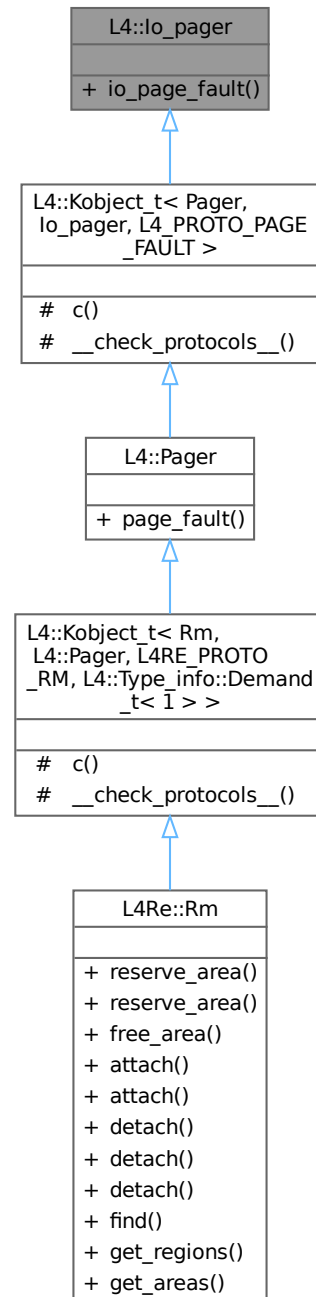
- [l4/cxx/exceptions](#)

## 15.109 L4::lo\_pager Class Reference

[lo\\_pager](#) interface.

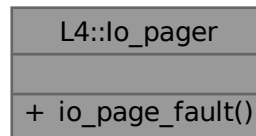
```
#include <pager>
```

Inheritance diagram for L4::lo\_pager:





Collaboration diagram for L4::io\_pager:



### Public Member Functions

- [l4\\_msgtag\\_t](#) [io\\_page\\_fault](#) ([l4\\_fpage\\_t](#) io\_pfa, [l4\\_umword\\_t](#) pc, [L4::lpc::Rcv\\_fpage](#) rwin, [L4::lpc::Opt<L4::lpc::Snd\\_fpage & >](#) fp)  
*IO page fault protocol message.*

## 15.109.1 Detailed Description

[io\\_pager](#) interface.

### Note

This interface is IA32 specific.

This class defines the interface for handling IO page faults. IO page faults happen when a thread tries to access an IO port that it does not currently have access to.

Depending on the microkernel's implementation, IO page faults can be handled in two ways.

If the microkernel does not support IO page faults, this IO pagefault interface is not used. Instead, the microkernel sends an exception IPC to the thread's exception handler ([L4::Exception](#)), indicating a GP (exception number 13). The exception handler must consult the faulting instruction to determine the cause of the exception. This is the default in Fiasco.OC.

In contrast, if the microkernel supports IO page faults, the microkernel will generate an IO page fault message and send it to the thread's page fault handler (pager). The page fault handler can implement this interface to handle the IO page faults.

### Note

A program may use this mechanism to implement a lazy IO port access scheme.

The page fault and exception handlers are set with the [L4::Thread::control](#) interface.

Definition at line 61 of file [pager](#).

## 15.109.2 Member Function Documentation

### 15.109.2.1 io\_page\_fault()

```

l4_msgtag_t L4::Io_pager::io_page_fault (
    l4_fpage_t io_pfa,
    l4_umword_t pc,
    L4::lpc::Rcv_fpage rwin,
    L4::lpc::Opt< L4::lpc::Snd_fpage & > fp )
  
```

IO page fault protocol message.

## Parameters

	<i>io_pfa</i>	Flex-page describing the faulting IO-port.
	<i>pc</i>	Faulting program counter.
	<i>rwin</i>	The receive window for a flex-page mapping.
out	<i>fp</i>	Optional: flex-page descriptor to send to the task raising the page fault.

## Returns

System call message tag; use [l4\\_error\(\)](#) to check for errors.

IO-port fault messages are usually generated by the kernel and an IO-page-fault handler needs to be in place to handle such faults and generate a reply, potentially filling in *fp*.

The documentation for this class was generated from the following file:

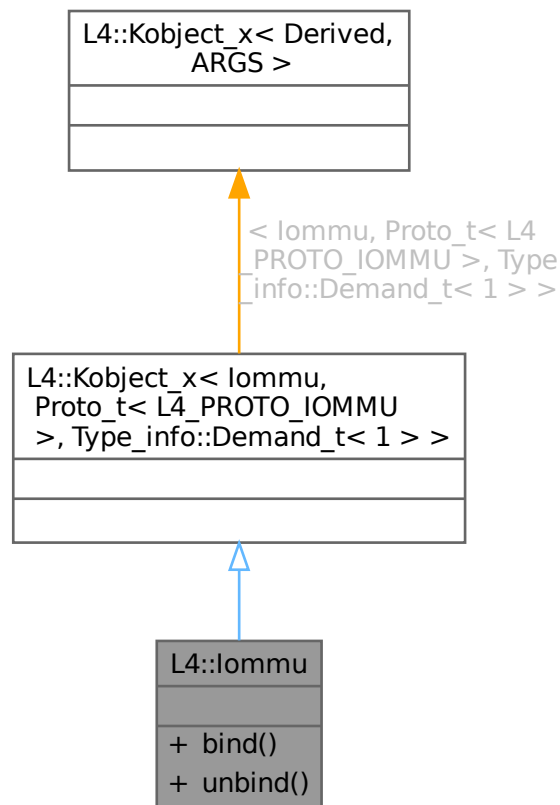
- [l4/sys/pager](#)

## 15.110 L4::lommu Class Reference

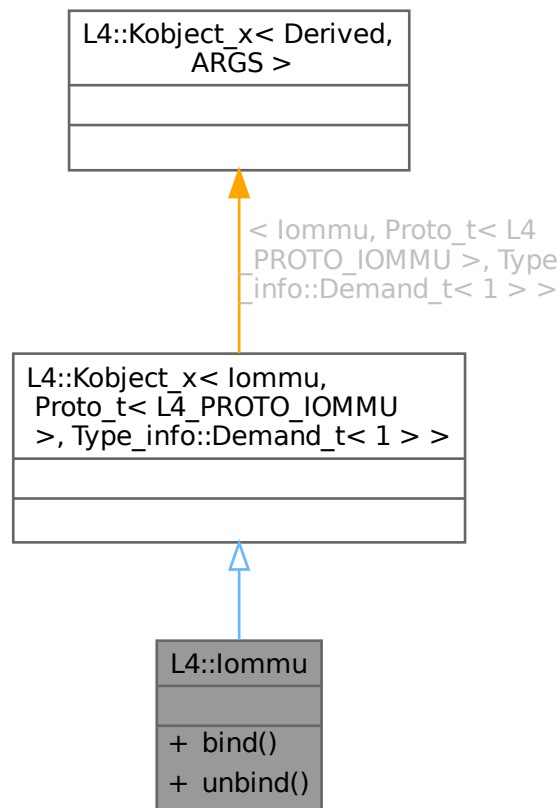
Interface for IO-MMUs used for DMA remapping.

```
#include <iommu>
```

Inheritance diagram for L4::lommu:



Collaboration diagram for L4::lommu:



### Public Member Functions

- `l4_msgtag_t bind (l4_uint64_t src_id, lpc::Cap< Task > dma_space)`  
Associate *dma\_space* with the set of device(s) specified by *src\_id*.
- `l4_msgtag_t unbind (l4_uint64_t src_id, lpc::Cap< Task > dma_space)`  
Remove the association of the given DMA address space from the device(s) specified by *src\_id*.

### 15.110.1 Detailed Description

Interface for IO-MMUs used for DMA remapping.

#### Note

This interface is only present in the kernel if the kernel detected an IOMMU during boot.

This interface allows to associate a DMA address space with a platform dependent set of devices. The kernel automatically keeps the memory spaces of associated DMA spaces in sync with the respective page table structures in the IOMMU.

Definition at line 21 of file [iommu](#).

## 15.110.2 Member Function Documentation

### 15.110.2.1 bind()

```
l4_msgtag_t L4::Iommu::bind (
    l4_uint64_t src_id,
    Ipc::Cap< Task > dma_space )
```

Associate `dma_space` with the set of device(s) specified by `src_id`.

Updates the respective page table structures in the IOMMU and keeps them in sync when memory is mapped to the `dma_space` or revoked from it.

#### Parameters

<i>src_id</i>	Platform dependent source ID specifying the set of devices that shall use <code>dma_space</code> for DMA remapping.
<i>dma_space</i>	The DMA space ( <a href="#">L4::Task</a> created with <code>L4_PROTO_DMA_SPACE</code> ) providing the mappings that shall be used for the device(s).

### 15.110.2.2 unbind()

```
l4_msgtag_t L4::Iommu::unbind (
    l4_uint64_t src_id,
    Ipc::Cap< Task > dma_space )
```

Remove the association of the given DMA address space from the device(s) specified by `src_id`.

Clear the respective page stable structures in the IOMMU.

#### Parameters

<i>src_id</i>	Platform dependent source ID specifying the set of devices that shall no longer use <code>dma_space</code> for DMA remapping.
<i>dma_space</i>	The DMA space formerly associated with <a href="#">bind()</a> .

The documentation for this class was generated from the following file:

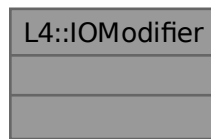
- `l4/sys/iommu`

## 15.111 L4::IOModifier Class Reference

Modifier class for the IO stream.

```
#include <basic_ostream>
```

Collaboration diagram for L4::IOModifier:



### 15.111.1 Detailed Description

Modifier class for the IO stream.

An IO Modifier can be used to change properties of an IO stream for example the number format.

Definition at line 33 of file [basic\\_ostream](#).

The documentation for this class was generated from the following file:

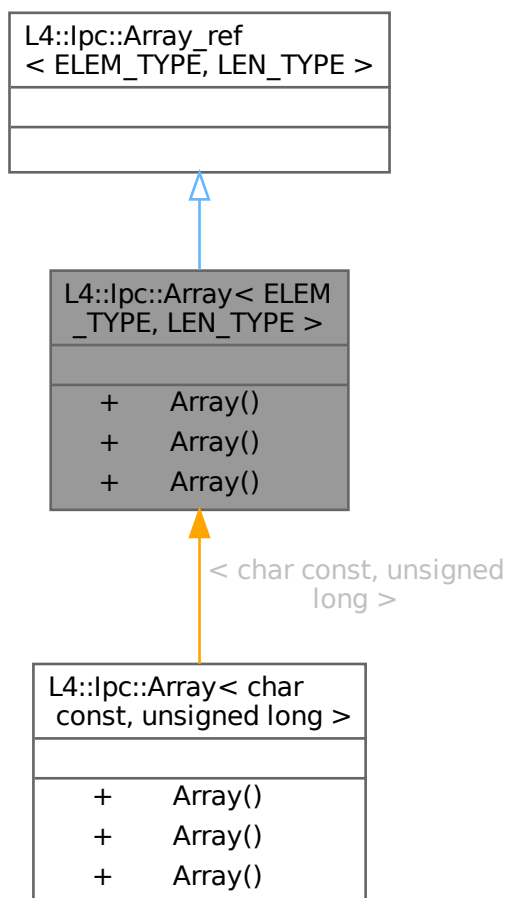
- l4/cxx/[basic\\_ostream](#)

## 15.112 L4::lpc::Array< ELEM\_TYPE, LEN\_TYPE > Struct Template Reference

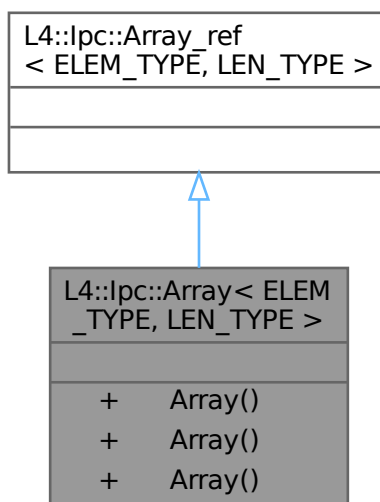
[Array](#) data type for dynamically sized arrays in RPCs.

```
#include <ipc_array>
```

Inheritance diagram for L4::lpc::Array< ELEM\_TYPE, LEN\_TYPE >:



Collaboration diagram for L4::lpc::Array< ELEM\_TYPE, LEN\_TYPE >:



## Public Member Functions

- **Array ()**  
*Make array.*
- **Array** (LEN\_TYPE length, ELEM\_TYPE \*data)  
*Make array from length and data pointer.*
- **Array** (typename Non\_const< ELEM\_TYPE >::type const &other)  
*Make a const array from a non-const array.*

### 15.112.1 Detailed Description

```
template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default>
struct L4::lpc::Array< ELEM_TYPE, LEN_TYPE >
```

[Array](#) data type for dynamically sized arrays in RPCs.

#### Template Parameters

<i>ELEM_TYPE</i>	The data type of an array element, should be 'const' when used as input.
<i>LEN_TYPE</i>	Data type used to store the number of elements in the array.

An [Array](#) generally encapsulates a data pointer and a length (number of elements). [Array](#) does *not* provide any storage for the data itself. The storage is either provided by a client-side caller or in the case of [Array\\_ref](#) is the message itself.

Arrays can be used as input or as output arguments, when used as input ELEM\_TYPE should be qualified *const*, when used as output a reference to an array must be used and the ELEM\_TYPE must *not* be qualified *const*. It is

the caller's responsibility to provide an array buffer of sufficient length. If a message from the server is too large it will be silently truncated.

If backward compatibility with `lpc::Stream` is required, then `LEN_TYPE` must be `unsigned long`.

Definition at line 92 of file [ipc\\_array](#).

The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_array`

### 15.113 L4::lpc::Array\_in\_buf< ELEM\_TYPE, LEN\_TYPE, MAX > Struct Template Reference

Server-side copy in buffer for [Array](#).

```
#include <ipc_array>
```

Collaboration diagram for `L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >`:

L4::lpc::Array_in_buf < ELEM_TYPE, LEN_TYPE, MAX >	
+	data
+	length
+	copy_in()
+	Array_in_buf()
+	Array_in_buf()

#### Public Member Functions

- void **copy\_in** (const\_array a)  
*copy in data from a source array*
- **Array\_in\_buf** (const\_array a)  
*Make [Array\\_in\\_buf](#) from a const array.*
- **Array\_in\_buf** (array a)  
*Make [Array\\_in\\_buf](#) from a non-const array.*

#### Data Fields

- `ELEM_TYPE` **data** [MAX]  
*The data elements.*
- `LEN_TYPE` **length**  
*The length of the array.*



### 15.113.1 Detailed Description

```
template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default, LEN_TYPE MAX = (L4_
UTCB_GENERIC_DATA_SIZE * sizeof(I4_umword_t)) / sizeof(ELEM_TYPE)>
struct L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >
```

Server-side copy in buffer for [Array](#).

#### Template Parameters

<i>ELEM_TYPE</i>	Data type of an array element.
<i>LEN_TYPE</i>	Data type for the number of elements in the array.
<i>MAX</i>	The maximum number of elements in the buffer. If the actual message is longer than the buffer, it will be silently truncated.

This type is assignment compatible to `Array_ref<ELEM_TYPE, LEN_TYPE>` and provides a transparent server-side copy-in mechanism for array parameters. The `Array_in_buf` provides the storage for the array data and receives a copy of the data passed to the server-function.

Definition at line 137 of file [ipc\\_array](#).

The documentation for this struct was generated from the following file:

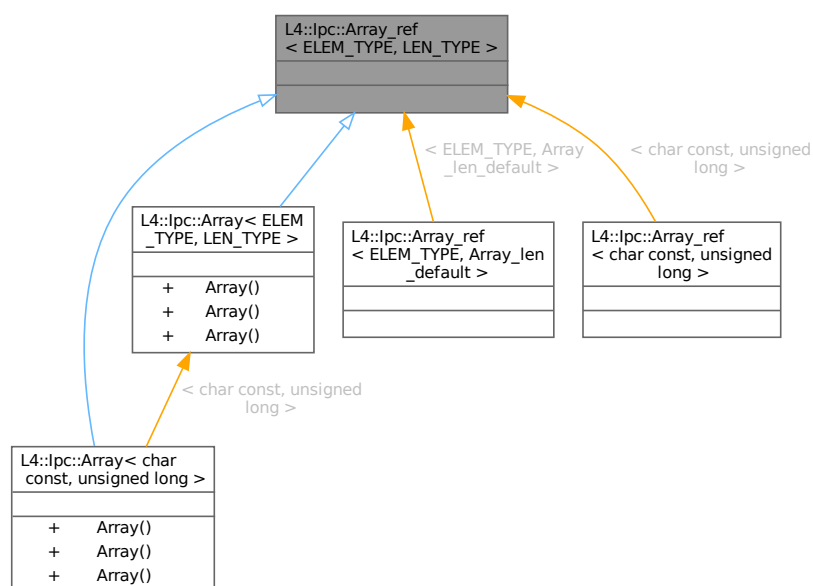
- `I4/sys/cxx/ipc_array`

## 15.114 L4::lpc::Array\_ref< ELEM\_TYPE, LEN\_TYPE > Struct Template Reference

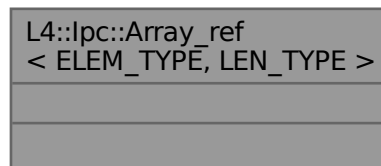
[Array](#) reference data type for arrays located in the message.

```
#include <ipc_array>
```

Inheritance diagram for `L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >`:



Collaboration diagram for L4::lpc::Array\_ref< ELEM\_TYPE, LEN\_TYPE >:



### 15.114.1 Detailed Description

```
template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default>
struct L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >
```

[Array](#) reference data type for arrays located in the message.

#### Note

Use [Array](#) for normal RPC interfaces, [Array\\_ref](#) is usually used as server-side argument, see [Array](#).

#### Template Parameters

<i>ELEM_TYPE</i>	The data type of an array element, should be 'const' when used as input.
<i>LEN_TYPE</i>	Data type used to store the number of elements in the array.

Definition at line 39 of file [ipc\\_array](#).

The documentation for this struct was generated from the following file:

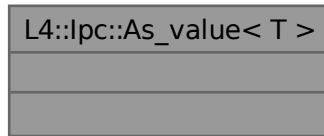
- l4/sys/cxx/ipc\_array

## 15.115 L4::lpc::As\_value< T > Struct Template Reference

Pass the argument as plain data value.

```
#include <ipc_types>
```

Collaboration diagram for L4::lpc::As\_value< T >:



### 15.115.1 Detailed Description

```
template<typename T>
struct L4::lpc::As_value< T >
```

Pass the argument as plain data value.

#### Template Parameters

<i>T</i>	The type of the original argument.
----------	------------------------------------

As\_value<T> is used when *T* would be otherwise interpreted specially, for example as flex page. When using As\_value<> then the argument is transmitted as plain data element.

Definition at line 127 of file [ipc\\_types](#).

The documentation for this struct was generated from the following file:

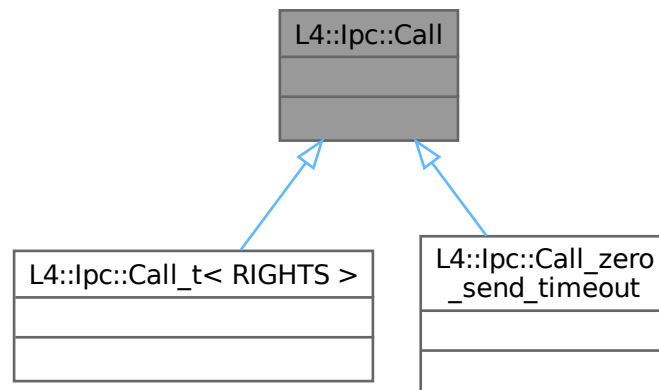
- [l4/sys/cxx/ipc\\_types](#)

## 15.116 L4::lpc::Call Struct Reference

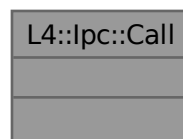
RPC attribute for a standard RPC call.

```
#include <ipc_iface>
```

Inheritance diagram for L4::Ipc::Call:



Collaboration diagram for L4::Ipc::Call:



### 15.116.1 Detailed Description

RPC attribute for a standard RPC call.

This is the default for the *FLAGS* parameter for L4::Ipc::Msg::Rpc\_call L4::Ipc::Msg::Rpc\_inline\_call templates and declares the RPC to have default call semantics and timeouts.

Examples:

```
L4_RPC(long, send, (unsigned value), L4::Ipc::Call);
```

which is equivalent to:

```
L4_RPC(long, send, (unsigned value));
```

Definition at line 226 of file [ipc\\_iface](#).

The documentation for this struct was generated from the following file:

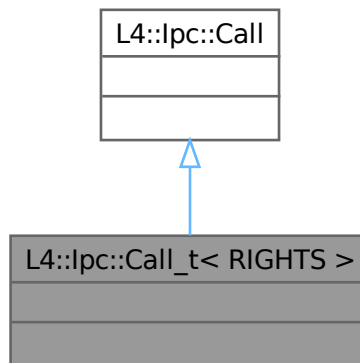
- [l4/sys/cxx/ipc\\_iface](#)

## 15.117 L4::lpc::Call\_t< RIGHTS > Struct Template Reference

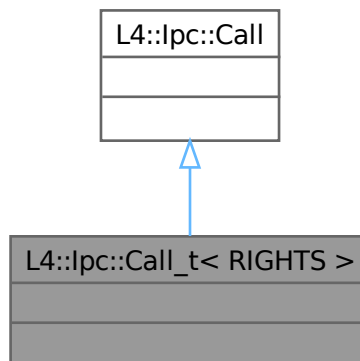
RPC attribute for an RPC call with required rights.

```
#include <ipc_iface>
```

Inheritance diagram for L4::lpc::Call\_t< RIGHTS >:



Collaboration diagram for L4::lpc::Call\_t< RIGHTS >:



### 15.117.1 Detailed Description

```
template<unsigned RIGHTS>  
struct L4::lpc::Call_t< RIGHTS >
```

RPC attribute for an RPC call with required rights.

## Template Parameters

<i>RIGHTS</i>	The capability rights required for this call. <a href="#">L4_CAP_FPAGE_W</a> and <a href="#">L4_CAP_FPAGE_S</a> are checked within the server (and <a href="#">-L4_EPERM</a> shall be returned if the caller has insufficient rights). <a href="#">L4_CAP_FPAGE_R</a> is always on but might be specified for documentation purposes. Other rights cannot be used in this context, because they cannot be checked at the server side.
---------------	---

## Examples:

```
L4_RPC(long, func, (unsigned value), L4::Ipc::Call_t<L4_CAP_FPAGE_RW>);
```

Definition at line 257 of file [ipc\\_iface](#).

The documentation for this struct was generated from the following file:

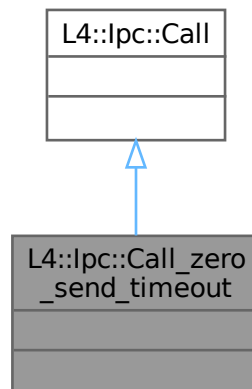
- [l4/sys/cxx/ipc\\_iface](#)

## 15.118 L4::lpc::Call\_zero\_send\_timeout Struct Reference

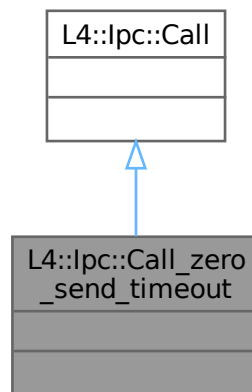
RPC attribute for an RPC call, with zero send timeout.

```
#include <ipc_iface>
```

Inheritance diagram for L4::lpc::Call\_zero\_send\_timeout:



Collaboration diagram for L4::lpc::Call\_zero\_send\_timeout:



### 15.118.1 Detailed Description

RPC attribute for an RPC call, with zero send timeout.

Definition at line 236 of file [ipc\\_iface](#).

The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc\\_iface](#)

## 15.119 L4::lpc::Cap< T > Class Template Reference

Capability type for RPC interfaces (see [L4::Cap<T>](#)).

```
#include <ipc_types>
```

Collaboration diagram for L4::lpc::Cap< T >:

L4::lpc::Cap< T >
<ul style="list-style-type: none"> <li>+ Cap()</li> <li>+ Cap()</li> <li>+ Cap()</li> <li>+ Cap()</li> <li>+ Cap()</li> <li>+ cap()</li> <li>+ rights()</li> <li>+ fpage()</li> <li>+ is_valid()</li> <li>+ from_ci()</li> </ul>

## Public Types

- enum { **Rights\_mask** = 0xff , **Cap\_mask** = L4\_CAP\_MASK }

## Public Member Functions

- template<typename O >  
**Cap** (**Cap**< O > const &o) noexcept  
*Make copy with conversion.*
- Cap** (L4::Cap< T > **cap**) noexcept  
*Make a **Cap** from L4::Cap< T >, with minimal rights.*
- template<typename O >  
**Cap** (L4::Cap< O > **cap**) noexcept  
*Make IPC **Cap** from L4::Cap with conversion (and minimal rights).*
- Cap** () noexcept  
*Make an invalid cap.*
- Cap** (L4::Cap< T > **cap**, unsigned char **rights**) noexcept  
*Make a **Cap** from L4::Cap< T > with the given rights.*
- L4::Cap< T > **cap** () const noexcept  
*Return the L4::Cap< T > of this **Cap**.*
- unsigned **rights** () const noexcept  
*Return the rights bits stored in this IPC cap.*
- L4::lpc::Snd\_fpage **fpage** () const noexcept  
*Return the send flexpage for this **Cap** (see [l4\\_fpage\\_t](#))*
- bool **is\_valid** () const noexcept  
*Return true if this **Cap** is valid.*



## Static Public Member Functions

- static [Cap from\\_ci](#) ([l4\\_cap\\_idx\\_t](#) c) noexcept  
*Create an IPC capability from a C capability index plus rights.*

## 15.119.1 Detailed Description

```
template<typename T>
class L4::lpc::Cap< T >
```

Capability type for RPC interfaces (see [L4 : :Cap<T>](#)).

### Template Parameters

<i>T</i>	type of the interface referenced by the capability.
----------	---

In contrast to [L4 : :Cap<T>](#) this type additionally stores a rights mask that shall be used when the capability is transferred to the receiver. This allows to apply restrictions to the transferred capability in the form of a subset of the rights possessed by the sender.

See also

[L4::lpc::make\\_cap\(\)](#)

Definition at line 562 of file [ipc\\_types](#).

## 15.119.2 Member Enumeration Documentation

### 15.119.2.1 anonymous enum

```
template<typename T >
anonymous enum
```

#### Enumerator

Rights_mask	Mask for rights bits stored internally. <a href="#">L4_FPAGE_RIGHTS_MASK</a>   <a href="#">L4_FPAGE_C_NO_REF_CNT</a>   <a href="#">L4_FPAGE_C_OBJ_RIGHTS</a> ).
Cap_mask	Mask for significant capability bits. (incl. the invalid bit to support invalid caps)

Definition at line 568 of file [ipc\\_types](#).

## 15.119.3 Constructor & Destructor Documentation

### 15.119.3.1 Cap()

```
template<typename T >
L4::Ipc::Cap< T >::Cap (
```

```
L4::Cap< T > cap,
unsigned char rights ) [inline], [noexcept]
```

Make a [Cap](#) from `L4::Cap<T>` with the given rights.

#### Parameters

<i>cap</i>	Capability to be sent.
<i>rights</i>	Rights to be sent. Consists of <a href="#">L4_fpage_rights</a> and <a href="#">L4_obj_fpage_ctl</a> .

Definition at line [614](#) of file [ipc\\_types](#).

## 15.119.4 Member Function Documentation

### 15.119.4.1 from\_ci()

```
template<typename T >
static Cap L4::Ipc::Cap< T >::from_ci (
    l4_cap_idx_t c ) [inline], [static], [noexcept]
```

Create an IPC capability from a C capability index plus rights.

#### Parameters

<i>c</i>	C capability index with the lowest 8 bits used as rights for the map operation (see <a href="#">L4_fpage_rights</a> ).
----------	--

Definition at line [622](#) of file [ipc\\_types](#).

References [L4::lpc::Cap< T >::Cap\\_mask](#), and [L4::lpc::Cap< T >::Rights\\_mask](#).

The documentation for this class was generated from the following file:

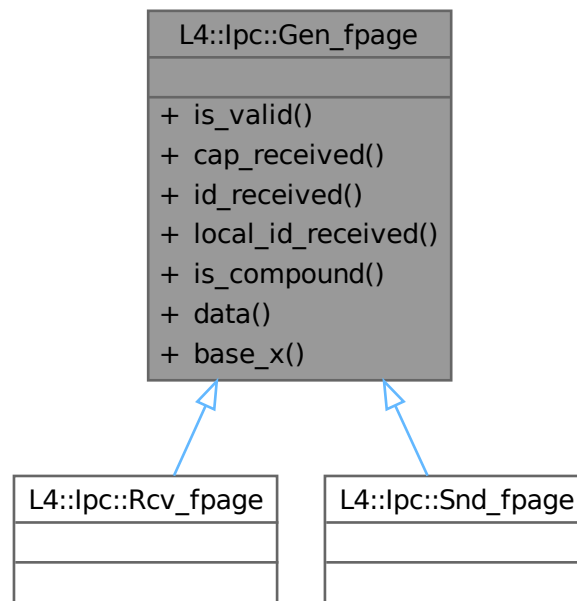
- [l4/sys/cxx/ipc\\_types](#)

## 15.120 L4::lpc::Gen\_fpage Class Reference

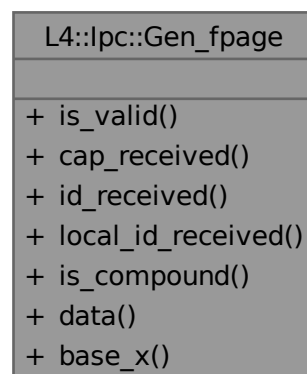
Generic RPC base for [L4](#) flex-pages.

```
#include <ipc_types>
```

Inheritance diagram for L4::lpc::Gen\_fpage:



Collaboration diagram for L4::lpc::Gen\_fpage:



## Public Types

- enum [Type](#)

*Type of mapping object, see L4\_fpage\_type.*

- enum [Map\\_type](#)  
*Kind of mapping.*
- enum [Cacheopt](#)  
*Caching options, see [l4\\_fpage\\_cacheability\\_opt\\_t](#).*

## Public Member Functions

- bool [is\\_valid](#) () const noexcept  
*Check if the capability is valid.*
- bool [cap\\_received](#) () const noexcept  
*Check if at least one capability has been mapped.*
- bool [id\\_received](#) () const noexcept  
*Check if a label was received instead of a mapping.*
- bool [local\\_id\\_received](#) () const noexcept  
*Check if a local capability id has been received.*
- bool [is\\_compound](#) () const noexcept  
*Check if the received item has the compound bit set.*
- [l4\\_umword\\_t](#) [data](#) () const noexcept  
*Return the raw flex page descriptor.*
- [l4\\_umword\\_t](#) [base\\_x](#) () const noexcept  
*Return the raw base descriptor.*

### 15.120.1 Detailed Description

Generic RPC base for [L4](#) flex-pages.

Definition at line [296](#) of file [ipc\\_types](#).

### 15.120.2 Member Function Documentation

#### 15.120.2.1 [cap\\_received\(\)](#)

```
bool L4::Rpc::Gen_fpage::cap_received ( ) const [inline], [noexcept]
```

Check if at least one capability has been mapped.

The capabilities themselves can then be retrieved from the cap slots that have been provided in the receive operation.

#### Note

If this function returns `true` and the receive window covers more than one capability slot, then it is not possible to determine which slots actually got capabilities mapped from the sender.

Definition at line [365](#) of file [ipc\\_types](#).

**15.120.2.2 id\_received()**

```
bool L4::Ipc::Gen_fpage::id_received ( ) const [inline], [noexcept]
```

Check if a label was received instead of a mapping.

For IPC gates, if the L4\_RCV\_ITEM\_LOCAL\_ID has been set, then only the label of the IPC gate will be provided if the gate is local to the receiver, i.e. the target thread of the IPC gate is in the same task as the receiving thread.

The label can be retrieved with [Gen\\_fpage::data\(\)](#).

Definition at line 377 of file [ipc\\_types](#).

**15.120.2.3 is\_compound()**

```
bool L4::Ipc::Gen_fpage::is_compound ( ) const [inline], [noexcept]
```

Check if the received item has the compound bit set.

A set compound bit means the next message item of the same type will be mapped to the same receive buffer as this message item.

Definition at line 395 of file [ipc\\_types](#).

**15.120.2.4 local\_id\_received()**

```
bool L4::Ipc::Gen_fpage::local_id_received ( ) const [inline], [noexcept]
```

Check if a local capability id has been received.

If the L4\_RCV\_ITEM\_LOCAL\_ID flag has been set by the receiver, and sender and receiver are in the same task, then only the capability index is transferred.

The capability can be retrieved with [Gen\\_fpage::data\(\)](#).

Definition at line 387 of file [ipc\\_types](#).

The documentation for this class was generated from the following file:

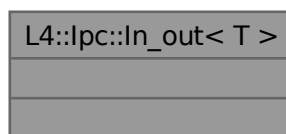
- [l4/sys/cxx/ipc\\_types](#)

**15.121 L4::lpc::ln\_out< T > Struct Template Reference**

Mark an argument as in-out argument.

```
#include <ipc_types>
```

Collaboration diagram for L4::lpc::ln\_out< T >:



### 15.121.1 Detailed Description

```
template<typename T>  
struct L4::lpc::In_out< T >
```

Mark an argument as in-out argument.

#### Template Parameters

<i>T</i>	The original argument type, usually a pointer or a reference.
----------	---

In\_out<> is used when an otherwise output-only value shall also be used as input value.

Definition at line 52 of file [ipc\\_types](#).

The documentation for this struct was generated from the following file:

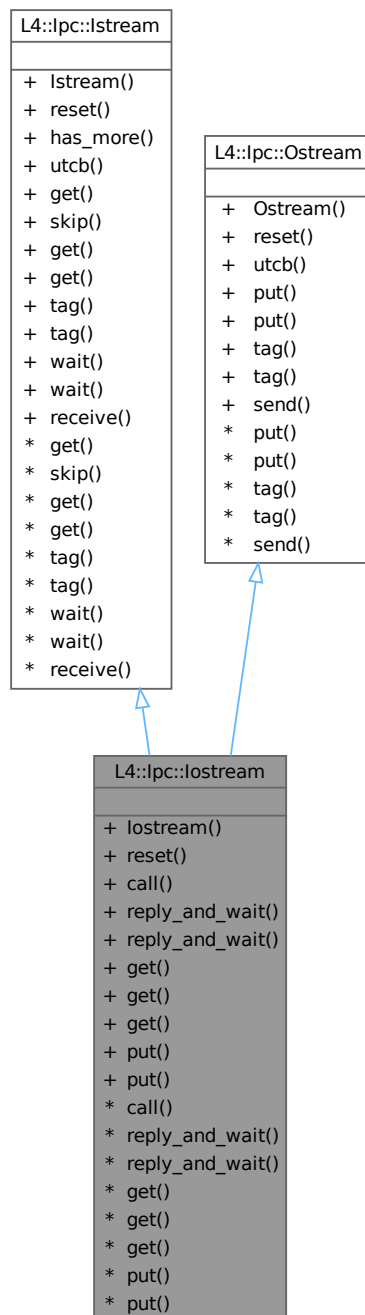
- [l4/sys/cxx/ipc\\_types](#)

## 15.122 L4::lpc::lostream Class Reference

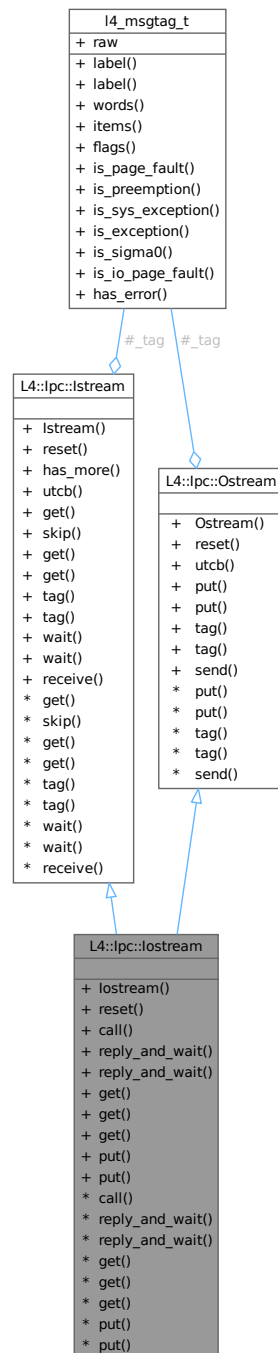
Input/Output stream for IPC [un]marshalling.

```
#include <ipc_stream>
```

Inheritance diagram for L4::lpc::lostream:



Collaboration diagram for L4::ipc::lostream:



## Public Member Functions

- `lostream` (`l4_utcb_t *utcb`)  
Create an IPC IO stream with a single message buffer.
- `void reset` ()  
Reset the stream to its initial state.



**IPC operations.**

- `l4_msgtag_t` call (`l4_cap_idx_t` dst, `l4_timeout_t` timeout, long proto=0)  
*Do an IPC call using the message in the output stream and receive the reply in the input stream.*
- `l4_msgtag_t` reply\_and\_wait (`l4_umword_t` \*src\_dst, long proto=0)  
*Do an IPC reply and wait.*
- `l4_msgtag_t` reply\_and\_wait (`l4_umword_t` \*src\_dst, `l4_timeout_t` timeout, long proto=0)  
*Do an IPC reply and wait.*

**Get/Put functions.**

These functions are basically used to implement the insertion operators (<<) and should not be called directly.

- template<typename T >  
unsigned long `get` (T \*buf, unsigned long elems)  
*Copy out an array of type T with size elements.*
- template<typename T >  
unsigned long `get` (Msg\_ptr< T > const &buf, unsigned long elems=1)  
*Read one size elements of type T from the stream and return a pointer.*
- template<typename T >  
bool `get` (T &v)  
*Extract a single element of type T from the stream.*
- template<typename T >  
bool `put` (T \*buf, unsigned long size)  
*Put an array with size elements of type T into the stream.*
- template<typename T >  
bool `put` (T const &v)  
*Insert an element of type T into the stream.*

**Public Member Functions inherited from L4::lpc::lstream**

- `lstream` (`l4_utcb_t` \*utcb)  
*Create an input stream for the given message buffer.*
- void `reset` ()  
*Reset the stream to empty, and ready for `receive()`/`wait()`.*
- template<typename T >  
bool `has_more` (unsigned long count=1)  
*Check whether a value of type T can be obtained from the stream.*
- `l4_utcb_t` \* `utcb` () const  
*Return utcb pointer.*
- template<typename T >  
unsigned long `get` (T \*buf, unsigned long elems)  
*Copy out an array of type T with size elements.*
- template<typename T >  
void `skip` (unsigned long elems)  
*Skip size elements of type T in the stream.*
- template<typename T >  
unsigned long `get` (Msg\_ptr< T > const &buf, unsigned long elems=1)  
*Read one size elements of type T from the stream and return a pointer.*
- template<typename T >  
bool `get` (T &v)  
*Extract a single element of type T from the stream.*
- `l4_msgtag_t` tag () const

- `l4_msgtag_t & tag ()`  
Get the message tag of a received IPC.
- `l4_msgtag_t wait (l4_umword_t *src)`  
Wait for an incoming message from any sender.
- `l4_msgtag_t wait (l4_umword_t *src, l4_timeout_t timeout)`  
Wait for an incoming message from any sender.
- `l4_msgtag_t receive (l4_cap_idx_t src)`  
Wait for a message from the specified sender.

## Public Member Functions inherited from `L4::lpc::Ostream`

- **Ostream** (`l4_utcb_t *utcb`)  
Create an IPC output stream using the given message buffer `utcb`.
- void **reset** ()  
Reset the stream to empty, same state as a newly created stream.
- `l4_utcb_t * utcb () const`  
Return `utcb` pointer.
- `template<typename T >`  
`bool put (T *buf, unsigned long size)`  
Put an array with `size` elements of type `T` into the stream.
- `template<typename T >`  
`bool put (T const &v)`  
Insert an element of type `T` into the stream.
- `l4_msgtag_t tag () const`  
Extract the `L4` message tag from the stream.
- `l4_msgtag_t & tag ()`  
Extract a reference to the `L4` message tag from the stream.
- `l4_msgtag_t send (l4_cap_idx_t dst, long proto=0, unsigned flags=0)`  
Send the message via IPC to the given receiver.

### 15.122.1 Detailed Description

Input/Output stream for IPC [un]marshalling.

The `lpc::lostream` is part of the AW Env IPC framework as well as `lpc::lstream` and `lpc::Ostream`. In particular an `lpc::lostream` is a combination of an `lpc::lstream` and an `lpc::Ostream`. It can use either a single message buffer for receiving and sending messages or a pair of a receive and a send buffer. The stream also supports combined IPC operations such as `call()` and `reply_and_wait()`, which can be used to implement RPC functionality.

#### Examples

`examples/libs/l4re/c++/shared_ds/ds_srv.cc`, `examples/libs/l4re/streammap/client.cc`, and `examples/libs/l4re/streammap/server.cc`.

Definition at line 800 of file `ipc_stream`.

### 15.122.2 Constructor & Destructor Documentation

#### 15.122.2.1 `lostream()`

```
L4::Ipc::lostream::lostream (
    l4_utcb_t * utcb ) [inline], [explicit]
```

Create an IPC IO stream with a single message buffer.

## Parameters

<i>utcb</i>	The message buffer used as backing store.
-------------	---

The created IO stream uses the same message buffer for sending and receiving IPC messages.

Definition at line 812 of file [ipc\\_stream](#).

### 15.122.3 Member Function Documentation

#### 15.122.3.1 call()

```
l4_msgtag_t L4::Ipc::Iostream::call (
    l4_cap_idx_t dst,
    l4_timeout_t timeout,
    long proto = 0 ) [inline]
```

Do an IPC call using the message in the output stream and receive the reply in the input stream.

## Parameters

<i>dst</i>	The destination to call.
<i>timeout</i>	The IPC timeout for the call.
<i>proto</i>	The protocol value to use in the message tag.

## Returns

The result tag of the IPC operation.

This is a combined IPC operation consisting of a send and a receive to/from the given destination *dst*.

A call is usually used by clients for RPCs to a server.

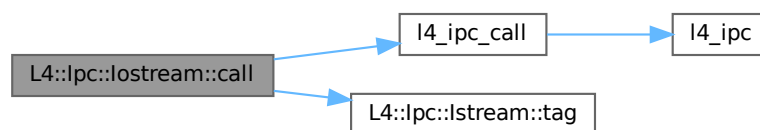
## Examples

[examples/libs/l4re/streammap/client.cc](#).

Definition at line 977 of file [ipc\\_stream](#).

References [l4\\_ipc\\_call\(\)](#), and [L4::lpc::Iostream::tag\(\)](#).

Here is the call graph for this function:



**15.122.3.2** `get()` [1/3]

```
template<typename T >
unsigned long L4::Ipc::Istream::get (
    Msg_ptr< T > const & buf,
    unsigned long elems = 1 ) [inline]
```

Read one size elements of type T from the stream and return a pointer.

**Parameters**

<i>buf</i>	A <a href="#">Msg_ptr</a> that is actually set to point to the element in the stream.
<i>elems</i>	Number of elements to extract (default is 1).

**Returns**

The number of elements extracted.

In contrast to a normal `get`, this version does actually not copy the data but returns a pointer to the data.

See [operator>>\(\)](#)

Definition at line 450 of file [ipc\\_stream](#).

**15.122.3.3** `get()` [2/3]

```
template<typename T >
bool L4::Ipc::Istream::get (
    T & v ) [inline]
```

Extract a single element of type T from the stream.

**Parameters**

<i>out</i>	<i>v</i>	The element.
------------	----------	--------------

**Return values**

<i>true</i>	An element was successfully extracted.
<i>false</i>	An element could not be extracted.

See [operator>>\(\)](#)

Definition at line 475 of file [ipc\\_stream](#).

**15.122.3.4** `get()` [3/3]

```
template<typename T >
unsigned long L4::Ipc::Istream::get (
```

```
T * buf,
unsigned long elems ) [inline]
```

Copy out an array of type `T` with `size` elements.

#### Parameters

<i>buf</i>	Pointer to a buffer for <code>size</code> elements of type <code>T</code> .
<i>elems</i>	Number of elements of type <code>T</code> to copy out.

#### Returns

The number of elements copied out.

See [operator>>\(\)](#)

Definition at line 405 of file [ipc\\_stream](#).

### 15.122.3.5 put() [1/2]

```
template<typename T >
bool L4::Ipc::Ostream::put (
    T * buf,
    unsigned long size ) [inline]
```

Put an array with `size` elements of type `T` into the stream.

#### Parameters

<i>buf</i>	A pointer to the array to insert into the buffer.
<i>size</i>	The number of elements in the array.

Definition at line 671 of file [ipc\\_stream](#).

### 15.122.3.6 put() [2/2]

```
template<typename T >
bool L4::Ipc::Ostream::put (
    T const & v ) [inline]
```

Insert an element of type `T` into the stream.

#### Parameters

<i>v</i>	The element to insert.
----------	------------------------

Definition at line 689 of file [ipc\\_stream](#).

### 15.122.3.7 `reply_and_wait()` [1/2]

```
l4_msgtag_t L4::Ipc::Iostream::reply_and_wait (
    l4_umword_t * src_dst,
    l4_timeout_t timeout,
    long proto = 0 ) [inline]
```

Do an IPC reply and wait.

#### Parameters

<code>in, out</code>	<code>src_dst</code>	Input: the destination for the send operation. Output: the source of the received message.
	<code>timeout</code>	Timeout used for IPC.
	<code>proto</code>	Protocol to use.

#### Returns

The result tag of the IPC operation.

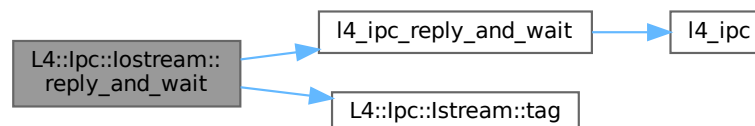
This is a combined IPC operation consisting of a send operation and an open wait for any message.

A reply and wait is usually used by servers that reply to a client and wait for the next request by any other client.

Definition at line 992 of file [ipc\\_stream](#).

References [l4\\_ipc\\_reply\\_and\\_wait\(\)](#), and [L4::Ipc::Iostream::tag\(\)](#).

Here is the call graph for this function:



### 15.122.3.8 `reply_and_wait()` [2/2]

```
l4_msgtag_t L4::Ipc::Iostream::reply_and_wait (
    l4_umword_t * src_dst,
    long proto = 0 ) [inline]
```

Do an IPC reply and wait.

## Parameters

<code>in, out</code>	<code>src_dst</code>	Input: the destination for the send operation. Output: the source of the received message.
	<code>proto</code>	Protocol to use.

## Returns

The result tag of the IPC operation.

This is a combined IPC operation consisting of a send operation and an open wait for any message.

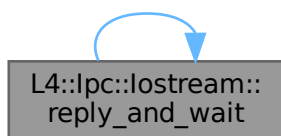
A reply and wait is usually used by servers that reply to a client and wait for the next request by any other client.

Definition at line 885 of file [ipc\\_stream](#).

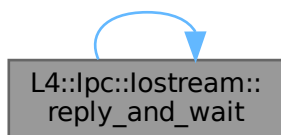
References [L4\\_IPC\\_SEND\\_TIMEOUT\\_0](#), and [reply\\_and\\_wait\(\)](#).

Referenced by [reply\\_and\\_wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.122.3.9 reset()

```
void L4::Ipc::Iostream::reset ( ) [inline]
```

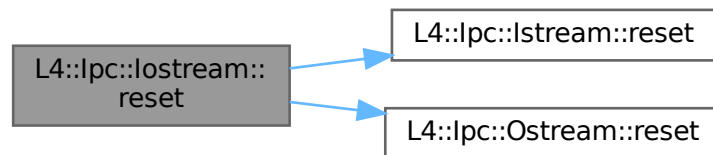
Reset the stream to its initial state.

Input as well as the output stream are reset.

Definition at line 826 of file [ipc\\_stream](#).

References [L4::Ipc::Istream::reset\(\)](#), and [L4::Ipc::Ostream::reset\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- [l4/cxx/ipc\\_stream](#)

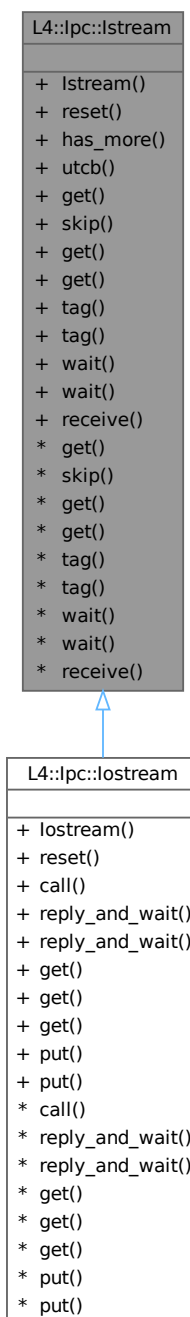
## 15.123 L4::Ipc::Istream Class Reference

Input stream for IPC unmarshalling.

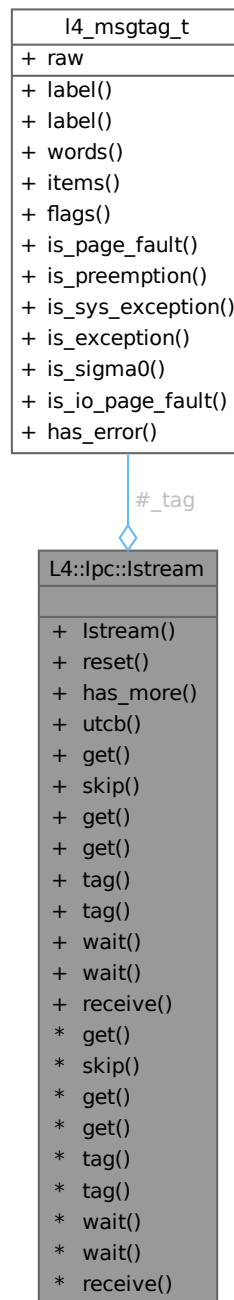
```
#include <ipc_stream>
```



Inheritance diagram for L4::lpc::Istream:



Collaboration diagram for L4::lpc::Istream:



## Public Member Functions

- [Istream](#) ([l4\\_utcb\\_t](#) \*utcb)  
*Create an input stream for the given message buffer.*
- void [reset](#) ()  
*Reset the stream to empty, and ready for [receive\(\)](#)/[wait\(\)](#).*

- `template<typename T >`  
`bool has_more (unsigned long count=1)`  
*Check whether a value of type T can be obtained from the stream.*
- `l4_utcb_t * utcb () const`  
*Return utcb pointer.*

#### Get/Put Functions.

- `template<typename T >`  
`unsigned long get (T *buf, unsigned long elems)`  
*Copy out an array of type T with size elements.*
- `template<typename T >`  
`void skip (unsigned long elems)`  
*Skip size elements of type T in the stream.*
- `template<typename T >`  
`unsigned long get (Msg_ptr< T > const &buf, unsigned long elems=1)`  
*Read one size elements of type T from the stream and return a pointer.*
- `template<typename T >`  
`bool get (T &v)`  
*Extract a single element of type T from the stream.*
- `l4_msgtag_t tag () const`  
*Get the message tag of a received IPC.*
- `l4_msgtag_t & tag ()`  
*Get the message tag of a received IPC.*

#### IPC operations.

- `l4_msgtag_t wait (l4_umword_t *src)`  
*Wait for an incoming message from any sender.*
- `l4_msgtag_t wait (l4_umword_t *src, l4_timeout_t timeout)`  
*Wait for an incoming message from any sender.*
- `l4_msgtag_t receive (l4_cap_idx_t src)`  
*Wait for a message from the specified sender.*

### 15.123.1 Detailed Description

Input stream for IPC unmarshalling.

[lpc::Istream](#) is part of the dynamic IPC marshalling infrastructure, as well as [lpc::Ostream](#) and [lpc::Iostream](#).

[lpc::Istream](#) is an input stream supporting extraction of values from an IPC message buffer. A received IPC message can be unmarshalled using the usual extraction operator (>>).

There exist some special wrapper classes to extract arrays (see `lpc_buf_cp_in` and `lpc_buf_in`) and indirect strings (see `Msg_in_buffer` and `Msg_io_buffer`).

Definition at line 345 of file [ipc\\_stream](#).

### 15.123.2 Constructor & Destructor Documentation

#### 15.123.2.1 Istream()

```
L4::Ipc::Istream::Istream (
    l4_utcb_t * utcb ) [inline]
```

Create an input stream for the given message buffer.

The given message buffer is used for IPC operations [wait\(\)](#)/[receive\(\)](#) and received data can be extracted using the >> operator afterwards. In the case of indirect message parts a buffer of type `Msg_in_buffer` must be inserted into the stream before the IPC operation and contains received data afterwards.

## Parameters

<i>utcb</i>	The message buffer to receive IPC messages.
-------------	---

Definition at line 359 of file [ipc\\_stream](#).

### 15.123.3 Member Function Documentation

#### 15.123.3.1 `get()` [1/3]

```
template<typename T >
unsigned long L4::Ipc::Istream::get (
    Msg\_ptr< T > const & buf,
    unsigned long elems = 1 ) [inline]
```

Read one size elements of type T from the stream and return a pointer.

## Parameters

<i>buf</i>	A <a href="#">Msg_ptr</a> that is actually set to point to the element in the stream.
<i>elems</i>	Number of elements to extract (default is 1).

## Returns

The number of elements extracted.

In contrast to a normal `get`, this version does actually not copy the data but returns a pointer to the data.

See [operator>>\(\)](#)

Definition at line 450 of file [ipc\\_stream](#).

References [L4\\_UNLIKELY](#).

#### 15.123.3.2 `get()` [2/3]

```
template<typename T >
bool L4::Ipc::Istream::get (
    T & v ) [inline]
```

Extract a single element of type T from the stream.

## Parameters

<i>out</i>	<i>v</i>	The element.
------------	----------	--------------

## Return values

<i>true</i>	An element was successfully extracted.
-------------	--

## Return values

<i>false</i>	An element could not be extracted.
--------------	------------------------------------

See [operator>>\(\)](#)

Definition at line 475 of file [ipc\\_stream](#).

References [L4\\_UNLIKELY](#).

**15.123.3.3 get() [3/3]**

```
template<typename T >
unsigned long L4::Ipc::Istream::get (
    T * buf,
    unsigned long elems ) [inline]
```

Copy out an array of type T with `size` elements.

## Parameters

<i>buf</i>	Pointer to a buffer for size elements of type T.
<i>elems</i>	Number of elements of type T to copy out.

## Returns

The number of elements copied out.

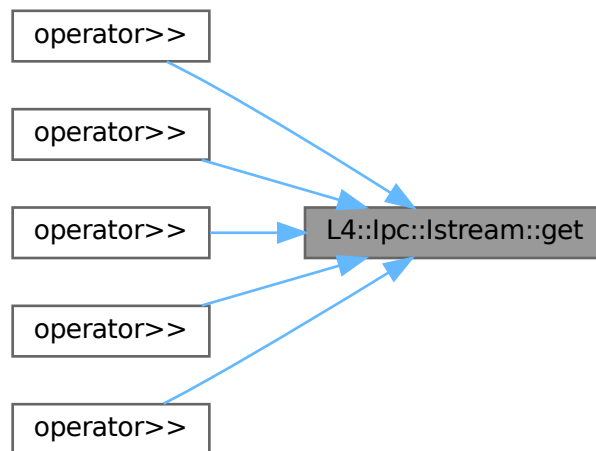
See [operator>>\(\)](#)

Definition at line 405 of file [ipc\\_stream](#).

References [L4\\_UNLIKELY](#).

Referenced by [operator>>\(\)](#), [operator>>\(\)](#), [operator>>\(\)](#), [operator>>\(\)](#), and [operator>>\(\)](#).

Here is the caller graph for this function:



#### 15.123.3.4 receive()

```
l4_msgtag_t L4::Ipc::Istream::receive (
    l4_cap_idx_t src ) [inline]
```

Wait for a message from the specified sender.

##### Parameters

<code>src</code>	The sender id to receive from.
------------------	--------------------------------

##### Returns

The IPC result tag ([l4\\_msgtag\\_t](#)).

This is commonly known as 'closed wait'.

Definition at line 583 of file [ipc\\_stream](#).

References [L4\\_IPC\\_NEVER](#), and [receive\(\)](#).

Referenced by [receive\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.123.3.5 reset()

```
void L4::Ipc::Istream::reset ( ) [inline]
```

Reset the stream to empty, and ready for [receive\(\)/wait\(\)](#).

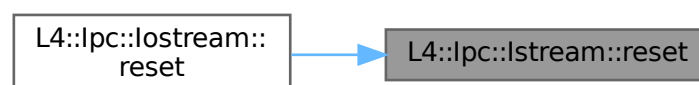
The stream is reset to the same state as on its creation.

Definition at line [369](#) of file [ipc\\_stream](#).

References [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [L4::lpc::lostream::reset\(\)](#).

Here is the caller graph for this function:



### 15.123.3.6 skip()

```
template<typename T >
void L4::Ipc::Istream::skip (
    unsigned long elems ) [inline]
```

Skip size elements of type T in the stream.

#### Parameters

<i>elems</i>	Number of elements to skip.
--------------	-----------------------------

Definition at line 425 of file [ipc\\_stream](#).

References [L4\\_UNLIKELY](#).

### 15.123.3.7 tag() [1/2]

```
l4\_msgtag\_t & L4::Ipc::Istream::tag ( ) [inline]
```

Get the message tag of a received IPC.

#### Returns

A reference to the [L4](#) message tag for the received IPC.

This is in particular useful for handling page faults or exceptions.

See [operator>>\(\)](#)

Definition at line 528 of file [ipc\\_stream](#).

### 15.123.3.8 tag() [2/2]

```
l4\_msgtag\_t L4::Ipc::Istream::tag ( ) const [inline]
```

Get the message tag of a received IPC.



**Returns**

The [L4](#) message tag for the received IPC.

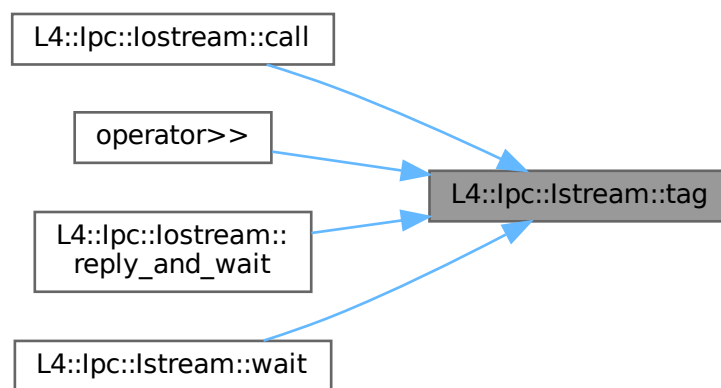
This is in particular useful for handling page faults or exceptions.

See [operator>>\(\)](#)

Definition at line [516](#) of file [ipc\\_stream](#).

Referenced by [L4::lpc::lostream::call\(\)](#), [operator>>\(\)](#), [L4::lpc::lostream::reply\\_and\\_wait\(\)](#), and [wait\(\)](#).

Here is the caller graph for this function:

**15.123.3.9 wait() [1/2]**

```
l4_msgtag_t L4::Ipc::Istream::wait (
    l4_umword_t * src ) [inline]
```

Wait for an incoming message from any sender.

**Parameters**

out	src	Contains the sender after a successful IPC operation.
-----	-----	---

**Returns**

Syscall return tag.

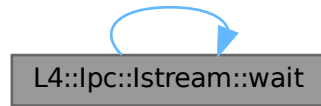
This wait is actually known as 'open wait'.

Definition at line [559](#) of file [ipc\\_stream](#).

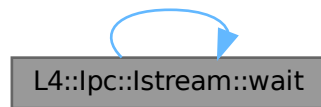
References [L4\\_IPC\\_NEVER](#), and [wait\(\)](#).

Referenced by [wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 15.123.3.10 wait() [2/2]

```
l4_msgtag_t L4::Ipc::Istream::wait (
    l4_umword_t * src,
    l4_timeout_t timeout ) [inline]
```

Wait for an incoming message from any sender.

##### Parameters

out	src	Contains the sender after a successful IPC operation.
	timeout	Timeout used for IPC.

##### Returns

The IPC result tag ([l4\\_msgtag\\_t](#)).

This wait is actually known as 'open wait'.

Definition at line [1024](#) of file [ipc\\_stream](#).

References [l4\\_ipc\\_wait\(\)](#), and [tag\(\)](#).

Here is the call graph for this function:



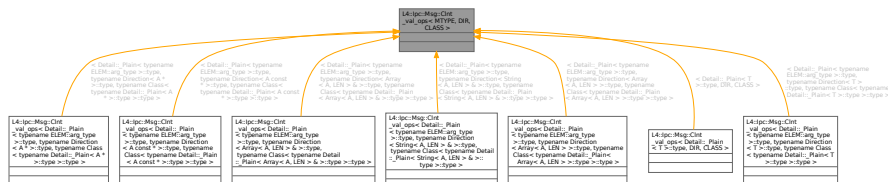
The documentation for this class was generated from the following file:

- [l4/cxx/ipc\\_stream](#)

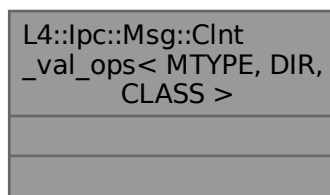
## 15.124 L4::ipc::Msg::Cnt\_val\_ops< MTYPE, DIR, CLASS > Struct Template Reference

Defines client-side handling of 'MTYPE' as RPC argument.

Inheritance diagram for L4::ipc::Msg::Cnt\_val\_ops< MTYPE, DIR, CLASS >:



Collaboration diagram for L4::ipc::Msg::Cnt\_val\_ops< MTYPE, DIR, CLASS >:



### 15.124.1 Detailed Description

```
template<typename MTYPE, typename DIR, typename CLASS>
struct L4::lpc::Msg::CInt_val_ops< MTYPE, DIR, CLASS >
```

Defines client-side handling of 'MTYPE' as RPC argument.

Template Parameters

<i>MTYPE</i>	Elem<T>::arg_type (where T is the type used in the RPC definition)
<i>DIR</i>	<a href="#">Dir_in</a> (client -> server), or <a href="#">Dir_out</a> (server -> client)
<i>CLASS</i>	<a href="#">Cls_data</a> , <a href="#">Cls_item</a> , or <a href="#">Cls_buffer</a>

Definition at line 221 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

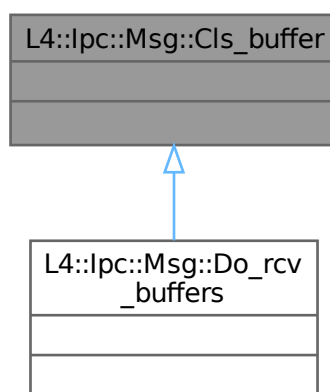
- `l4/sys/cxx/ipc_basics`

### 15.125 L4::lpc::Msg::Cls\_buffer Struct Reference

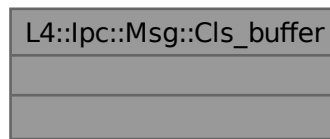
Marker type for receive buffer values.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Cls\_buffer:



Collaboration diagram for L4::lpc::Msg::Cls\_buffer:



### 15.125.1 Detailed Description

Marker type for receive buffer values.

Definition at line 165 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

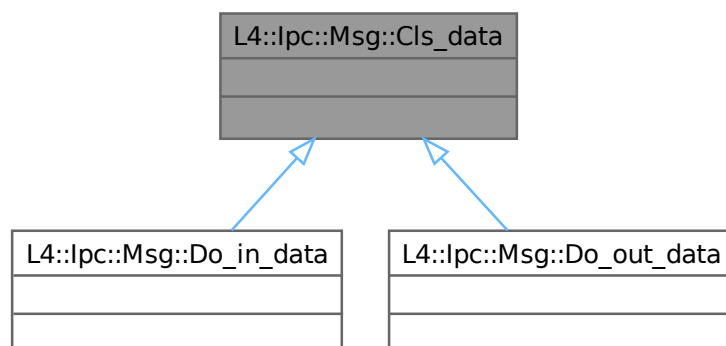
- `l4/sys/cxx/ipc_basics`

## 15.126 L4::lpc::Msg::Cls\_data Struct Reference

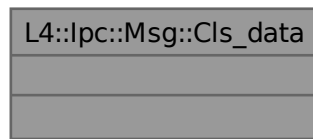
Marker type for data values.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Cls\_data:



Collaboration diagram for L4::lpc::Msg::Cls\_data:



### 15.126.1 Detailed Description

Marker type for data values.

Definition at line 161 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

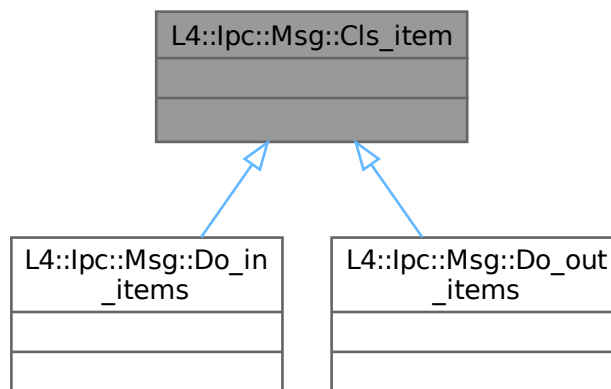
- l4/sys/cxx/ipc\_basics

## 15.127 L4::lpc::Msg::Cls\_item Struct Reference

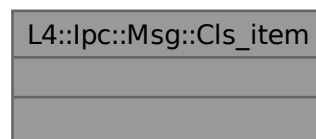
Marker type for item values.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Cls\_item:



Collaboration diagram for L4::lpc::Msg::Cls\_item:



### 15.127.1 Detailed Description

Marker type for item values.

Definition at line 163 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

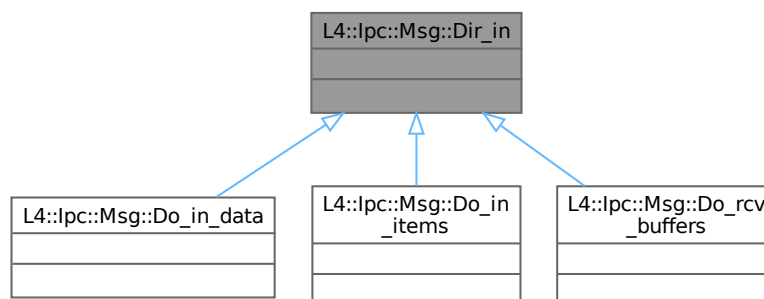
- `l4/sys/cxx/ipc_basics`

## 15.128 L4::lpc::Msg::Dir\_in Struct Reference

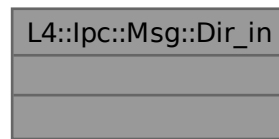
Marker type for input values.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Dir\_in:



Collaboration diagram for L4::lpc::Msg::Dir\_in:



### 15.128.1 Detailed Description

Marker type for input values.

Definition at line 156 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

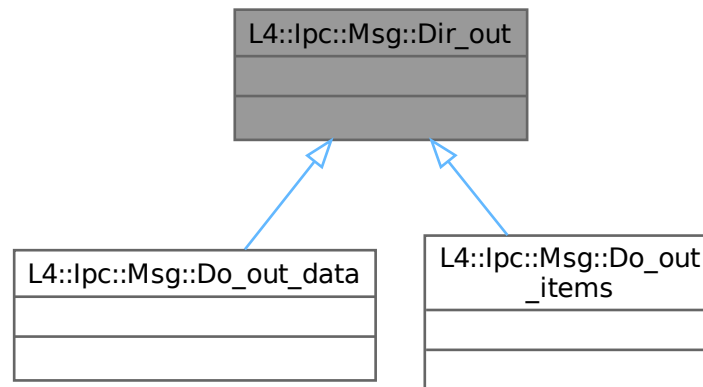
- [l4/sys/cxx/ipc\\_basics](#)

## 15.129 L4::lpc::Msg::Dir\_out Struct Reference

Marker type for output values.

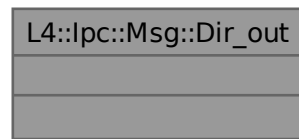
```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Dir\_out:





Collaboration diagram for L4::lpc::Msg::Dir\_out:



### 15.129.1 Detailed Description

Marker type for output values.

Definition at line 158 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

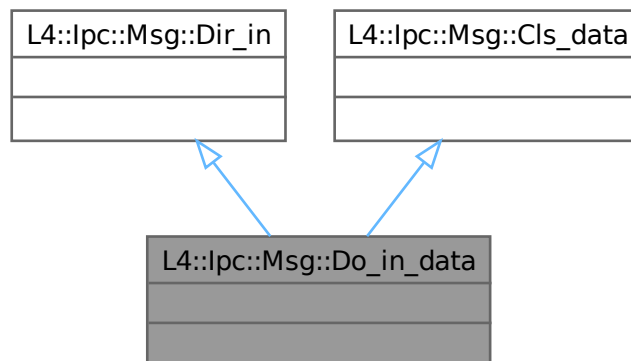
- l4/sys/cxx/ipc\_basics

## 15.130 L4::lpc::Msg::Do\_in\_data Struct Reference

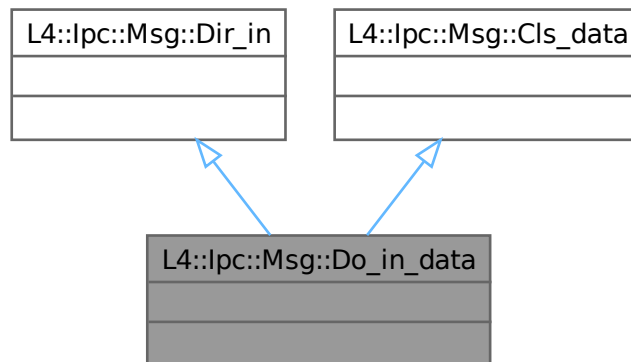
Marker for Input data.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do\_in\_data:



Collaboration diagram for L4::lpc::Msg::Do\_in\_data:



### 15.130.1 Detailed Description

Marker for Input data.

Definition at line 169 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

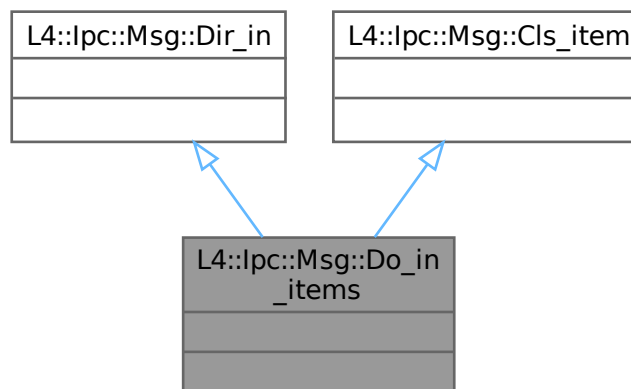
- `l4/sys/cxx/ipc_basics`

## 15.131 L4::lpc::Msg::Do\_in\_items Struct Reference

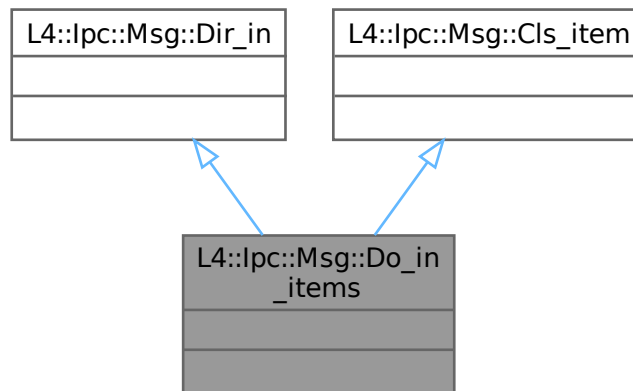
Marker for Input items.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do\_in\_items:



Collaboration diagram for L4::lpc::Msg::Do\_in\_items:



### 15.131.1 Detailed Description

Marker for Input items.

Definition at line 173 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

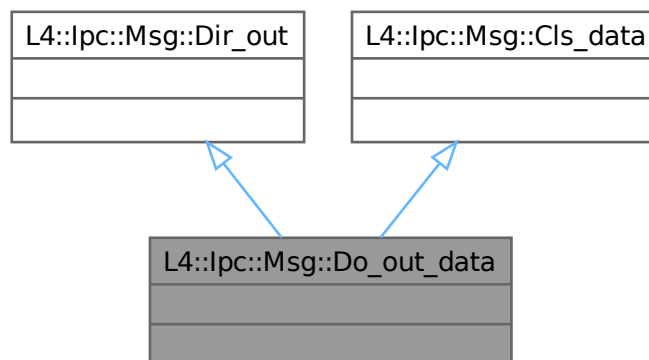
- `l4/sys/cxx/ipc_basics`

## 15.132 L4::lpc::Msg::Do\_out\_data Struct Reference

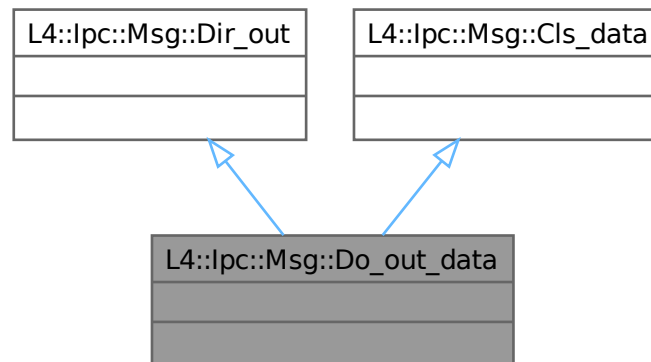
Marker for Output data.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do\_out\_data:



Collaboration diagram for L4::lpc::Msg::Do\_out\_data:



### 15.132.1 Detailed Description

Marker for Output data.

Definition at line 171 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

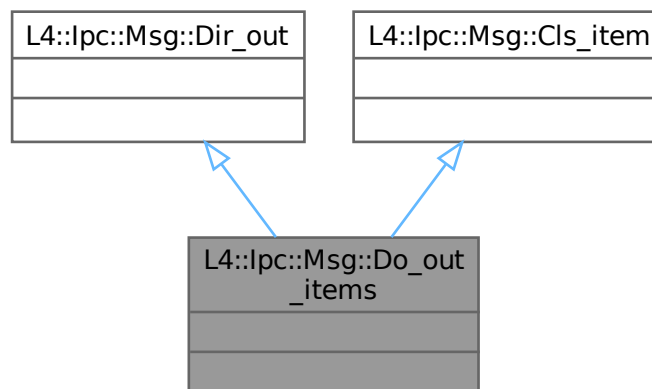
- `l4/sys/cxx/ipc_basics`

## 15.133 L4::lpc::Msg::Do\_out\_items Struct Reference

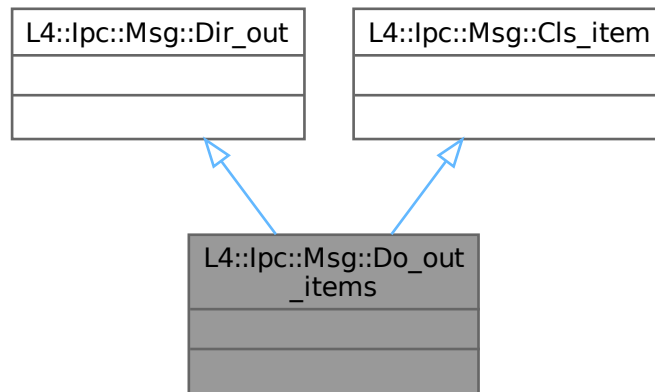
Marker for Output items.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do\_out\_items:



Collaboration diagram for L4::lpc::Msg::Do\_out\_items:



### 15.133.1 Detailed Description

Marker for Output items.

Definition at line 175 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

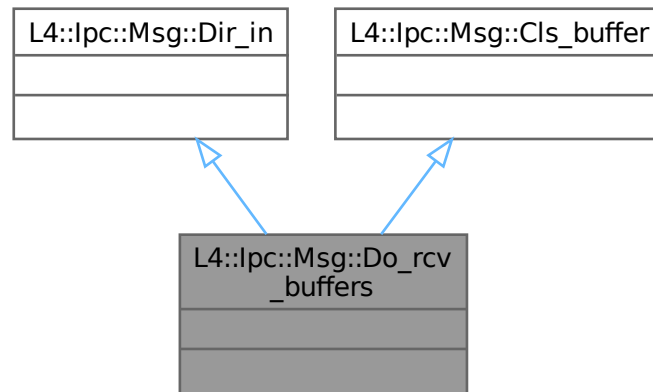
- `l4/sys/cxx/ipc_basics`

## 15.134 L4::lpc::Msg::Do\_rcv\_buffers Struct Reference

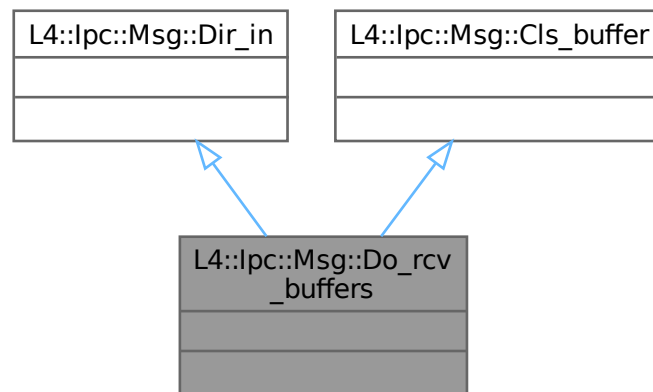
Marker for receive buffers.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do\_rcv\_buffers:



Collaboration diagram for L4::lpc::Msg::Do\_rcv\_buffers:



### 15.134.1 Detailed Description

Marker for receive buffers.

Definition at line 177 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

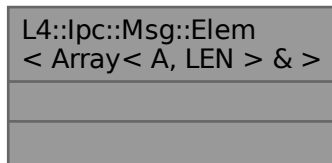
- `l4/sys/cxx/ipc_basics`

## 15.135 L4::lpc::Msg::Elem< Array< A, LEN > & > Struct Template Reference

[Array](#) as output argument.

```
#include <ipc_array>
```

Collaboration diagram for L4::lpc::Msg::Elem< Array< A, LEN > & >:



### Public Types

- typedef [Array](#)< A, LEN > & **arg\_type**  
*Array<> & at the interface.*
- typedef [Array\\_ref](#)< A, LEN > **svr\_type**  
*Array\_ref<> as server storage type.*
- typedef [svr\\_type](#) & **svr\_arg\_type**  
*Array\_ref<> & at the server side.*

### 15.135.1 Detailed Description

```
template<typename A, typename LEN>
struct L4::lpc::Msg::Elem< Array< A, LEN > & >
```

[Array](#) as output argument.

Definition at line [181](#) of file [ipc\\_array](#).

The documentation for this struct was generated from the following file:

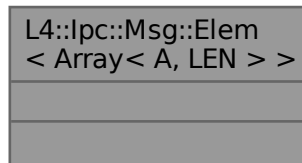
- `l4/sys/cxx/ipc_array`

## 15.136 L4::lpc::Msg::Elem< Array< A, LEN > > Struct Template Reference

[Array](#) as input arguments.

```
#include <ipc_array>
```

Collaboration diagram for L4::lpc::Msg::Elem< Array< A, LEN > >:



### Public Types

- typedef [Array](#)< A, LEN > **arg\_type**  
*Array<> as argument at the interface.*
- typedef [Array\\_ref](#)< A, LEN > **svr\_type**  
*Array\_ref<> at the server side.*

### 15.136.1 Detailed Description

```
template<typename A, typename LEN>
struct L4::lpc::Msg::Elem< Array< A, LEN > >
```

[Array](#) as input arguments.

Definition at line 169 of file [ipc\\_array](#).

The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_array`

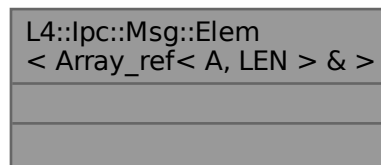


## 15.137 L4::lpc::Msg::Elem< Array\_ref< A, LEN > & > Struct Template Reference

[Array\\_ref](#) as output argument.

```
#include <ipc_array>
```

Collaboration diagram for L4::lpc::Msg::Elem< Array\_ref< A, LEN > & >:



### Public Types

- typedef [Array\\_ref](#)< A, LEN > & **arg\_type**  
*Array\_ref<> at the interface.*
- typedef [Array\\_ref](#)< typename L4::Types::Remove\_const< A >::type, LEN > **svr\_type**  
*Array\_ref<> as server storage.*
- typedef [svr\\_type](#) & **svr\_arg\_type**  
*Array\_ref<> & as server argument.*

### 15.137.1 Detailed Description

```
template<typename A, typename LEN>
struct L4::lpc::Msg::Elem< Array_ref< A, LEN > & >
```

[Array\\_ref](#) as output argument.

Definition at line 194 of file [ipc\\_array](#).

The documentation for this struct was generated from the following file:

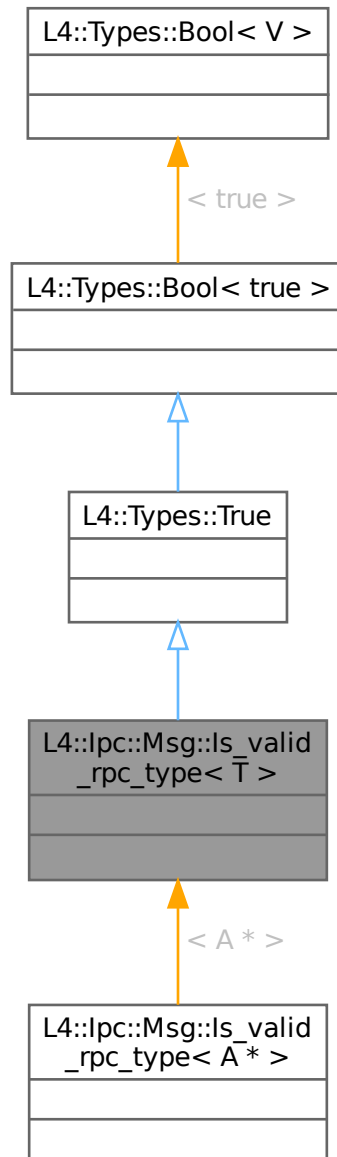
- l4/sys/cxx/ipc\_array

## 15.138 L4::lpc::Msg::ls\_valid\_rpc\_type< T > Struct Template Reference

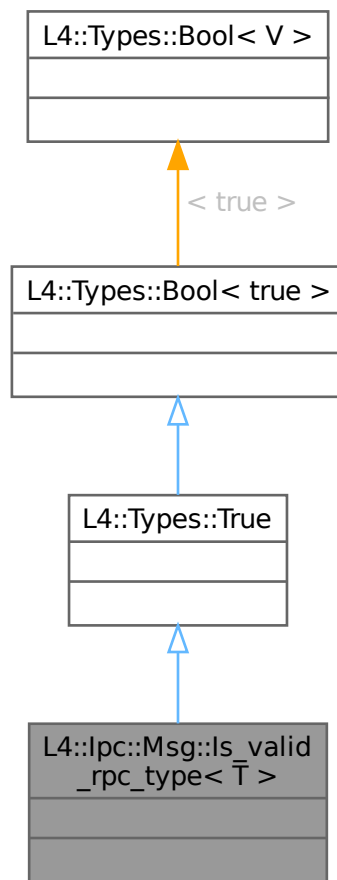
Type trait defining a valid RPC parameter type.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::ls\_valid\_rpc\_type< T >:



Collaboration diagram for L4::lpc::Msg::ls\_valid\_rpc\_type< T >:



### Additional Inherited Members

### Public Types inherited from [L4::Types::Bool< true >](#)

- typedef [Bool< V >](#) **type**  
*The meta type itself.*

### 15.138.1 Detailed Description

```
template<typename T>
struct L4::lpc::Msg::ls_valid_rpc_type< T >
```

Type trait defining a valid RPC parameter type.

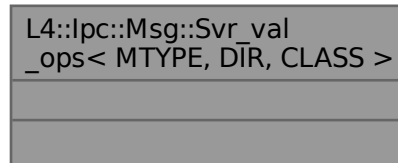
Definition at line 350 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

- `I4/sys/cxx/ipc_basics`



Collaboration diagram for L4::lpc::Msg::Svr\_val\_ops< MTYPE, DIR, CLASS >:



### 15.140.1 Detailed Description

**template<typename MTYPE, typename DIR, typename CLASS>**  
**struct L4::lpc::Msg::Svr\_val\_ops< MTYPE, DIR, CLASS >**

Defines server-side handling for MTYPE server arguments.

#### Template Parameters

<i>MTYPE</i>	Elem<T>::svr_type (where T is the type used in the RPC definition)
<i>DIR</i>	<a href="#">Dir_in</a> (client -> server), or <a href="#">Dir_out</a> (server -> client)
<i>CLASS</i>	<a href="#">Cls_data</a> , <a href="#">Cls_item</a> , or <a href="#">Cls_buffer</a>

Definition at line 275 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

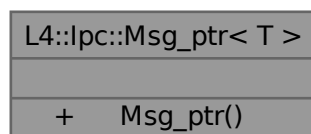
- l4/sys/cxx/ipc\_basics

## 15.141 L4::lpc::Msg\_ptr< T > Class Template Reference

Pointer to an element of type T in an [lpc::lstream](#).

```
#include <ipc_stream>
```

Collaboration diagram for L4::lpc::Msg\_ptr< T >:



## Public Member Functions

- [Msg\\_ptr](#) (T \*&p)

Create a [Msg\\_ptr](#) object that set pointer *p* to point into the message buffer.

### 15.141.1 Detailed Description

```
template<typename T>
class L4::lpc::Msg_ptr< T >
```

Pointer to an element of type *T* in an [lpc::lstream](#).

This wrapper can be used to extract an element of type *T* from an [lpc::lstream](#), whereas the data is not copied out, but a pointer into the message buffer itself is returned. With is mechanism it is possible to avoid an extra copy of large data structures from a received IPC message, instead the returned pointer gives direct access to the data in the message.

See [msg\\_ptr\(\)](#).

Definition at line 240 of file [ipc\\_stream](#).

### 15.141.2 Constructor & Destructor Documentation

#### 15.141.2.1 Msg\_ptr()

```
template<typename T >
L4::lpc::Msg_ptr< T >::Msg_ptr (
    T *& p ) [inline], [explicit]
```

Create a [Msg\\_ptr](#) object that set pointer *p* to point into the message buffer.

#### Parameters

<i>p</i>	The pointer that is adjusted to point into the message buffer.
----------	--

Definition at line 251 of file [ipc\\_stream](#).

The documentation for this class was generated from the following file:

- [l4/cxx/ipc\\_stream](#)

## 15.142 L4::lpc::Opt< T > Struct Template Reference

Attribute for defining an optional RPC argument.

```
#include <ipc_types>
```

Collaboration diagram for L4::lpc::Opt< T >:

L4::lpc::Opt< T >
+ _value
+ _valid
+ Opt()
+ Opt()
+ operator=()
+ set_valid()
+ operator->()
+ operator->()
+ value()
+ value()
+ is_valid()

## Public Member Functions

- **Opt** () noexcept  
*Make an absent optional argument.*
- **Opt** (T **value**) noexcept  
*Make a present optional argument with the given value.*
- **Opt** & **operator=** (T **value**) noexcept  
*Assign a value to the optional argument (makes the argument present)*
- void **set\_valid** (bool valid=true) noexcept  
*Set the argument to present or absent.*
- T \* **operator->** () noexcept  
*Get the pointer to the value.*
- T const \* **operator->** () const noexcept  
*Get the const pointer to the value.*
- T **value** () const noexcept  
*Get the value.*
- T & **value** () noexcept  
*Get the value.*
- bool **is\_valid** () const noexcept  
*Get true if present, false if not.*

## Data Fields

- T **\_value**  
*The value.*
- bool **\_valid**  
*True if the optional argument is present, false else.*

### 15.142.1 Detailed Description

```
template<typename T>  
struct L4::lpc::Opt< T >
```

Attribute for defining an optional RPC argument.

Definition at line 147 of file [ipc\\_types](#).

The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc\\_types](#)

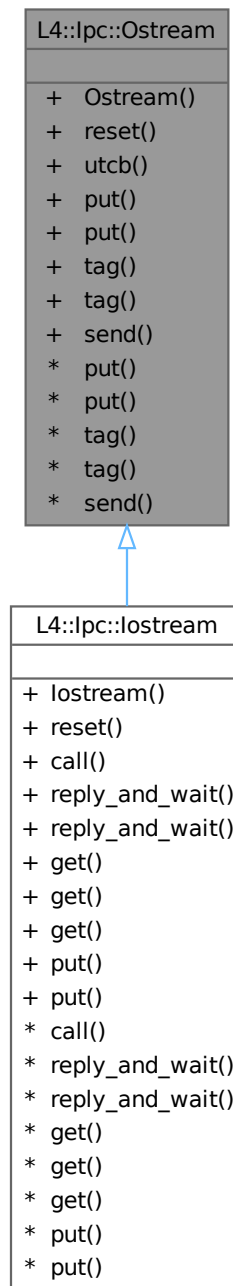
## 15.143 L4::lpc::Ostream Class Reference

Output stream for IPC marshalling.

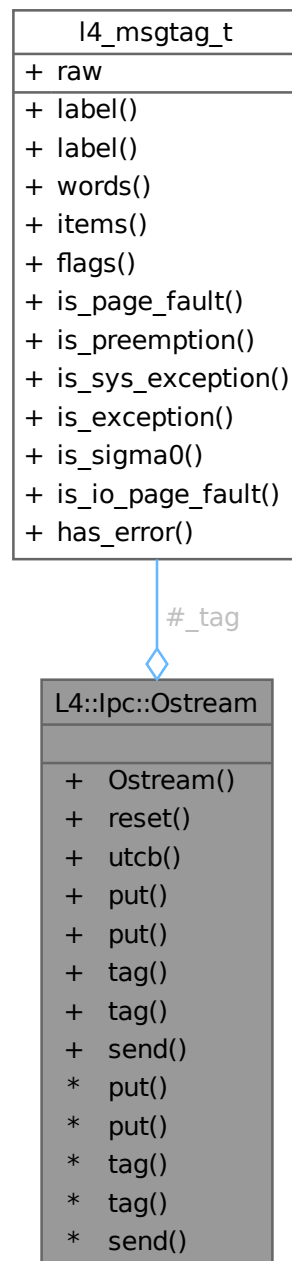
```
#include <ipc_stream>
```



Inheritance diagram for L4::lpc::Ostream:



Collaboration diagram for L4::lpc::Ostream:



## Public Member Functions

- **Ostream** ([l4\\_utcb\\_t](#) \*utcb)  
*Create an IPC output stream using the given message buffer `utcb`.*
- void **reset** ()  
*Reset the stream to empty, same state as a newly created stream.*
- [l4\\_utcb\\_t](#) \* **utcb** () const

*Return utcb pointer.*

### Get/Put functions.

*These functions are basically used to implement the insertion operators (<<) and should not be called directly.*

- `template<typename T >`  
`bool put (T *buf, unsigned long size)`  
*Put an array with `size` elements of type `T` into the stream.*
- `template<typename T >`  
`bool put (T const &v)`  
*Insert an element of type `T` into the stream.*
- `l4_msgtag_t tag () const`  
*Extract the `L4` message tag from the stream.*
- `l4_msgtag_t & tag ()`  
*Extract a reference to the `L4` message tag from the stream.*

### IPC operations.

- `l4_msgtag_t send (l4_cap_idx_t dst, long proto=0, unsigned flags=0)`  
*Send the message via IPC to the given receiver.*

## 15.143.1 Detailed Description

Output stream for IPC marshalling.

`lpc::Ostream` is part of the dynamic IPC marshalling infrastructure, as well as `lpc::Istream` and `lpc::lostream`.

`lpc::Ostream` is an output stream supporting insertion of values into an IPC message buffer. A IPC message can be marshalled using the usual insertion operator <<, see [IPC stream operators](#) .

There exist some special wrapper classes to insert arrays (see `lpc::Buf_cp_out`) and indirect strings (see `Msg_↔out_buffer` and `Msg_io_buffer`).

Definition at line 634 of file [ipc\\_stream](#).

## 15.143.2 Member Function Documentation

### 15.143.2.1 put() [1/2]

```
template<typename T >
bool L4::Ipc::Ostream::put (
    T * buf,
    unsigned long size ) [inline]
```

Put an array with `size` elements of type `T` into the stream.

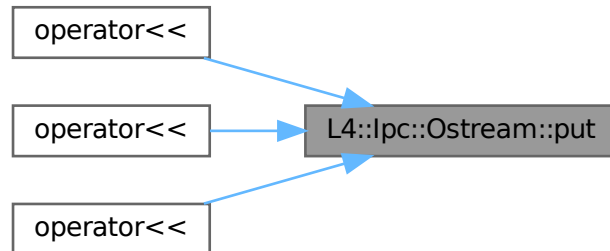
#### Parameters

<i>buf</i>	A pointer to the array to insert into the buffer.
<i>size</i>	The number of elements in the array.

Definition at line 671 of file [ipc\\_stream](#).

Referenced by [operator<<\(\)](#), [operator<<\(\)](#), and [operator<<\(\)](#).

Here is the caller graph for this function:



#### 15.143.2.2 put() [2/2]

```
template<typename T >
bool L4::Ipc::Ostream::put (
    T const & v ) [inline]
```

Insert an element of type T into the stream.

##### Parameters

<i>v</i>	The element to insert.
----------	------------------------

Definition at line 689 of file [ipc\\_stream](#).

#### 15.143.2.3 send()

```
l4\_msgtag\_t L4::Ipc::Ostream::send (
    l4\_cap\_idx\_t dst,
    long proto = 0,
    unsigned flags = 0 ) [inline]
```

Send the message via IPC to the given receiver.

##### Parameters

<i>dst</i>	The destination for the message.
<i>proto</i>	Protocol to use.
<i>flags</i>	Flags to use.

**Returns**

The syscall return tag.

Definition at line 970 of file [ipc\\_stream](#).

References [L4\\_IPC\\_NEVER](#), [l4\\_ipc\\_send\(\)](#), [L4\\_MSGTAG\\_FLAGS](#), and [tag\(\)](#).

Here is the call graph for this function:

**15.143.2.4 tag() [1/2]**

```
l4_msgtag_t & L4::Ipc::Ostream::tag ( ) [inline]
```

Extract a reference to the [L4](#) message tag from the stream.

**Returns**

A reference to the [L4](#) message tag.

Definition at line 724 of file [ipc\\_stream](#).

**15.143.2.5 tag() [2/2]**

```
l4_msgtag_t L4::Ipc::Ostream::tag ( ) const [inline]
```

Extract the [L4](#) message tag from the stream.

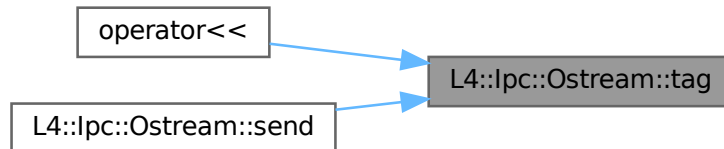
**Returns**

The extracted [L4](#) message tag.

Definition at line [717](#) of file [ipc\\_stream](#).

Referenced by [operator<<\(\)](#), and [send\(\)](#).

Here is the caller graph for this function:



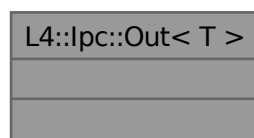
The documentation for this class was generated from the following file:

- [l4/cxx/ipc\\_stream](#)

## 15.144 L4::ipc::Out< T > Struct Template Reference

Mark an argument as a output value in an RPC signature.

Collaboration diagram for `L4::ipc::Out< T >`:



### 15.144.1 Detailed Description

```
template<typename T>
struct L4::ipc::Out< T >
```

Mark an argument as a output value in an RPC signature.

## Template Parameters

<i>T</i>	The original type of the argument.
----------	------------------------------------

## Note

The use of Out<> is usually not needed, because typical out-put data types in C++ (pointers to non-const objects or non-const references are interpreted as output values anyway. However, there are some data types, such as returned capabilities that can be marked as such by using Out<>.

Definition at line 42 of file [ipc\\_types](#).

The documentation for this struct was generated from the following file:

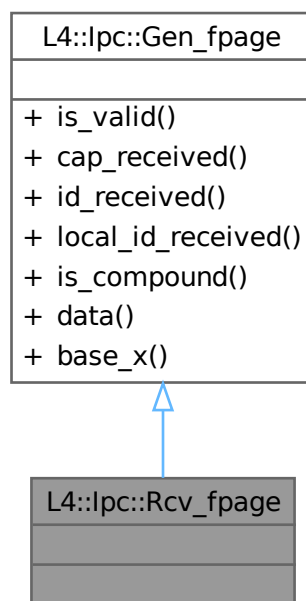
- [l4/sys/cxx/ipc\\_types](#)

## 15.145 L4::lpc::Rcv\_fpage Class Reference

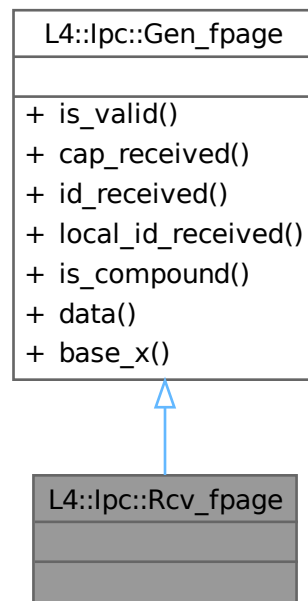
Rcv flex-page.

```
#include <ipc_types>
```

Inheritance diagram for L4::lpc::Rcv\_fpage:



Collaboration diagram for L4::lpc::Rcv\_fpage:



### Additional Inherited Members

### Public Types inherited from [L4::lpc::Gen\\_fpage](#)

- enum [Type](#)  
*Type of mapping object, see `L4_fpage_type`.*
- enum [Map\\_type](#)  
*Kind of mapping.*
- enum [Cacheopt](#)  
*Caching options, see `l4_fpage_cacheability_opt_t`.*

### Public Member Functions inherited from [L4::lpc::Gen\\_fpage](#)

- bool **`is_valid()`** const noexcept  
*Check if the capability is valid.*
- bool **`cap_received()`** const noexcept  
*Check if at least one capability has been mapped.*
- bool **`id_received()`** const noexcept  
*Check if a label was received instead of a mapping.*
- bool **`local_id_received()`** const noexcept  
*Check if a local capability id has been received.*
- bool **`is_compound()`** const noexcept  
*Check if the received item has the compound bit set.*
- **`l4_umword_t data()`** const noexcept  
*Return the raw flex page descriptor.*
- **`l4_umword_t base_x()`** const noexcept  
*Return the raw base descriptor.*



### 15.145.1 Detailed Description

Rcv flex-page.

Definition at line 459 of file [ipc\\_types](#).

The documentation for this class was generated from the following file:

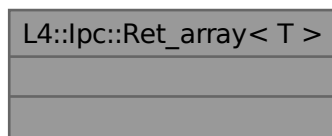
- [l4/sys/cxx/ipc\\_types](#)

## 15.146 L4::lpc::Ret\_array< T > Struct Template Reference

Dynamically sized output array of type T.

```
#include <ipc_ret_array>
```

Collaboration diagram for L4::lpc::Ret\_array< T >:



### 15.146.1 Detailed Description

```
template<typename T>
struct L4::lpc::Ret_array< T >
```

Dynamically sized output array of type T.

#### Template Parameters

<i>T</i>	The data-type of each array element.
----------	--------------------------------------

Ret\_array<> is a special dynamically sized output array where the number of transmitted elements is passed in the return value of the call (if positive)

Definition at line 34 of file [ipc\\_ret\\_array](#).

The documentation for this struct was generated from the following file:

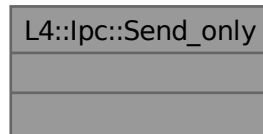
- [l4/sys/cxx/ipc\\_ret\\_array](#)

## 15.147 L4::lpc::Send\_only Struct Reference

RPC attribute for a send-only RPC.

```
#include <ipc_iface>
```

Collaboration diagram for L4::lpc::Send\_only:



### 15.147.1 Detailed Description

RPC attribute for a send-only RPC.

This class can be used as FLAGS parameter to L4::lpc::Msg::Rpc\_call and L4::lpc::Msg::Rpc\_inline\_call templates and declares the RPC to use send-only semantics and timeouts.

Examples:

```
L4_RPC(long, send, (unsigned value), L4::lpc::Send_only);
```

Definition at line 274 of file [ipc\\_iface](#).

The documentation for this struct was generated from the following file:

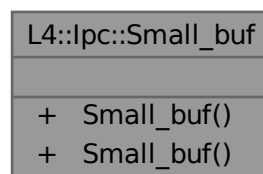
- [l4/sys/cxx/ipc\\_iface](#)

## 15.148 L4::lpc::Small\_buf Class Reference

A receive item for receiving a single object capability.

```
#include <ipc_types>
```

Collaboration diagram for L4::lpc::Small\_buf:



## Public Member Functions

- [Small\\_buf](#) ([L4::Cap](#)< void > cap, unsigned long flags=0) noexcept  
*Create a receive item from a C++ cap.*
- [Small\\_buf](#) ([l4\\_cap\\_idx\\_t](#) cap, unsigned long flags=0) noexcept  
*Create a receive item from a C cap.*

### 15.148.1 Detailed Description

A receive item for receiving a single object capability.

This class is the main abstraction for receiving object capabilities via [lpc::lstream](#). To receive an object capability, an instance of [Small\\_buf](#) that refers to an empty capability slot must be inserted into the [lpc::lstream](#) before the receive operation.

Definition at line 268 of file [ipc\\_types](#).

### 15.148.2 Constructor & Destructor Documentation

#### 15.148.2.1 Small\_buf() [1/2]

```
L4::Ipc::Small_buf::Small_buf (
    L4::Cap< void > cap,
    unsigned long flags = 0 ) [inline], [explicit], [noexcept]
```

Create a receive item from a C++ cap.

##### Parameters

<i>cap</i>	Capability slot where to save the capability.
<i>flags</i>	Receive buffer flags, see <a href="#">l4_msg_item_consts_t</a> . L4_RCV_ITEM_SINGLE_CAP will always be set.

Definition at line 278 of file [ipc\\_types](#).

#### 15.148.2.2 Small\_buf() [2/2]

```
L4::Ipc::Small_buf::Small_buf (
    l4\_cap\_idx\_t cap,
    unsigned long flags = 0 ) [inline], [explicit], [noexcept]
```

Create a receive item from a C cap.

##### Parameters

<i>cap</i>	Capability slot where to save the capability.
<i>flags</i>	Receive buffer flags, see <a href="#">l4_msg_item_consts_t</a> . L4_RCV_ITEM_SINGLE_CAP will always be set.

Definition at line 285 of file [ipc\\_types](#).

The documentation for this class was generated from the following file:

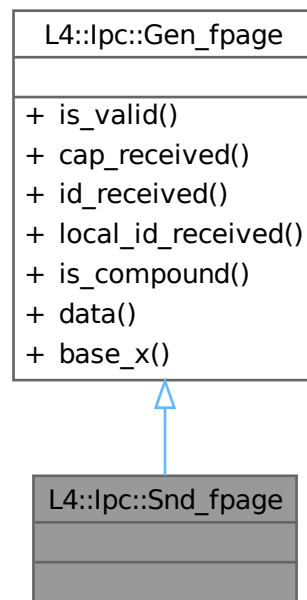
- [l4/sys/cxx/ipc\\_types](#)

## 15.149 L4::lpc::Snd\_fpage Class Reference

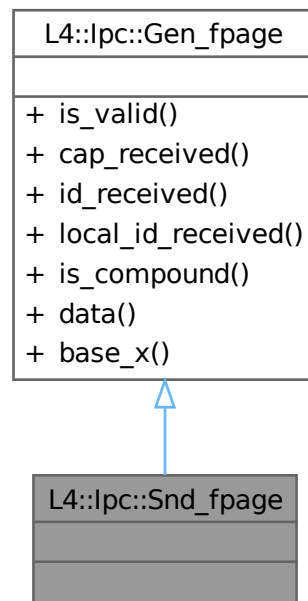
Send flex-page.

```
#include <ipc_types>
```

Inheritance diagram for L4::lpc::Snd\_fpage:



Collaboration diagram for L4::lpc::Snd\_fpage:



### Additional Inherited Members

### Public Types inherited from [L4::lpc::Gen\\_fpage](#)

- enum [Type](#)  
*Type of mapping object, see `L4_fpage_type`.*
- enum [Map\\_type](#)  
*Kind of mapping.*
- enum [Cacheopt](#)  
*Caching options, see `l4_fpage_cacheability_opt_t`.*

### Public Member Functions inherited from [L4::lpc::Gen\\_fpage](#)

- bool **`is_valid()`** const noexcept  
*Check if the capability is valid.*
- bool **`cap_received()`** const noexcept  
*Check if at least one capability has been mapped.*
- bool **`id_received()`** const noexcept  
*Check if a label was received instead of a mapping.*
- bool **`local_id_received()`** const noexcept  
*Check if a local capability id has been received.*
- bool **`is_compound()`** const noexcept  
*Check if the received item has the compound bit set.*
- **`l4_umword_t data()`** const noexcept  
*Return the raw flex page descriptor.*
- **`l4_umword_t base_x()`** const noexcept  
*Return the raw base descriptor.*

### 15.149.1 Detailed Description

Send flex-page.

Definition at line 407 of file [ipc\\_types](#).

The documentation for this class was generated from the following file:

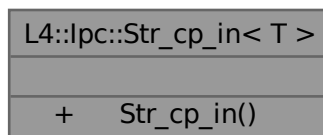
- [l4/sys/cxx/ipc\\_types](#)

## 15.150 L4::lpc::Str\_cp\_in< T > Class Template Reference

Abstraction for extracting a zero-terminated string from an [lpc::lstream](#).

```
#include <ipc_stream>
```

Collaboration diagram for L4::lpc::Str\_cp\_in< T >:



### Public Member Functions

- [Str\\_cp\\_in](#) (T \*v, unsigned long &size)  
Create a buffer for extracting an array from an [lpc::lstream](#).

### 15.150.1 Detailed Description

```
template<typename T>
class L4::lpc::Str_cp_in< T >
```

Abstraction for extracting a zero-terminated string from an [lpc::lstream](#).

An instance of [Str\\_cp\\_in](#) can be used to extract a zero-terminated string an [lpc::lstream](#). The data from the received message is thereby copied to the given buffer and size is set to the number of characters found in the stream. The string is zero terminated in any circumstances. When the given buffer is smaller than the received string the last byte in the buffer will be the zero terminator. In the case the received string is shorter than the given buffer the zero termination will be placed behind the received data. This provides a zero-terminated result even in cases where the sender did not provide proper termination or in cases of too small receiver buffers.

See also

[str\\_cp\\_in\(\)](#).

Definition at line 189 of file [ipc\\_stream](#).

## 15.150.2 Constructor & Destructor Documentation

### 15.150.2.1 Str\_cp\_in()

```
template<typename T >
L4::Ipc::Str_cp_in< T >::Str_cp_in (
    T * v,
    unsigned long & size ) [inline]
```

Create a buffer for extracting an array from an [lpc::lstream](#).

#### Parameters

	<i>v</i>	The buffer for string.
<i>in, out</i>	<i>size</i>	Input: The number of bytes available in <i>v</i> Output: The number of bytes received (including the terminator).

Definition at line 200 of file [ipc\\_stream](#).

The documentation for this class was generated from the following file:

- [l4/cxx/ipc\\_stream](#)

## 15.151 L4::lpc::Varg Class Reference

Variably sized RPC argument.

```
#include <ipc_varg>
```

Inherited by `L4::lpc::Varg_t< T >`.

Collaboration diagram for L4::lpc::Varg:

L4::lpc::Varg
<ul style="list-style-type: none"> <li>+ type()</li> <li>+ length()</li> <li>+ tag()</li> <li>+ tag()</li> <li>+ data()</li> <li>+ data()</li> <li>+ Varg()</li> <li>+ Varg()</li> <li>+ value()</li> <li>+ is_of()</li> <li>and 8 more...</li> </ul>

## Public Types

- typedef [l4\\_umword\\_t](#) **Tag**

*The data type for the tag.*

## Public Member Functions

- [L4\\_varg\\_type](#) [type](#) () const
- unsigned [length](#) () const  
*Get the size of the RPC argument.*
- [Tag](#) [tag](#) () const
- void **tag** ([Tag](#) tag)  
*Set [Varg](#) tag (usually from message)*
- void **data** (char const \*d)  
*Set [Varg](#) to indirect data value (usually in UTCB)*
- char const \* [data](#) () const
- **Varg** ()=default  
*Make uninitialized [Varg](#).*
- **Varg** ([L4\\_varg\\_type](#) t, void const \*v, int len)  
*Make an indirect varg.*
- template<typename V >  
  [Va\\_type](#)< V >::Ret\_value [value](#) () const
- template<typename T >  
  bool [is\\_of](#) () const
- bool [is\\_nil](#) () const
- bool [is\\_of\\_int](#) () const
- template<typename T >  
  bool [get\\_value](#) (typename [Va\\_type](#)< T >::Value \*v) const  
*Get the value of the [Varg](#) as type T.*
- template<typename T >  
  void **set\_value** (void const \*d)  
*Set to indirect value of type T.*
- template<typename T >  
  void **set\_direct\_value** (T val, typename [L4::Types::Enable\\_if](#)< sizeof(T)<=sizeof(char const \*), bool >::type=true)  
*Set to directly stored value of type T.*
- template<typename T >  
  **Varg** (T const \*[data](#))  
*Make [Varg](#) from indirect value (pointer)*
- **Varg** (char const \*[data](#))  
*Make [Varg](#) from null-terminated string.*
- template<typename T >  
  **Varg** (T [data](#), typename [L4::Types::Enable\\_if](#)< sizeof(T)<=sizeof(char const \*), bool >::type=true)  
*Make [Varg](#) from direct value.*

### 15.151.1 Detailed Description

Variably sized RPC argument.

Definition at line 107 of file [ipc\\_varg](#).



## 15.151.2 Member Function Documentation

### 15.151.2.1 data()

```
char const * L4::Ipc::Varg::data ( ) const [inline]
```

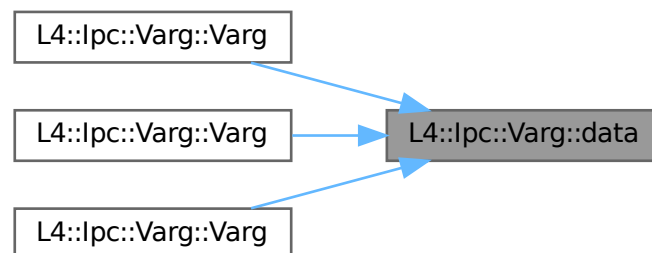
#### Returns

pointer to the data, also safe for direct data

Definition at line 134 of file [ipc\\_varg](#).

Referenced by [Varg\(\)](#), [Varg\(\)](#), and [Varg\(\)](#).

Here is the caller graph for this function:



### 15.151.2.2 get\_value()

```
template<typename T >
bool L4::Ipc::Varg::get_value (
    typename Va_type< T >::Value * v ) const [inline]
```

Get the value of the [Varg](#) as type T.

#### Template Parameters

<i>T</i>	The expected type of the <a href="#">Varg</a> .
----------	---

#### Parameters

<i>v</i>	Pointer to store the value
----------	----------------------------

**Returns**

true when the [Varg](#) is of type T, false if not

Definition at line 196 of file [ipc\\_varg](#).

**15.151.2.3 is\_nil()**

```
bool L4::Ipc::Varg::is_nil ( ) const [inline]
```

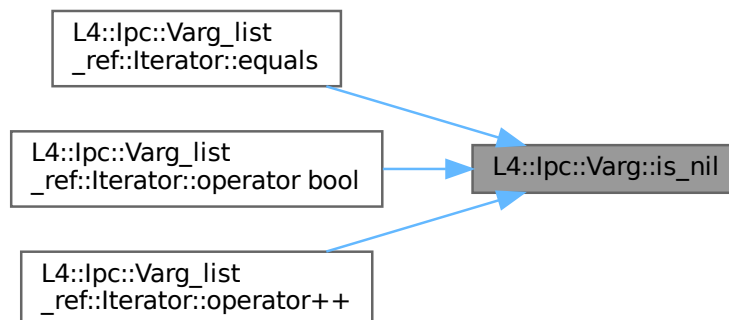
**Returns**

true if the [Varg](#) is of nil type.

Definition at line 183 of file [ipc\\_varg](#).

Referenced by [L4::Ipc::Varg\\_list\\_ref::Iterator::equals\(\)](#), [L4::Ipc::Varg\\_list\\_ref::Iterator::operator bool\(\)](#), and [L4::Ipc::Varg\\_list\\_ref::Iterator::operator++\(\)](#).

Here is the caller graph for this function:

**15.151.2.4 is\_of()**

```
template<typename T >
bool L4::Ipc::Varg::is_of ( ) const [inline]
```

**Returns**

true if the [Varg](#) is of type T

Definition at line 180 of file [ipc\\_varg](#).

References [type\(\)](#).

Here is the call graph for this function:

**15.151.2.5 is\_of\_int()**

```
bool L4::Ipc::Varg::is_of_int ( ) const [inline]
```

**Returns**

true if the [Varg](#) is an integer type (signed or unsigned).

Definition at line 186 of file [ipc\\_varg](#).

References [type\(\)](#).

Here is the call graph for this function:

**15.151.2.6 length()**

```
unsigned L4::Ipc::Varg::length ( ) const [inline]
```

Get the size of the RPC argument.

**Returns**

The size of the RPC argument

Definition at line 125 of file [ipc\\_varg](#).

**15.151.2.7 tag()**

```
Tag L4::Ipc::Varg::tag ( ) const [inline]
```

**Returns**

the tag value (the Direct\_data bit masked)

Definition at line 127 of file [ipc\\_varg](#).

**15.151.2.8 type()**

```
L4_varg_type L4::Ipc::Varg::type ( ) const [inline]
```

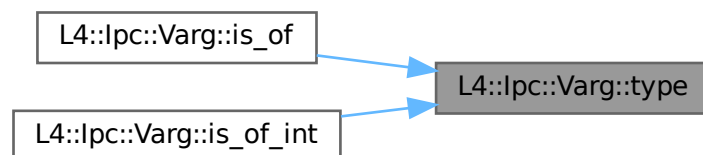
**Returns**

the type field of the tag

Definition at line 120 of file [ipc\\_varg](#).

Referenced by [is\\_of\(\)](#), and [is\\_of\\_int\(\)](#).

Here is the caller graph for this function:

**15.151.2.9 value()**

```
template<typename V >
Va_type< V >::Ret_value L4::Ipc::Varg::value ( ) const [inline]
```

**Template Parameters**

V	The data type of the value to retrieve.
---	---

**Precondition**

The [Varg](#) must be of type *V* (otherwise the result is unpredictable).

### Returns

The value of the [Varg](#) as type V.

Definition at line 166 of file [ipc\\_varg](#).

The documentation for this class was generated from the following file:

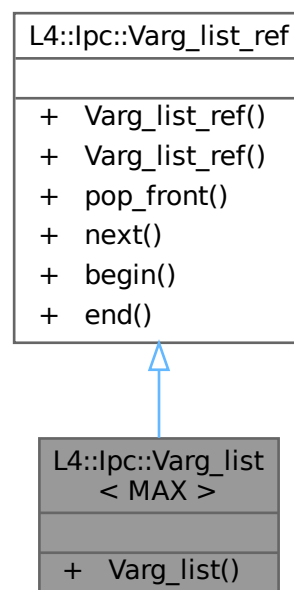
- l4/sys/cxx/ipc\_varg

## 15.152 L4::lpc::Varg\_list< MAX > Class Template Reference

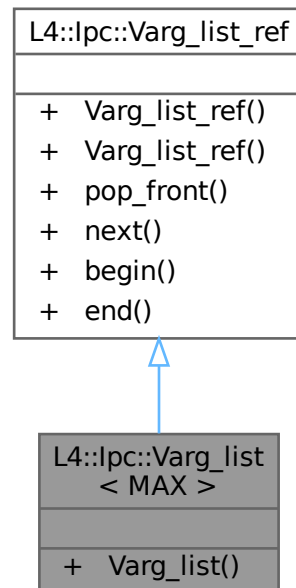
Self-contained list of variable-sized RPC parameters.

```
#include <ipc_varg>
```

Inheritance diagram for L4::lpc::Varg\_list< MAX >:



Collaboration diagram for L4::lpc::Varg\_list< MAX >:



### Public Member Functions

- **Varg\_list** ([Varg\\_list\\_ref](#) const &r)  
*Create a parameter list as a copy from a referencing list.*

### Public Member Functions inherited from [L4::lpc::Varg\\_list\\_ref](#)

- **Varg\_list\_ref** ()=default  
*Create an empty parameter list.*
- [Varg\\_list\\_ref](#) (void const \*start, void const \*end)  
*Create a parameter list over a given memory region.*
- [Varg](#) **pop\_front** ()  
*Get the next parameter in the list.*
- [Varg](#) **next** ()  
*Get the next parameter in the list.*
- [Iterator](#) **begin** () const  
*Returns an iterator to the first [Varg](#).*
- [Iterator](#) **end** () const  
*Returns the end of the list.*

### 15.152.1 Detailed Description

```
template<unsigned MAX>
class L4::lpc::Varg_list< MAX >
```

Self-contained list of variable-sized RPC parameters.

Works like [Varg\\_list\\_ref](#) but contains a full copy of the data. Use this as a parameter in server functions, if the handler function needs to use the UTCB (e.g. while sending further IPC).

Definition at line 422 of file [ipc\\_varg](#).

The documentation for this class was generated from the following file:

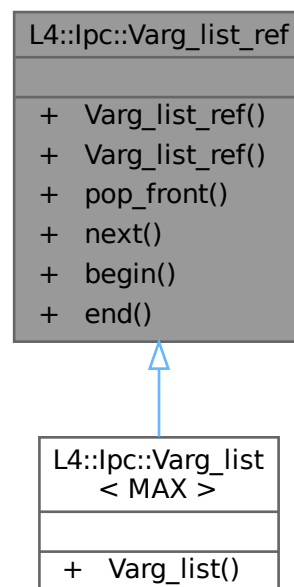
- [l4/sys/cxx/ipc\\_varg](#)

## 15.153 L4::lpc::Varg\_list\_ref Class Reference

List of variable-sized RPC parameters as received by the server.

```
#include <ipc_varg>
```

Inheritance diagram for L4::lpc::Varg\_list\_ref:



Collaboration diagram for L4::lpc::Varg\_list\_ref:

L4::lpc::Varg_list_ref
<ul style="list-style-type: none"> <li>+ Varg_list_ref()</li> <li>+ Varg_list_ref()</li> <li>+ pop_front()</li> <li>+ next()</li> <li>+ begin()</li> <li>+ end()</li> </ul>

## Data Structures

- class [Iterator](#)  
*Iterator for Valists.*

## Public Member Functions

- **Varg\_list\_ref** ()=default  
*Create an empty parameter list.*
- **Varg\_list\_ref** (void const \*start, void const \*end)  
*Create a parameter list over a given memory region.*
- **Varg pop\_front** ()  
*Get the next parameter in the list.*
- **Varg next** ()  
*Get the next parameter in the list.*
- **Iterator begin** () const  
*Returns an iterator to the first Varg.*
- **Iterator end** () const  
*Returns the end of the list.*

### 15.153.1 Detailed Description

List of variable-sized RPC parameters as received by the server.

The list can be traversed exactly once using [next\(\)](#).

This is a reference list, where the returned [Varg](#) point to data in the underlying storage, conventionally the UTCB. This type should only be used in server functions when the implementation can ensure that all content is read before the UTCB is reused (e.g. for IPC), otherwise use [Varg\\_list](#).

Definition at line 264 of file [ipc\\_varg](#).



## 15.153.2 Constructor & Destructor Documentation

### 15.153.2.1 Varg\_list\_ref()

```
L4::Ipc::Varg_list_ref::Varg_list_ref (
    void const * start,
    void const * end ) [inline]
```

Create a parameter list over a given memory region.

#### Parameters

<i>start</i>	Pointer to start of the parameter list.
<i>end</i>	Pointer to end of the list (inclusive).

Definition at line 343 of file [ipc\\_varg](#).

The documentation for this class was generated from the following file:

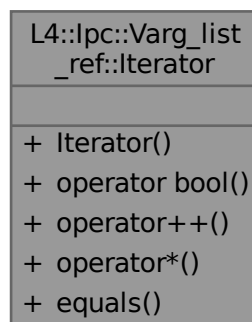
- [l4/sys/cxx/ipc\\_varg](#)

## 15.154 L4::lpc::Varg\_list\_ref::Iterator Class Reference

[Iterator](#) for Valists.

```
#include <ipc_varg>
```

Collaboration diagram for L4::lpc::Varg\_list\_ref::Iterator:



## Public Member Functions

- **Iterator** (Iter\_state const &s)  
*Create a new iterator.*
- **operator bool** () const  
*validity check for the iterator*
- **Iterator** & **operator++** ()  
*increment iterator to the next arg*
- **Varg** **operator\*** () const  
*dereference the iterator, get [Varg](#)*
- bool **equals** (**Iterator** const &o) const  
*check for equality*

### 15.154.1 Detailed Description

[Iterator](#) for Valists.

Definition at line 349 of file [ipc\\_varg](#).

The documentation for this class was generated from the following file:

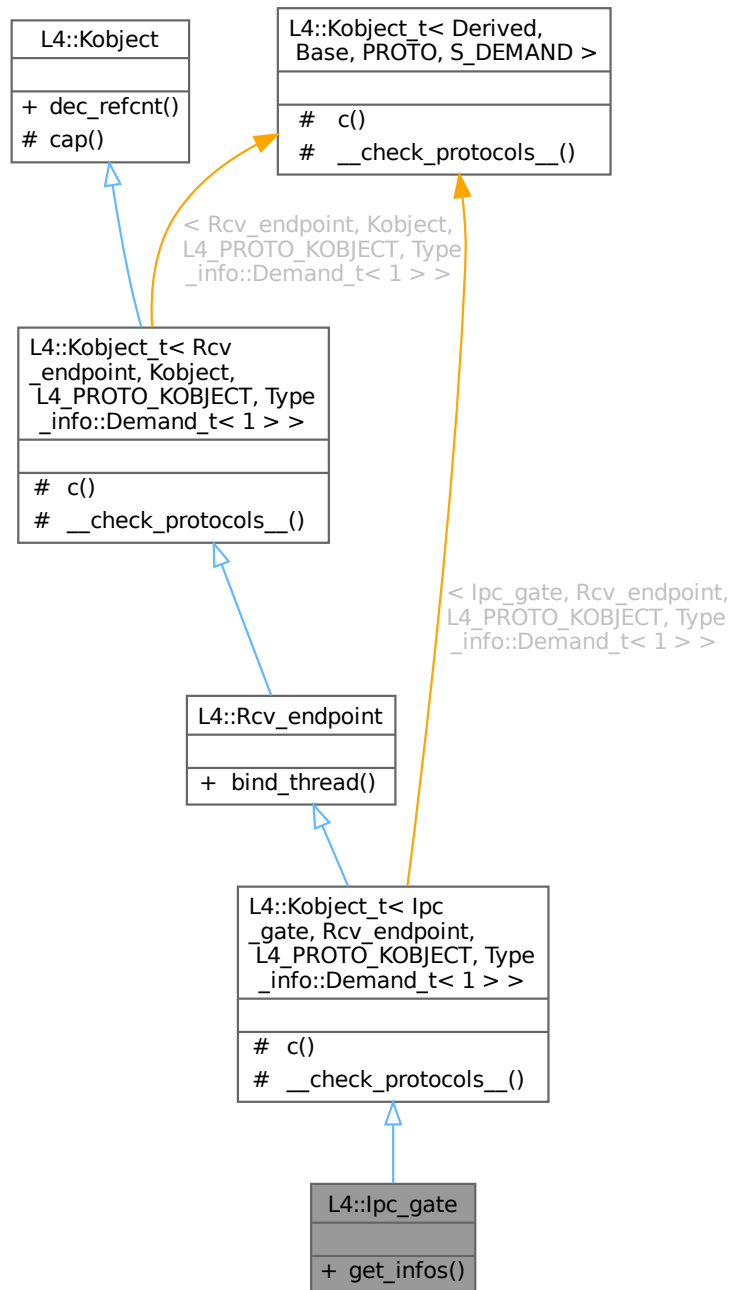
- l4/sys/cxx/ipc\_varg

## 15.155 L4::lpc\_gate Class Reference

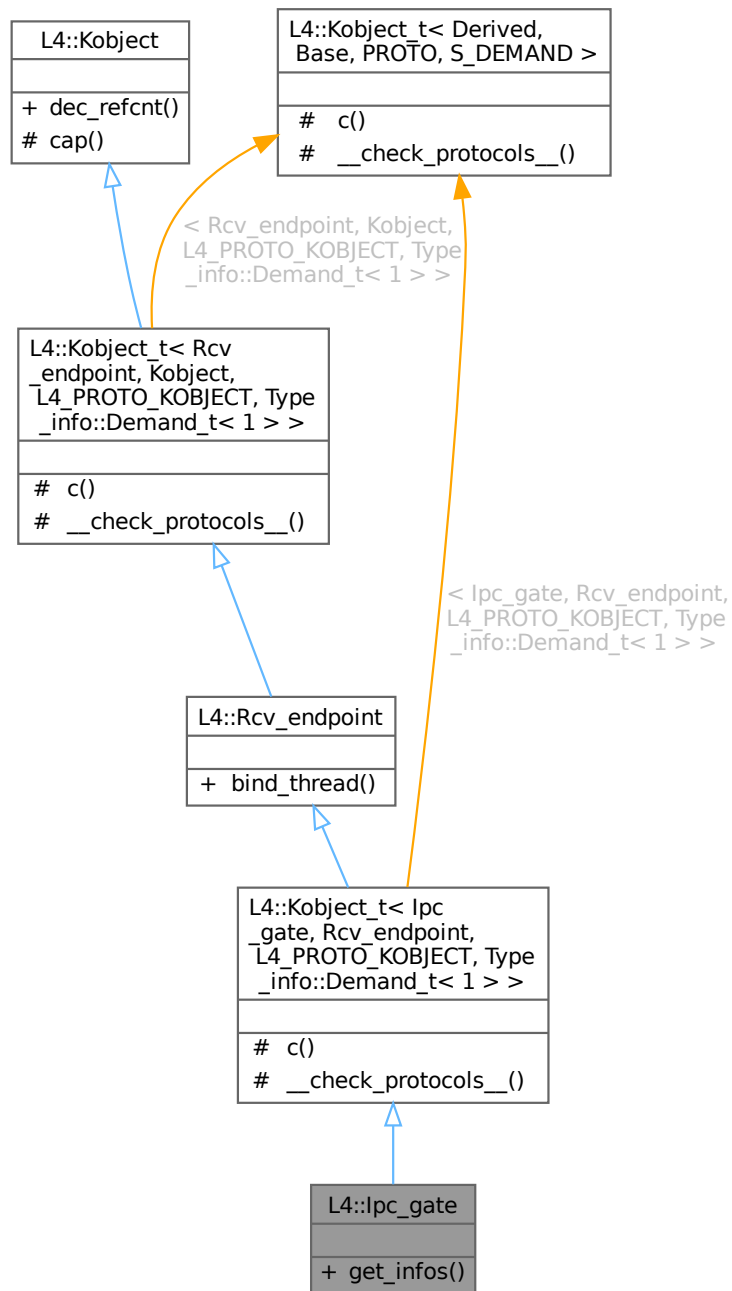
The C++ IPC gate interface, see [IPC-Gate API](#) for the C interface.

```
#include <ipc_gate>
```

Inheritance diagram for L4::lpc\_gate:



Collaboration diagram for L4::lpc\_gate:



### Public Member Functions

- `l4_msgtag_t get_infos (l4_umword_t *label)`  
Get information about the IPC-gate.

### Public Member Functions inherited from L4::Rcv\_endpoint

- `l4_msgtag_t bind_thread (lpc::Cap< Thread > t, l4_umword_t label)`

*Bind a thread to an IPC receive endpoint.*

## Public Member Functions inherited from L4::Kobject

- [l4\\_msgtag\\_t dec\\_refcnt](#) ([l4\\_mword\\_t](#) diff, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)())  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

## Protected Types inherited from

[L4::Kobject\\_t](#)< [lpc\\_gate](#), [Rcv\\_endpoint](#), [L4\\_PROTO\\_KOBJECT](#), [Type\\_info::Demand\\_t](#)< 1 > >

- typedef [lpc\\_gate](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef [Typeid::Iface](#)< [PROTO](#), [lpc\\_gate](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef [Typeid::Merge\\_list](#)< [Typeid::Iface\\_list](#)< **\_\_Iface** >, typename [Base::\\_\\_Iface\\_list](#) > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Types inherited from

[L4::Kobject\\_t](#)< [Rcv\\_endpoint](#), [Kobject](#), [L4\\_PROTO\\_KOBJECT](#), [Type\\_info::Demand\\_t](#)< 1 > >

- typedef [Rcv\\_endpoint](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef [Typeid::Iface](#)< [PROTO](#), [Rcv\\_endpoint](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef [Typeid::Merge\\_list](#)< [Typeid::Iface\\_list](#)< **\_\_Iface** >, typename [Base::\\_\\_Iface\\_list](#) > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [lpc\\_gate](#), [Rcv\\_endpoint](#), [L4\\_PROTO\\_KOBJECT](#), [Type\\_info::Demand\\_t](#)< 1 > >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept  
*Get the capability to ourselves.*

## Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Rcv\\_endpoint](#), [Kobject](#), [L4\\_PROTO\\_KOBJECT](#), [Type\\_info::Demand\\_t](#)< 1 > >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept  
*Get the capability to ourselves.*

## Protected Member Functions inherited from L4::Kobject

- [l4\\_cap\\_idx\\_t cap](#) () const noexcept  
*Return capability selector.*

**Static Protected Member Functions inherited from****[L4::Kobject\\_t< Ipc\\_gate, Rcv\\_endpoint, L4\\_PROTO\\_KOBJECT, Type\\_info::Demand\\_t< 1 > >](#)**

- static void `__check_protocols__()` noexcept  
*Helper to check for protocol conflicts.*

**Static Protected Member Functions inherited from****[L4::Kobject\\_t< Rcv\\_endpoint, Kobject, L4\\_PROTO\\_KOBJECT, Type\\_info::Demand\\_t< 1 > >](#)**

- static void `__check_protocols__()` noexcept  
*Helper to check for protocol conflicts.*

**15.155.1 Detailed Description**

The C++ IPC gate interface, see [IPC-Gate API](#) for the C interface.

IPC gates are used to create secure communication channels between protection domains. An IPC gate can be created using the [L4::Factory](#) interface.

Depending on the permissions of the capability used, an IPC gate forwards IPC to the [L4::Thread](#) that is *bound* to the IPC gate (cf. [bind\\_thread\(\)](#)). If the capability has the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission, only IPC using a protocol different from the [L4\\_PROTO\\_KOBJECT](#) protocol is forwarded. Without the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission, all IPC is forwarded. The latter is the usual case for a client in a client/server scenario. When no thread is bound yet, the forwarded IPC blocks until a thread is bound or the IPC times out.

Forwarded IPC is always forwarded to the userland of the bound thread. That means, the [L4::Thread](#) interface of the bound thread is not accessible via an IPC gate. The [L4::ipc\\_gate](#) interface of an IPC gate is only accessible if the capability used has the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission (cf. previous paragraph). Conversely that means, if the capability used lacks the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission, [L4::ipc\\_gate](#) interface calls are forwarded to the bound thread instead of being processed by the IPC gate itself. In a client/server scenario, a client should only get IPC gate capabilities without [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission so the client cannot tamper with the IPC gate.

When binding a thread to an IPC gate, a user-defined, kernel protected, machine-word sized payload called the IPC gate's *label* is assigned to the IPC gate (cf. [bind\\_thread\(\)](#)). When a send-only IPC or call IPC is forwarded via an IPC gate, the label provided by the sender is ignored and replaced by the IPC gate's label where the two least significant bits are the result of bitwise disjunction of the corresponding label bits with the [L4\\_CAP\\_FPAGE\\_S](#) and [L4\\_CAP\\_FPAGE\\_W](#) permissions of the capability used. Hence, the label provided via [bind\\_thread\(\)](#) should usually have its two least significant bits set to zero. The replaced label is only visible to the bound thread upon receive. However, the configured label of an IPC gate can also be queried via [get\\_infos\(\)](#) if the capability used has the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission.

When deleting an IPC gate or when unbinding it from a thread, the label of IPC already in flight won't be changed. To ensure that no IPC from this IPC gate is received by a thread with an unexpected label, [L4::Thread::modify\\_senders\(\)](#) shall be used to change the labels of every pending IPC to that gate. This is also required if the label of an already bound IPC gate is changed. It is not necessary after binding the IPC gate to a thread for the first time.

When binding a new thread to an IPC gate that is currently bound, the same label should be used that was used with the old thread. Otherwise the old and the new thread need to synchronize to avoid IPC messages with unexpected labels.

**Include File**

```
#include <l4/sys/ipc_gate>
```

For the C interface refer to the C [IPC-Gate API](#).

See also

[Object Invocation](#)

Definition at line 94 of file [ipc\\_gate](#).

## 15.155.2 Member Function Documentation

### 15.155.2.1 get\_infos()

```
l4_msgtag_t L4::Ipc_gate::get_infos (
    l4_umword_t * label )
```

Get information about the IPC-gate.

#### Parameters

out	<i>label</i>	The label of the IPC gate is returned here.
-----	--------------	---

#### Returns

System call return tag.

#### Precondition

If the IPC gate capability used to invoke this operation does not possess the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) right, the kernel will not perform the operation. Instead, the underlying IPC message will be forwarded to the thread bound to the IPC gate, blocking the caller if no thread is bound yet.

The documentation for this class was generated from the following file:

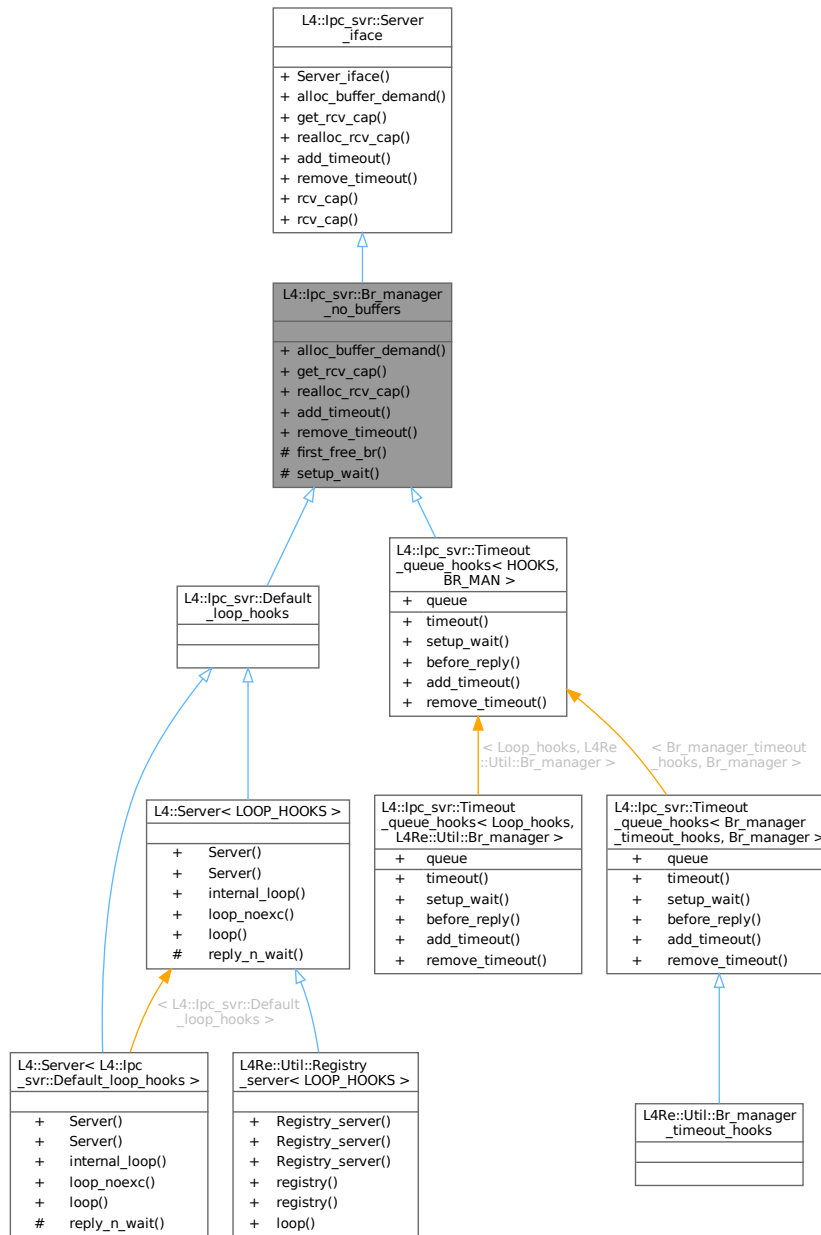
- [l4/sys/ipc\\_gate](#)

## 15.156 L4::lpc\_svr::Br\_manager\_no\_buffers Class Reference

Empty implementation of [Server\\_iface](#).

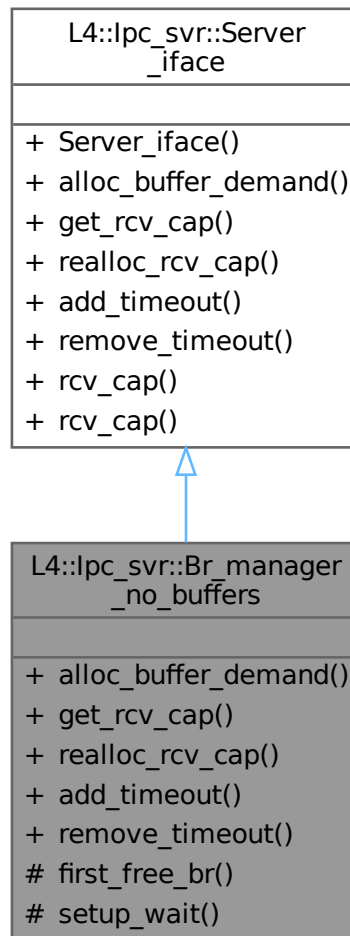
```
#include <ipc_server_loop>
```

Inheritance diagram for L4::ipc\_svr::Br\_manager\_no\_buffers:





Collaboration diagram for L4::lpc\_svr::Br\_manager\_no\_buffers:



## Public Member Functions

- int `alloc_buffer_demand` (`Demand` const &demand) override  
*Tells the server to allocate buffers for the given demand.*
- L4::Cap< void > `get_rcv_cap` (int) const override  
*Returns L4::Cap<void>::Invalid, we have no buffer management.*
- int `realloc_rcv_cap` (int) override  
*Returns -L4\_ENOMEM, we have no buffer management.*
- int `add_timeout` (`Timeout` \*, `l4_kernel_clock_t`) override  
*Returns -L4\_ENOSYS, we have no timeout queue.*
- int `remove_timeout` (`Timeout` \*) override  
*Returns -L4\_ENOSYS, we have no timeout queue.*

## Public Member Functions inherited from [L4::lpc\\_svr::Server\\_iface](#)

- **Server\_iface** ()  
*Make a server interface.*
- `template<typename T >`  
[L4::Cap](#)< T > [rcv\\_cap](#) (int index) const  
*Get given receive buffer as typed capability.*
- [L4::Cap](#)< void > [rcv\\_cap](#) (int index) const  
*Get receive cap with the given index as generic (void) type.*

## Protected Member Functions

- unsigned **first\_free\_br** () const  
*Returns 1 as first free buffer.*
- void **setup\_wait** ([l4\\_utcb\\_t](#) \*utcb, [L4::lpc\\_svr::Reply\\_mode](#))  
*Setup wait function for the server loop (Server<>).*

## Additional Inherited Members

## Public Types inherited from [L4::lpc\\_svr::Server\\_iface](#)

- typedef [L4::Type\\_info::Demand](#) **Demand**  
*Data type expressing server-side demand for receive buffers.*

### 15.156.1 Detailed Description

Empty implementation of [Server\\_iface](#).

This implementation of [Server\\_iface](#) provides no buffer or timeout management at all it just returns errors for all calls that express other than empty demands. However, this may be useful for very simple servers that serve simple server objects only.

Definition at line 184 of file [ipc\\_server\\_loop](#).

### 15.156.2 Member Function Documentation

#### 15.156.2.1 [alloc\\_buffer\\_demand\(\)](#)

```
int L4::Ipc_svr::Br_manager_no_buffers::alloc_buffer_demand (
    Demand const & demand ) [inline], [override], [virtual]
```

Tells the server to allocate buffers for the given demand.

#### Parameters

<i>demand</i>	The total server-side demand of receive buffers needed for a given interface, see <a href="#">Demand</a> .
---------------	--

This function is not called by user applications directly. Usually the server implementation or the registry implementation calls this function whenever a new object is registered at the server.

#### Returns

success (0) if demand is empty, -L4\_ENOMEM else.

Implements [L4::lpc\\_svr::Server\\_iface](#).

Definition at line 191 of file [ipc\\_server\\_loop](#).

References [L4\\_ENOMEM](#), [L4\\_EOK](#), and [L4::Type\\_info::Demand::no\\_demand\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

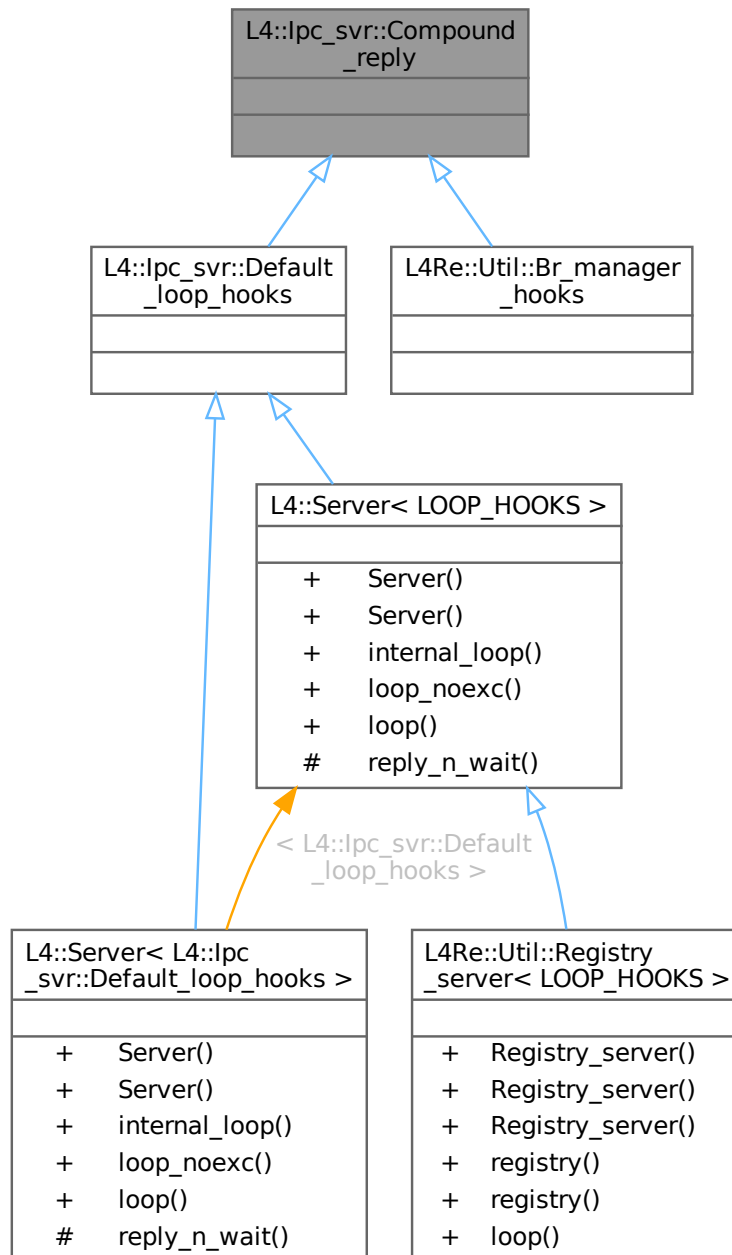
- `l4/sys/cxx/ipc_server_loop`

## 15.157 L4::lpc\_svr::Compound\_reply Struct Reference

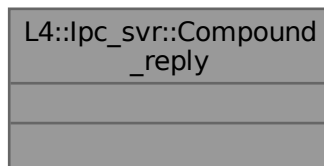
Mix in for `LOOP_HOOKS` to always use compound reply and wait.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc\_svr::Compound\_reply:



Collaboration diagram for L4::lpc\_svr::Compound\_reply:



### 15.157.1 Detailed Description

Mix in for LOOP\_HOOKS to always use compound reply and wait.

Definition at line 77 of file [ipc\\_server\\_loop](#).

The documentation for this struct was generated from the following file:

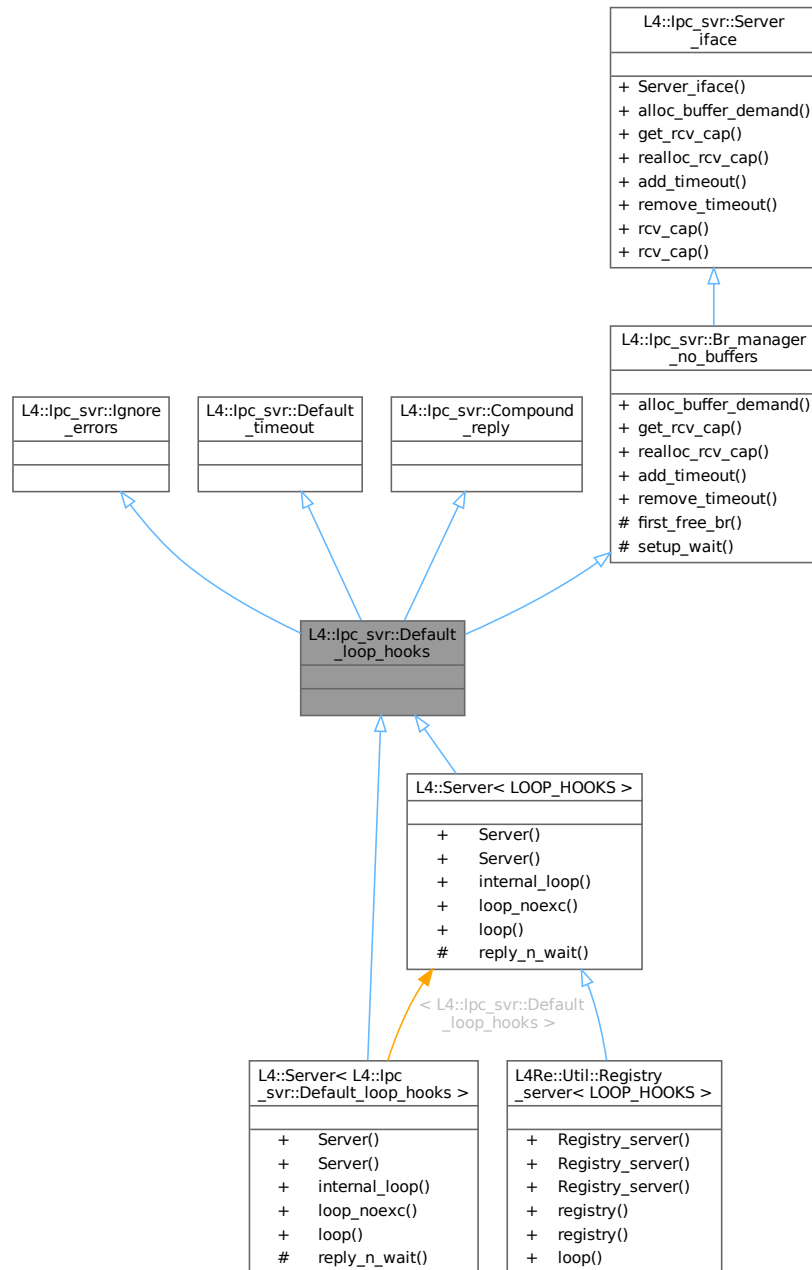
- l4/sys/cxx/ipc\_server\_loop

## 15.158 L4::lpc\_svr::Default\_loop\_hooks Struct Reference

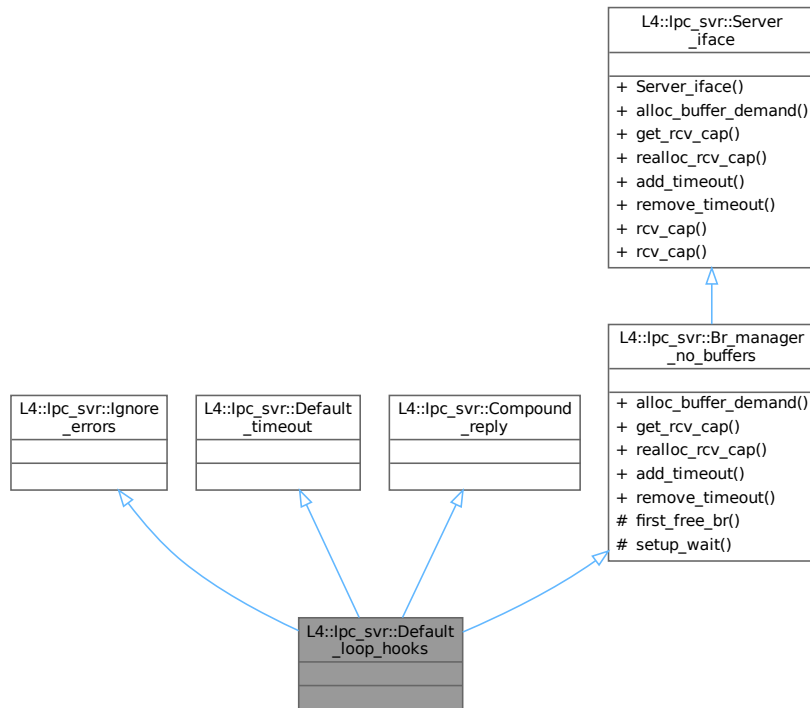
Default LOOP\_HOOKS.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc\_svr::Default\_loop\_hooks:



Collaboration diagram for L4::lpc\_svr::Default\_loop\_hooks:



### Additional Inherited Members

### Public Types inherited from L4::lpc\_svr::Server\_iface

- typedef L4::Type\_info::Demand Demand  
*Data type expressing server-side demand for receive buffers.*

### Public Member Functions inherited from L4::lpc\_svr::Br\_manager\_no\_buffers

- int **alloc\_buffer\_demand** (Demand const &demand) override  
*Tells the server to allocate buffers for the given demand.*
- L4::Cap< void > **get\_rcv\_cap** (int) const override  
*Returns L4::Cap< void >::Invalid, we have no buffer management.*
- int **realloc\_rcv\_cap** (int) override  
*Returns -L4\_ENOMEM, we have no buffer management.*
- int **add\_timeout** (Timeout \*, l4\_kernel\_clock\_t) override  
*Returns -L4\_ENOSYS, we have no timeout queue.*
- int **remove\_timeout** (Timeout \*) override  
*Returns -L4\_ENOSYS, we have no timeout queue.*

## Public Member Functions inherited from [L4::lpc\\_svr::Server\\_iface](#)

- **Server\_iface** ()  
*Make a server interface.*
- `template<typename T >`  
[L4::Cap](#)< T > [rcv\\_cap](#) (int index) const  
*Get given receive buffer as typed capability.*
- [L4::Cap](#)< void > [rcv\\_cap](#) (int index) const  
*Get receive cap with the given index as generic (void) type.*

## Protected Member Functions inherited from [L4::lpc\\_svr::Br\\_manager\\_no\\_buffers](#)

- unsigned **first\_free\_br** () const  
*Returns 1 as first free buffer.*
- void **setup\_wait** ([l4\\_utcb\\_t](#) \*utcb, [L4::lpc\\_svr::Reply\\_mode](#))  
*Setup wait function for the server loop (Server<>).*

### 15.158.1 Detailed Description

Default LOOP\_HOOKS.

Combination of [Ignore\\_errors](#), [Default\\_timeout](#), [Compound\\_reply](#), and [Br\\_manager\\_no\\_buffers](#).

Definition at line 236 of file [ipc\\_server\\_loop](#).

The documentation for this struct was generated from the following file:

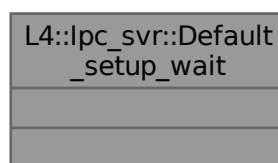
- `l4/sys/cxx/ipc_server_loop`

## 15.159 [L4::lpc\\_svr::Default\\_setup\\_wait](#) Struct Reference

Mix in for LOOP\_HOOKS for setup\_wait no op.

```
#include <ipc_server_loop>
```

Collaboration diagram for [L4::lpc\\_svr::Default\\_setup\\_wait](#):





### 15.159.1 Detailed Description

Mix in for LOOP\_HOOKS for setup\_wait no op.

Definition at line 88 of file [ipc\\_server\\_loop](#).

The documentation for this struct was generated from the following file:

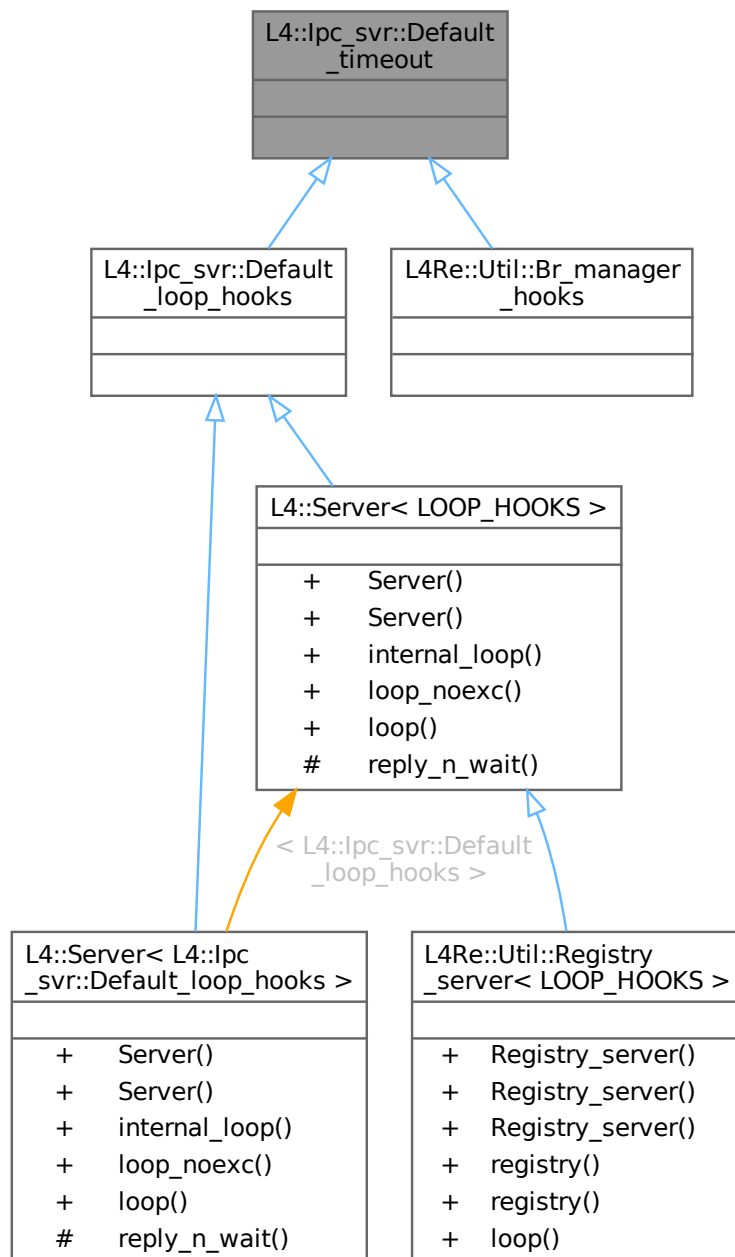
- l4/sys/cxx/ipc\_server\_loop

## 15.160 L4::lpc\_svr::Default\_timeout Struct Reference

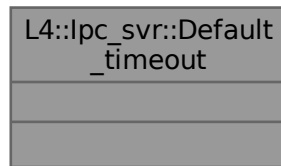
Mix in for LOOP\_HOOKS to use a 0 send and a infinite receive timeout.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc\_svr::Default\_timeout:



Collaboration diagram for L4::lpc\_svr::Default\_timeout:



### 15.160.1 Detailed Description

Mix in for LOOP\_HOOKS to use a 0 send and a infinite receive timeout.

Definition at line 69 of file [ipc\\_server\\_loop](#).

The documentation for this struct was generated from the following file:

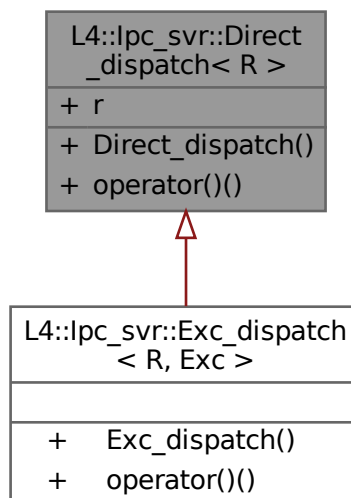
- l4/sys/cxx/ipc\_server\_loop

## 15.161 L4::lpc\_svr::Direct\_dispatch< R > Struct Template Reference

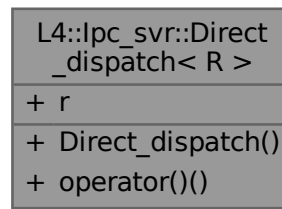
Direct dispatch helper, for forwarding dispatch calls to a registry *R*.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc\_svr::Direct\_dispatch< R >:



Collaboration diagram for L4::lpc\_svr::Direct\_dispatch< R >:



### Public Member Functions

- **Direct\_dispatch** (R &r)  
*Make a direct dispatcher.*
- **l4\_msgtag\_t operator()** (l4\_msgtag\_t tag, l4\_umword\_t obj, l4\_utcb\_t \*utcb)  
*call operator forwarding to r.dispatch()*

### Data Fields

- **R & r**  
*stores a reference to the registry object*

## 15.161.1 Detailed Description

```
template<typename R>
struct L4::lpc_svr::Direct_dispatch< R >
```

Direct dispatch helper, for forwarding dispatch calls to a registry *R*.

#### Template Parameters

<i>R</i>	Data type of the registry that is used for dispatching to different server objects, usually based on the protected IPC label.
----------	---

Definition at line 99 of file [ipc\\_server\\_loop](#).

The documentation for this struct was generated from the following file:

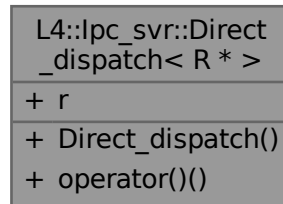
- l4/sys/cxx/ipc\_server\_loop

## 15.162 L4::lpc\_svr::Direct\_dispatch< R \* > Struct Template Reference

Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry *R*.

```
#include <ipc_server_loop>
```

Collaboration diagram for L4::lpc\_svr::Direct\_dispatch< R \* >:



### Public Member Functions

- **Direct\_dispatch** (R \*r)  
*Make a direct dispatcher.*
- **l4\_msgtag\_t operator()** (l4\_msgtag\_t tag, l4\_umword\_t obj, l4\_utcb\_t \*utcb)  
*call operator forwarding to r->dispatch()*

### Data Fields

- **R \* r**  
*stores a pointer to the registry object*

### 15.162.1 Detailed Description

```
template<typename R>
struct L4::lpc_svr::Direct_dispatch< R * >
```

Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry *R*.

#### Template Parameters

<i>R</i>	Data type of the registry that is used for dispatching to different server objects, usually based on the protected IPC label.
----------	---

Definition at line 120 of file [ipc\\_server\\_loop](#).

The documentation for this struct was generated from the following file:

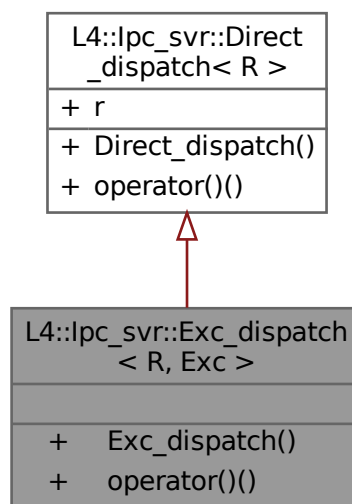
- l4/sys/cxx/ipc\_server\_loop

### 15.163 L4::lpc\_svr::Exc\_dispatch< R, Exc > Struct Template Reference

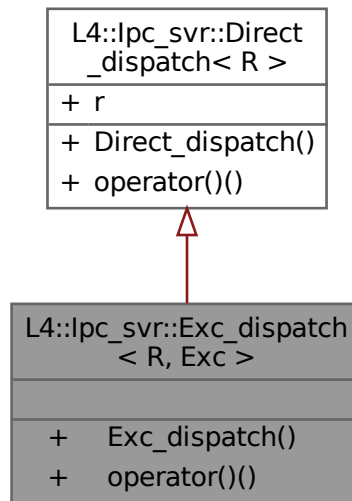
Dispatch helper wrapping try {} catch {} around the dispatch call.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc\_svr::Exc\_dispatch< R, Exc >:



Collaboration diagram for L4::lpc\_svr::Exc\_dispatch< R, Exc >:



## Public Member Functions

- **Exc\_dispatch** (R r)  
*Make an exception handling dispatcher.*
- **l4\_msgtag\_t operator()** (l4\_msgtag\_t tag, l4\_umword\_t obj, l4\_utcb\_t \*utcb)  
*Dispatch the call via Direct\_dispatch<R>() and handle exceptions.*

### 15.163.1 Detailed Description

```
template<typename R, typename Exc>
struct L4::lpc_svr::Exc_dispatch< R, Exc >
```

Dispatch helper wrapping try {} catch {} around the dispatch call.

#### Template Parameters

<i>R</i>	Data type of the registry used for dispatching to objects.
<i>Exc</i>	Data type of the exceptions that shall be caught. This data type must provide a member <code>err_no()</code> that returns the negative integer (int) error code for the exception.

This dispatcher wraps `Direct_dispatch<R>` with a try-catch (`Exc`).

Definition at line 144 of file [ipc\\_server\\_loop](#).

The documentation for this struct was generated from the following file:

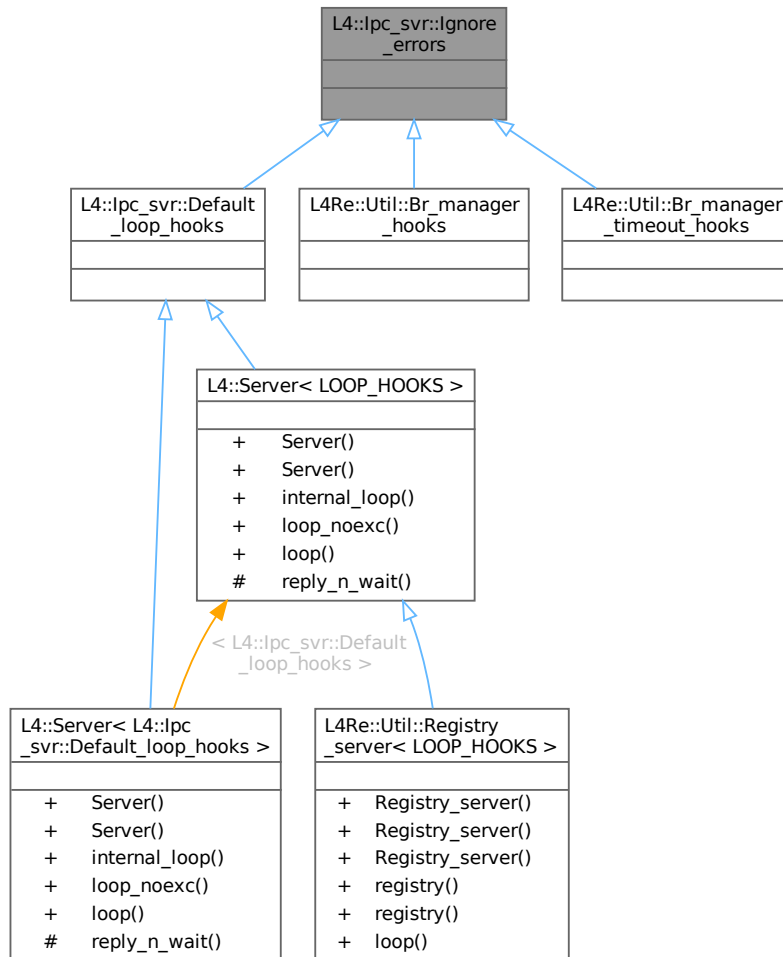
- `l4/sys/cxx/ipc_server_loop`

## 15.164 L4::lpc\_svr::Ignore\_errors Struct Reference

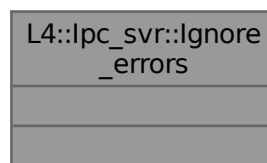
Mix in for LOOP\_HOOKS to ignore IPC errors.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc\_svr::Ignore\_errors:



Collaboration diagram for L4::lpc\_svr::Ignore\_errors:





### 15.164.1 Detailed Description

Mix in for LOOP\_HOOKS to ignore IPC errors.

Definition at line 61 of file [ipc\\_server\\_loop](#).

The documentation for this struct was generated from the following file:

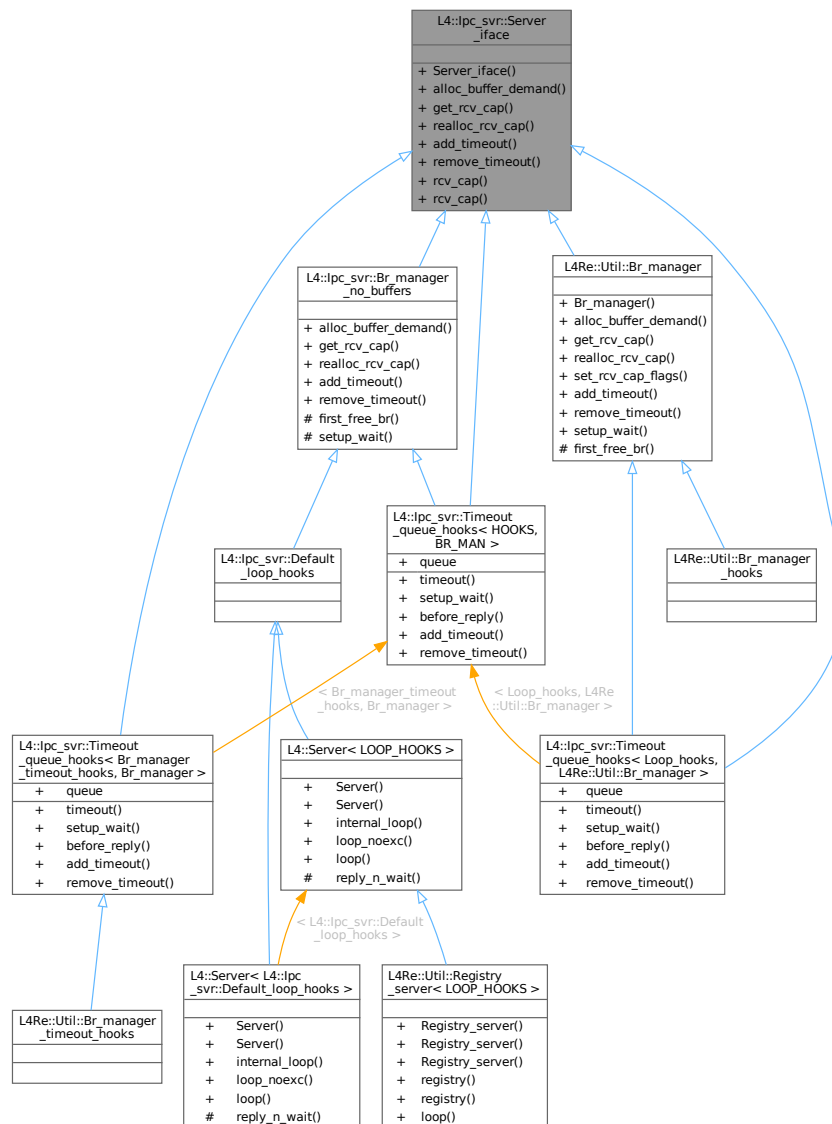
- l4/sys/cxx/ipc\_server\_loop

## 15.165 L4::lpc\_svr::Server\_iface Class Reference

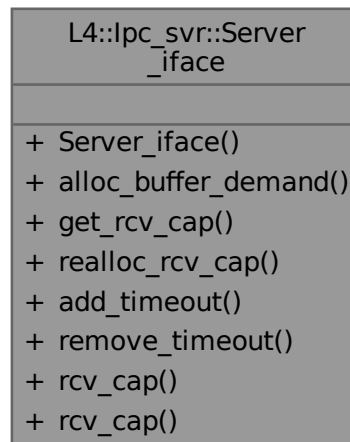
Interface for server-loop related functions.

```
#include <ipc_epiface>
```

Inheritance diagram for L4::lpc\_svr::Server\_iface:



Collaboration diagram for L4::lpc\_svr::Server\_iface:



## Public Types

- typedef [L4::Type\\_info::Demand](#) **Demand**  
*Data type expressing server-side demand for receive buffers.*

## Public Member Functions

- **Server\_iface ()**  
*Make a server interface.*
- virtual int [alloc\\_buffer\\_demand](#) ([Demand](#) const &demand)=0  
*Tells the server to allocate buffers for the given demand.*
- virtual [L4::Cap](#)< void > [get\\_rcv\\_cap](#) (int index) const =0  
*Get capability slot allocated to the given receive buffer.*
- virtual int [realloc\\_rcv\\_cap](#) (int index)=0  
*Allocate a new capability for the given receive buffer.*
- virtual int [add\\_timeout](#) ([Timeout](#) \*timeout, [l4\\_kernel\\_clock\\_t](#) time)=0  
*Add a timeout to the server internal timeout queue.*
- virtual int [remove\\_timeout](#) ([Timeout](#) \*timeout)=0  
*Remove the given timeout from the timer queue.*
- template<typename T >  
[L4::Cap](#)< T > [rcv\\_cap](#) (int index) const  
*Get given receive buffer as typed capability.*
- [L4::Cap](#)< void > [rcv\\_cap](#) (int index) const  
*Get receive cap with the given index as generic (void) type.*

### 15.165.1 Detailed Description

Interface for server-loop related functions.

This interface provides access to high-level server-loop related functions, such as management of receive buffers and timeouts.

Definition at line 47 of file [ipc\\_epiface](#).

### 15.165.2 Member Function Documentation

#### 15.165.2.1 add\_timeout()

```
virtual int L4::Ipc_svr::Server_iface::add_timeout (
    Timeout * timeout,
    l4_kernel_clock_t time ) [pure virtual]
```

Add a timeout to the server internal timeout queue.

##### Parameters

<i>timeout</i>	The timeout object to register.
<i>time</i>	The time (absolute) at which the timeout shall expire.

##### Precondition

*timeout* must not be in any queue.

##### Returns

0 on success, 1 if timeout is already expired, < 0 on error.

Implemented in [L4Re::Util::Br\\_manager](#), [L4::lpc\\_svr::Br\\_manager\\_no\\_buffers](#), [L4::lpc\\_svr::Timeout\\_queue\\_hooks<HOOKS, BR\\_M](#), [L4::lpc\\_svr::Timeout\\_queue\\_hooks<Br\\_manager\\_timeout\\_hooks, Br\\_manager>](#), and [L4::lpc\\_svr::Timeout\\_queue\\_hooks<Loop](#)

#### 15.165.2.2 alloc\_buffer\_demand()

```
virtual int L4::Ipc_svr::Server_iface::alloc_buffer_demand (
    Demand const & demand ) [pure virtual]
```

Tells the server to allocate buffers for the given demand.

##### Parameters

<i>demand</i>	The total server-side demand of receive buffers needed for a given interface, see Demand.
---------------	---

This function is not called by user applications directly. Usually the server implementation or the registry implementation calls this function whenever a new object is registered at the server.

Implemented in [L4Re::Util::Br\\_manager](#), and [L4::lpc\\_svr::Br\\_manager\\_no\\_buffers](#).

### 15.165.2.3 get\_rcv\_cap()

```
virtual L4::Cap< void > L4::Ipc_svr::Server_iface::get_rcv_cap (
    int index ) const [pure virtual]
```

Get capability slot allocated to the given receive buffer.

#### Parameters

<i>index</i>	The receive buffer index of the expected capability argument ( $0 \leq \text{index} < \text{caps}$ registered with <a href="#">alloc_buffer_demand()</a> ).
--------------	---

#### Precondition

$0 \leq \text{index} < \text{caps}$  registered with [alloc\\_buffer\\_demand\(\)](#)

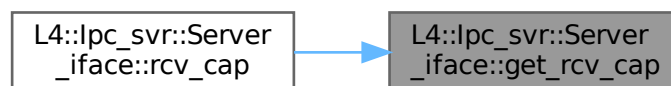
#### Returns

Capability slot currently allocated to the given receive buffer.

Implemented in [L4Re::Util::Br\\_manager](#), and [L4::lpc\\_svr::Br\\_manager\\_no\\_buffers](#).

Referenced by [rcv\\_cap\(\)](#).

Here is the caller graph for this function:



### 15.165.2.4 rcv\_cap() [1/2]

```
template<typename T >
L4::Cap< T > L4::Ipc_svr::Server_iface::rcv_cap (
    int index ) const [inline]
```

Get given receive buffer as typed capability.

#### See also

[get\\_rcv\\_cap\(\)](#)

## Parameters

<i>index</i>	The receive buffer index of the expected capability argument. ( $0 \leq \text{index} < \text{caps}$ registered with <a href="#">alloc_buffer_demand()</a> .)
--------------	--

## Precondition

$0 \leq \text{index} < \text{caps}$  registered with [alloc\\_buffer\\_demand\(\)](#)

## Returns

Capability slot currently allocated to the given receive buffer.

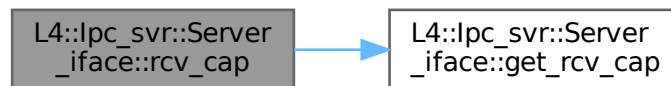
## Note

This is a convenience wrapper for [get\\_rcv\\_cap\(\)](#) to avoid [L4::cap\\_cast<>\(\)](#).

Definition at line 125 of file [ipc\\_epiface](#).

References [get\\_rcv\\_cap\(\)](#).

Here is the call graph for this function:



## 15.165.2.5 rcv\_cap() [2/2]

```

L4::Cap< void > L4::lpc_svr::Server_iface::rcv_cap (
    int index ) const [inline]
  
```

Get receive cap with the given index as generic (void) type.

## Parameters

<i>index</i>	The index of the cap receive buffer of the expected capability. ( $0 \leq \text{index} < \text{caps}$ registered with <a href="#">alloc_buffer_demand()</a> .)
--------------	--

## Returns

Capability slot currently allocated to the given capability buffer.

**Note**

This is a convenience wrapper for [get\\_rcv\\_cap\(\)](#).

Definition at line 137 of file [ipc\\_epiface](#).

References [get\\_rcv\\_cap\(\)](#).

Here is the call graph for this function:

**15.165.2.6 realloc\_rcv\_cap()**

```
virtual int L4::Ipc_svr::Server_iface::realloc_rcv_cap (
    int index ) [pure virtual]
```

Allocate a new capability for the given receive buffer.

**Parameters**

<i>index</i>	The receive buffer index of the expected capability argument ( $0 \leq \text{index} < \text{caps}$ registered with <a href="#">alloc_buffer_demand()</a> ).
--------------	---

**Precondition**

$0 \leq \text{index} < \text{caps}$  registered with [alloc\\_buffer\\_demand\(\)](#)

**Returns**

0 on success, < 0 on error.

Implemented in [L4Re::Util::Br\\_manager](#), and [L4::lpc\\_svr::Br\\_manager\\_no\\_buffers](#).

**15.165.2.7 remove\_timeout()**

```
virtual int L4::Ipc_svr::Server_iface::remove_timeout (
    Timeout * timeout ) [pure virtual]
```

Remove the given timeout from the timer queue.

## Parameters

<i>timeout</i>	The timeout object to remove.
----------------	-------------------------------

## Returns

0 on success, < 0 on error.

Implemented in [L4Re::Util::Br\\_manager](#), [L4::lpc\\_svr::Br\\_manager\\_no\\_buffers](#), [L4::lpc\\_svr::Timeout\\_queue\\_hooks< HOOKS, BR\\_M](#), [L4::lpc\\_svr::Timeout\\_queue\\_hooks< Br\\_manager\\_timeout\\_hooks, Br\\_manager >](#), and [L4::lpc\\_svr::Timeout\\_queue\\_hooks< Loop\\_](#)

The documentation for this class was generated from the following file:

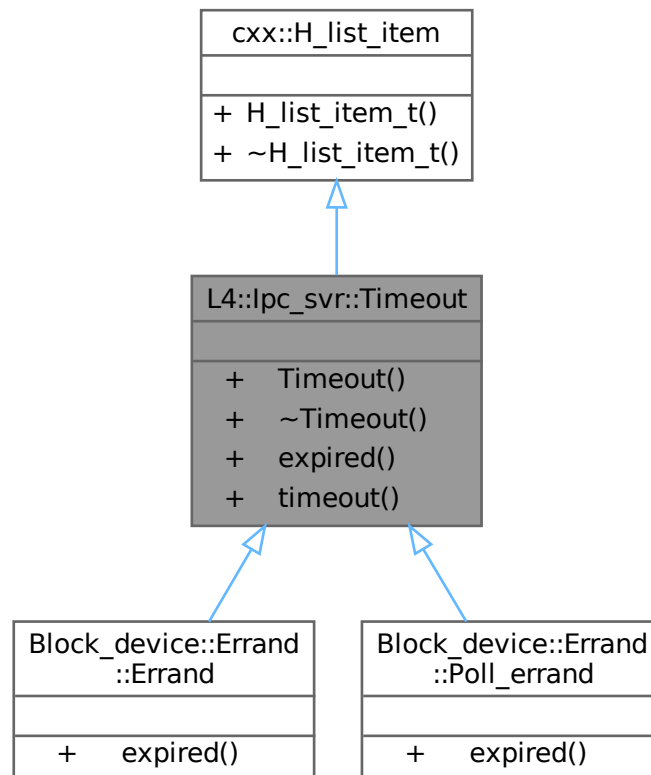
- [l4/sys/cxx/ipc\\_epiface](#)

## 15.166 L4::lpc\_svr::Timeout Class Reference

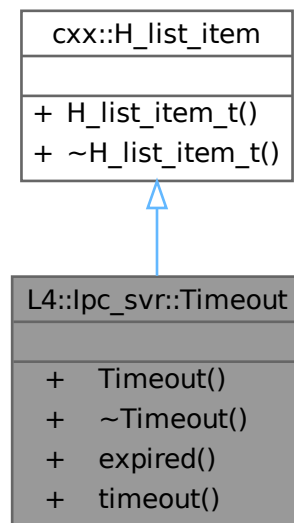
Callback interface for [Timeout\\_queue](#).

```
#include <ipc_timeout_queue>
```

Inheritance diagram for L4::lpc\_svr::Timeout:



Collaboration diagram for L4::lpc\_svr::Timeout:



### Public Member Functions

- **Timeout ()**  
*Make a timeout.*
- virtual **~Timeout ()=0**  
*Destroy a timeout.*
- virtual void **expired ()=0**  
*callback function to be called when timeout happened*
- **l4\_kernel\_clock\_t timeout () const**  
*return absolute timeout of this callback.*

### Public Member Functions inherited from **cxx::H\_list\_item\_t< ELEM\_TYPE >**

- **H\_list\_item\_t ()**  
*Constructor.*
- **~H\_list\_item\_t () noexcept**  
*Destructor.*

#### 15.166.1 Detailed Description

Callback interface for [Timeout\\_queue](#).

Definition at line 28 of file [ipc\\_timeout\\_queue](#).



## 15.166.2 Member Function Documentation

### 15.166.2.1 expired()

```
virtual void L4::Ipc_svr::Timeout::expired ( ) [pure virtual]
```

callback function to be called when timeout happened

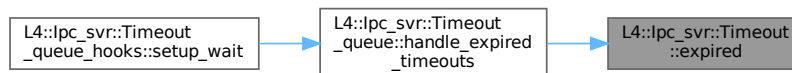
#### Note

The timeout object is already dequeued when this function is called, this means the timeout may be safely queued again within the [expired\(\)](#) function.

Implemented in [Block\\_device::Errand::Poll\\_errand](#), and [Block\\_device::Errand::Errand](#).

Referenced by [L4::lpc\\_svr::Timeout\\_queue::handle\\_expired\\_timeouts\(\)](#).

Here is the caller graph for this function:



### 15.166.2.2 timeout()

```
l4_kernel_clock_t L4::Ipc_svr::Timeout::timeout ( ) const [inline]
```

return absolute timeout of this callback.

#### Returns

absolute timeout for this instance of the timeout.

#### Precondition

The timeout object must have been in a queue before, otherwise the timeout is not set.

Definition at line 52 of file [ipc\\_timeout\\_queue](#).

The documentation for this class was generated from the following file:

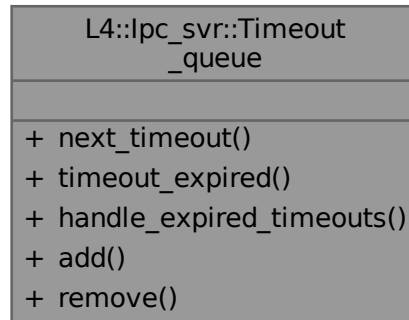
- `I4/cxx/ipc_timeout_queue`

## 15.167 L4::lpc\_svr::Timeout\_queue Class Reference

[Timeout](#) queue to be used in l4re server loop.

```
#include <ipc_timeout_queue>
```

Collaboration diagram for L4::lpc\_svr::Timeout\_queue:



### Public Types

- typedef [L4::lpc\\_svr::Timeout](#) **Timeout**  
Provide a local definition of [Timeout](#) for backward compatibility.

### Public Member Functions

- [l4\\_kernel\\_clock\\_t](#) **next\_timeout** () const  
Get the time for the next timeout.
- bool **timeout\_expired** ([l4\\_kernel\\_clock\\_t](#) now) const  
Determine if a timeout has happened.
- void **handle\_expired\_timeouts** ([l4\\_kernel\\_clock\\_t](#) now)  
run the callbacks of expired timeouts
- void **add** ([Timeout](#) \*timeout, [l4\\_kernel\\_clock\\_t](#) time)  
Add a timeout to the queue.
- void **remove** ([Timeout](#) \*timeout)  
Remove timeout from the queue.

### 15.167.1 Detailed Description

[Timeout](#) queue to be used in l4re server loop.

Definition at line 65 of file [ipc\\_timeout\\_queue](#).

## 15.167.2 Member Function Documentation

### 15.167.2.1 add()

```
void L4::Ipc_svr::Timeout_queue::add (
    Timeout * timeout,
    l4_kernel_clock_t time ) [inline]
```

Add a timeout to the queue.

#### Parameters

<i>timeout</i>	timeout object to add
<i>time</i>	the time when the timeout expires

#### Precondition

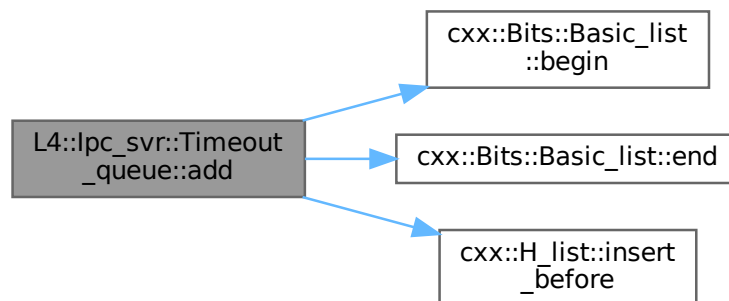
*timeout* must not be in any queue already

Definition at line 121 of file [ipc\\_timeout\\_queue](#).

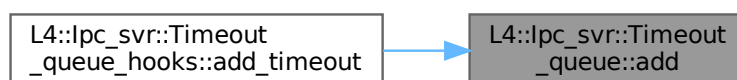
References [cxx::Bits::Basic\\_list< POLICY >::begin\(\)](#), [cxx::Bits::Basic\\_list< POLICY >::end\(\)](#), and [cxx::H\\_list< T, POLICY >::insert\\_](#)

Referenced by [L4::lpc\\_svr::Timeout\\_queue\\_hooks< HOOKS, BR\\_MAN >::add\\_timeout\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.167.2.2 handle\_expired\_timeouts()

```
void L4::Ipc_svr::Timeout_queue::handle_expired_timeouts (
    l4_kernel_clock_t now ) [inline]
```

run the callbacks of expired timeouts

#### Parameters

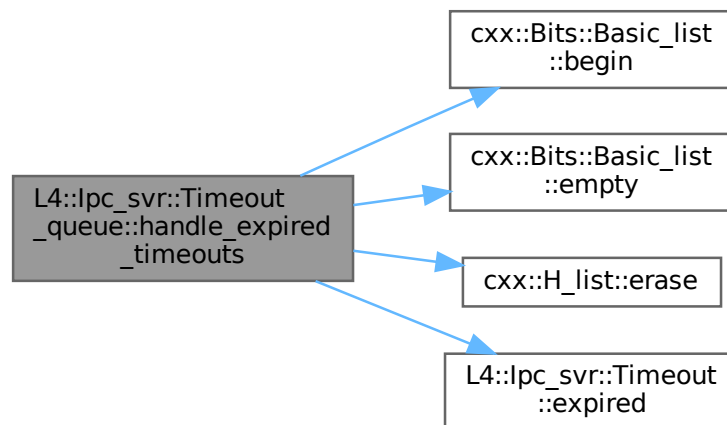
<i>now</i>	the current time.
------------	-------------------

Definition at line 101 of file [ipc\\_timeout\\_queue](#).

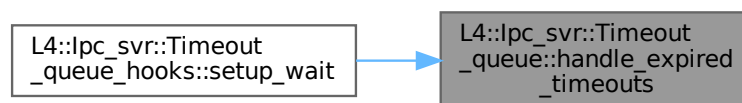
References [cxx::Bits::Basic\\_list< POLICY >::begin\(\)](#), [cxx::Bits::Basic\\_list< POLICY >::empty\(\)](#), [cxx::H\\_list< T, POLICY >::erase\(\)](#), and [L4::lpc\\_svr::Timeout::expired\(\)](#).

Referenced by [L4::lpc\\_svr::Timeout\\_queue\\_hooks< HOOKS, BR\\_MAN >::setup\\_wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.167.2.3 next\_timeout()

```
l4_kernel_clock_t L4::Ipc_svr::Timeout_queue::next_timeout ( ) const [inline]
```

Get the time for the next timeout.

#### Returns

the time for the next timeout or 0 if there is none

Definition at line 75 of file [ipc\\_timeout\\_queue](#).

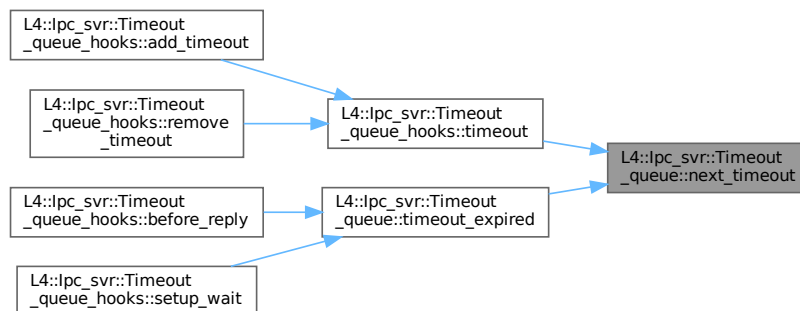
References [cxx::Bits::Basic\\_list< POLICY >::front\(\)](#).

Referenced by [L4::lpc\\_svr::Timeout\\_queue\\_hooks< HOOKS, BR\\_MAN >::timeout\(\)](#), and [timeout\\_expired\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.167.2.4 remove()

```
void L4::Ipc_svr::Timeout_queue::remove (
    Timeout * timeout ) [inline]
```

Remove *timeout* from the queue.

## Parameters

<i>timeout</i>	timeout to remove from timeout queue
----------------	--------------------------------------

## Precondition

*timeout* must be in this queue

Definition at line 136 of file `ipc_timeout_queue`.

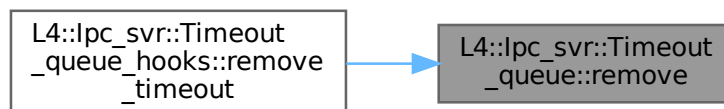
References `cxx::H_list< T, POLICY >::remove()`.

Referenced by `L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::remove_timeout()`.

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.167.2.5 timeout\_expired()

```
bool L4::Ipc_svr::Timeout_queue::timeout_expired (
    l4_kernel_clock_t now ) const [inline]
```

Determine if a timeout has happened.

## Parameters

<i>now</i>	The current time.
------------	-------------------

## Return values

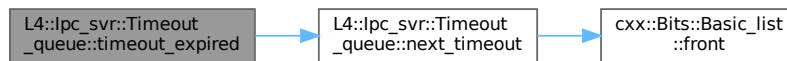
<code>true</code>	There is at least one expired timeout in the queue.
<code>false</code>	No expired timeout in the queue.

Definition at line 91 of file [ipc\\_timeout\\_queue](#).

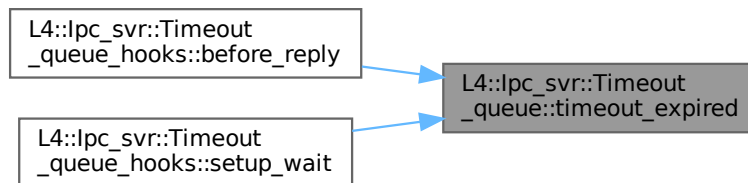
References [next\\_timeout\(\)](#).

Referenced by [L4::lpc\\_svr::Timeout\\_queue\\_hooks< HOOKS, BR\\_MAN >::before\\_reply\(\)](#), and [L4::lpc\\_svr::Timeout\\_queue\\_hooks<](#)

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

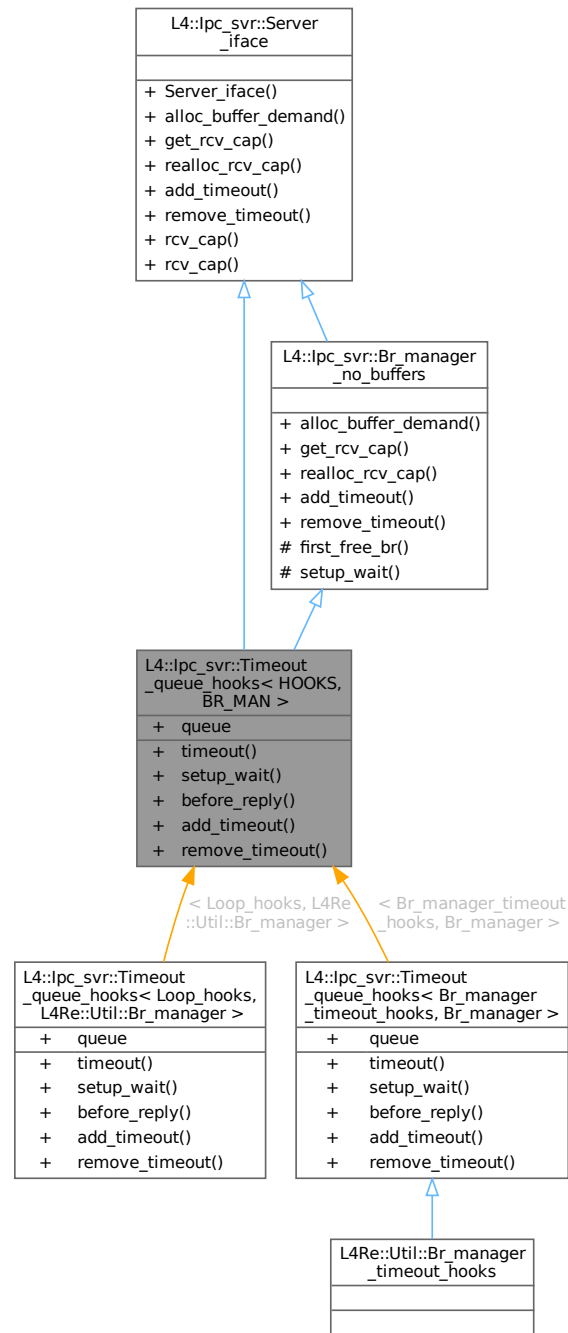
- `I4/cxx/ipc_timeout_queue`

## 15.168 L4::lpc\_svr::Timeout\_queue\_hooks< HOOKS, BR\_MAN > Class Template Reference

Loop hooks mixin for integrating a timeout queue into the server loop.

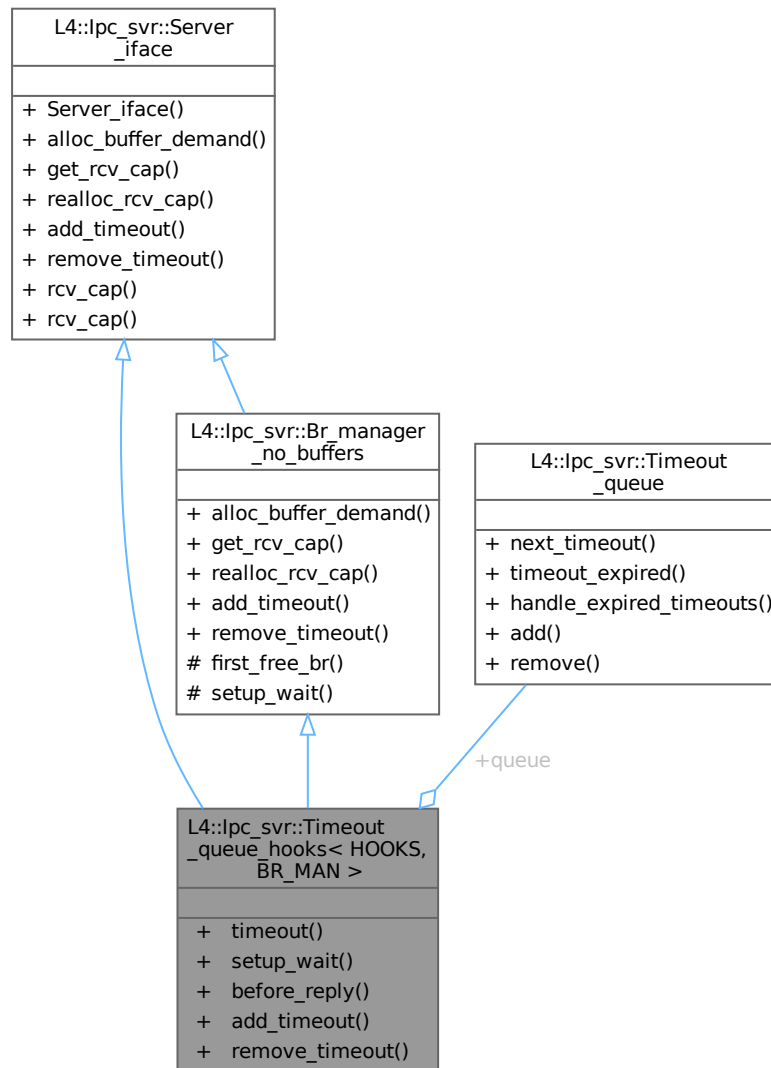
```
#include <ipc_timeout_queue>
```

Inheritance diagram for L4::lpc\_svr::Timeout\_queue\_hooks< HOOKS, BR\_MAN >:





Collaboration diagram for L4::lpc\_svr::Timeout\_queue\_hooks< HOOKS, BR\_MAN >:



## Public Member Functions

- `l4_timeout_t timeout ()`  
*get the time for the next timeout*
- `void setup_wait (l4_utcb_t *utcb, L4::lpc_svr::Reply_mode mode)`  
*setup\_wait() for the server loop*
- `L4::lpc_svr::Reply_mode before_reply (l4_msgtag_t, l4_utcb_t *)`  
*server loop hook*
- `int add_timeout (Timeout *timeout, l4_kernel_clock_t time) override`  
*Add a timeout to the queue for time time.*
- `int remove_timeout (Timeout *timeout) override`  
*Remove timeout from the queue.*

## Public Member Functions inherited from [L4::lpc\\_svr::Server\\_iface](#)

- **Server\_iface** ()  
*Make a server interface.*
- `template<typename T >`  
[L4::Cap](#)< T > [rcv\\_cap](#) (int index) const  
*Get given receive buffer as typed capability.*
- [L4::Cap](#)< void > [rcv\\_cap](#) (int index) const  
*Get receive cap with the given index as generic (void) type.*

## Public Member Functions inherited from [L4::lpc\\_svr::Br\\_manager\\_no\\_buffers](#)

- int [alloc\\_buffer\\_demand](#) ([Demand](#) const &demand) override  
*Tells the server to allocate buffers for the given demand.*
- [L4::Cap](#)< void > [get\\_rcv\\_cap](#) (int) const override  
*Returns [L4::Cap](#)<void>::Invalid, we have no buffer management.*
- int [realloc\\_rcv\\_cap](#) (int) override  
*Returns -L4\_ENOMEM, we have no buffer management.*

## Data Fields

- [Timeout\\_queue](#) queue  
*Use this timeout queue.*

## Additional Inherited Members

## Public Types inherited from [L4::lpc\\_svr::Server\\_iface](#)

- typedef [L4::Type\\_info::Demand](#) Demand  
*Data type expressing server-side demand for receive buffers.*

## Protected Member Functions inherited from [L4::lpc\\_svr::Br\\_manager\\_no\\_buffers](#)

- unsigned [first\\_free\\_br](#) () const  
*Returns 1 as first free buffer.*
- void [setup\\_wait](#) ([l4\\_utcb\\_t](#) \*utcb, [L4::lpc\\_svr::Reply\\_mode](#))  
*Setup wait function for the server loop (Server<>).*

### 15.168.1 Detailed Description

```
template<typename HOOKS, typename BR_MAN = Br_manager_no_buffers>
class L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >
```

Loop hooks mixin for integrating a timeout queue into the server loop.

## Template Parameters

<i>HOOKS</i>	has to inherit from Timeout_queue_hooks<> and provide the functions now() that has to return the current time.
<i>BR_MAN</i>	This used as a base class for and provides the API for selecting the buffer register (BR) that is used to store the timeout value. This is usually <a href="#">L4Re::Util::Br_manager</a> or <a href="#">L4::lpc_svr::Br_manager_no_buffers</a> .

Definition at line 160 of file [ipc\\_timeout\\_queue](#).

## 15.168.2 Member Function Documentation

### 15.168.2.1 add\_timeout()

```
template<typename HOOKS , typename BR_MAN = Br_manager_no_buffers>
int L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::add_timeout (
    Timeout * timeout,
    l4_kernel_clock_t time ) [inline], [override], [virtual]
```

Add a timeout to the queue for time *time*.

## Parameters

<i>timeout</i>	The timeout object to add into the queue (must not be in any queue currently).
<i>time</i>	The time when the timeout shall expire.

## Precondition

*timeout* must not be in any queue.

## Note

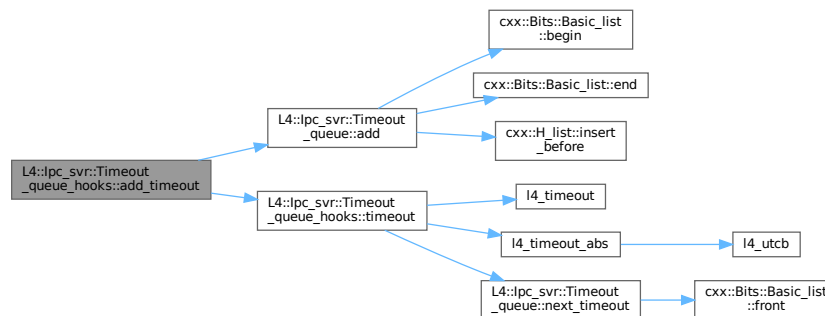
The timeout is automatically dequeued before the [Timeout::expired\(\)](#) function is called

Implements [L4::lpc\\_svr::Server\\_iface](#).

Definition at line 212 of file [ipc\\_timeout\\_queue](#).

References [L4::lpc\\_svr::Timeout\\_queue::add\(\)](#), [L4::lpc\\_svr::Timeout\\_queue\\_hooks< HOOKS, BR\\_MAN >::queue](#), and [L4::lpc\\_svr::Timeout\\_queue\\_hooks< HOOKS, BR\\_MAN >::timeout\(\)](#).

Here is the call graph for this function:



### 15.168.2.2 remove\_timeout()

```

template<typename HOOKS , typename BR_MAN = Br_manager_no_buffers>
int L4::ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::remove_timeout (
    Timeout * timeout ) [inline], [override], [virtual]
  
```

Remove timeout from the queue.

#### Parameters

<i>timeout</i>	The timeout object to be removed from the queue.
----------------	--

#### Note

This function may be safely called even if the timeout is not currently enqueued.

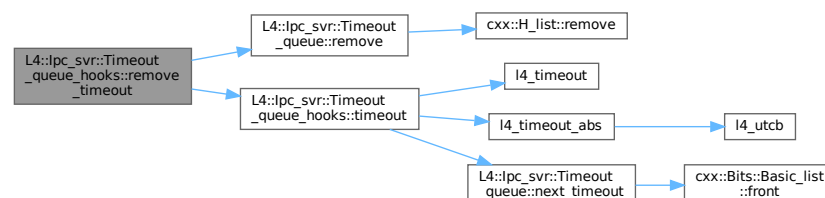
in [Timeout::expired\(\)](#) the timeout is already dequeued!

Implements [L4::ipc\\_svr::Server\\_iface](#).

Definition at line 225 of file [ipc\\_timeout\\_queue](#).

References [L4::ipc\\_svr::Timeout\\_queue\\_hooks< HOOKS, BR\\_MAN >::queue](#), [L4::ipc\\_svr::Timeout\\_queue::remove\(\)](#), and [L4::ipc\\_svr::Timeout\\_queue\\_hooks< HOOKS, BR\\_MAN >::timeout\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

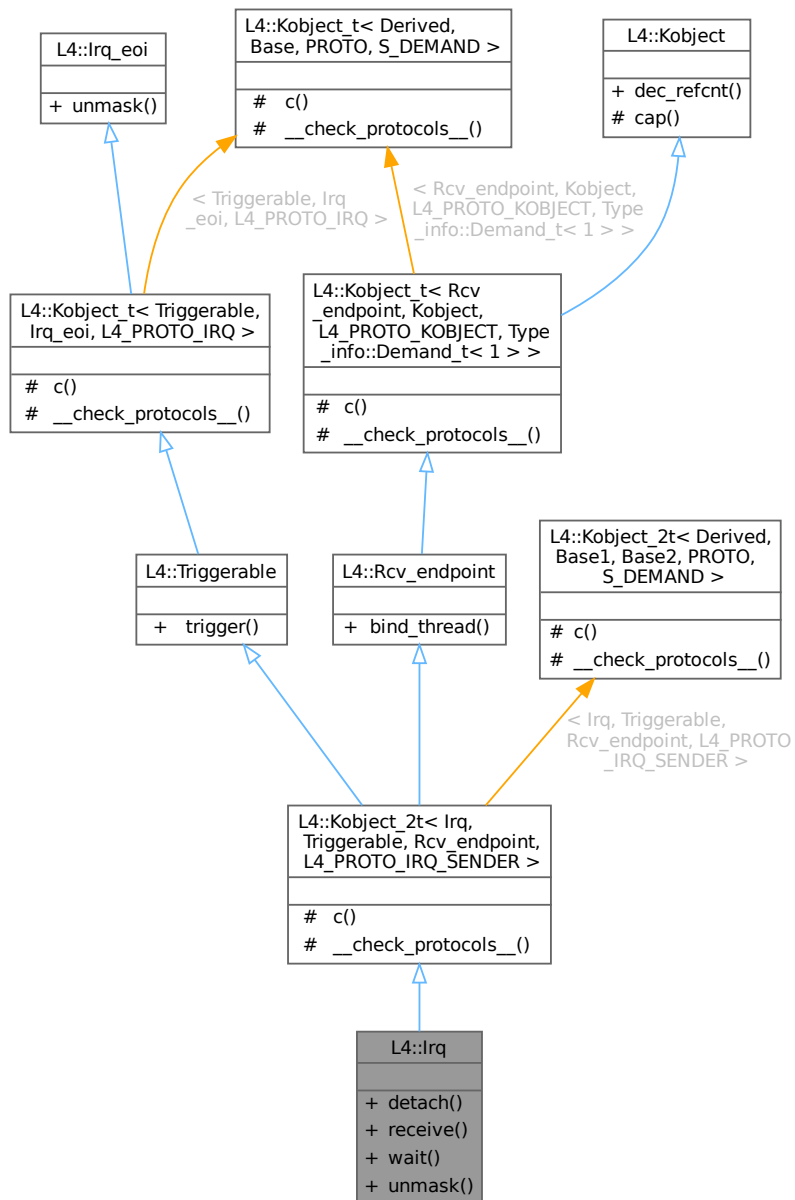
- [I4/cxx/ipc\\_timeout\\_queue](#)

## 15.169 L4::Irq Class Reference

C++ [Irq](#) interface, see [IRQs](#) for the C interface.

```
#include <irq>
```

Inheritance diagram for L4::Irq:





## Public Member Functions inherited from [L4::Triggerable](#)

- [l4\\_msgtag\\_t trigger](#) ([l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Trigger the object.*

## Public Member Functions inherited from [L4::Irq\\_eoi](#)

- [l4\\_msgtag\\_t unmask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Unmask the given interrupt line.*

## Public Member Functions inherited from [L4::Rcv\\_endpoint](#)

- [l4\\_msgtag\\_t bind\\_thread](#) ([lpc::Cap](#)< [Thread](#) > t, [l4\\_umword\\_t](#) label)  
*Bind a thread to an IPC receive endpoint.*

## Public Member Functions inherited from [L4::Kobject](#)

- [l4\\_msgtag\\_t dec\\_refcnt](#) ([l4\\_mword\\_t](#) diff, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#))  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

## Protected Types inherited from

### [L4::Kobject\\_2t](#)< [Irq](#), [Triggerable](#), [Rcv\\_endpoint](#), [L4\\_PROTO\\_IRQ\\_SENDER](#) >

- typedef [Irq Class](#)  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef [Typeid::Iface](#)< [PROTO](#), [Irq](#) > [\\_\\_Iface](#)  
*The interface description for the derived class.*
- typedef [Typeid::Merge\\_list](#)< [Typeid::Iface\\_list](#)< [\\_\\_Iface](#) >, [Typeid::Merge\\_list](#)< typename [Base1::\\_\\_Iface](#)<\_\_  
\_list, typename [Base2::\\_\\_Iface\\_list](#) > > [\\_\\_Iface\\_list](#)  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Types inherited from [L4::Kobject\\_t](#)< [Triggerable](#), [Irq\\_eoi](#), [L4\\_PROTO\\_IRQ](#) >

- typedef [Triggerable Class](#)  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef [Typeid::Iface](#)< [PROTO](#), [Triggerable](#) > [\\_\\_Iface](#)  
*The interface description for the derived class.*
- typedef [Typeid::Merge\\_list](#)< [Typeid::Iface\\_list](#)< [\\_\\_Iface](#) >, typename [Base::\\_\\_Iface\\_list](#) > [\\_\\_Iface\\_list](#)  
*The list of all RPC interfaces provided directly or through inheritance.*

**Protected Types inherited from****L4::Kobject\_t**< Rcv\_endpoint, Kobject, L4\_PROTO\_KOBJECT, Type\_info::Demand\_t< 1 > >

- typedef **Rcv\_endpoint** **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, **Rcv\_endpoint** > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

**Protected Member Functions inherited from****L4::Kobject\_2t**< Irq, Triggerable, Rcv\_endpoint, L4\_PROTO\_IRQ\_SENDER >

- **L4::Cap**< **Class** > **c** () const noexcept  
*Get the capability to ourselves.*

**Protected Member Functions inherited from****L4::Kobject\_t**< Triggerable, Irq\_eoi, L4\_PROTO\_IRQ >

- **L4::Cap**< **Class** > **c** () const noexcept  
*Get the capability to ourselves.*

**Protected Member Functions inherited from****L4::Kobject\_t**< Rcv\_endpoint, Kobject, L4\_PROTO\_KOBJECT, Type\_info::Demand\_t< 1 > >

- **L4::Cap**< **Class** > **c** () const noexcept  
*Get the capability to ourselves.*

**Protected Member Functions inherited from [L4::Kobject](#)**

- **l4\_cap\_idx\_t** **cap** () const noexcept  
*Return capability selector.*

**Static Protected Member Functions inherited from****L4::Kobject\_2t**< Irq, Triggerable, Rcv\_endpoint, L4\_PROTO\_IRQ\_SENDER >

- static void **\_\_check\_protocols\_\_** () noexcept  
*Helper to check for protocol conflicts.*

**Static Protected Member Functions inherited from****L4::Kobject\_t**< Triggerable, Irq\_eoi, L4\_PROTO\_IRQ >

- static void **\_\_check\_protocols\_\_** () noexcept  
*Helper to check for protocol conflicts.*



**Static Protected Member Functions inherited from****[L4::Kobject\\_t< Rcv\\_endpoint, Kobject, L4\\_PROTO\\_KOBJECT, Type\\_info::Demand\\_t< 1 > >](#)**

- static void `__check_protocols__()` noexcept  
*Helper to check for protocol conflicts.*

**15.169.1 Detailed Description**

C++ [Irq](#) interface, see [IRQs](#) for the C interface.

**Note**

"IRQ" is short for "interrupt request". This is often used interchangeably for "interrupt"

The [Irq](#) class provides access to abstract interrupts provided by the microkernel. Interrupts may be

- hardware interrupts provided by the platform interrupt controller,
- virtual device interrupts provided by the microkernel's virtual devices (virtual serial or trace buffer) or
- virtual interrupts that can be triggered by user programs (IRQs) via the inherited method [L4::Triggerable::trigger\(\)](#).

For hardware and virtual device interrupts the [Irq](#) object must be bound to an interrupt source, see [L4::Icu](#). To receive interrupts, the [Irq](#) object must be bound to a thread, see [L4::Rcv\\_endpoint](#).

[Irq](#) objects can be created using a factory, see the [L4::Factory](#) API ([L4::Factory::create\(\)](#)).

**Include File**

```
#include <l4/sys/irq>
```

For the C interface refer to the [IRQs](#) API for an overview.

**Examples**

[examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#).

Definition at line 131 of file [irq](#).

**15.169.2 Member Function Documentation****15.169.2.1 detach()**

```
l4\_msgtag\_t L4::Irq::detach (
    l4\_utcb\_t * utcb = l4\_utcb\(\) ) [inline], [noexcept]
```

Detach from this interrupt.

## Parameters

<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
-------------	--

## Returns

Syscall return tag

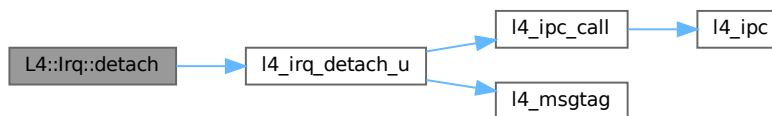
## Return values

<i>0</i>	Successfully detached, there was no interrupt pending.
<i>1</i>	Successfully detached, there was an interrupt pending.
<i>-L4_EPERM</i>	No <a href="#">L4_CAP_FPAGE_S</a> rights on the capability used to invoke this operation.

Definition at line 148 of file [irq](#).

References [l4\\_irq\\_detach\\_u\(\)](#).

Here is the call graph for this function:



## 15.169.2.2 receive()

```

l4_msgtag_t L4::Irq::receive (
    l4_timeout_t timeout = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]

```

Unmask and wait for this IRQ.

## Parameters

<i>timeout</i>	Timeout.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

## Returns

Syscall return tag

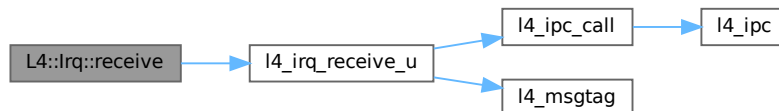
**Note**

If this is the function normally used for your IRQs consider using [L4::Semaphore](#) instead of [L4::Irq](#).

Definition at line 163 of file [irq](#).

References [l4\\_irq\\_receive\\_u\(\)](#).

Here is the call graph for this function:

**15.169.2.3 unmask()**

```

l4_msgtag_t L4::Irq::unmask (
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Unmask this IRQ.

**Parameters**

<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
-------------	--

**Returns**

Syscall return tag for a send-only operation, this means there is no return value except [L4\\_MSGTAG\\_ERROR](#) indicating success or failure of the send operation. Use [l4\\_ipc\\_error\(\)](#) to check for errors and **do not** use [l4\\_error\(\)](#).

[Irq::wait\(\)](#) and [Irq::receive\(\)](#) operations already include an [unmask\(\)](#), do not use an extra [unmask\(\)](#) in these cases.

Definition at line 193 of file [irq](#).

References [L4\\_IPC\\_NEVER](#), and [unmask\(\)](#).

Referenced by [unmask\(\)](#), and [wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 15.169.2.4 wait()

```

l4_msgtag_t L4::Irq::wait (
    l4_umword_t * label,
    l4_timeout_t timeout = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Unmask IRQ and (open) wait for any message.

##### Parameters

<i>label</i>	The <i>protected label</i> shall be received here.
<i>timeout</i>	Timeout.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

##### Returns

Syscall return tag

Definition at line [176](#) of file [irq](#).

References [unmask\(\)](#).

Here is the call graph for this function:

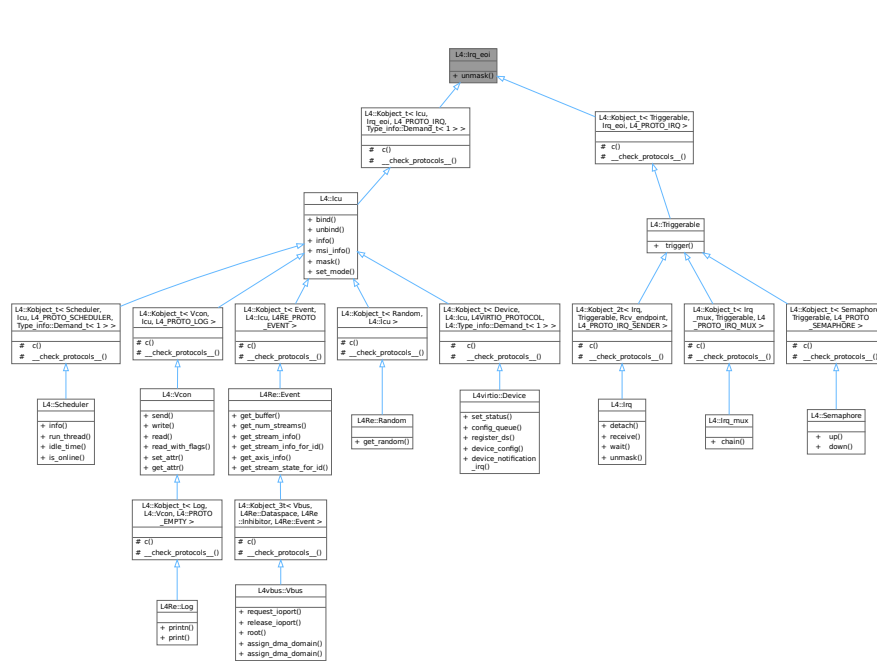


The documentation for this class was generated from the following file:

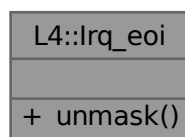
- [l4/sys/irq](#)

Interface for sending an unmask message to an object.

Inheritance diagram for L4::lrq\_eoi:



Collaboration diagram for L4::Irq\_eoi:



- `l4_msgtag_t unmask` (unsigned irqnum, `l4_umword_t` \*label=0, `l4_timeout_t` to=L4\_IPC\_NEVER, `l4_utcb_t` \*utcb=l4\_utcb()) noexcept

*Unmask the given interrupt line.*

### 15.170.1 Detailed Description

Interface for sending an unmask message to an object.

The object is usually an ICU or an IRQ.

When the kernel receives an IRQ, it masks the interrupt line at the interrupt controller and immediately acknowledges the interrupt. This interface is used to let the kernel know that userspace has dealt with the IRQ. The kernel will unmask the interrupt line and further IRQs can then be delivered.

See also

[L4::lcu](#), [L4::irq](#)

Definition at line 48 of file [irq](#).

### 15.170.2 Member Function Documentation

#### 15.170.2.1 unmask()

```
l4_msgtag_t L4::Irq_eoi::unmask (
    unsigned irqnum,
    l4_umword_t * label = 0,
    l4_timeout_t to = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

Unmask the given interrupt line.

When the object is an IRQ, the given interrupt line is ignored and instead the line which the IRQ is bound to (if any) is unmasked.

Its counterpart for explicitly masking an interrupt line is [L4::lcu::mask\(\)](#).

Parameters

	<i>irqnum</i>	The interrupt line that shall be unmasked. Ignored if the object is an IRQ.
out	<i>label</i>	If NULL, this is a send-only unmask. If not NULL, this operation enters an open wait and the <i>protected label</i> shall be received here.
	<i>to</i>	The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non-NULL <i>label</i> only.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

Returns

Syscall return tag. If *label* is NULL, this function performs an IPC send-only operation and there is no return value except [L4\\_MSGTAG\\_ERROR](#) indicating success or failure of the send operation. In this case use [l4\\_ipc\\_error\(\)](#) to check for errors and **do not** use [l4\\_error\(\)](#).

Definition at line 75 of file [irq](#).

The documentation for this class was generated from the following file:

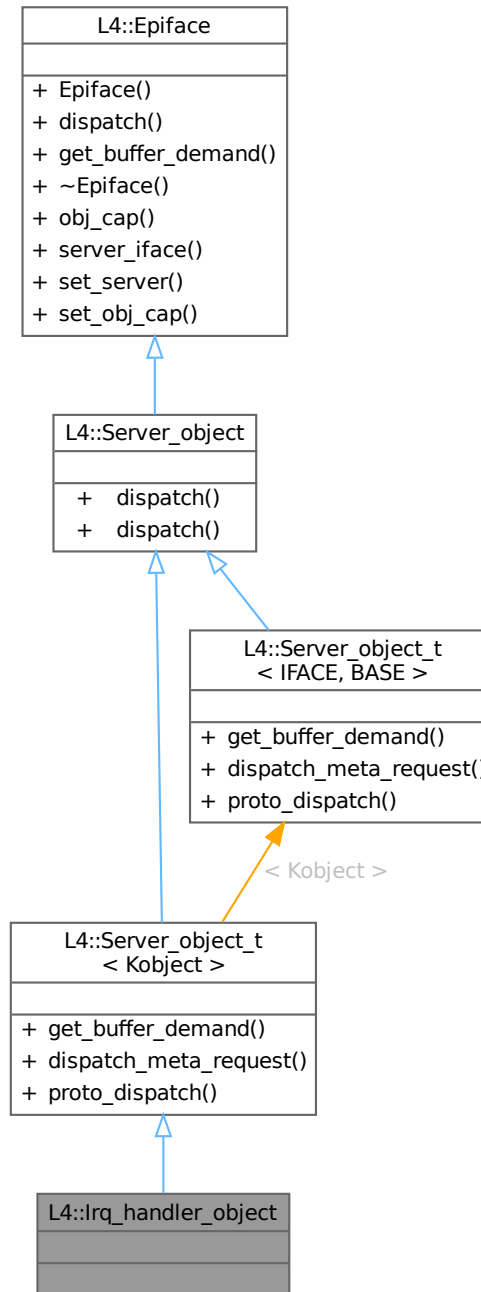
- [l4/sys/irq](#)

## 15.171 L4::lrq\_handler\_object Struct Reference

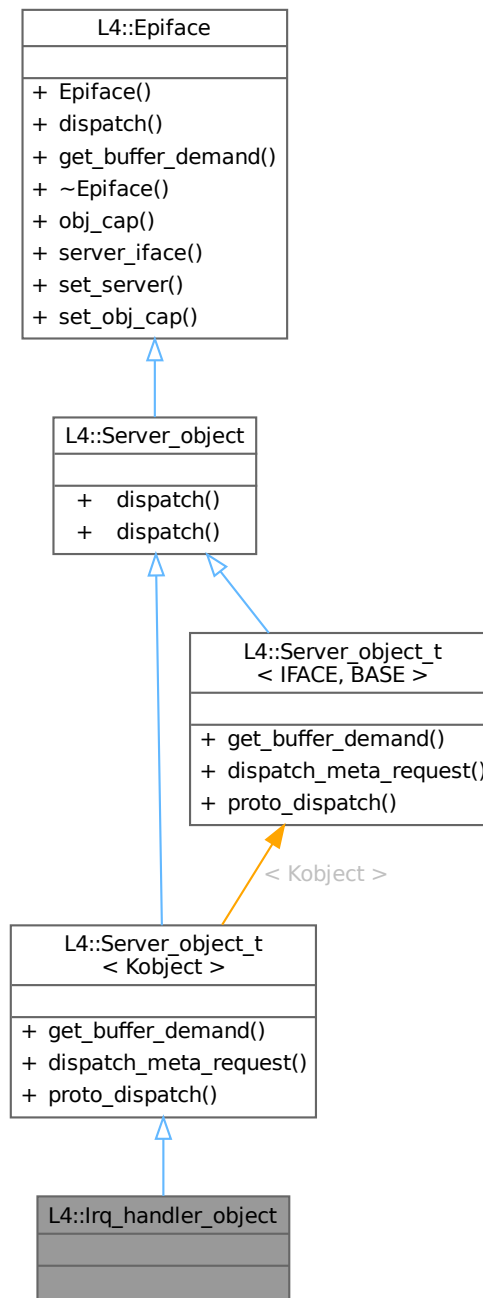
[Server](#) object base class for handling IRQ messages.

```
#include <ipc_server>
```

Inheritance diagram for L4::lrq\_handler\_object:



Collaboration diagram for L4::Irq\_handler\_object:



#### Additional Inherited Members

#### Public Types inherited from **L4::Server\_object\_t< Kobject >**

- typedef **Kobject** Interface

*Data type of the IPC interface definition.*



**Public Types inherited from L4::Epiface**

- typedef [lpc\\_svr::Server\\_iface](#) **Server\_iface**  
*Type for abstract server interface.*
- typedef [lpc\\_svr::Server\\_iface::Demand](#) **Demand**  
*Type for server-side receive buffer demand.*

**Public Member Functions inherited from L4::Server\_object\_t< Kobject >**

- BASE::Demand [get\\_buffer\\_demand](#) () const override
- int [dispatch\\_meta\\_request](#) (L4::lpc::lostream &ios)  
*Implementation of the meta protocol based on IFACE.*

**Public Member Functions inherited from L4::Server\_object**

- virtual int [dispatch](#) (unsigned long rights, [lpc::lostream](#) &ios)=0  
*The abstract handler for client requests to the object.*
- [l4\\_msgtag\\_t](#) [dispatch](#) ([l4\\_msgtag\\_t](#) tag, unsigned rights, [l4\\_utcb\\_t](#) \*utcb) override  
*The abstract handler for client requests to the object.*

**Public Member Functions inherited from L4::Epiface**

- **Epiface** ()  
*Make a server object.*
- virtual ~**Epiface** ()=0  
*Destroy the object.*
- Stored\_cap [obj\\_cap](#) () const  
*Get the capability to the kernel object belonging to this object.*
- [Server\\_iface](#) \* [server\\_iface](#) () const  
*Get pointer to server interface at which the object is currently registered.*
- int [set\\_server](#) ([Server\\_iface](#) \*srv, [Cap](#)< void > cap, bool managed=false)  
*Set server registration info for the object.*
- void [set\\_obj\\_cap](#) ([Cap](#)< void > const &cap)  
*Deprecated server registration function.*

**Static Public Member Functions inherited from L4::Server\_object\_t< Kobject >**

- static int [proto\\_dispatch](#) (THIS \*self, [l4\\_umword\\_t](#) rights, L4::lpc::lostream &ios)  
*Implementation of protocol-based dispatch for this server object.*

**15.171.1 Detailed Description**

[Server](#) object base class for handling IRQ messages.

This server object base class implements the empty interface ([L4::Kobject](#)). The implementation of [Server\\_object::dispatch\(\)](#) must return [-L4\\_ENOREPLY](#), because IRQ messages do not handle replies.

**Examples**

[examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#).

Definition at line 172 of file [ipc\\_server](#).

The documentation for this struct was generated from the following file:

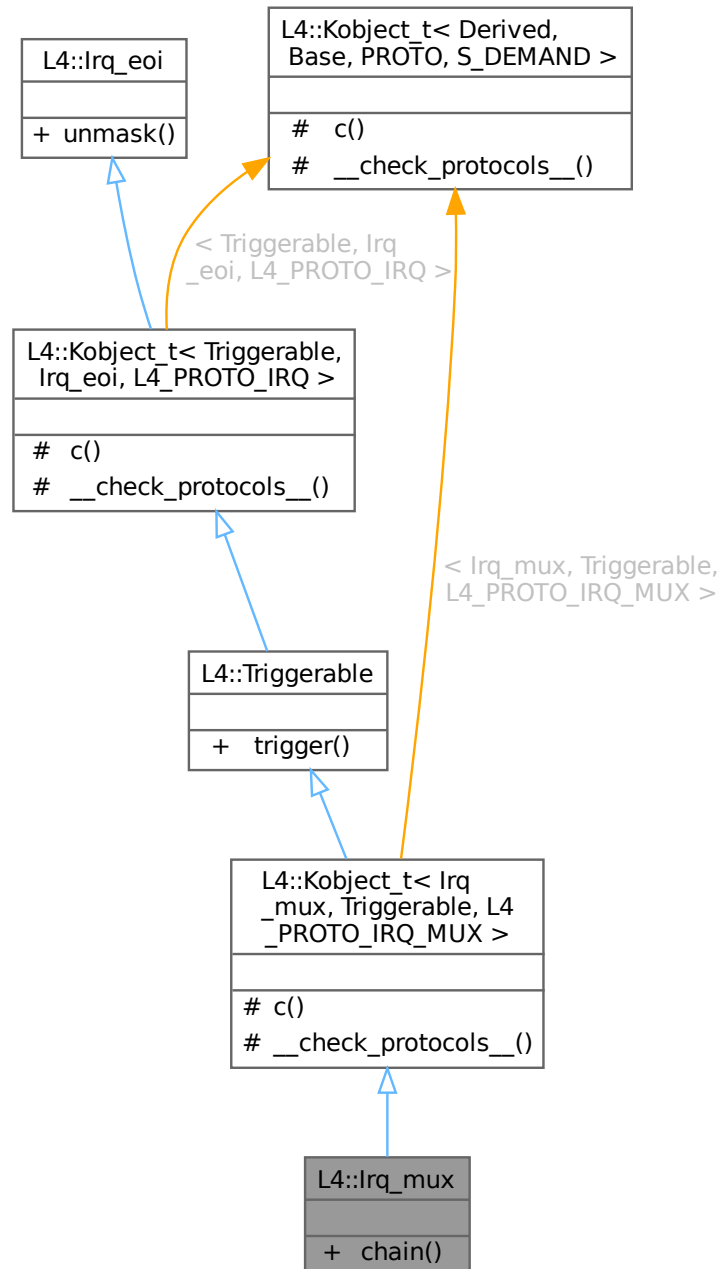
- [l4/cxx/ipc\\_server](#)

## 15.172 L4::Irq\_mux Struct Reference

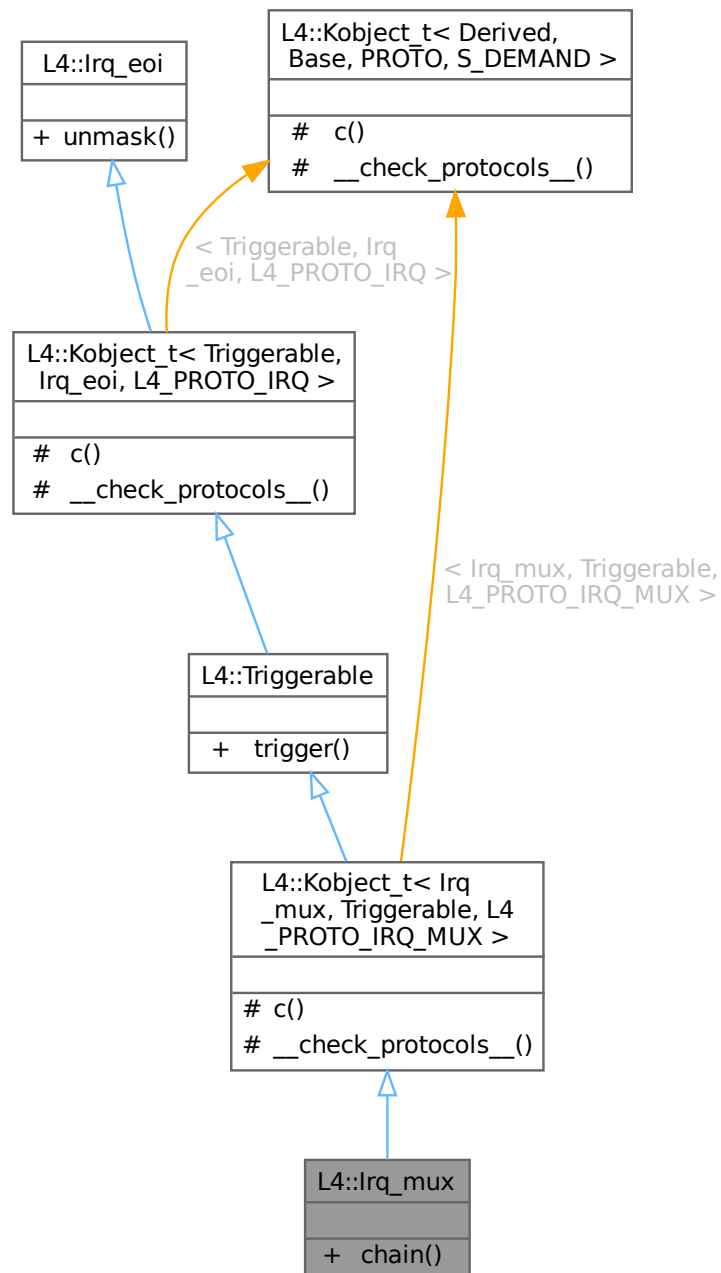
IRQ multiplexer for shared IRQs.

```
#include <irq>
```

Inheritance diagram for L4::Irq\_mux:



Collaboration diagram for L4::Irq\_mux:



### Public Member Functions

- `l4_msgtag_t chain (Cap< Triggerable > const &slave, l4_utcb_t *utcb=l4_utcb()) noexcept`  
Attach an IRQ to this multiplexer.

### Public Member Functions inherited from L4::Triggerable

- `l4_msgtag_t trigger (l4_utcb_t *utcb=l4_utcb()) noexcept`

*Trigger the object.*

## Public Member Functions inherited from [L4::Irq\\_eoi](#)

- [l4\\_msgtag\\_t](#) [unmask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=L4\_IPC\_NEVER, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\\_t](#)()) noexcept

*Unmask the given interrupt line.*

## Additional Inherited Members

## Protected Types inherited from

### [L4::Kobject\\_t](#)< [Irq\\_mux](#), [Triggerable](#), [L4\\_PROTO\\_IRQ\\_MUX](#) >

- typedef [Irq\\_mux](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, [Irq\\_mux](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Types inherited from [L4::Kobject\\_t](#)< [Triggerable](#), [Irq\\_eoi](#), [L4\\_PROTO\\_IRQ](#) >

- typedef [Triggerable](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, [Triggerable](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from

### [L4::Kobject\\_t](#)< [Irq\\_mux](#), [Triggerable](#), [L4\\_PROTO\\_IRQ\\_MUX](#) >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept  
*Get the capability to ourselves.*

## Protected Member Functions inherited from

### [L4::Kobject\\_t](#)< [Triggerable](#), [Irq\\_eoi](#), [L4\\_PROTO\\_IRQ](#) >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept  
*Get the capability to ourselves.*

## Static Protected Member Functions inherited from

### [L4::Kobject\\_t](#)< [Irq\\_mux](#), [Triggerable](#), [L4\\_PROTO\\_IRQ\\_MUX](#) >

- static void **\_\_check\_protocols\_\_** () noexcept  
*Helper to check for protocol conflicts.*

## Static Protected Member Functions inherited from [L4::Kobject\\_t](#)<[Triggerable](#), [Irq\\_eoi](#), [L4\\_PROTO\\_IRQ](#)>

- static void `__check_protocols__()` noexcept  
*Helper to check for protocol conflicts.*

### 15.172.1 Detailed Description

IRQ multiplexer for shared IRQs.

**Deprecated** IRQ muxer objects are no longer supported by the kernel.

This interface allows broadcasting of shared IRQs to multiple triggerables. The IRQ multiplexer is responsible for the correct mask and unmask logic for such shared IRQs.

The semantics are that each of the slave IRQs is triggered whenever the multiplexer IRQ is triggered. As shared IRQs are usually level-triggered, the real IRQ source will be masked automatically when an IRQ is delivered and shall be unmasked when all attached slave IRQs are unmasked.

Definition at line 212 of file [irq](#).

### 15.172.2 Member Function Documentation

#### 15.172.2.1 `chain()`

```
l4_msgtag_t L4::Irq_mux::chain (
    Cap< Triggerable > const & slave,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

Attach an IRQ to this multiplexer.

##### Parameters

<i>slave</i>	The slave that shall be attached to the master.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

##### Returns

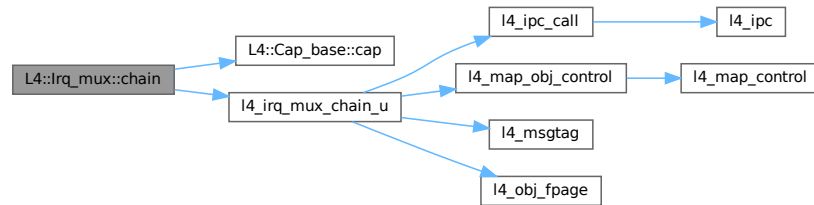
Syscall return tag

The chaining feature of IRQ objects allows to deal with shared IRQs. For chaining IRQs there must be an IRQ multiplexer ([Irq\\_mux](#)) bound to the real IRQ source. This function allows to add slave IRQs to this multiplexer.

Definition at line 227 of file [irq](#).

References [L4::Cap\\_base::cap\(\)](#), and [l4\\_irq\\_mux\\_chain\\_u\(\)](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

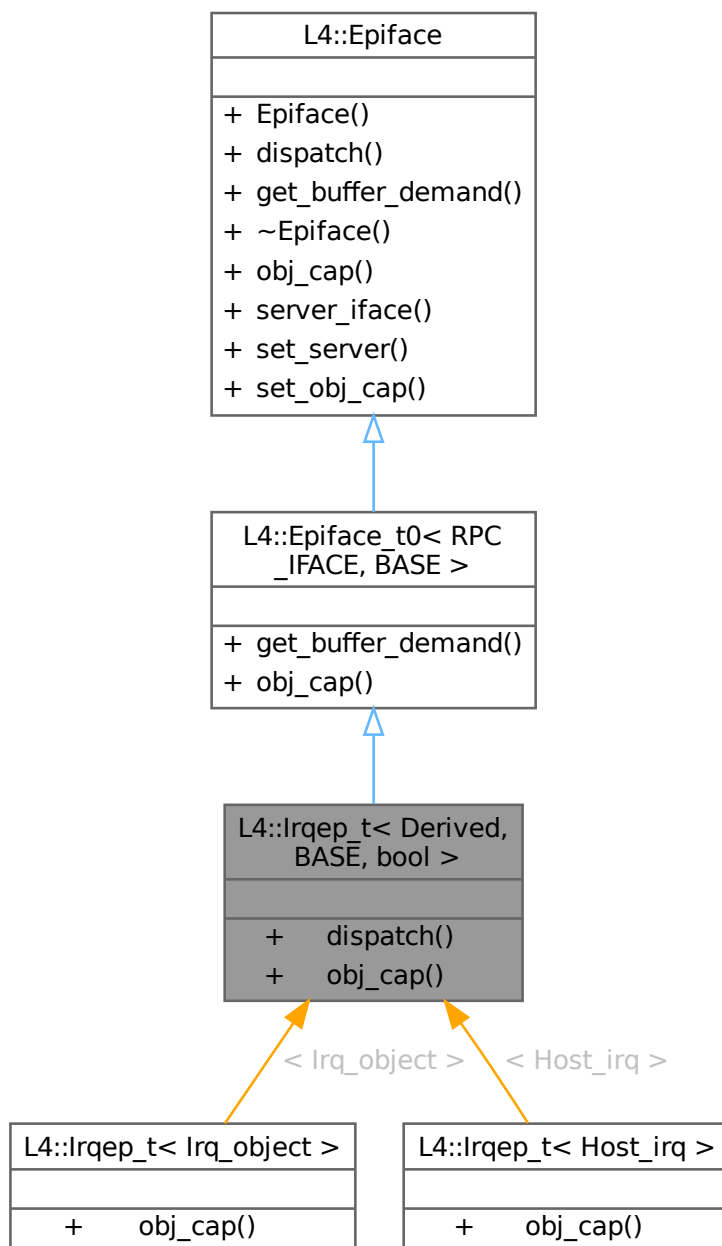
- [l4/sys/irq](#)

## 15.173 L4::lrqep\_t< Derived, BASE, bool > Struct Template Reference

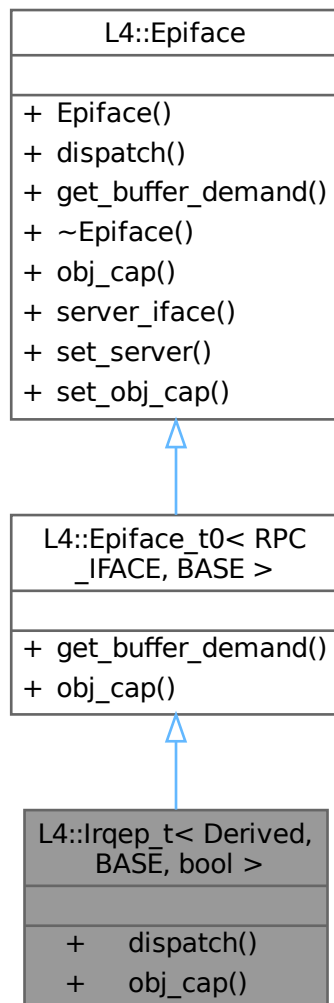
[Epiface](#) implementation for interrupt handlers.

```
#include <ipc_epiface>
```

Inheritance diagram for L4::lrqep\_t< Derived, BASE, bool >:



Collaboration diagram for L4::lrqep\_t< Derived, BASE, bool >:



### Public Member Functions

- `l4_msgtag_t dispatch (l4_msgtag_t, unsigned, l4_utcb_t *)` final  
*The abstract handler for client requests to the object.*
- `Cap< L4::lrq > obj_cap ()` const  
*Get the (typed) capability to this object.*

### Public Member Functions inherited from `L4::Epiface_t0< RPC_IFACE, BASE >`

- `Type_info::Demand get_buffer_demand ()` const  
*Get the server-side buffer demand based in IFACE.*
- `Cap< RPC_IFACE > obj_cap ()` const  
*Get the (typed) capability to this object.*



## Public Member Functions inherited from L4::Epiface

- **Epiface** ()  
*Make a server object.*
- virtual **~Epiface** ()=0  
*Destroy the object.*
- Stored\_cap **obj\_cap** () const  
*Get the capability to the kernel object belonging to this object.*
- **Server\_iface** \* **server\_iface** () const  
*Get pointer to server interface at which the object is currently registered.*
- int **set\_server** (**Server\_iface** \*srv, **Cap**< void > cap, bool managed=false)  
*Set server registration info for the object.*
- void **set\_obj\_cap** (**Cap**< void > const &cap)  
*Deprecated server registration function.*

## Additional Inherited Members

## Public Types inherited from L4::Epiface\_t0< RPC\_IFACE, BASE >

- typedef **RPC\_IFACE** **Interface**  
*Data type of the IPC interface definition.*

## Public Types inherited from L4::Epiface

- typedef **lpc\_svr::Server\_iface** **Server\_iface**  
*Type for abstract server interface.*
- typedef **lpc\_svr::Server\_iface::Demand** **Demand**  
*Type for server-side receive buffer demand.*

## 15.173.1 Detailed Description

```
template<typename Derived, typename BASE = Epiface, bool = cxx::is_polymorphic<BASE>::value>
struct L4::Irqep_t< Derived, BASE, bool >
```

**Epiface** implementation for interrupt handlers.

### Template Parameters

<i>Derived</i>	<b>Irq</b> handler implementation class. The class must provide a single function <code>handle_irq()</code> .
<i>BASE</i>	Base <b>Epiface</b> class.

Definition at line 293 of file `ipc_epiface`.

## 15.173.2 Member Function Documentation

### 15.173.2.1 dispatch()

```
template<typename Derived , typename BASE = Epiface, bool = cxx::is_polymorphic<BASE>::value>
l4_msgtag_t L4::Irqep_t< Derived, BASE, bool >::dispatch (
    l4_msgtag_t tag,
    unsigned rights,
    l4_utcb_t * utcb ) [inline], [final], [virtual]
```

The abstract handler for client requests to the object.

#### Parameters

<i>tag</i>	The message tag for this invocation.
<i>rights</i>	The rights bits in the invoked capability.
<i>utcb</i>	The UTCB used for the invocation.

#### Return values

<code>-L4_ENOREPLY</code>	No reply message is send.
<code>&lt;0</code>	Error, reply with error code.
<code>&gt;=0</code>	Success, reply with return value.

This function must be implemented by application specific server objects.

Implements [L4::Epiface](#).

Definition at line 295 of file [ipc\\_epiface](#).

References [L4\\_ENOREPLY](#), and [l4\\_msgtag\(\)](#).

Here is the call graph for this function:



### 15.173.2.2 obj\_cap()

```
template<typename Derived , typename BASE = Epiface, bool = cxx::is_polymorphic<BASE>::value>
Cap< L4::Irq > L4::Irqep_t< Derived, BASE, bool >::obj_cap ( ) const [inline]
```

Get the (typed) capability to this object.

**Returns**

[lirq](#) capability for the kernel object behind the server.

Definition at line 305 of file [ipc\\_epiface](#).

The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc\\_epiface](#)

## 15.174 L4::Kip::Mem\_desc Class Reference

Memory descriptors stored in the kernel interface page.

```
#include <kip>
```

Collaboration diagram for L4::Kip::Mem\_desc:

L4::Kip::Mem_desc
<ul style="list-style-type: none"> <li>+ Mem_desc()</li> <li>+ start()</li> <li>+ end()</li> <li>+ size()</li> <li>+ type()</li> <li>+ sub_type()</li> <li>+ is_virtual()</li> <li>+ eager_map()</li> <li>+ set()</li> <li>+ first()</li> <li>+ count()</li> <li>+ count()</li> <li>+ all()</li> <li>+ all()</li> </ul>

**Public Types**

- enum [Mem\\_type](#) {  
[Undefined](#) = 0x0 , [Conventional](#) = 0x1 , [Reserved](#) = 0x2 , [Dedicated](#) = 0x3 ,  
[Shared](#) = 0x4 , [Info](#) = 0xd , [Bootloader](#) = 0xe , [Arch](#) = 0xf }  
*Memory types.*
- enum [Info\\_sub\\_type](#) { [Info\\_acpi\\_rsdp](#) = 0 }  
*Memory sub types for the [Mem\\_type::Info](#) type.*
- enum [Arch\\_sub\\_type\\_common](#) { [Arch\\_acpi\\_tables](#) = 3 , [Arch\\_acpi\\_nvs](#) = 4 }  
*Common sub types across all architectures for the [Mem\\_type::Arch](#) type.*

## Public Member Functions

- `Mem_desc` (unsigned long `start`, unsigned long `end`, `Mem_type` `t`, unsigned char `st=0`, bool `virt=false`, bool `eager=false`) noexcept  
*Initialize memory descriptor.*
- unsigned long `start` () const noexcept  
*Return start address of memory descriptor.*
- unsigned long `end` () const noexcept  
*Return end address of memory descriptor.*
- unsigned long `size` () const noexcept  
*Return size of region described by the memory descriptor.*
- `Mem_type` `type` () const noexcept  
*Return type of the memory descriptor.*
- unsigned char `sub_type` () const noexcept  
*Return sub-type of the memory descriptor.*
- unsigned `is_virtual` () const noexcept  
*Return whether the memory descriptor describes a virtual or physical region.*
- unsigned `eager_map` () const noexcept  
*Return whether the region shall be eligible for eager mapping in sigma0 or the root task.*
- void `set` (unsigned long `start`, unsigned long `end`, `Mem_type` `t`, unsigned char `st=0`, bool `virt=false`, bool `eager=false`) noexcept  
*Set values of a memory descriptor.*

## Static Public Member Functions

- static `Mem_desc * first` (void \*`kip`) noexcept  
*Get first memory descriptor.*
- static unsigned long `count` (void const \*`kip`) noexcept  
*Return number of memory descriptors stored in the kernel info page.*
- static void `count` (void \*`kip`, unsigned `count`) noexcept  
*Set number of memory descriptors.*
- static `cxx::static_vector< Mem_desc const > all` (void const \*`kip`)  
*Return enumerable list of memory descriptors.*
- static `cxx::static_vector< Mem_desc > all` (void \*`kip`)  
*Return enumerable list of memory descriptors.*

### 15.174.1 Detailed Description

Memory descriptors stored in the kernel interface page.

#### Include File

```
#include <l4/sys/kip>
```

Definition at line 53 of file `kip`.

### 15.174.2 Member Enumeration Documentation

#### 15.174.2.1 Arch\_sub\_type\_common

```
enum L4::Kip::Mem_desc::Arch_sub_type_common
```

Common sub types across all architectures for the `Mem_type::Arch` type.

## Enumerator

Arch_acpi_tables	Firmware ACPI tables.
Arch_acpi_nvs	Firmware reserved address space.

Definition at line 83 of file [kip](#).

### 15.174.2.2 Info\_sub\_type

```
enum L4::Kip::Mem_desc::Info_sub_type
```

Memory sub types for the [Mem\\_type::Info](#) type.

## Enumerator

Info_acpi_rsdp	Physical address of the ACPI root pointer.
----------------	--

Definition at line 75 of file [kip](#).

### 15.174.2.3 Mem\_type

```
enum L4::Kip::Mem_desc::Mem_type
```

Memory types.

## Enumerator

Undefined	Undefined memory.
Conventional	Conventional memory.
Reserved	Reserved region, do not use this memory.
Dedicated	Dedicated.
Shared	Shared.
Info	Info by boot loader.
Bootloader	Memory belongs to the boot loader.
Arch	Architecture specific memory.

Definition at line 59 of file [kip](#).

## 15.174.3 Constructor & Destructor Documentation

### 15.174.3.1 Mem\_desc()

```
L4::Kip::Mem_desc::Mem_desc (
    unsigned long start,
    unsigned long end,
    Mem_type t,
```

```

    unsigned char st = 0,
    bool virt = false,
    bool eager = false ) [inline], [noexcept]

```

Initialize memory descriptor.

#### Parameters

<i>start</i>	Start address
<i>end</i>	End address
<i>t</i>	Memory type
<i>st</i>	Memory subtype, defaults to 0
<i>virt</i>	True for virtual memory, false for physical memory, defaults to physical
<i>eager</i>	The region shall be eligible for eager mapping in sigma0 or the root task. This is just an optimization to prevent on-demand paging.

Definition at line 179 of file [kip](#).

## 15.174.4 Member Function Documentation

### 15.174.4.1 all() [1/2]

```

static cxx::static_vector< Mem_desc > L4::Kip::Mem_desc::all (
    void * kip ) [inline], [static]

```

Return enumerable list of memory descriptors.

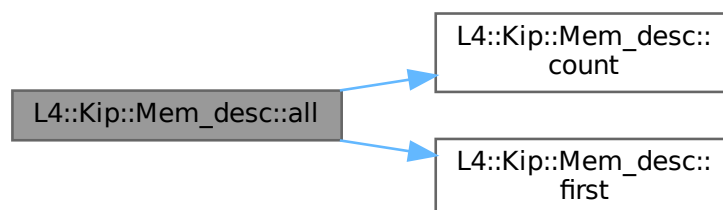
#### Parameters

<i>kip</i>	Pointer to the kernel info page.
------------	----------------------------------

Definition at line 160 of file [kip](#).

References [count\(\)](#), and [first\(\)](#).

Here is the call graph for this function:



### 15.174.4.2 all() [2/2]

```
static cxx::static_vector< Mem_desc const > L4::Kip::Mem_desc::all (
    void const * kip ) [inline], [static]
```

Return enumerable list of memory descriptors.

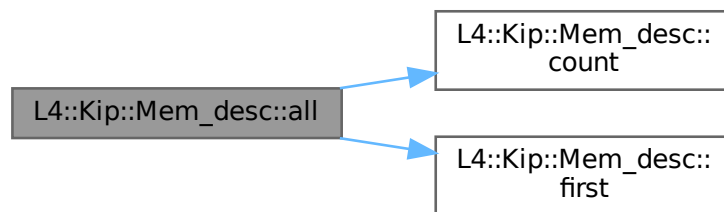
#### Parameters

<i>kip</i>	Pointer to the kernel info page.
------------	----------------------------------

Definition at line 149 of file [kip](#).

References [count\(\)](#), and [first\(\)](#).

Here is the call graph for this function:



### 15.174.4.3 count() [1/2]

```
static void L4::Kip::Mem_desc::count (
    void * kip,
    unsigned count ) [inline], [static], [noexcept]
```

Set number of memory descriptors.

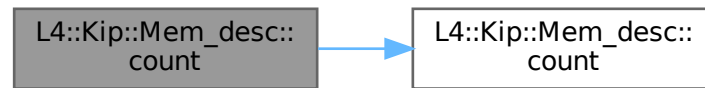
#### Parameters

<i>kip</i>	Pointer to the kernel info page
<i>count</i>	Number of memory descriptors

Definition at line 138 of file [kip](#).

References [count\(\)](#).

Here is the call graph for this function:



#### 15.174.4.4 count() [2/2]

```
static unsigned long L4::Kip::Mem_desc::count (
    void const * kip ) [inline], [static], [noexcept]
```

Return number of memory descriptors stored in the kernel info page.

##### Parameters

<i>kip</i>	Pointer to the kernel info page
------------	---------------------------------

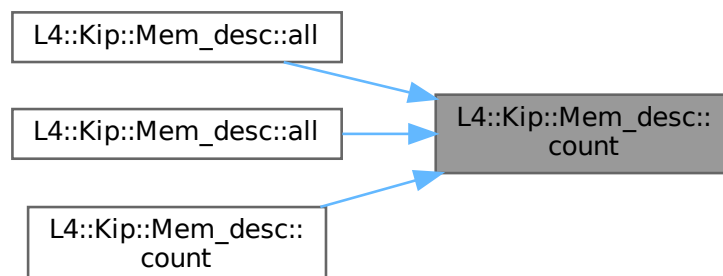
##### Returns

Number of memory descriptors in the kernel info page.

Definition at line 126 of file [kip](#).

Referenced by [all\(\)](#), [all\(\)](#), and [count\(\)](#).

Here is the caller graph for this function:





#### 15.174.4.5 end()

```
unsigned long L4::Kip::Mem_desc::end ( ) const [inline], [noexcept]
```

Return end address of memory descriptor.

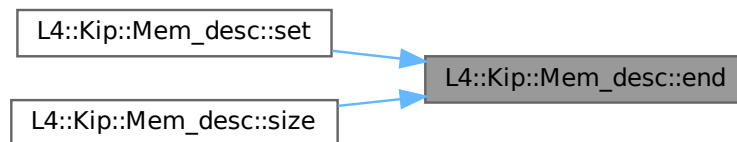
##### Returns

End address of memory descriptor

Definition at line 198 of file [kip](#).

Referenced by [set\(\)](#), and [size\(\)](#).

Here is the caller graph for this function:



#### 15.174.4.6 first()

```
static Mem_desc * L4::Kip::Mem_desc::first (
    void * kip ) [inline], [static], [noexcept]
```

Get first memory descriptor.

##### Parameters

<i>kip</i>	Pointer to the kernel info page
------------	---------------------------------

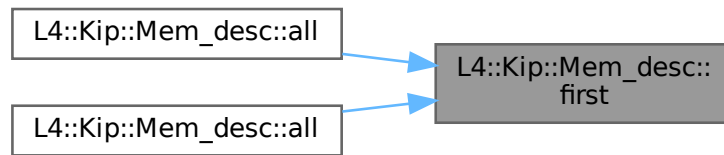
##### Returns

First memory descriptor stored in the kernel info page

Definition at line 106 of file [kip](#).

Referenced by [all\(\)](#), and [all\(\)](#).

Here is the caller graph for this function:



#### 15.174.4.7 is\_virtual()

```
unsigned L4::Kip::Mem_desc::is_virtual ( ) const [inline], [noexcept]
```

Return whether the memory descriptor describes a virtual or physical region.

##### Returns

True for virtual region, false for physical region.

Definition at line 230 of file [kip](#).

#### 15.174.4.8 set()

```
void L4::Kip::Mem_desc::set (
    unsigned long start,
    unsigned long end,
    Mem_type t,
    unsigned char st = 0,
    bool virt = false,
    bool eager = false ) [inline], [noexcept]
```

Set values of a memory descriptor.

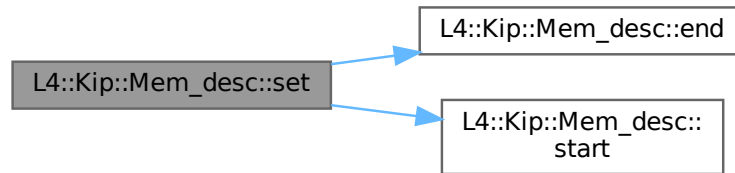
##### Parameters

<i>start</i>	Start address
<i>end</i>	End address
<i>t</i>	Memory type
<i>st</i>	Sub-type, defaults to 0
<i>virt</i>	Virtual or physical memory region, defaults to physical
<i>eager</i>	The region shall be eligible for eager mapping in sigma0 or the root task. This is just an optimization to prevent on-demand paging.

Definition at line 250 of file [kip](#).

References [end\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



#### 15.174.4.9 size()

```
unsigned long L4::Kip::Mem_desc::size ( ) const [inline], [noexcept]
```

Return size of region described by the memory descriptor.

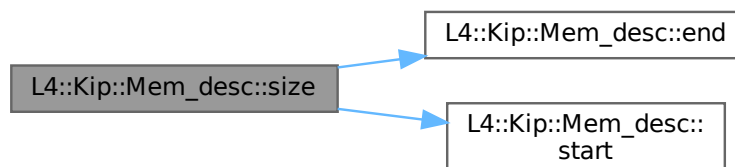
##### Returns

Size of the region described by the memory descriptor

Definition at line [205](#) of file [kip](#).

References [end\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



#### 15.174.4.10 start()

```
unsigned long L4::Kip::Mem_desc::start ( ) const [inline], [noexcept]
```

Return start address of memory descriptor.

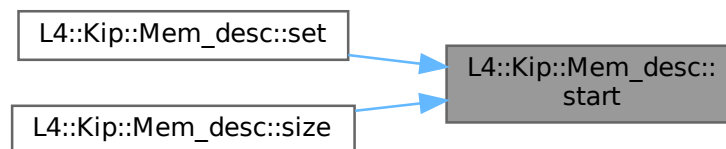
##### Returns

Start address of memory descriptor

Definition at line 191 of file [kip](#).

Referenced by [set\(\)](#), and [size\(\)](#).

Here is the caller graph for this function:



#### 15.174.4.11 sub\_type()

```
unsigned char L4::Kip::Mem_desc::sub_type ( ) const [inline], [noexcept]
```

Return sub-type of the memory descriptor.

##### Returns

Sub-type of the memory descriptor

Definition at line 222 of file [kip](#).

#### 15.174.4.12 type()

```
Mem\_type L4::Kip::Mem_desc::type ( ) const [inline], [noexcept]
```

Return type of the memory descriptor.

##### Returns

Type of the memory descriptor

Definition at line 212 of file [kip](#).

The documentation for this class was generated from the following file:

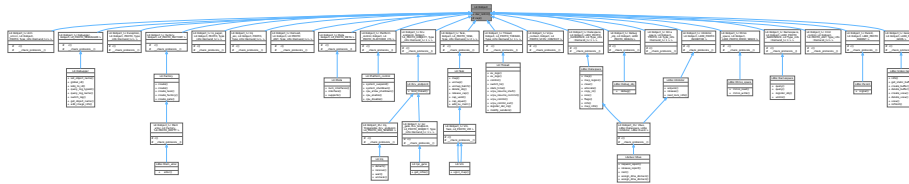
- [l4/sys/kip](#)

## 15.175 L4::Kobject Class Reference

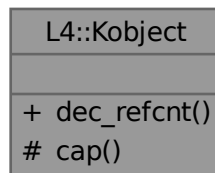
Base class for all kinds of kernel objects and remote objects, referenced by capabilities.

```
#include <kobject>
```

Inheritance diagram for L4::Kobject:



Collaboration diagram for L4::Kobject:



### Public Member Functions

- [l4\\_msgtag\\_t dec\\_refcnt](#) ([l4\\_mword\\_t diff](#), [l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)](#))  
*Decrement the in kernel reference counter for the object.*

### Protected Member Functions

- [l4\\_cap\\_idx\\_t cap](#) () const noexcept  
*Return capability selector.*

### 15.175.1 Detailed Description

Base class for all kinds of kernel objects and remote objects, referenced by capabilities.

#### Include File

```
#include <l4/sys/capability>
```

This is the base class for all remote objects accessible using RPC. However, subclasses do not directly inherit from [L4::Kobject](#) but *must* use [L4::Kobject\\_t](#) ([L4::Kobject\\_0t](#), [L4::Kobject\\_2t](#), [L4::Kobject\\_3t](#), or [L4::Kobject\\_x](#)) for inheritance, otherwise these classes cannot be used as RPC interfaces.

#### Attention

Objects derived from [Kobject](#) *must* never add any data to those objects. Kobjects can act only as proxy object for encapsulating object invocations.

Definition at line 46 of file [kobject](#).

## 15.175.2 Member Function Documentation

### 15.175.2.1 cap()

```
l4_cap_idx_t L4::Kobject::cap ( ) const [inline], [protected], [noexcept]
```

Return capability selector.

#### Returns

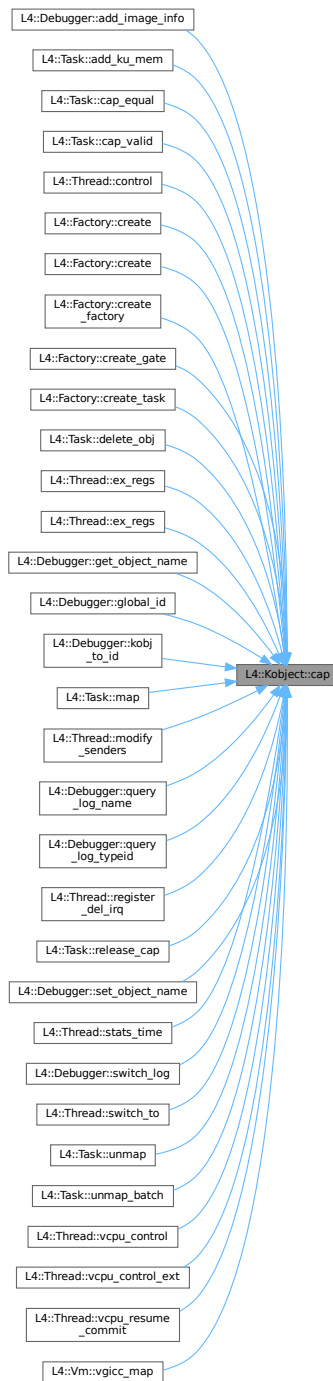
Capability selector.

This method is for derived classes to gain access to the actual capability selector.

Definition at line 79 of file [kobject](#).

Referenced by [L4::Debugger::add\\_image\\_info\(\)](#), [L4::Task::add\\_ku\\_mem\(\)](#), [L4::Task::cap\\_equal\(\)](#), [L4::Task::cap\\_valid\(\)](#), [L4::Thread::control\(\)](#), [L4::Factory::create\(\)](#), [L4::Factory::create\(\)](#), [L4::Factory::create\\_factory\(\)](#), [L4::Factory::create\\_gate\(\)](#), [L4::Factory::create\\_task\(\)](#), [L4::Task::delete\\_obj\(\)](#), [L4::Thread::ex\\_regs\(\)](#), [L4::Thread::ex\\_regs\(\)](#), [L4::Debugger::get\\_object\\_name\(\)](#), [L4::Debugger::global\\_id\(\)](#), [L4::Debugger::kobj\\_to\\_id\(\)](#), [L4::Task::map\(\)](#), [L4::Thread::modify\\_senders\(\)](#), [L4::Debugger::query\\_log\\_name\(\)](#), [L4::Debugger::query\\_log\\_typeid\(\)](#), [L4::Thread::register\\_del\\_irq\(\)](#), [L4::Task::release\\_cap\(\)](#), [L4::Debugger::set\\_object\\_name\(\)](#), [L4::Thread::stats\\_time\(\)](#), [L4::Debugger::switch\\_log\(\)](#), [L4::Thread::switch\\_to\(\)](#), [L4::Task::unmap\(\)](#), [L4::Task::unmap\\_batch\(\)](#), [L4::Thread::vcpu\\_control\(\)](#), [L4::Thread::vcpu\\_control\\_ext\(\)](#), [L4::Thread::vcpu\\_resume\\_commit\(\)](#), and [L4::Vm::vgicc\\_map\(\)](#).

Here is the caller graph for this function:



### 15.175.2.2 dec\_refcnt()

```

14_msgtag_t L4::Kobject::dec_refcnt (
    14_mword_t diff,
    14_utcb_t * utcb = 14_utcb() ) [inline]
  
```

Decrement the in kernel reference counter for the object.

## Parameters

<i>diff</i>	The delta that shall be subtracted from the reference count.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

## Returns

Syscall return tag

This function is intended for servers to be able to remove the servers own capability from the counted references. This leads to the semantics that the kernel will delete the object even if the capability of the server is valid. The server can detect the deletion by polling its capabilities or by using the IPC-gate deletion IRQs. And to cleanup if the clients dropped the last reference (capability) to the object.

This function only succeeds on a kernel object of type [L4::lpc\\_gate](#) which has the server right ([L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#)). For other kernel objects, -L4\_ENOSYS is returned.

Definition at line 110 of file [kobject](#).

The documentation for this class was generated from the following file:

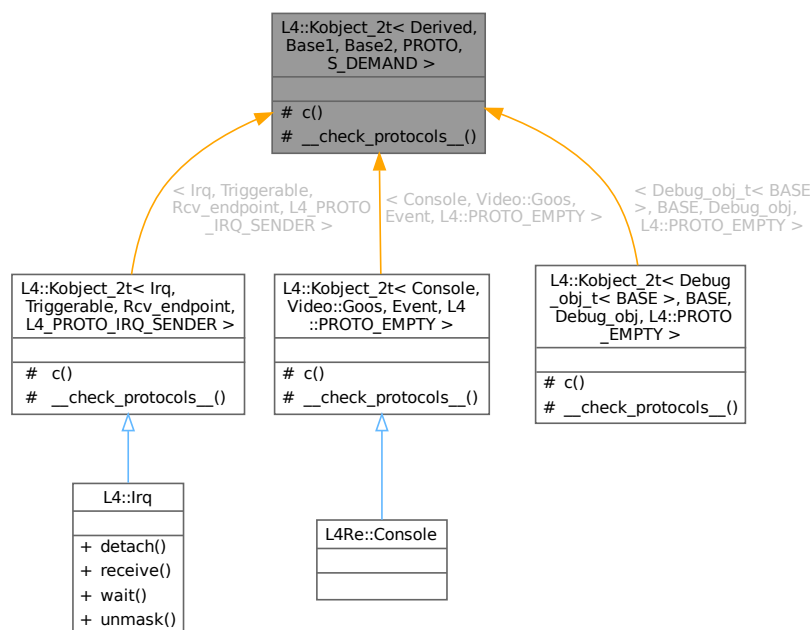
- [l4/sys/kobject](#)

## 15.176 L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_DEMAND > Class Template Reference

Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject\\_t](#)).

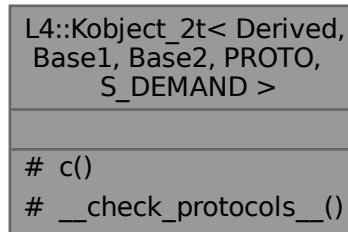
```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_DEMAND >:





Collaboration diagram for L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_DEMAND >:



### Protected Types

- typedef Derived [Class](#)  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, Derived > [\\_\\_iface](#)  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_iface](#) >, Typeid::Merge\_list< typename Base1::\_\_iface\_list, typename Base2::\_\_iface\_list > > [\\_\\_iface\\_list](#)  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

### Static Protected Member Functions

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

## 15.176.1 Detailed Description

```

template<typename Derived, typename Base1, typename Base2, long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
class L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >
  
```

Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject\\_t](#)).

### Template Parameters

<i>Derived</i>	is the name of the new interface.
<i>Base1</i>	is the name of the interface's first base class.
<i>Base2</i>	is the name of the interface's second base class.
<i>PROTO</i>	may be set to the statically assigned protocol number used to communicate with this interface.
<i>S_DEMAND</i>	type defining the demand of server-side resources for this interface, usually a <a href="#">L4::Type_info::Demand_t</a> . This value must describe the server-side resources needed by the interface itself, the resource demand of the base interfaces (Base1 and Base2) are automatically included.

The typical usage pattern is shown in the following code snippet. The semantics of this example is an interface `My_iface` that is derived from `L4::Icu` and `L4Re::Dataspace`.

```
class My_iface : public L4::Kobject_2t<My_iface, L4::Icu, L4Re::Dataspace>
{
    ...
};
```

Definition at line 838 of file `__typeinfo.h`.

## 15.176.2 Member Typedef Documentation

### 15.176.2.1 `__Iface`

```
template<typename Derived , typename Base1 , typename Base2 , long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Iface<PROTO, Derived> L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND
>::__Iface [protected]
```

The interface description for the derived class.

Definition at line 844 of file `__typeinfo.h`.

### 15.176.2.2 `__Iface_list`

```
template<typename Derived , typename Base1 , typename Base2 , long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Merge_list< Typeid::Iface_list<__Iface>, Typeid::Merge_list< typename Base1↔
::__Iface_list, typename Base2::__Iface_list > > L4::Kobject_2t< Derived, Base1, Base2, PROTO,
S_DEMAND >::__Iface_list [protected]
```

The list of all RPC interfaces provided directly or through inheritance.

Definition at line 852 of file `__typeinfo.h`.

### 15.176.2.3 `Class`

```
template<typename Derived , typename Base1 , typename Base2 , long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
typedef Derived L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >::Class [protected]
```

The target interface type (inheriting from `Kobject_t`)

Definition at line 842 of file `__typeinfo.h`.

## 15.176.3 Member Function Documentation

### 15.176.3.1 `__check_protocols__()`

```
template<typename Derived , typename Base1 , typename Base2 , long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
static void L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >::__check_protocols__ ( )
[inline], [static], [protected], [noexcept]
```

Helper to check for protocol conflicts.

Definition at line 855 of file `__typeinfo.h`.

## 15.176.3.2 c()

```
template<typename Derived , typename Base1 , typename Base2 , long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
L4::Cap< Class > L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >::c ( ) const [inline],
[protected], [noexcept]
```

Get the capability to ourselves.

Definition at line 874 of file [\\_\\_typeinfo.h](#).

The documentation for this class was generated from the following file:

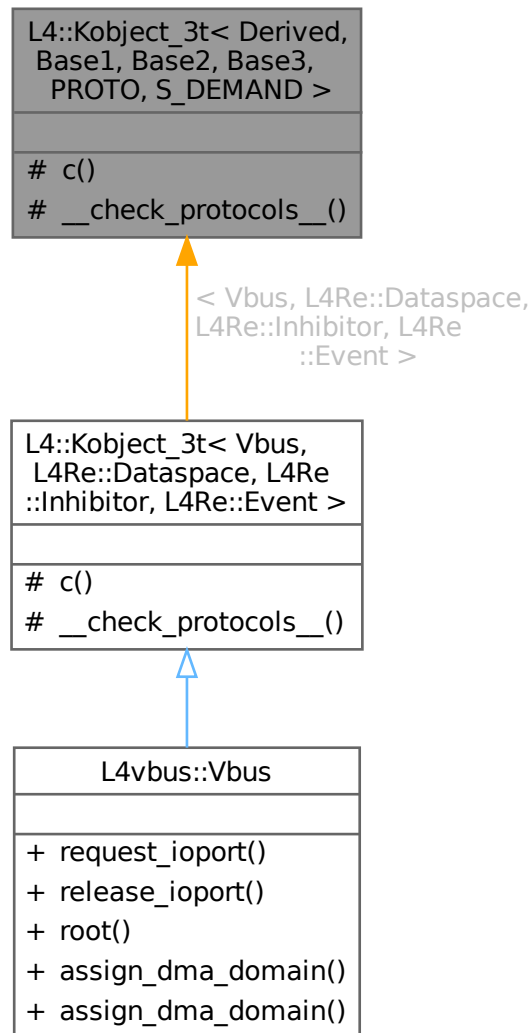
- [l4/sys/\\_\\_typeinfo.h](#)

## 15.177 L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S\_DEMAND > Struct Template Reference

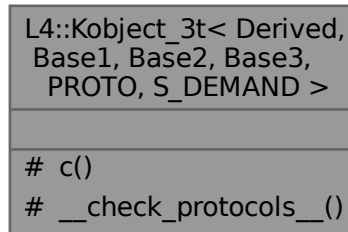
Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject\\_t](#)).

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S\_DEMAND >:



Collaboration diagram for L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S\_DEMAND >:



### Protected Types

- typedef Derived [Class](#)  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, Derived > [\\_\\_iface](#)  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_iface](#) >, Typeid::Merge\_list< typename Base1::\_\_iface\_list, Typeid::Merge\_list< typename Base2::\_\_iface\_list, typename Base3::\_\_iface\_list > > > [\\_\\_iface\\_list](#)  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

### Static Protected Member Functions

- static void [\\_\\_check\\_protocols\\_\\_](#) () noexcept  
*Helper to check for protocol conflicts.*

## 15.177.1 Detailed Description

```

template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO_
_ANY, typename S_DEMAND = Type_info::Demand_t<>>
struct L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >
  
```

Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4 : Kobject\\_t](#)).

### Template Parameters

<i>Derived</i>	is the name of the new interface.
<i>Base1</i>	is the name of the interface's first base class.
<i>Base2</i>	is the name of the interface's second base class.
<i>Base3</i>	is the name of the interfaces third base class.
<i>PROTO</i>	may be set to the statically assigned protocol number used to communicate with this interface.
<i>S_DEMAND</i>	type defining the demand on server-side resources for this interface, usually a <a href="#">L4::Type_info::Demand_t</a> . This value must describe the server-side resources needed by the interface itself. the resource demand of the base interfaces (Base1 and Base2) are

See also

[L4::Kobject\\_t](#), [L4::Kobject\\_2t](#), [L4::Kobject\\_0t](#), [L4::Kobject\\_x](#)

Definition at line 941 of file [\\_\\_typeinfo.h](#).

## 15.177.2 Member Typedef Documentation

### 15.177.2.1 \_\_Iface

```
template<typename Derived , typename Base1 , typename Base2 , typename Base3 , long PROTO =
PROTO_ANY, typename S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Iface<PROTO, Derived> L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO,
S_DEMAND >::__Iface [protected]
```

The interface description for the derived class.

Definition at line 947 of file [\\_\\_typeinfo.h](#).

### 15.177.2.2 \_\_Iface\_list

```
template<typename Derived , typename Base1 , typename Base2 , typename Base3 , long PROTO =
PROTO_ANY, typename S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Merge_list< Typeid::Iface_list<\_\_Iface>, Typeid::Merge_list< typename Base1↵
::__Iface_list, Typeid::Merge_list< typename Base2::__Iface_list, typename Base3::__Iface↵
_list > > > L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::__Iface_list
[protected]
```

The list of all RPC interfaces provided directly or through inheritance.

Definition at line 958 of file [\\_\\_typeinfo.h](#).

### 15.177.2.3 Class

```
template<typename Derived , typename Base1 , typename Base2 , typename Base3 , long PROTO =
PROTO_ANY, typename S_DEMAND = Type_info::Demand_t<>>
typedef Derived L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::Class [protected]
```

The target interface type (inheriting from [Kobject\\_t](#))

Definition at line 945 of file [\\_\\_typeinfo.h](#).

## 15.177.3 Member Function Documentation

### 15.177.3.1 \_\_check\_protocols\_\_()

```
template<typename Derived , typename Base1 , typename Base2 , typename Base3 , long PROTO =
PROTO_ANY, typename S_DEMAND = Type_info::Demand_t<>>
static void L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::__check_protocols↵
__ ( ) [inline], [static], [protected], [noexcept]
```

Helper to check for protocol conflicts.

Definition at line 961 of file [\\_\\_typeinfo.h](#).

**15.177.3.2 c()**

```
template<typename Derived , typename Base1 , typename Base2 , typename Base3 , long PROTO =
PROTO_ANY, typename S_DEMAND = Type_info::Demand_t<>>
L4::Cap< Class > L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::c ( ) const
[inline], [protected], [noexcept]
```

Get the capability to ourselves.

Definition at line 989 of file [\\_\\_typeinfo.h](#).

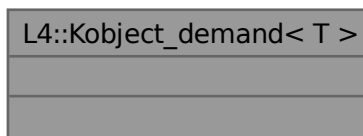
The documentation for this struct was generated from the following file:

- [l4/sys/\\_\\_typeinfo.h](#)

**15.178 L4::Kobject\_demand< T > Struct Template Reference**

Get the combined server-side resource requirements for all type T...

Collaboration diagram for L4::Kobject\_demand< T >:

**15.178.1 Detailed Description**

```
template<typename ... T>
struct L4::Kobject_demand< T >
```

Get the combined server-side resource requirements for all type T...

**Template Parameters**

<b>T</b>	List of IPC interface types for which the combined server-side resource requirements shall be calculated.
----------	---

Definition at line 1042 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

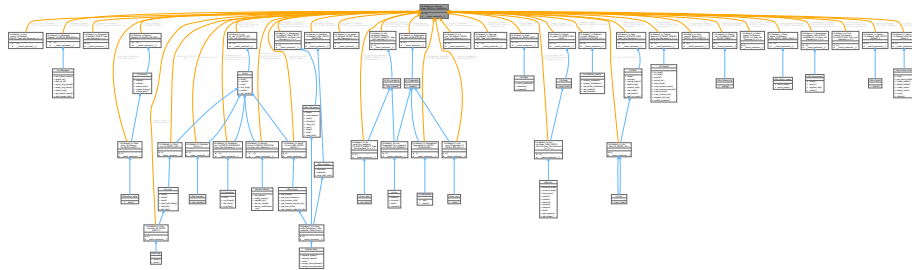
- [l4/sys/\\_\\_typeinfo.h](#)

## 15.179 L4::Kobject\_t< Derived, Base, PROTO, S\_DEMAND > Class Template Reference

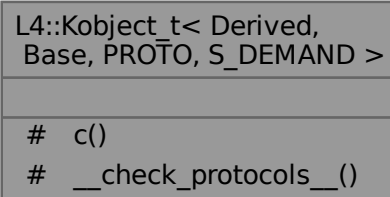
Helper class to create an [L4Re](#) interface class that is derived from a single base class.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Kobject\_t< Derived, Base, PROTO, S\_DEMAND >:



Collaboration diagram for L4::Kobject\_t< Derived, Base, PROTO, S\_DEMAND >:



### Protected Types

- typedef Derived **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, Derived > **\_\_iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_iface](#) >, typename Base::\_\_iface\_list > **\_\_iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions

- [L4::Cap](#)< [Class](#) > **c** () const noexcept  
*Get the capability to ourselves.*



### Static Protected Member Functions

- static void `__check_protocols__()` noexcept  
*Helper to check for protocol conflicts.*

#### 15.179.1 Detailed Description

```
template<typename Derived, typename Base, long PROTO = PROTO_ANY, typename S_DEMAND = Type_
_info::Demand_t<>>
class L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >
```

Helper class to create an [L4Re](#) interface class that is derived from a single base class.

#### Template Parameters

<i>Derived</i>	is the name of the new interface.
<i>Base</i>	is the name of the interfaces single base class.
<i>PROTO</i>	may be set to the statically assigned protocol number used to communicate with this interface.
<i>S_DEMAND</i>	type defining the demand on server-side resources for this interface, usually a <a href="#">L4::Type_info::Demand_t</a> . This value must describe the server-side resources needed by the interface itself, the resource demand of the base interface <i>Base</i> is automatically included.

The typical usage pattern is shown in the following code snippet. The semantics of this example is an interface `My_iface` that is derived from [L4::Kobject](#).

```
class My_iface : public L4::Kobject_t<My_iface, L4::Kobject>
{
    ...
};
```

Definition at line 760 of file [\\_\\_typeinfo.h](#).

The documentation for this class was generated from the following file:

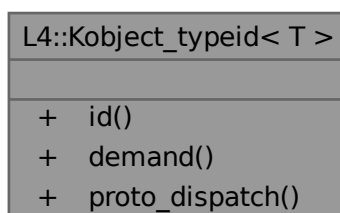
- [l4/sys/\\_\\_typeinfo.h](#)

## 15.180 L4::Kobject\_typeid< T > Struct Template Reference

[Meta](#) object for handling access to type information of Kobjects.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Kobject\_typeid< T >:



## Public Types

- typedef T::\_\_Kobject\_typeid::Demand [Demand](#)

*Data type expressing the static demand of receive buffers in a server.*

## Static Public Member Functions

- static [Type\\_info](#) const \* [id](#) () noexcept  
*Get a pointer to the [Kobject](#) type information of T.*
- static [Type\\_info::Demand](#) [demand](#) () noexcept  
*Get the receive-buffer demand for the server providing the interface T.*
- template<typename THIS , typename A1 , typename A2 >  
static int [proto\\_dispatch](#) (THIS \*self, long proto, A1 a1, A2 &a2)  
*Protocol based server-side dispatch function.*

### 15.180.1 Detailed Description

```
template<typename T>
struct L4::Kobject_typeid< T >
```

[Meta](#) object for handling access to type information of Kobjects.

#### Template Parameters

<a href="#">T</a>	The data type derived from <a href="#">Kobject</a> , usually using <a href="#">Kobject_t</a> .
-------------------	--

Definition at line 621 of file [\\_\\_typeinfo.h](#).

### 15.180.2 Member Typedef Documentation

#### 15.180.2.1 Demand

```
template<typename T >
typedef T::__Kobject_typeid::Demand L4::Kobject\_typeid< T >::Demand
```

Data type expressing the static demand of receive buffers in a server.

This information is the combined demand of all base interfaces for T and the buffer demand of T itself. The buffer demand of T is usually specified as the S\_DEMAND argument of the [Kobject\\_t](#) or [Kobject\\_2t](#) inheritance helpers. S\_DEMAND is usually of type [L4::Type\\_info::Demand\\_t](#), or [L4::Type\\_info::Demand\\_union\\_t](#).

Definition at line 633 of file [\\_\\_typeinfo.h](#).

### 15.180.3 Member Function Documentation

#### 15.180.3.1 demand()

```
template<typename T >
static Type_info::Demand L4::Kobject_typeid< T >::demand ( ) [inline], [static], [noexcept]
```

Get the receive-buffer demand for the server providing the interface T.

##### Returns

A demand value describing the minimum receive buffers needed for handling server side requests for interface T.

Definition at line 650 of file [\\_\\_typeinfo.h](#).

#### 15.180.3.2 id()

```
template<typename T >
static Type_info const * L4::Kobject_typeid< T >::id ( ) [inline], [static], [noexcept]
```

Get a pointer to the [Kobject](#) type information of T.

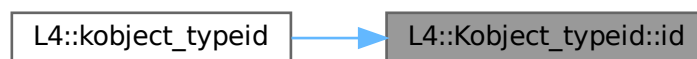
##### Returns

a pointer to the [Kobject](#) typeinfo of T.

Definition at line 641 of file [\\_\\_typeinfo.h](#).

Referenced by [L4::kobject\\_typeid\(\)](#).

Here is the caller graph for this function:



#### 15.180.3.3 proto\_dispatch()

```
template<typename T >
template<typename THIS , typename A1 , typename A2 >
static int L4::Kobject_typeid< T >::proto_dispatch (
    THIS * self,
    long proto,
    A1 a1,
    A2 & a2 ) [inline], [static]
```

Protocol based server-side dispatch function.

## Template Parameters

<i>THIS</i>	Data type of the server-side object implementing the interface T.
<i>A1</i>	Data type of second argument for p_dispatch()
<i>A2</i>	Data type of third argument for p_dispatch()

## Parameters

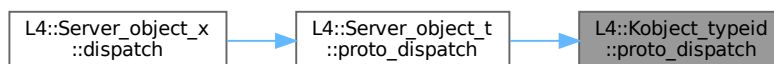
<i>self</i>	The pointer to the server object
<i>proto</i>	The protocol number used by the caller
<i>a1</i>	The second argument passed to self->p_dispatch()
<i>a2</i>	The third argument passed to self->p_dispatch()

This function forwards the call to the overloaded p\_dispatch() function of self. The data type of the first argument for p\_dispatch is determined by the given protocol number.

Definition at line 671 of file [\\_\\_typeinfo.h](#).

Referenced by [L4::Server\\_object\\_t< IFACE, BASE >::proto\\_dispatch\(\)](#).

Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

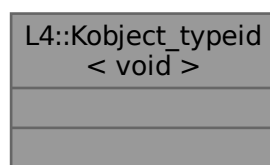
- [l4/sys/\\_\\_typeinfo.h](#)

## 15.181 L4::Kobject\_typeid< void > Struct Reference

Minimalistic ID for void interface.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Kobject\_typeid< void >:



### 15.181.1 Detailed Description

Minimalistic ID for `void` interface.

Definition at line 678 of file [\\_\\_typeinfo.h](#).

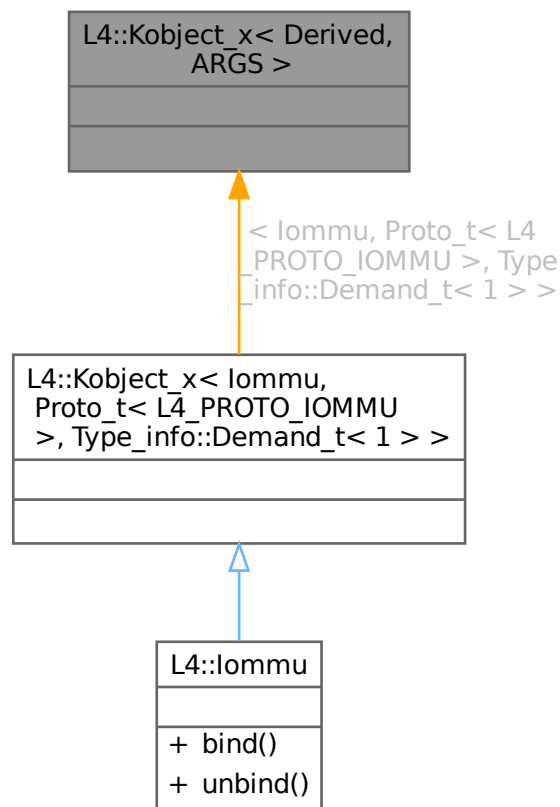
The documentation for this struct was generated from the following file:

- [l4/sys/\\_\\_typeinfo.h](#)

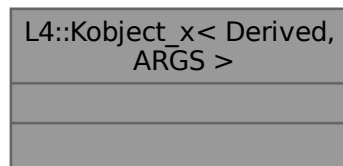
## 15.182 L4::Kobject\_x< Derived, ARGS > Struct Template Reference

Generic [Kobject](#) inheritance template.

Inheritance diagram for L4::Kobject\_x< Derived, ARGS >:



Collaboration diagram for L4::Kobject\_x< Derived, ARGS >:



### 15.182.1 Detailed Description

```
template<typename Derived, typename ... ARGS>
struct L4::Kobject_x< Derived, ARGS >
```

Generic [Kobject](#) inheritance template.

#### Template Parameters

<i>Derived</i>	The class name that derives from <a href="#">Kobject_x</a> .
<i>ARGS</i>	An optional protocol number via <a href="#">L4::Proto_t</a> , followed by an optional server-side requirement passed as <a href="#">L4::Type_info::Demand_t</a> , followed by the list of base classes.

Definition at line [1208](#) of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

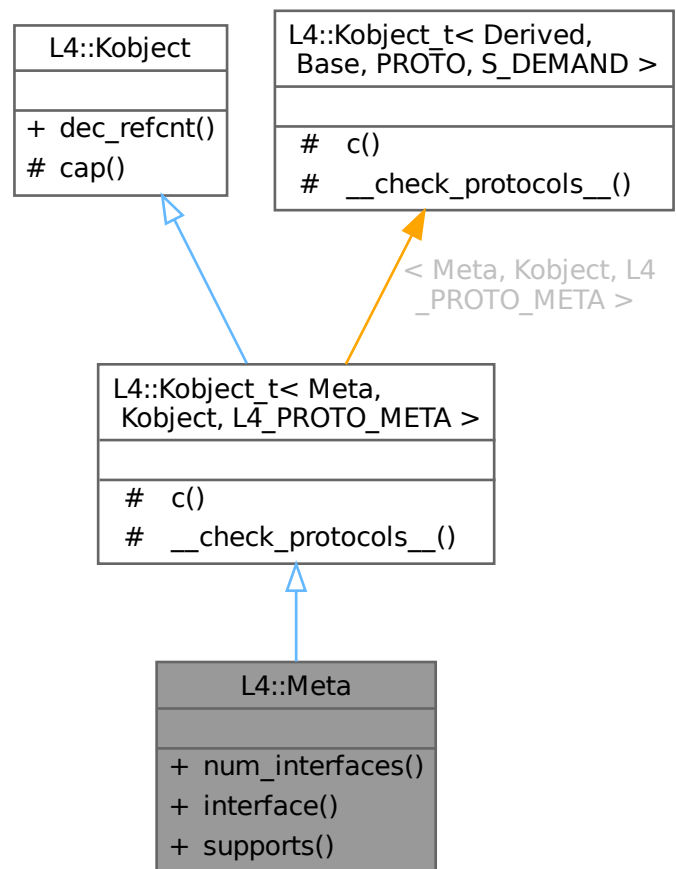
- [l4/sys/\\_\\_typeinfo.h](#)

## 15.183 L4::Meta Class Reference

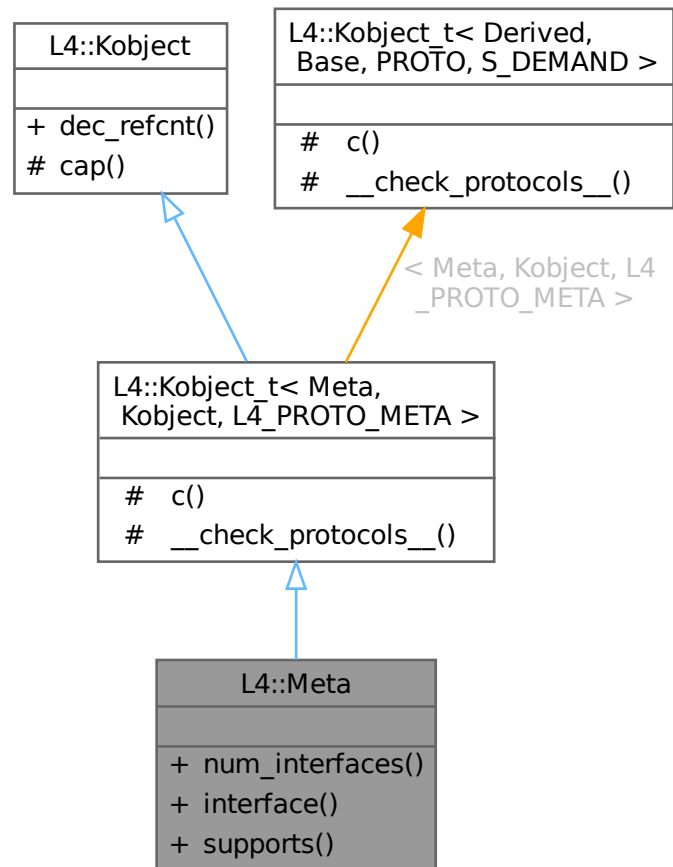
[Meta](#) interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.

```
#include <meta>
```

Inheritance diagram for L4::Meta:



Collaboration diagram for L4::Meta:



## Public Member Functions

- [l4\\_msgtag\\_t num\\_interfaces \(\)](#)  
Get the number of interfaces implemented by this object.
- [l4\\_msgtag\\_t interface \(l4\\_umword\\_t idx, long \\*proto, L4::lpc::String< char > \\*name\)](#)  
Get the protocol number that must be used for the interface with the number *idx*.
- [l4\\_msgtag\\_t supports \(l4\\_mword\\_t protocol\)](#)  
Figure out if the object supports the given protocol (number).

## Public Member Functions inherited from [L4::Kobject](#)

- [l4\\_msgtag\\_t dec\\_refcnt \(l4\\_mword\\_t diff, l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)\)](#)  
Decrement the in kernel reference counter for the object.



## Additional Inherited Members

### Protected Types inherited from [L4::Kobject\\_t< Meta, Kobject, L4\\_PROTO\\_META >](#)

- typedef [Meta](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, [Meta](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from [L4::Kobject\\_t< Meta, Kobject, L4\\_PROTO\\_META >](#)

- [L4::Cap< Class > c](#) () const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t](#) [cap](#) () const noexcept  
*Return capability selector.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< Meta, Kobject, L4\\_PROTO\\_META >](#)

- static void [\\_\\_check\\_protocols\\_\\_](#) () noexcept  
*Helper to check for protocol conflicts.*

## 15.183.1 Detailed Description

[Meta](#) interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.

Definition at line 37 of file [meta](#).

## 15.183.2 Member Function Documentation

### 15.183.2.1 interface()

```
l4\_msgtag\_t L4::Meta::interface (
    l4\_umword\_t idx,
    long * proto,
    L4::Ipc::String< char > * name )
```

Get the protocol number that must be used for the interface with the number *idx*.

## Parameters

	<i>idx</i>	The index of the interface to get the protocol number for. <i>idx</i> must be $\geq 0$ and $<$ the return value of <a href="#">num_interfaces()</a> .
out	<i>proto</i>	The protocol number for interface <i>idx</i> .
out	<i>name</i>	The protocol name for interface <i>idx</i> .

## Return values

<i>l4_msgtag_t::label()</i> == 0	Successful; see `proto` and `name`.
<i>l4_msgtag_t::label()</i> < 0	Error code.

**15.183.2.2 num\_interfaces()**

```
l4_msgtag_t L4::Meta::num_interfaces ( )
```

Get the number of interfaces implemented by this object.

## Return values

<i>l4_msgtag_t::label()</i> $\geq 0$	The number of supported interfaces.
<i>l4_msgtag_t::label()</i> < 0	Error code of the occurred error.

**15.183.2.3 supports()**

```
l4_msgtag_t L4::Meta::supports (
    l4_mword_t protocol )
```

Figure out if the object supports the given protocol (number).

## Parameters

<i>protocol</i>	The protocol number to check for.
-----------------	-----------------------------------

## Return values

<i>l4_msgtag_t::label()</i> == 1	protocol is supported.
<i>l4_msgtag_t::label()</i> == 0	protocol is not supported.

This method is intended to be used for statically assigned protocol numbers.

The documentation for this class was generated from the following file:

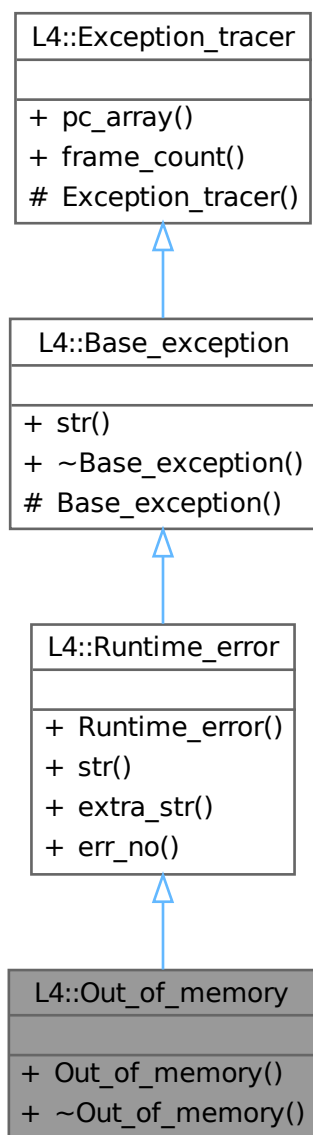
- [l4/sys/meta](#)

## 15.184 L4::Out\_of\_memory Class Reference

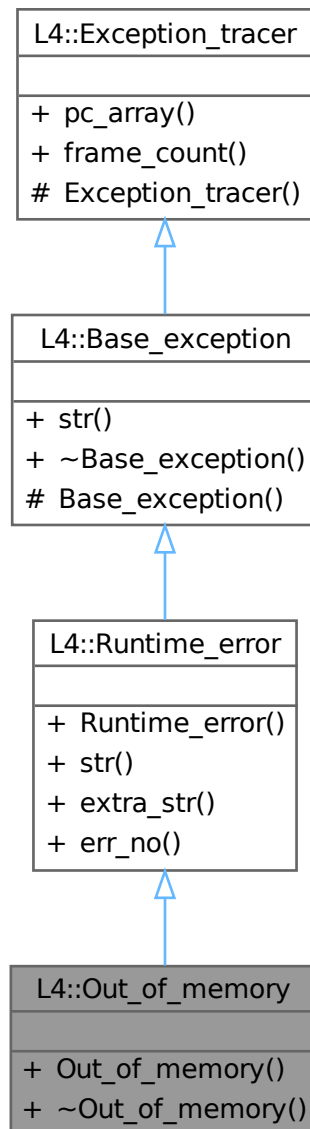
[Exception](#) signalling insufficient memory.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Out\_of\_memory:



Collaboration diagram for L4::Out\_of\_memory:



### Public Member Functions

- **Out\_of\_memory** (char const \*extra="") noexcept  
*Create an out-of-memory exception.*
- **~Out\_of\_memory** () noexcept  
*Destruction.*

### Public Member Functions inherited from [L4::Runtime\\_error](#)

- [Runtime\\_error](#) (long err\_no, char const \*extra=0) noexcept

Create a new [Runtime\\_error](#).

- char const \* **str** () const noexcept override  
Return a human readable string for the exception.
- char const \* **extra\_str** () const  
Get the description text for this runtime error.
- long **err\_no** () const noexcept  
Get the error value for this runtime error.

## Public Member Functions inherited from [L4::Base\\_exception](#)

- virtual ~**Base\_exception** () noexcept  
Destruction.

## Public Member Functions inherited from [L4::Exception\\_tracer](#)

- void const \*const \* **pc\_array** () const noexcept  
Get the array containing the call trace.
- int **frame\_count** () const noexcept  
Get the number of entries that are valid in the call trace.

## Additional Inherited Members

## Protected Member Functions inherited from [L4::Base\\_exception](#)

- **Base\_exception** () noexcept  
Create a base exception.

## Protected Member Functions inherited from [L4::Exception\\_tracer](#)

- **Exception\_tracer** () noexcept  
Create a back trace.

### 15.184.1 Detailed Description

[Exception](#) signalling insufficient memory.

Definition at line 188 of file [exceptions](#).

The documentation for this class was generated from the following file:

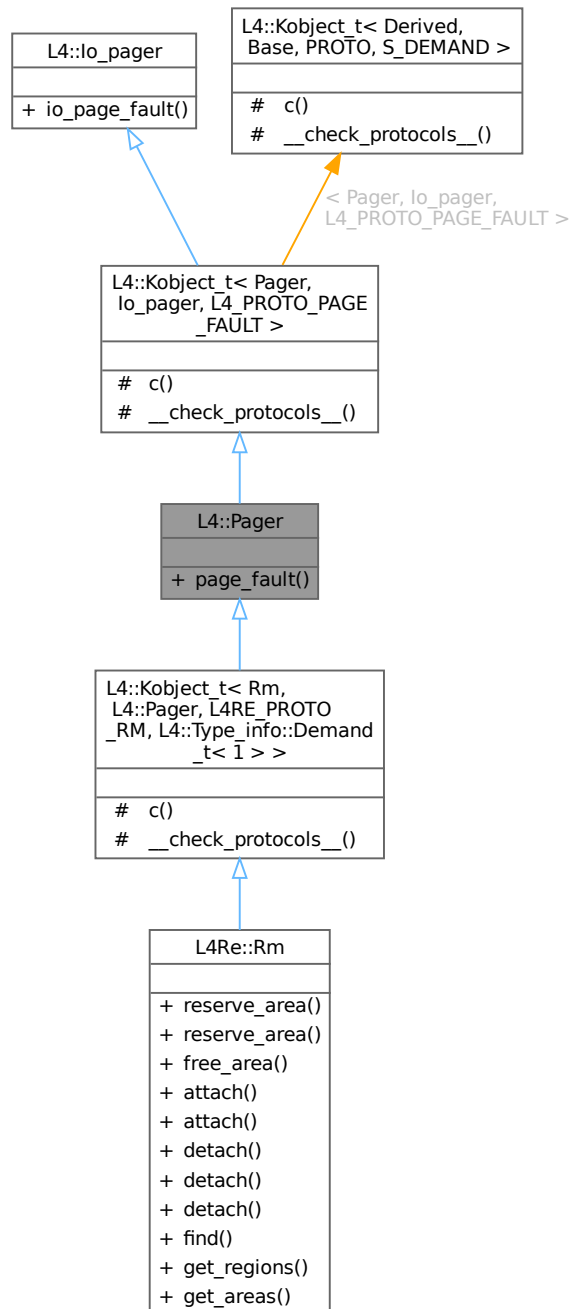
- [l4/cxx/exceptions](#)

## 15.185 L4::Pager Class Reference

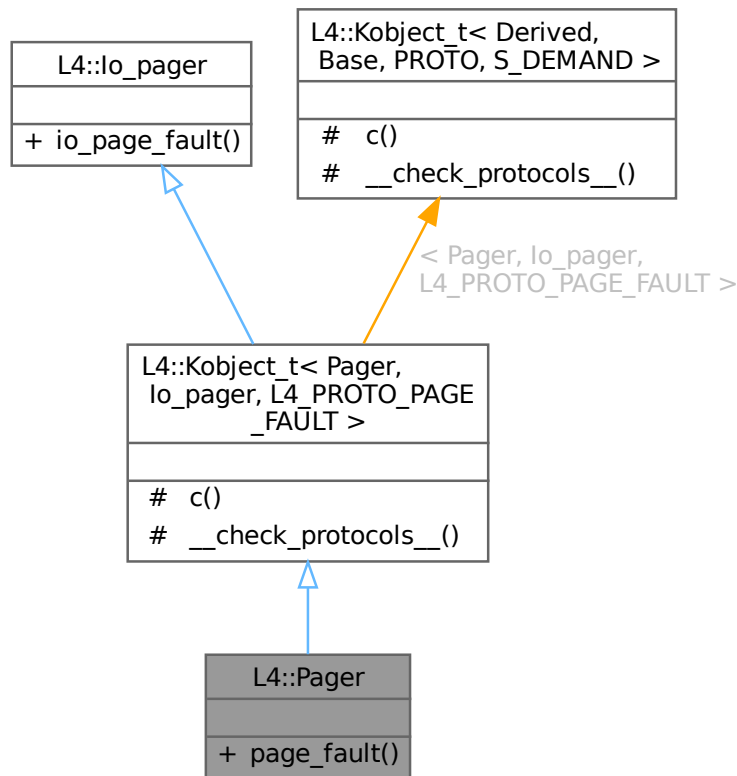
[Pager](#) interface including the [lo\\_pager](#) interface.

```
#include <pager>
```

Inheritance diagram for L4::Pager:



Collaboration diagram for L4::Pager:



### Public Member Functions

- [l4\\_msgtag\\_t page\\_fault](#) ([l4\\_umword\\_t](#) pfa, [l4\\_umword\\_t](#) pc, [L4::lpc::Rcv\\_fpage](#) rwin, [L4::lpc::Opt<L4::lpc::Snd\\_fpage & >](#) fp)  
*Page-fault protocol message.*

### Public Member Functions inherited from [L4::lo\\_pager](#)

- [l4\\_msgtag\\_t io\\_page\\_fault](#) ([l4\\_fpage\\_t](#) io\_pfa, [l4\\_umword\\_t](#) pc, [L4::lpc::Rcv\\_fpage](#) rwin, [L4::lpc::Opt<L4::lpc::Snd\\_fpage & >](#) fp)  
*IO page fault protocol message.*

### Additional Inherited Members

### Protected Types inherited from

[L4::Kobject\\_t<Pager, lo\\_pager, L4\\_PROTO\\_PAGE\\_FAULT >](#)

- typedef [Pager](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef [Typeid::Iface<PROTO, Pager >](#) **\_\_Iface**  
*The interface description for the derived class.*
- typedef [Typeid::Merge\\_list<Typeid::Iface\\_list<\\_\\_Iface >, typename Base::\\_\\_Iface\\_list >](#) **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from [L4::Kobject\\_t<Pager, lo\\_pager, L4\\_PROTO\\_PAGE\\_FAULT>](#)

- [L4::Cap<Class>c\(\)](#) const noexcept  
*Get the capability to ourselves.*

## Static Protected Member Functions inherited from [L4::Kobject\\_t<Pager, lo\\_pager, L4\\_PROTO\\_PAGE\\_FAULT>](#)

- static void [\\_\\_check\\_protocols\\_\\_\(\)](#) noexcept  
*Helper to check for protocol conflicts.*

### 15.185.1 Detailed Description

[Pager](#) interface including the [lo\\_pager](#) interface.

This class defines the interface for handling page fault IPC. If a thread causes a page fault, the microkernel synthesises a page fault IPC message and sends it to the thread's page fault handler (pager). The pager can then handle the message, for example by establishing a suitable page mapping.

The page fault handler is set with the [L4::Thread::control](#) interface.

Definition at line 98 of file [pager](#).

### 15.185.2 Member Function Documentation

#### 15.185.2.1 [page\\_fault\(\)](#)

```
l4_msgtag_t L4::Pager::page_fault (
    l4_umword_t pfa,
    l4_umword_t pc,
    L4::Ipc::Rcv_fpage rwin,
    L4::Ipc::Opt< L4::Ipc::Snd_fpage & > fp )
```

Page-fault protocol message.

#### Parameters

	<i>pfa</i>	Faulting address including failure reason: bits [0:2].
	<i>pc</i>	Faulting program counter.
	<i>rwin</i>	Receive window for a flex-page mapping resolving the page fault.
out	<i>fp</i>	Optional: flex-page descriptor to send to the task raising the page fault.

#### Returns

System call message tag; use [l4\\_error\(\)](#) to check for errors.

Page-fault messages are usually generated by the kernel and need to be handled by an appropriate handler function, potentially filling in *fp* for the reply.



pfa encoding is as shown:

[63/31 .. 3]	2	1	0
PFA	X	W	r

- **PFA** Bits 63/31..3 of `pfa` are the page fault address bits 63/31 to 3, bits 2..0 are masked.
- **X** Bit 2 of `pfa` if set, indicates a page fault during instruction fetch. Note, this bit is implementation-defined and might always be clear. Therefore, if this bit is clear it does not imply that the page fault is not due to an instruction fetch.
- **W** Bit 1 of `pfa` is set to 1 for a page fault due to a write operation.
- **r** Bit0: reserved, undefined.

The documentation for this class was generated from the following file:

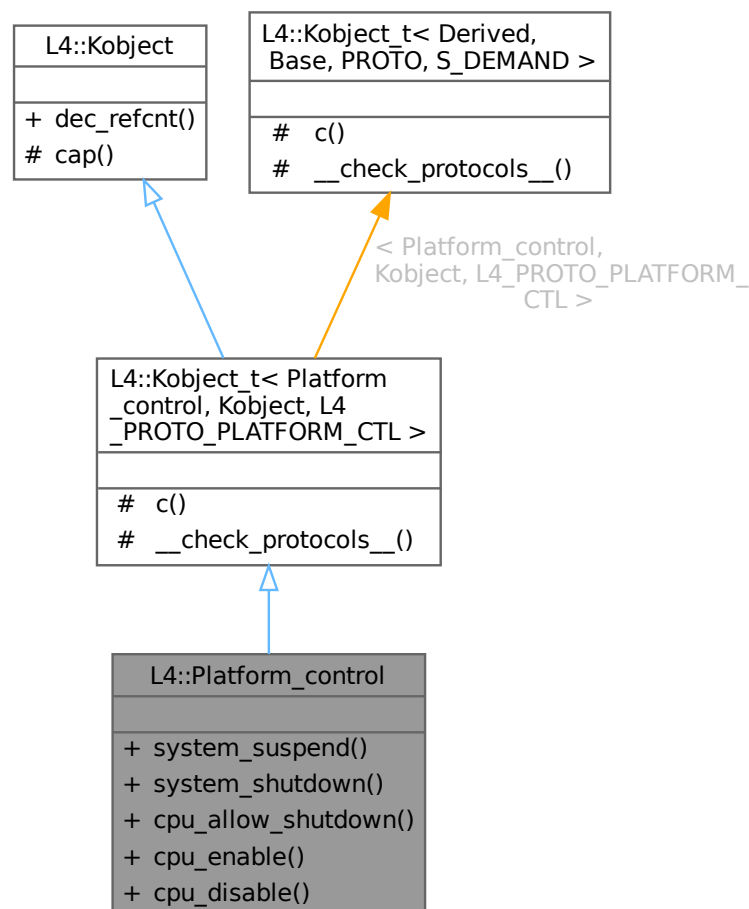
- [l4/sys/pager](#)

## 15.186 L4::Platform\_control Class Reference

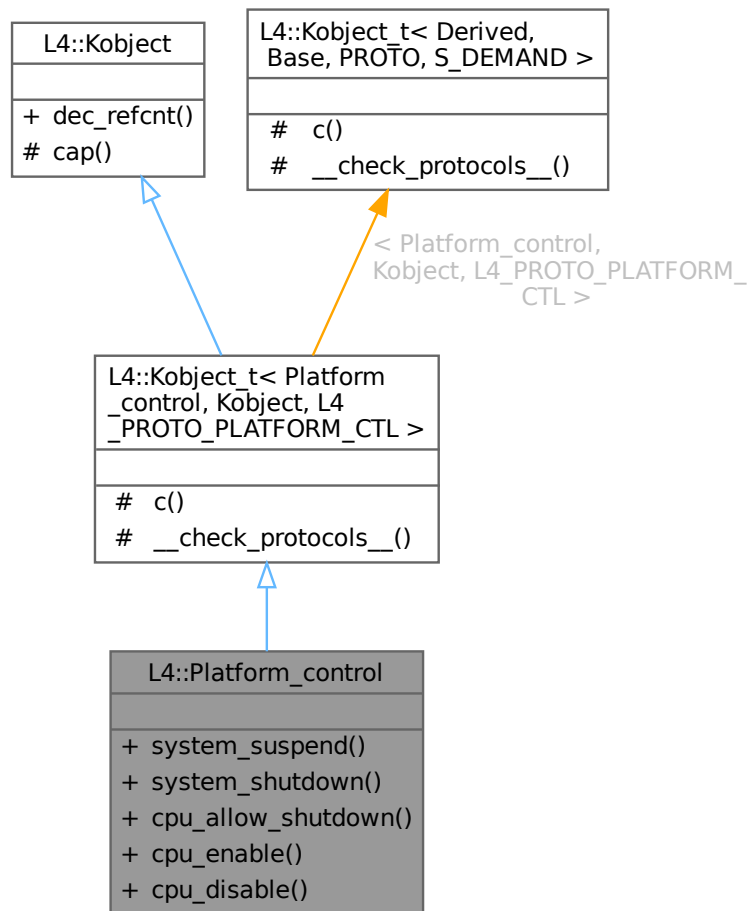
[L4](#) C++ interface for controlling platform-wide properties, see [Platform Control C API](#) for the C interface.

```
#include <platform_control>
```

Inheritance diagram for L4::Platform\_control:



Collaboration diagram for L4::Platform\_control:



## Public Member Functions

- [l4\\_msgtag\\_t system\\_suspend](#) ([l4\\_umword\\_t](#) extras)  
*Enter suspend to RAM.*
- [l4\\_msgtag\\_t system\\_shutdown](#) ([l4\\_umword\\_t](#) reboot)  
*Shutdown/Reboot the system.*
- [l4\\_msgtag\\_t cpu\\_allow\\_shutdown](#) ([l4\\_umword\\_t](#) phys\_id, [l4\\_umword\\_t](#) enable)  
*Allow CPU shutdown.*
- [l4\\_msgtag\\_t cpu\\_enable](#) ([l4\\_umword\\_t](#) phys\_id)  
*Enable an offline CPU.*
- [l4\\_msgtag\\_t cpu\\_disable](#) ([l4\\_umword\\_t](#) phys\_id)  
*Disable an online CPU.*

## Public Member Functions inherited from L4::Kobject

- [l4\\_msgtag\\_t dec\\_refcnt](#) ([l4\\_mword\\_t](#) diff, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#))  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

### Protected Types inherited from

[L4::Kobject\\_t](#)< [Platform\\_control](#), [Kobject](#), [L4\\_PROTO\\_PLATFORM\\_CTL](#) >

- typedef [Platform\\_control](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, [Platform\\_control](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Platform\\_control](#), [Kobject](#), [L4\\_PROTO\\_PLATFORM\\_CTL](#) >

- [L4::Cap](#)< **Class** > **c** () const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t](#) **cap** () const noexcept  
*Return capability selector.*

### Static Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Platform\\_control](#), [Kobject](#), [L4\\_PROTO\\_PLATFORM\\_CTL](#) >

- static void **\_\_check\_protocols** () noexcept  
*Helper to check for protocol conflicts.*

## 15.186.1 Detailed Description

[L4](#) C++ interface for controlling platform-wide properties, see [Platform Control C API](#) for the C interface.

Add

```
#include <l4/sys/platform_control>
```

to your code to use the platform control functions. The API allows a client to suspend, reboot or shutdown the system.

For the C interface refer to the [Platform Control C API](#).

Definition at line 47 of file [platform\\_control](#).

## 15.186.2 Member Function Documentation

### 15.186.2.1 [cpu\\_allow\\_shutdown\(\)](#)

```
l4\_msgtag\_t L4::Platform\_control::cpu\_allow\_shutdown (  
    l4\_umword\_t phys_id,  
    l4\_umword\_t enable )
```

Allow CPU shutdown.

## Parameters

<i>phys↔ _id</i>	Physical CPU id of CPU (e.g. local APIC id) to disable.
<i>enable</i>	Allow shutdown when 1, disallow when 0.

Sets or unsets a hint that a CPU that is not currently used may be powered down.

**15.186.2.2 cpu\_disable()**

```
l4_msgtag_t L4::Platform_control::cpu_disable (
    l4_umword_t phys_id )
```

Disable an online CPU.

## Parameters

<i>phys↔ _id</i>	Physical CPU id of CPU (e.g. local APIC id) to disable.
----------------------	---

## Returns

System call message tag

This function is currently only supported on the ARM EXYNOS platform.

**15.186.2.3 cpu\_enable()**

```
l4_msgtag_t L4::Platform_control::cpu_enable (
    l4_umword_t phys_id )
```

Enable an offline CPU.

## Parameters

<i>phys↔ _id</i>	Physical CPU id of CPU (e.g. local APIC id) to enable.
----------------------	--

## Returns

System call message tag

This function is currently only supported on the ARM EXYNOS platform.

**15.186.2.4 system\_shutdown()**

```
l4_msgtag_t L4::Platform_control::system_shutdown (
    l4_umword_t reboot )
```

Shutdown/Reboot the system.

## Parameters

<i>reboot</i>	1 for reboot, 0 for power off
---------------	-------------------------------

**15.186.2.5 system\_suspend()**

```
l4_msgtag_t L4::Platform_control::system_suspend (
    l4_umword_t extras )
```

Enter suspend to RAM.

## Precondition

Must only be invoked on the boot CPU. Furthermore it must be ensured that the invoking thread is not migrated to a different CPU during the suspend.

## Parameters

<i>extras</i>	Some extra platform-specific information needed to enter suspend to RAM. On x86 platforms and when using the <a href="#">Platform_control</a> object provided by Fiasco, the value defines the sleep state. The sleep states are defined in the ACPI table. Other platforms as well as Io's <a href="#">Platform_control</a> object don't make use of this value at the moment.
---------------	---

The documentation for this class was generated from the following file:

- [l4/sys/platform\\_control](#)

**15.187 L4::Poll\_timeout\_counter Class Reference**

Evaluate an expression for a maximum number of times.

```
#include <poll_timeout_counter.h>
```

Collaboration diagram for L4::Poll\_timeout\_counter:

L4::Poll_timeout_counter
<ul style="list-style-type: none"> <li>+ Poll_timeout_counter()</li> <li>+ set()</li> <li>+ test()</li> <li>+ timed_out()</li> </ul>

## Public Member Functions

- [Poll\\_timeout\\_counter](#) (unsigned counter\_val)  
*Constructor.*
- void [set](#) (unsigned counter\_val)  
*Set the counter to a certain value.*
- bool [test](#) (bool expression=true)  
*Evaluate the expression for a maximum number of times.*
- bool [timed\\_out](#) () const  
*Indicator if the maximum number of tests was required.*

### 15.187.1 Detailed Description

Evaluate an expression for a maximum number of times.

A typical use case is testing for a bit change in a hardware register for a maximum number of times (polling). For example:

```
{c++}
Mmio_register_block regs;
Poll_timeout_counter i(3000000);
while (i.test(!(regs.read<14_uint32_t>(0x04) & 1)))
;
```

The following usage is **wrong**:

```
{c++}
...
Poll_timeout_counter i(3000000);
while (!i.test((regs.read<14_uint32_t>(0x04) & 1)))
;
```

This loop would never terminate if the hardware register doesn't change!

Definition at line 36 of file [poll\\_timeout\\_counter.h](#).

### 15.187.2 Constructor & Destructor Documentation

#### 15.187.2.1 Poll\_timeout\_counter()

```
L4::Poll_timeout_counter::Poll_timeout_counter (
    unsigned counter_val ) [inline]
```

Constructor.

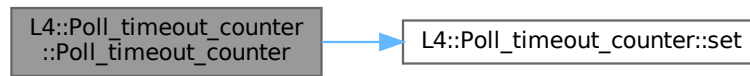
#### Parameters

<i>counter_val</i>	Maximum number of times to repeat the test.
--------------------	---

Definition at line 44 of file [poll\\_timeout\\_counter.h](#).

References [set\(\)](#).

Here is the call graph for this function:



## 15.187.3 Member Function Documentation

### 15.187.3.1 set()

```
void L4::Poll_timeout_counter::set (
    unsigned counter_val ) [inline]
```

Set the counter to a certain value.

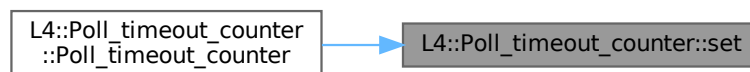
#### Parameters

<i>counter_val</i>	New counter value for maximum number of times to repeat the test.
--------------------	---

Definition at line 55 of file [poll\\_timeout\\_counter.h](#).

Referenced by [Poll\\_timeout\\_counter\(\)](#).

Here is the caller graph for this function:



### 15.187.3.2 timed\_out()

```
bool L4::Poll_timeout_counter::timed_out ( ) const [inline]
```

Indicator if the maximum number of tests was required.

#### Return values

<i>true,if</i>	the maximum number of tests was required or if the counter was initialized to zero.
----------------	---



Definition at line 83 of file [poll\\_timeout\\_counter.h](#).

The documentation for this class was generated from the following file:

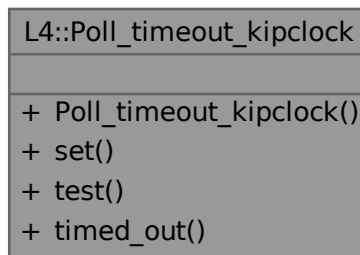
- [pkg/drivers-frst/include/poll\\_timeout\\_counter.h](#)

## 15.188 L4::Poll\_timeout\_kipclock Class Reference

A polling timeout based on the [L4Re](#) clock.

```
#include <poll_timeout_kipclock>
```

Collaboration diagram for L4::Poll\_timeout\_kipclock:



### Public Member Functions

- [Poll\\_timeout\\_kipclock](#) (unsigned poll\_time\_us)  
*Initialise relative timeout in microseconds.*
- void [set](#) (unsigned poll\_time\_us)  
*(Re-)Set relative timeout in microseconds*
- bool [test](#) (bool expression=true)  
*Test whether timeout has expired.*
- bool [timed\\_out](#) () const  
*Query whether timeout has expired.*

### 15.188.1 Detailed Description

A polling timeout based on the [L4Re](#) clock.

This class allows to conveniently add a timeout to a polling loop.

The original

```
while (device.read(State) & Busy)
;
```

is converted to

```
Poll_timeout_kipclock timeout(10000);
while (timeout.test(device.read(State) & Busy))
;
if (timeout.timed_out())
    printf("ERROR: Device does not respond.\n");
```

Definition at line 38 of file [poll\\_timeout\\_kipclock](#).

## 15.188.2 Constructor & Destructor Documentation

### 15.188.2.1 Poll\_timeout\_kipclock()

```
L4::Poll_timeout_kipclock::Poll_timeout_kipclock (
    unsigned poll_time_us ) [inline]
```

Initialise relative timeout in microseconds.

#### Parameters

<i>poll_time_us</i>	Polling timeout in microseconds.
---------------------	----------------------------------

Definition at line 45 of file [poll\\_timeout\\_kipclock](#).

References [set\(\)](#).

Here is the call graph for this function:



## 15.188.3 Member Function Documentation

### 15.188.3.1 set()

```
void L4::Poll_timeout_kipclock::set (
    unsigned poll_time_us ) [inline]
```

(Re-)Set relative timeout in microseconds

#### Parameters

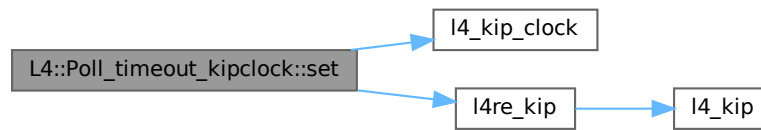
<i>poll_time_us</i>	Polling timeout in microseconds.
---------------------	----------------------------------

Definition at line 54 of file [poll\\_timeout\\_kipclock](#).

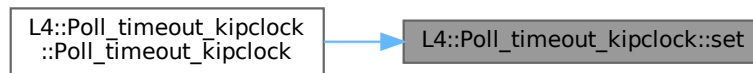
References [l4\\_kip\\_clock\(\)](#), and [l4re\\_kip\(\)](#).

Referenced by [Poll\\_timeout\\_kipclock\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.188.3.2 test()

```
bool L4::Poll_timeout_kipclock::test (
    bool expression = true ) [inline]
```

Test whether timeout has expired.

#### Parameters

<i>expression</i>	Optional expression.
-------------------	----------------------

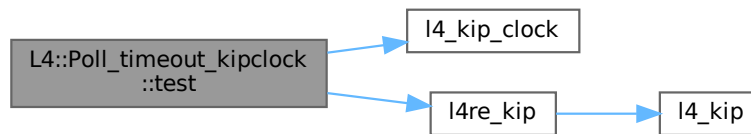
#### Return values

<i>false</i>	The timeout has expired or the given expression returned false.
<i>true</i>	The timeout has not expired and the optionally given expression returns true.

Definition at line 68 of file [poll\\_timeout\\_kipclock](#).

References [l4\\_kip\\_clock\(\)](#), and [l4re\\_kip\(\)](#).

Here is the call graph for this function:



### 15.188.3.3 timed\_out()

```
bool L4::Poll_timeout_kipclock::timed_out ( ) const [inline]
```

Query whether timeout has expired.

#### Returns

Expiry state of timeout

Definition at line 80 of file [poll\\_timeout\\_kipclock](#).

The documentation for this class was generated from the following file:

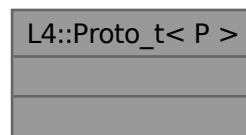
- `l4/re/util/poll_timeout_kipclock`

## 15.189 L4::Proto\_t< P > Struct Template Reference

Data type for defining protocol numbers.

```
#include <__typeinfo.h>
```

Collaboration diagram for `L4::Proto_t< P >`:



### 15.189.1 Detailed Description

```
template<long P = PROTO_EMPTY>
struct L4::Proto_t< P >
```

Data type for defining protocol numbers.

## Template Parameters

<i>P</i>	The protocol number itself
----------	----------------------------

This type must be used when specifying a protocol number with [Kobject\\_x](#).

Definition at line 1192 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

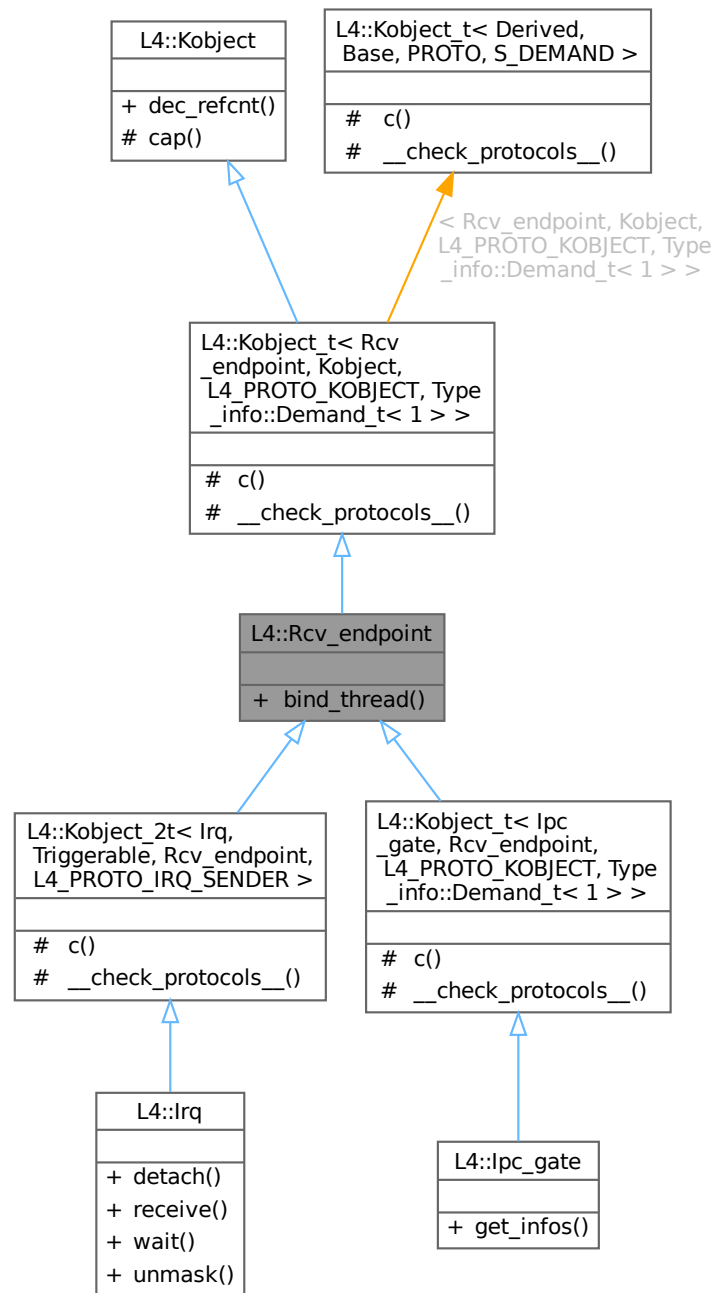
- [l4/sys/\\_\\_typeinfo.h](#)

## 15.190 L4::Rcv\_endpoint Class Reference

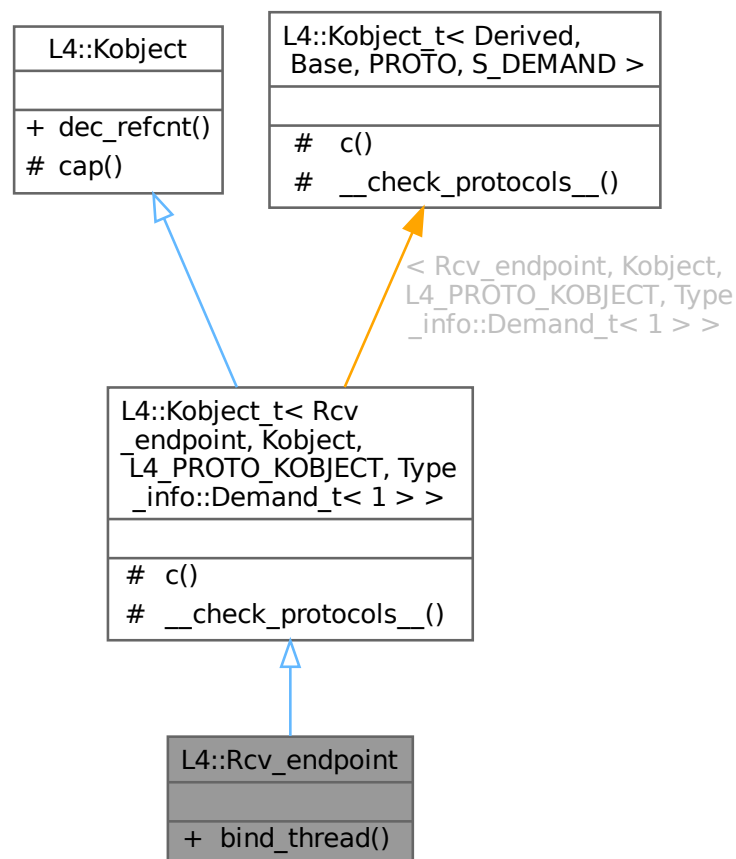
Interface for kernel objects that allow to receive IPC from them.

```
#include <rcv_endpoint>
```

Inheritance diagram for L4::Rcv\_endpoint:



Collaboration diagram for L4::Rcv\_endpoint:



## Public Member Functions

- [l4\\_msgtag\\_t bind\\_thread](#) ([ipc::Cap](#)< [Thread](#) > t, [l4\\_umword\\_t](#) label)  
*Bind a thread to an IPC receive endpoint.*

## Public Member Functions inherited from [L4::Kobject](#)

- [l4\\_msgtag\\_t dec\\_refcnt](#) ([l4\\_mword\\_t](#) diff, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#))  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

## Protected Types inherited from

[L4::Kobject\\_t](#)< [Rcv\\_endpoint](#), [Kobject](#), [L4\\_PROTO\\_KOBJECT](#), [Type\\_info::Demand\\_t](#)< 1 > >

- typedef [Rcv\\_endpoint](#) **Class**

The target interface type (inheriting from [Kobject\\_t](#))

- typedef Typeid::Iface< PROTO, [Rcv\\_endpoint](#) > \_\_Iface

The interface description for the derived class.

- typedef Typeid::Merge\_list< Typeid::Iface\_list< \_\_Iface >, typename Base::\_\_Iface\_list > \_\_Iface\_list

The list of all RPC interfaces provided directly or through inheritance.

### Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Rcv\\_endpoint](#), [Kobject](#), [L4\\_PROTO\\_KOBJECT](#), [Type\\_info::Demand\\_t](#)< 1 > >

- [L4::Cap](#)< [Class](#) > [c](#) () const noexcept

Get the capability to ourselves.

### Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t](#) [cap](#) () const noexcept

Return capability selector.

### Static Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Rcv\\_endpoint](#), [Kobject](#), [L4\\_PROTO\\_KOBJECT](#), [Type\\_info::Demand\\_t](#)< 1 > >

- static void \_\_check\_protocols\_\_ () noexcept

Helper to check for protocol conflicts.

## 15.190.1 Detailed Description

Interface for kernel objects that allow to receive IPC from them.

Such an object is for example an [lpc\\_gate](#) (with server rights) or an [lrq](#). Those objects allow to bind a thread that shall receive IPC from these object via [bind\\_thread\(\)](#).

Definition at line 40 of file [rcv\\_endpoint](#).

## 15.190.2 Member Function Documentation

### 15.190.2.1 [bind\\_thread\(\)](#)

```
l4_msgtag_t L4::Rcv_endpoint::bind_thread (
    Ipc::Cap< Thread > t,
    l4_umword_t label )
```

Bind a thread to an IPC receive endpoint.

#### Parameters

<i>t</i>	<a href="#">Thread</a> object that shall be bound to this receive endpoint.
<i>label</i>	Label to assign to <code>this</code> receive endpoint. The two least significant bits should usually be set to zero.



### Returns

Syscall return tag containing one of the following return codes.

### Return values

<code>L4_EOK</code>	Operation successful.
<code>-L4_EINVAL</code>	<code>t</code> is not a thread object or other arguments were malformed.
<code>-L4_EPERM</code>	No <a href="#">L4_CAP_FPAGE_S</a> right on <code>t</code> or the capability used to invoke this operation.

### Precondition

If this operation is invoked using an IPC gate capability without the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) right, the kernel will not perform the operation. Instead, the underlying IPC message will be forwarded to the thread bound to the IPC gate, blocking the caller if no thread is bound yet.

The specified `label` is passed to the receiver of the incoming IPC. It is possible to re-bind a receive endpoint to the same or a different thread. In this case, IPC already in flight will be delivered with the old label to the previously bound thread unless [L4::Thread::modify\\_senders\(\)](#) is used to change these labels.

The documentation for this class was generated from the following file:

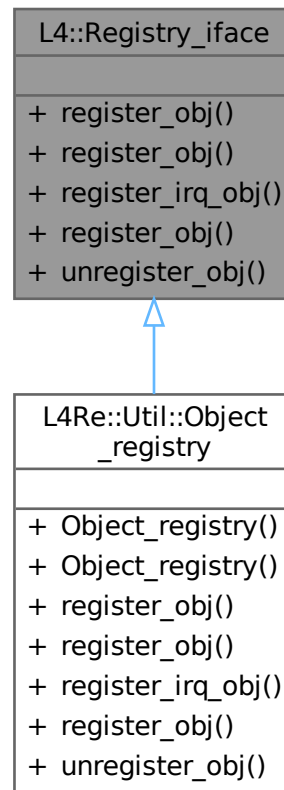
- [l4/sys/rcv\\_endpoint](#)

## 15.191 L4::Registry\_iface Class Reference

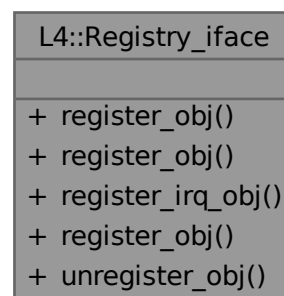
Abstract interface for object registries.

```
#include <ipc_epiface>
```

Inheritance diagram for L4::Registry\_iface:



Collaboration diagram for L4::Registry\_iface:



## Public Member Functions

- virtual [L4::Cap](#)< void > [register\\_obj](#) ([L4::Epiface](#) \*o, char const \*service)=0

- Register an [L4::Epiface](#) for an IPC gate available in the applications environment under the name *service*.*
- virtual [L4::Cap](#)< void > [register\\_obj](#) ([L4::Epiface](#) \*o)=0  
*Register o as server-side object for synchronous RPC.*
- virtual [L4::Cap](#)< [L4::Irq](#) > [register\\_irq\\_obj](#) ([L4::Epiface](#) \*o)=0  
*Register o as server-side object for asynchronous IRQs.*
- virtual [L4::Cap](#)< [L4::Rcv\\_endpoint](#) > [register\\_obj](#) ([L4::Epiface](#) \*o, [L4::Cap](#)< [L4::Rcv\\_endpoint](#) > ep)=0  
*Register o as server-side object for a pre-allocated capability.*
- virtual void [unregister\\_obj](#) ([L4::Epiface](#) \*o, bool unmap=true)=0  
*Unregister the given object o from the server.*

### 15.191.1 Detailed Description

Abstract interface for object registries.

An object registry allows to register [L4::Epiface](#) objects at a server loop either for synchronous RPC messages or for asynchronous IRQ messages.

Definition at line 333 of file [ipc\\_epiface](#).

### 15.191.2 Member Function Documentation

#### 15.191.2.1 [register\\_irq\\_obj\(\)](#)

```
virtual L4::Cap< L4::Irq > L4::Registry\_iface::register\_irq\_obj (  
    L4::Epiface * o ) [pure virtual]
```

Register o as server-side object for asynchronous IRQs.

##### Parameters

<i>o</i>	Pointer to an <a href="#">Epiface</a> object that shall be registered as server-side object for IRQs.
----------	---

##### Return values

<a href="#">L4::Cap</a> < <a href="#">L4::Irq</a> >	Capability to a new IRQ object on success.
<a href="#">L4::Cap</a> < <a href="#">L4::Irq</a> >:: <a href="#">Invalid</a>	The allocation of the IRQ has failed.

After successful registration `o->obj_cap()` will be the capability of the allocated IRQ object.

The function may allocate a capability slot for the object. In that case [unregister\\_obj\(\)](#) is responsible for freeing the slot as well.

Implemented in [L4Re::Util::Object\\_registry](#).

#### 15.191.2.2 [register\\_obj\(\)](#) [1/3]

```
virtual L4::Cap< void > L4::Registry\_iface::register\_obj (  
    L4::Epiface * o ) [pure virtual]
```

Register o as server-side object for synchronous RPC.

## Parameters

<i>o</i>	Pointer to an <a href="#">Epiface</a> object that shall be registered as server-side object for RPC.
----------	--

## Return values

<i>L4::Cap&lt;void&gt;</i>	A valid capability to a new IPC gate.
<i>L4::Cap&lt;void&gt;::Invalid</i>	The allocation of the IPC gate has failed.

After successful registration `o->obj_cap()` will be the capability of the allocated IPC gate.

The function may allocate a capability slot for the object. In that case [unregister\\_obj\(\)](#) is responsible for freeing the slot as well.

Implemented in [L4Re::Util::Object\\_registry](#).

**15.191.2.3 register\_obj()** [2/3]

```
virtual L4::Cap< void > L4::Registry_iface::register_obj (
    L4::Epiface * o,
    char const * service ) [pure virtual]
```

Register an [L4::Epiface](#) for an IPC gate available in the applications environment under the name `service`.

## Parameters

<i>o</i>	Pointer to an <a href="#">Epiface</a> object that shall be registered.
<i>service</i>	Name of the capability that shall be used to connect <code>o</code> to as a server-side object.

## Return values

<i>L4::Cap&lt;void&gt;</i>	The capability known as <code>service</code> on success.
<i>L4::Cap&lt;void&gt;::Invalid</i>	No capability with the given name found.

After a successful call to this function `o->obj_cap()` is equal to the capability in the environment with the name given by `service`.

Implemented in [L4Re::Util::Object\\_registry](#).

**15.191.2.4 register\_obj()** [3/3]

```
virtual L4::Cap< L4::Rcv_endpoint > L4::Registry_iface::register_obj (
    L4::Epiface * o,
    L4::Cap< L4::Rcv_endpoint > ep ) [pure virtual]
```

Register `o` as server-side object for a pre-allocated capability.

## Parameters

<i>o</i>	Pointer to an <a href="#">Epiface</a> object that shall be registered as server-side object.
<i>ep</i>	Capability to an already allocated capability where <i>o</i> shall be attached as server-side handler. The capability may point to an IPC gate or an IRQ.

## Return values

<i>L4::Cap&lt;L4::Rcv_endpoint&gt;</i>	Capability <i>ep</i> on success.
<i>L4::Cap&lt;L4::Rcv_endpoint&gt;::Invalid</i>	The IRQ attach operation has failed.

After successful registration *o*→*obj\_cap()* will be equal to *ep*.

Implemented in [L4Re::Util::Object\\_registry](#).

## 15.191.2.5 unregister\_obj()

```
virtual void L4::Registry_iface::unregister_obj (
    L4::Epiface * o,
    bool unmap = true ) [pure virtual]
```

Unregister the given object *o* from the server.

## Parameters

<i>o</i>	Pointer to the <a href="#">Epiface</a> object that shall be unregistered. The object must have been registered with any of the register methods if <a href="#">Registry_iface</a> .
<i>unmap</i>	If true the capability <i>o</i> → <i>obj_cap()</i> shall be unmapped from the local object space.

The function always unmaps and frees the capability if it was allocated by either [Registry\\_iface::register\\_irq\\_obj\(L4::Epiface \\*\)](#), or by [Registry\\_iface::register\\_obj\(L4::Epiface \\*\)](#).

Implemented in [L4Re::Util::Object\\_registry](#).

The documentation for this class was generated from the following file:

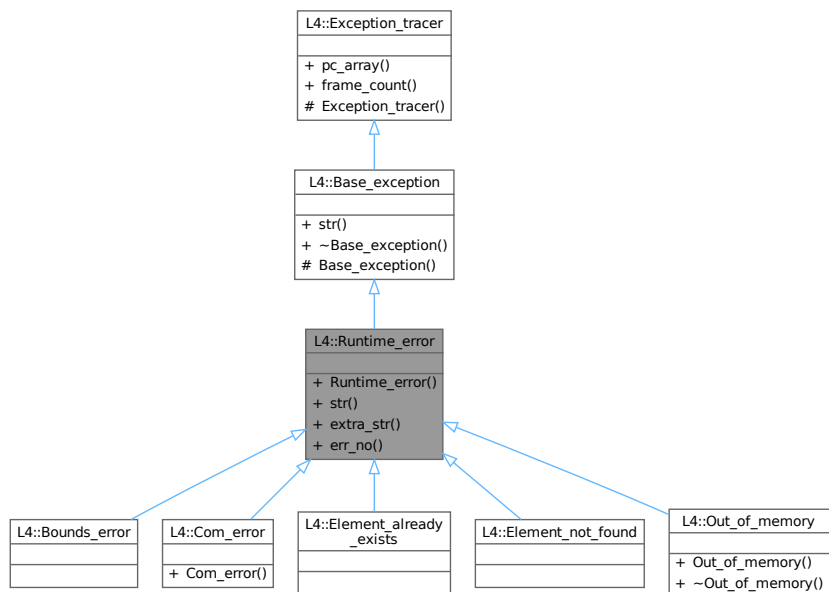
- `l4/sys/cxx/ipc_epiface`

## 15.192 L4::Runtime\_error Class Reference

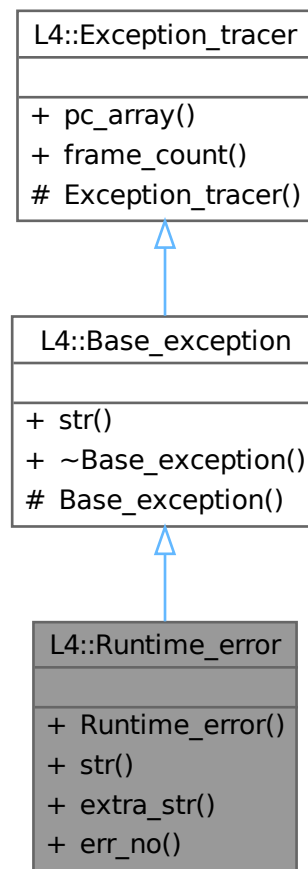
[Exception](#) for an abstract runtime error.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Runtime\_error:



Collaboration diagram for L4::Runtime\_error:



### Public Member Functions

- [Runtime\\_error](#) (long [err\\_no](#), char const \*extra=0) noexcept  
Create a new [Runtime\\_error](#).
- char const \* [str](#) () const noexcept override  
Return a human readable string for the exception.
- char const \* [extra\\_str](#) () const  
Get the description text for this runtime error.
- long [err\\_no](#) () const noexcept  
Get the error value for this runtime error.

### Public Member Functions inherited from [L4::Base\\_exception](#)

- virtual `~Base_exception` () noexcept  
Destruction.

## Public Member Functions inherited from [L4::Exception\\_tracer](#)

- `void const *const * pc_array () const noexcept`  
*Get the array containing the call trace.*
- `int frame_count () const noexcept`  
*Get the number of entries that are valid in the call trace.*

## Additional Inherited Members

## Protected Member Functions inherited from [L4::Base\\_exception](#)

- `Base_exception () noexcept`  
*Create a base exception.*

## Protected Member Functions inherited from [L4::Exception\\_tracer](#)

- `Exception_tracer () noexcept`  
*Create a back trace.*

### 15.192.1 Detailed Description

[Exception](#) for an abstract runtime error.

This is the base class for a set of exceptions that cover all errors that have a C error value (see [l4\\_error\\_code\\_t](#)).

Definition at line [139](#) of file [exceptions](#).

### 15.192.2 Constructor & Destructor Documentation

#### 15.192.2.1 `Runtime_error()`

```
L4::Runtime_error::Runtime_error (
    long err_no,
    char const * extra = 0 ) [inline], [explicit], [noexcept]
```

Create a new [Runtime\\_error](#).

#### Parameters

<i>err_no</i>	Error value for this runtime error.
<i>extra</i>	Description of what was happening while the error occurred.

Definition at line [152](#) of file [exceptions](#).



### 15.192.3 Member Function Documentation

#### 15.192.3.1 err\_no()

```
long L4::Runtime_error::err_no ( ) const [inline], [noexcept]
```

Get the error value for this runtime error.

##### Returns

Error value.

Definition at line 181 of file [exceptions](#).

#### 15.192.3.2 extra\_str()

```
char const * L4::Runtime_error::extra_str ( ) const [inline]
```

Get the description text for this runtime error.

##### Returns

Pointer to the description string.

Definition at line 173 of file [exceptions](#).

The documentation for this class was generated from the following file:

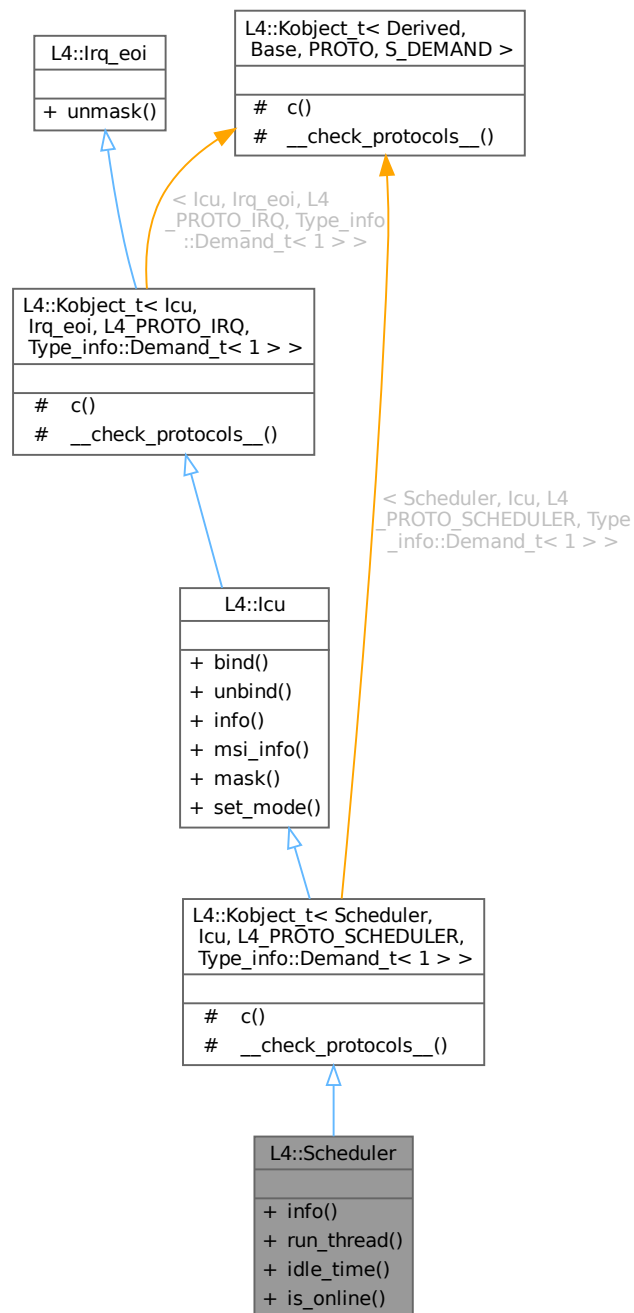
- [l4/cxx/exceptions](#)

## 15.193 L4::Scheduler Class Reference

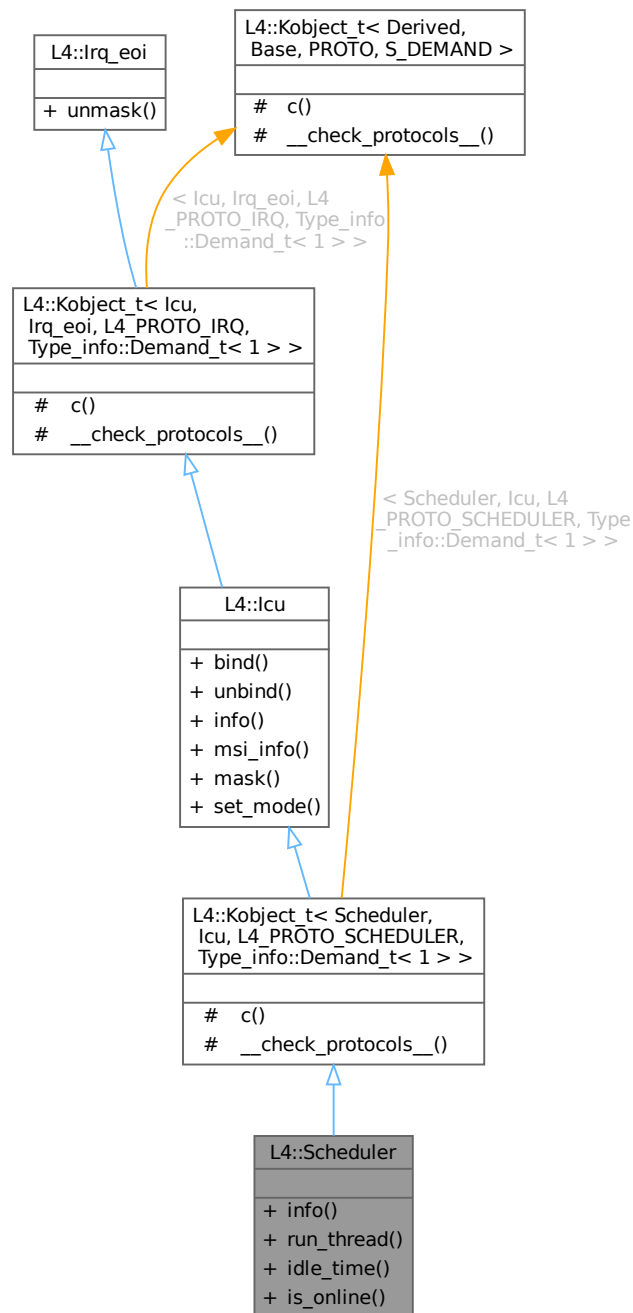
C++ interface of the [Scheduler](#) kernel object, see [Scheduler](#) for the C interface.

```
#include <scheduler>
```

Inheritance diagram for L4::Scheduler:



Collaboration diagram for L4::Scheduler:



## Public Member Functions

- `l4_msgtag_t info (l4_umword_t *cpu_max, l4_sched_cpu_set_t *cpus, l4_umword_t *sched_classes=nullptr, l4_utcb_t *utcb=l4_utcb()) const noexcept`  
Get scheduler information.
- `l4_msgtag_t run_thread (lpc::Cap< Thread > thread, l4_sched_param_t const &sp)`  
Run a thread on a *Scheduler*.

- `l4_msgtag_t idle_time` (`l4_sched_cpu_set_t` const &cpus, `l4_kernel_clock_t` \*us)  
*Query the idle time (in  $\mu$ s) of a CPU.*
- `bool is_online` (`l4_umword_t` cpu, `l4_utcb_t` \*utcb=`l4_utcb`()) const noexcept  
*Query if a CPU is online.*

## Public Member Functions inherited from `L4::lcu`

- `l4_msgtag_t bind` (unsigned irqnum, `L4::Cap`< `Triggerable` > irq, `l4_utcb_t` \*utcb=`l4_utcb`()) noexcept  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- `l4_msgtag_t unbind` (unsigned irqnum, `L4::Cap`< `Triggerable` > irq, `l4_utcb_t` \*utcb=`l4_utcb`()) noexcept  
*Remove binding of an interrupt line from the interrupt controller object.*
- `l4_msgtag_t info` (`l4_icu_info_t` \*info, `l4_utcb_t` \*utcb=`l4_utcb`()) noexcept  
*Get information about the ICU features.*
- `l4_msgtag_t msi_info` (`l4_umword_t` irqnum, `l4_uint64_t` source, `l4_icu_msi_info_t` \*msi\_info)  
*Get MSI info about IRQ.*
- `l4_msgtag_t mask` (unsigned irqnum, `l4_umword_t` \*label=0, `l4_timeout_t` to=`L4_IPC_NEVER`, `l4_utcb_t` \*utcb=`l4_utcb`()) noexcept  
*Mask an IRQ line.*
- `l4_msgtag_t set_mode` (unsigned irqnum, `l4_umword_t` mode, `l4_utcb_t` \*utcb=`l4_utcb`()) noexcept  
*Set interrupt mode.*

## Public Member Functions inherited from `L4::lrq_eoi`

- `l4_msgtag_t unmask` (unsigned irqnum, `l4_umword_t` \*label=0, `l4_timeout_t` to=`L4_IPC_NEVER`, `l4_utcb_t` \*utcb=`l4_utcb`()) noexcept  
*Unmask the given interrupt line.*

## Additional Inherited Members

## Protected Types inherited from

`L4::Kobject_t`< `Scheduler`, `lcu`, `L4_PROTO_SCHEDULER`, `Type_info::Demand_t`< 1 > >

- typedef `Scheduler` **Class**  
*The target interface type (inheriting from `Kobject_t`)*
- typedef `Typeid::Iface`< `PROTO`, `Scheduler` > **\_\_Iface**  
*The interface description for the derived class.*
- typedef `Typeid::Merge_list`< `Typeid::Iface_list`< **\_\_Iface** >, typename `Base::__Iface_list` > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Types inherited from

`L4::Kobject_t`< `lcu`, `lrq_eoi`, `L4_PROTO_IRQ`, `Type_info::Demand_t`< 1 > >

- typedef `lcu` **Class**  
*The target interface type (inheriting from `Kobject_t`)*
- typedef `Typeid::Iface`< `PROTO`, `lcu` > **\_\_Iface**  
*The interface description for the derived class.*
- typedef `Typeid::Merge_list`< `Typeid::Iface_list`< **\_\_Iface** >, typename `Base::__Iface_list` > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

**Protected Member Functions inherited from****L4::Kobject\_t< Scheduler, Icu, L4\_PROTO\_SCHEDULER, Type\_info::Demand\_t< 1 > >**

- **L4::Cap< Class > c ()** const noexcept

*Get the capability to ourselves.***Protected Member Functions inherited from****L4::Kobject\_t< Icu, Irq\_eoi, L4\_PROTO\_IRQ, Type\_info::Demand\_t< 1 > >**

- **L4::Cap< Class > c ()** const noexcept

*Get the capability to ourselves.***Static Protected Member Functions inherited from****L4::Kobject\_t< Scheduler, Icu, L4\_PROTO\_SCHEDULER, Type\_info::Demand\_t< 1 > >**

- **static void \_\_check\_protocols\_\_ ()** noexcept

*Helper to check for protocol conflicts.***Static Protected Member Functions inherited from****L4::Kobject\_t< Icu, Irq\_eoi, L4\_PROTO\_IRQ, Type\_info::Demand\_t< 1 > >**

- **static void \_\_check\_protocols\_\_ ()** noexcept

*Helper to check for protocol conflicts.***15.193.1 Detailed Description**

C++ interface of the [Scheduler](#) kernel object, see [Scheduler](#) for the C interface.

The [Scheduler](#) interface allows a client to manage CPU resources. The API provides functions to query scheduler information, check the online state of CPUs, query CPU idle time and to start threads on defined CPU sets.

The scheduler offers IRQ number 0, which triggers when the number of online cores changes, e.g. due to hotplug events.

The [Scheduler](#) interface inherits from [L4::Icu](#) and [L4::Irq\\_eoi](#) for managing the scheduler virtual device IRQ which, in contrast to hardware IRQs, implements a limited functionality:

- Only IRQ line 0 is supported, no MSI vectors.
- The IRQ is edge-triggered and the IRQ mode cannot be changed.
- As the IRQ is edge-triggered, it does not have to be explicitly unmasked.

It depends on the platform, which hotplug events actually trigger the IRQ. Many platforms only support triggering the IRQ when a CPU core different from the boot CPU goes online.

**Include File**

```
#include <l4/sys/scheduler>
```

Definition at line 57 of file [scheduler](#).

**15.193.2 Member Function Documentation****15.193.2.1 idle\_time()**

```
l4_msgtag_t L4::Scheduler::idle_time (
    l4_sched_cpu_set_t const & cpus,
    l4_kernel_clock_t * us )
```

Query the idle time (in µs) of a CPU.

## Parameters

	<i>cpus</i>	Set of CPUs to query. Only the idle time of the first selected CPU in <code>cpus.map</code> is queried.
out	<i>us</i>	Idle time of queried CPU in $\mu$ s.

## Return values

0	Success.
-L4_EINVAL	Invalid CPU requested in cpu set.

This function retrieves the idle time in  $\mu$ s of the first selected CPU in `cpus.map`. The idle time is the accumulated time a CPU has spent in the idle thread since its last reset. To calculate a load estimate  $l$  one has to retrieve the idle time at the beginning ( $i1$ ) and the end ( $i2$ ) of a known time interval  $t$ . The load is then calculated as  $l = 1 - (i2 - i1)/t$ .

The idle time is only defined for online CPUs. Reading the idle time from offline CPUs is undefined and may result in either getting -L4\_EINVAL or calculating an estimated (incorrect) load of 1.

## Note

The idle time statistics of remote CPUs is updated on context switch events only, hence may not be up-to-date when requested cross-CPU. To get up-to-date idle time you should use a thread running on the same CPU of which the idle time is requested.

## 15.193.2.2 info()

```
l4_msgtag_t L4::Scheduler::info (
    l4_umword_t * cpu_max,
    l4_sched_cpu_set_t * cpus,
    l4_umword_t * sched_classes = nullptr,
    l4_utcb_t * utcb = l4_utcb() ) const [inline], [noexcept]
```

Get scheduler information.

## Parameters

out	<i>cpu_max</i>	Maximum number of CPUs ever available. Optional, can be nullptr.
in, out	<i>cpus</i>	<i>cpus.offset</i> is first CPU of interest. <i>cpus.granularity</i> (see <a href="#">l4_sched_cpu_set_t</a> ). <i>cpus.map</i> Bitmap of online CPUs. Pass nullptr to omit this information.
out	<i>sched_classes</i>	A bitmap of available scheduling classes (see <code>L4_scheduler_classes</code> ). Pass nullptr to omit this information.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

## Return values

0	Success.
-L4_ERANGE	The given CPU offset is larger than the maximum number of CPUs.

Definition at line 85 of file [scheduler](#).

References [l4\\_sched\\_cpu\\_set\\_t::gran\\_offset](#), and [l4\\_sched\\_cpu\\_set\\_t::map](#).

### 15.193.2.3 is\_online()

```
bool L4::Scheduler::is_online (
    l4_umword_t cpu,
    l4_utcb_t * utcb = l4_utcb() ) const [inline], [noexcept]
```

Query if a CPU is online.

#### Parameters

<i>cpu</i>	CPU number whose online status should be queried.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

#### Return values

<i>true</i>	The CPU is online.
<i>false</i>	The CPU is offline

Definition at line 165 of file [scheduler](#).

### 15.193.2.4 run\_thread()

```
l4_msgtag_t L4::Scheduler::run_thread (
    Ipc::Cap< Thread > thread,
    l4_sched_param_t const & sp )
```

Run a thread on a [Scheduler](#).

#### Parameters

<i>thread</i>	Capability of the thread to run.
<i>sp</i>	Scheduling parameters.

#### Return values

<i>0</i>	Success.
<i>-L4_EINVAL</i>	Invalid size of the scheduling parameter.

This function launches a thread on a CPU determined by the scheduling parameter `sp.affinity`. A thread can be intentionally stopped by migrating it on an offline or an invalid CPU. The thread is only guaranteed to run if the CPU it is migrated to is currently online.

#### Note

If the target CPU is currently not online, there is no guarantee that the thread will ever run, even if the CPU comes online later on.

A scheduler may impose a policy with regard to selecting CPUs. However the scheduler is required to ensure the following two properties:

- Two threads with disjoint CPU sets must be scheduled to different CPUs.
- Two threads with identical CPU sets selecting only a single CPU must be scheduled to the same CPU.

The documentation for this class was generated from the following file:

- [l4/sys/scheduler](#)

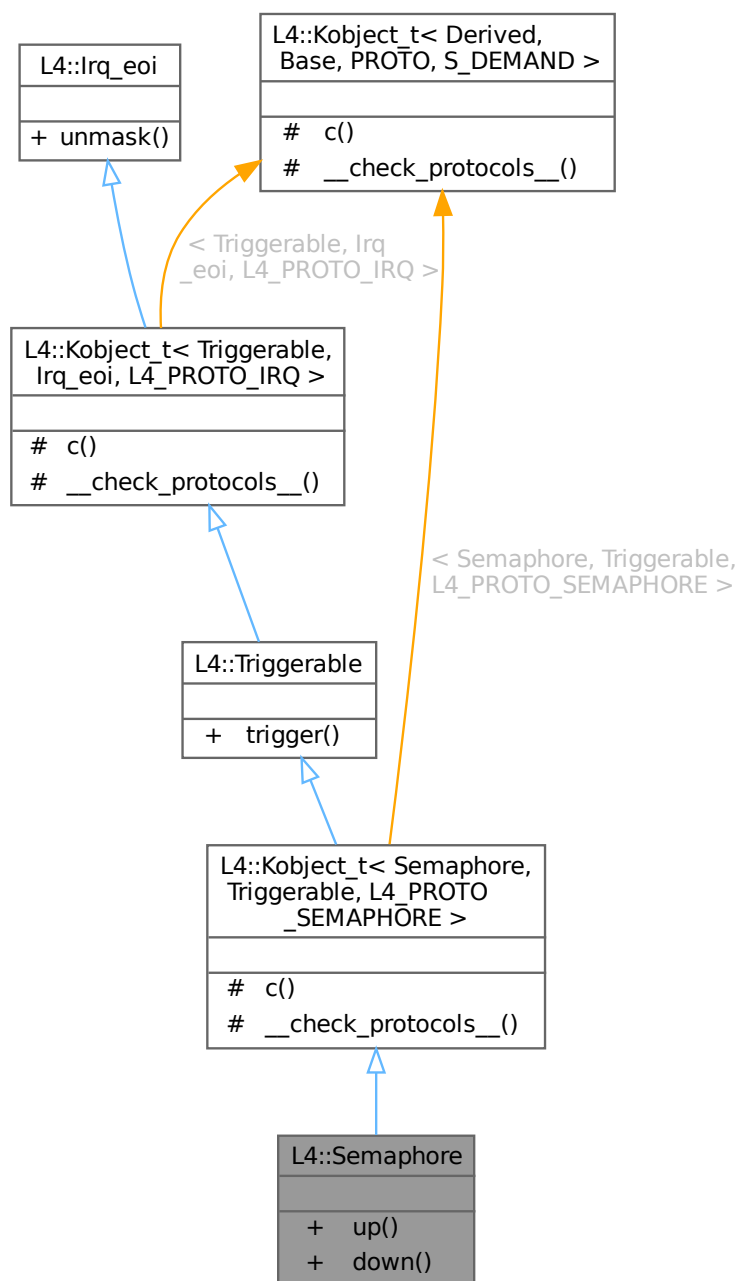
## 15.194 L4::Semaphore Struct Reference

C++ Kernel-provided semaphore interface, see [Kernel-provided semaphore](#) for the C interface.

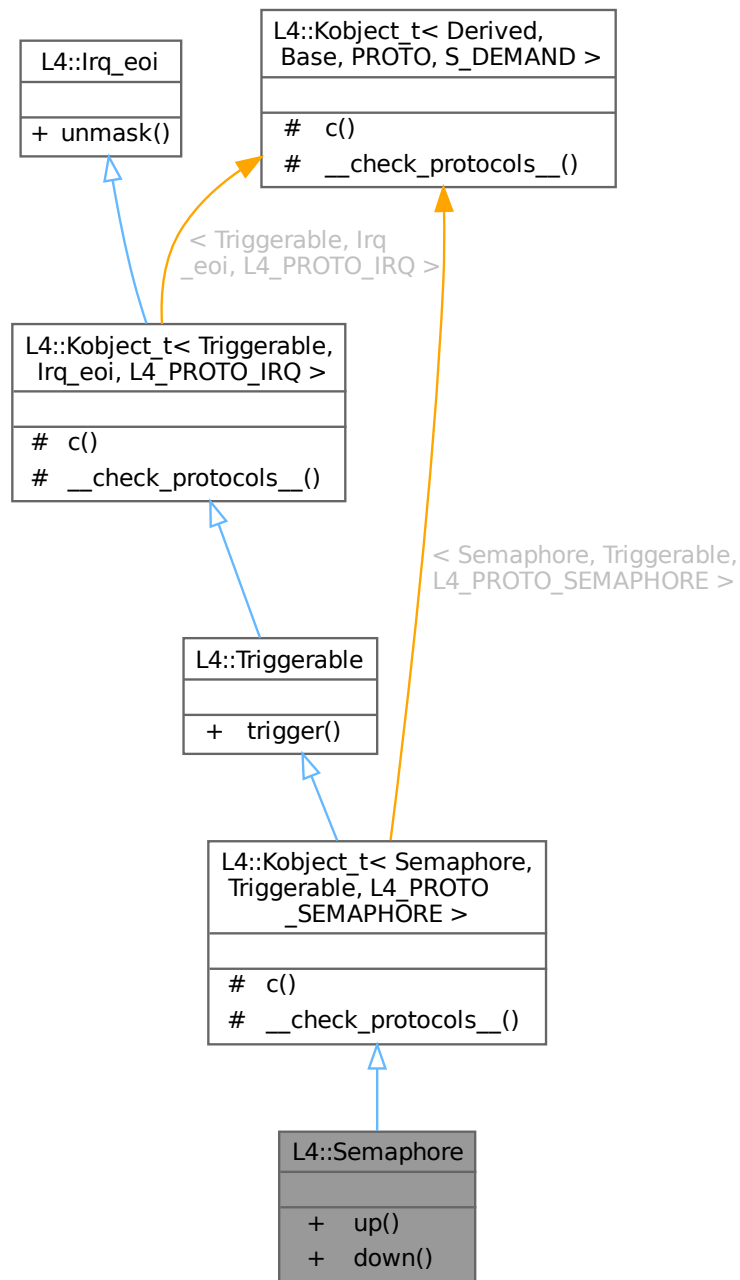
```
#include <semaphore>
```



Inheritance diagram for L4::Semaphore:



Collaboration diagram for L4::Semaphore:



## Public Member Functions

- `l4_msgtag_t up (l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Semaphore up operation (wrapper for `trigger()`).*
- `l4_msgtag_t down (l4_timeout_t timeout=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Semaphore down operation.*

## Public Member Functions inherited from [L4::Triggerable](#)

- [l4\\_msgtag\\_t trigger](#) ([l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Trigger the object.*

## Public Member Functions inherited from [L4::Irq\\_eoi](#)

- [l4\\_msgtag\\_t unmask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Unmask the given interrupt line.*

## Additional Inherited Members

### Protected Types inherited from [L4::Kobject\\_t](#)< [Semaphore](#), [Triggerable](#), [L4\\_PROTO\\_SEMAPHORE](#) >

- typedef [Semaphore](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< [PROTO](#), [Semaphore](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Types inherited from [L4::Kobject\\_t](#)< [Triggerable](#), [Irq\\_eoi](#), [L4\\_PROTO\\_IRQ](#) >

- typedef [Triggerable](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< [PROTO](#), [Triggerable](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from [L4::Kobject\\_t](#)< [Semaphore](#), [Triggerable](#), [L4\\_PROTO\\_SEMAPHORE](#) >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject\\_t](#)< [Triggerable](#), [Irq\\_eoi](#), [L4\\_PROTO\\_IRQ](#) >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept  
*Get the capability to ourselves.*

### Static Protected Member Functions inherited from

[L4::Kobject\\_t](#) < [Semaphore](#), [Triggerable](#), [L4\\_PROTO\\_SEMAPHORE](#) >

- static void `__check_protocols__()` noexcept

*Helper to check for protocol conflicts.*

### Static Protected Member Functions inherited from

[L4::Kobject\\_t](#) < [Triggerable](#), [Irq\\_eoi](#), [L4\\_PROTO\\_IRQ](#) >

- static void `__check_protocols__()` noexcept

*Helper to check for protocol conflicts.*

## 15.194.1 Detailed Description

C++ Kernel-provided semaphore interface, see [Kernel-provided semaphore](#) for the C interface.

This is the interface for kernel-provided semaphore objects. The object provides the classical functions `up()` and `down()` for counting the semaphore and blocking. The semaphore is a [Triggerable](#) with respect to the `up()` function, this means that a semaphore can be bound to an interrupt line at an ICU ([L4::lcu](#)) and incoming interrupts increment the semaphore counter.

The `down()` method decrements the semaphore counter and blocks if the counter is already zero. Blocking on a semaphore may—as all blocking operations—either return successfully, or be aborted due to an expired timeout provided to the `down()` operation, or due to an [L4::Thread::ex\\_regs\(\)](#) operation with the [L4\\_THREAD\\_EX\\_REGS\\_CANCEL](#) flag set.

A semaphore object is initialized with counter value 0.

The main reason for using a semaphore instead of an [L4::Irq](#) is to ensure that incoming trigger signals do not interfere with any open-wait operations, as used for example in a server loop.

Definition at line 54 of file [semaphore](#).

## 15.194.2 Member Function Documentation

### 15.194.2.1 `down()`

```
l4_msgtag_t L4::Semaphore::down (
    l4_timeout_t timeout = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

[Semaphore](#) down operation.

#### Parameters

<i>timeout</i>	Timeout for blocking the semaphore down operation. Note: The receive timeout of this timeout-pair is significant for blocking, the send part is usually non-blocking.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

## Returns

Syscall return tag. Use [l4\\_error\(\)](#) to check for errors.

## Return values

<code>-L4_EPERM</code>	No <a href="#">L4_CAP_FPAGE_S</a> right on invoked semaphore capability.
------------------------	--

This method decrements the semaphore counter by one, or blocks if the counter is already zero, until either a timeout or cancel condition hits or the counter is increased by an [up\(\)](#) operation.

Definition at line 90 of file [semaphore](#).

## 15.194.2.2 up()

```
l4_msgtag_t L4::Semaphore::up (
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

[Semaphore](#) up operation (wrapper for [trigger\(\)](#)).

## Parameters

<code>utcb</code>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
-------------------	--

## Returns

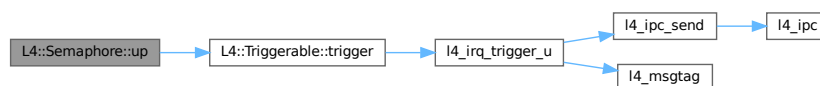
Syscall return tag for a send-only operation, this means there is no return value except [L4\\_MSGTAG\\_ERROR](#) indicating success or failure of the send operation. Use [l4\\_ipc\\_error\(\)](#) to check for errors and **do not** use [l4\\_error\(\)](#).

Increases the semaphore counter by one if it is smaller than an unspecified limit. The unspecified limit is guaranteed to be at least  $2^{31}-1$ .

Definition at line 70 of file [semaphore](#).

References [L4::Triggerable::trigger\(\)](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

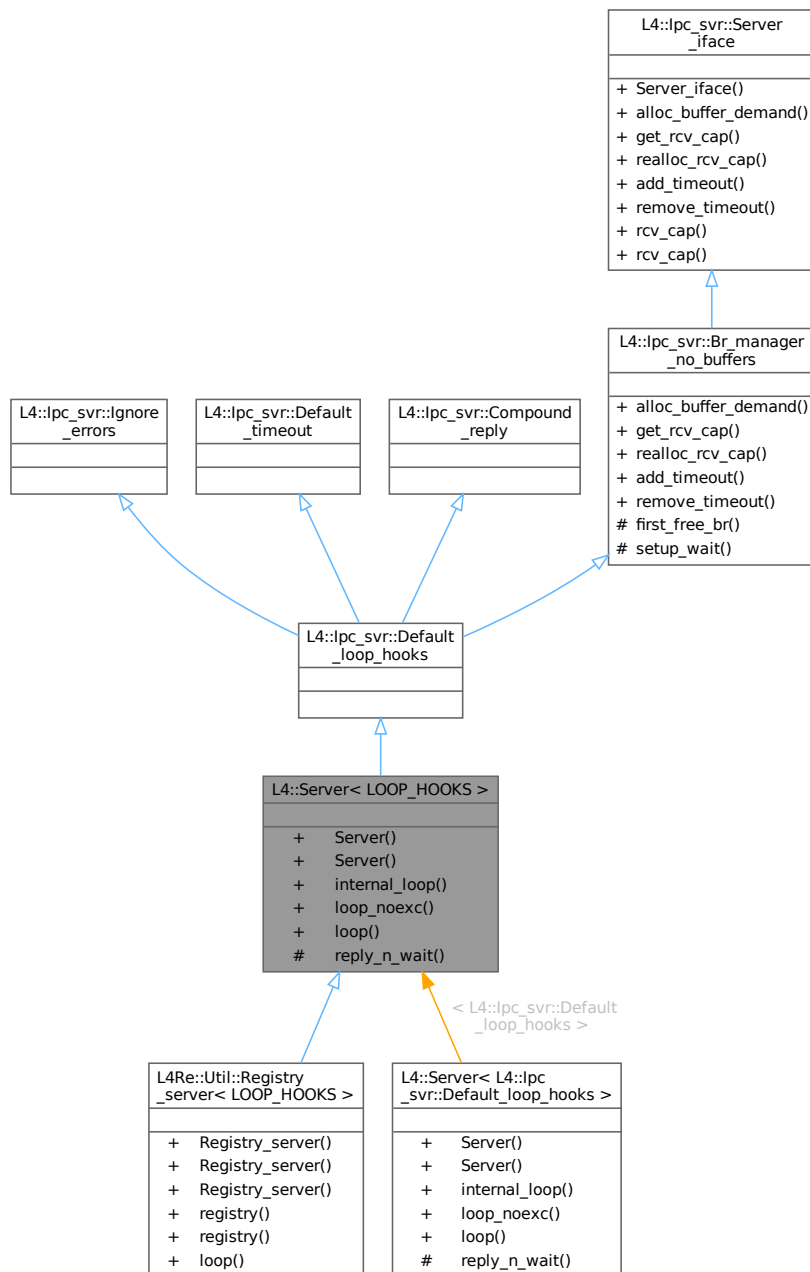
- [l4/sys/semaphore](#)

## 15.195 L4::Server< LOOP\_HOOKS > Class Template Reference

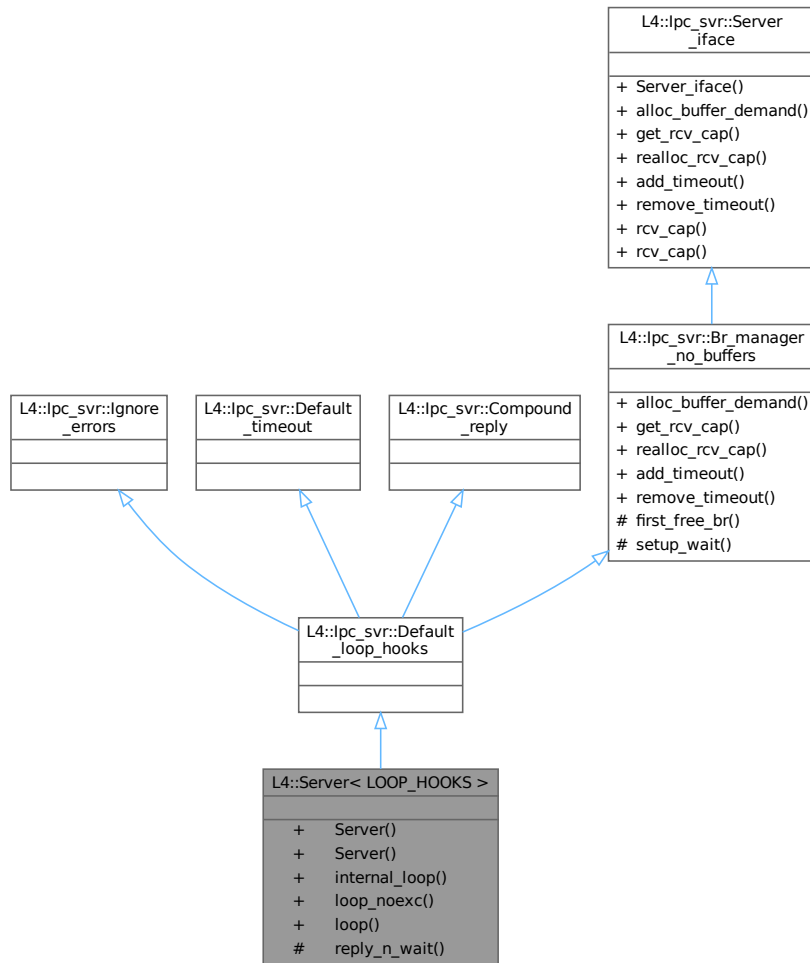
Basic server loop for handling client requests.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::Server< LOOP\_HOOKS >:



Collaboration diagram for L4::Server< LOOP\_HOOKS >:



## Public Member Functions

- **Server** (*I4\_utcb\_t* \*)  
*Initializes the server loop.*
- **Server** ()  
*Initializes the server loop.*
- `template<typename DISPATCH >`  
`L4_NORETURN void internal_loop (DISPATCH dispatch, I4_utcb_t *)`  
*The server loop.*
- `template<typename R >`  
`L4_NORETURN void loop_noexc (R r, I4_utcb_t *u=I4_utcb())`  
*Server loop without exception handling.*
- `template<typename EXC , typename R >`  
`L4_NORETURN void loop (R r, I4_utcb_t *u=I4_utcb())`  
*Server loop with internal exception handling.*

## Public Member Functions inherited from [L4::lpc\\_svr::Br\\_manager\\_no\\_buffers](#)

- int [alloc\\_buffer\\_demand](#) ([Demand](#) const &demand) override  
*Tells the server to allocate buffers for the given demand.*
- [L4::Cap](#)< void > [get\\_rcv\\_cap](#) (int) const override  
*Returns [L4::Cap](#)< void > ::Invalid, we have no buffer management.*
- int [realloc\\_rcv\\_cap](#) (int) override  
*Returns -L4\_ENOMEM, we have no buffer management.*
- int [add\\_timeout](#) ([Timeout](#) \*, [l4\\_kernel\\_clock\\_t](#)) override  
*Returns -L4\_ENOSYS, we have no timeout queue.*
- int [remove\\_timeout](#) ([Timeout](#) \*) override  
*Returns -L4\_ENOSYS, we have no timeout queue.*

## Public Member Functions inherited from [L4::lpc\\_svr::Server\\_iface](#)

- [Server\\_iface](#) ()  
*Make a server interface.*
- template<typename T >  
[L4::Cap](#)< T > [rcv\\_cap](#) (int index) const  
*Get given receive buffer as typed capability.*
- [L4::Cap](#)< void > [rcv\\_cap](#) (int index) const  
*Get receive cap with the given index as generic (void) type.*

## Protected Member Functions

- [l4\\_msgtag\\_t](#) [reply\\_n\\_wait](#) ([l4\\_msgtag\\_t](#) reply, [l4\\_umword\\_t](#) \*p, [l4\\_utcb\\_t](#) \*)  
*Internal implementation for reply and wait.*

## Protected Member Functions inherited from [L4::lpc\\_svr::Br\\_manager\\_no\\_buffers](#)

- unsigned [first\\_free\\_br](#) () const  
*Returns 1 as first free buffer.*
- void [setup\\_wait](#) ([l4\\_utcb\\_t](#) \*utcb, [L4::lpc\\_svr::Reply\\_mode](#))  
*Setup wait function for the server loop (Server<>).*

## Additional Inherited Members

## Public Types inherited from [L4::lpc\\_svr::Server\\_iface](#)

- typedef [L4::Type\\_info::Demand](#) [Demand](#)  
*Data type expressing server-side demand for receive buffers.*

### 15.195.1 Detailed Description

```
template<typename LOOP_HOOKS = lpc_svr::Default_loop_hooks>
class L4::Server< LOOP_HOOKS >
```

Basic server loop for handling client requests.



## Parameters

<code>LOOP_HOOKS</code>	the server inherits from <code>LOOP_HOOKS</code> and calls the hooks defined in <code>LOOP_HOOKS</code> in the server loop. See <a href="#">lpc_svr::Default_loop_hooks</a> , <a href="#">lpc_svr::Ignore_errors</a> , <a href="#">lpc_svr::Default_timeout</a> , <a href="#">lpc_svr::Compound_reply</a> , and <a href="#">lpc_svr::Br_manager_no_buffers</a> .
-------------------------	--

This is basically a simple server loop that uses a single message buffer for receiving requests and sending replies. The dispatcher determines how incoming messages are handled.

Definition at line 258 of file [ipc\\_server\\_loop](#).

## 15.195.2 Constructor & Destructor Documentation

### 15.195.2.1 Server()

```
template<typename LOOP_HOOKS = Ipc_svr::Default_loop_hooks>
L4::Server< LOOP_HOOKS >::Server (
    l4_utcb_t * ) [inline], [explicit]
```

Initializes the server loop.

Note that this variant is deprecated. The UTCB can be specified with the server loop function ([l4\\_utcb\(\)](#) is the default). (2021-10)

Definition at line 270 of file [ipc\\_server\\_loop](#).

## 15.195.3 Member Function Documentation

### 15.195.3.1 internal\_loop()

```
template<typename L >
template<typename DISPATCH >
L4_NORETURN void L4::Server< L >::internal_loop (
    DISPATCH dispatch,
    l4_utcb_t * utcb ) [inline]
```

The server loop.

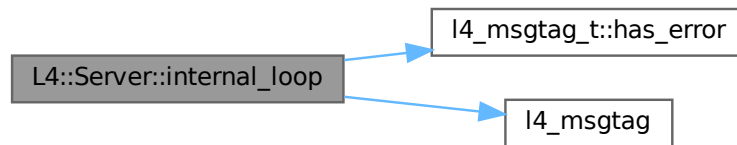
This function usually never returns, it waits for incoming messages calls the dispatcher, sends a reply and waits again.

Definition at line 343 of file [ipc\\_server\\_loop](#).

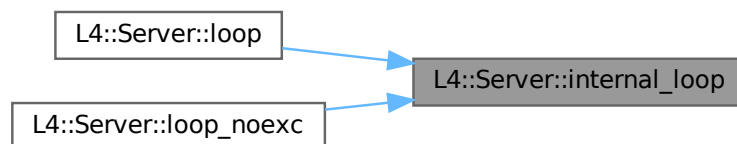
References [l4\\_msgtag\\_t::has\\_error\(\)](#), [L4\\_ENOREPLY](#), and [l4\\_msgtag\(\)](#).

Referenced by [L4::Server< LOOP\\_HOOKS >::loop\(\)](#), and [L4::Server< LOOP\\_HOOKS >::loop\\_noexc\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.195.3.2 loop()

```

template<typename LOOP_HOOKS = Ipc_svr::Default_loop_hooks>
template<typename EXC , typename R >
L4_NORETURN void L4::Server< LOOP_HOOKS >::loop (
    R r,
    l4_utcb_t * u = l4_utcb() ) [inline]
  
```

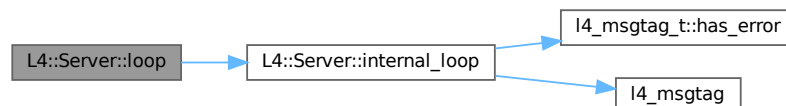
[Server](#) loop with internal exception handling.

This server loop translates [L4::Runtime\\_error](#) exceptions into negative error return codes sent to the caller.

Definition at line 306 of file [ipc\\_server\\_loop](#).

References [L4::Server< LOOP\\_HOOKS >::internal\\_loop\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

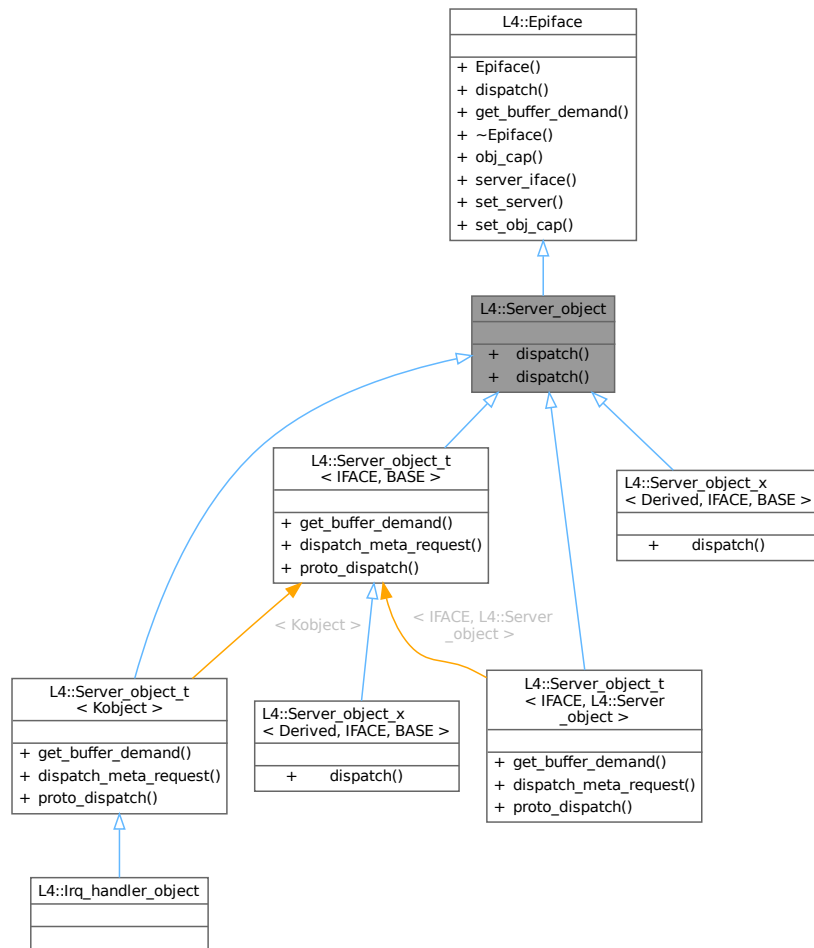
- [l4/sys/cxx/ipc\\_server\\_loop](#)

## 15.196 L4::Server\_object Class Reference

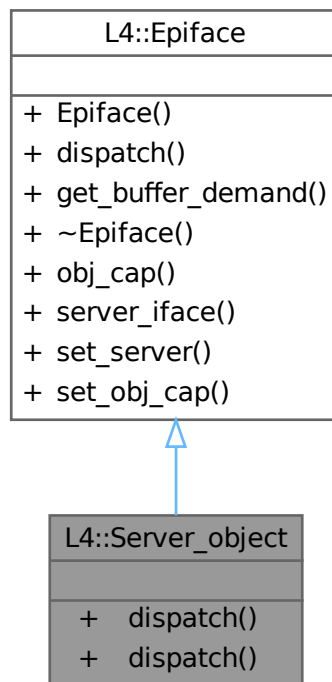
Abstract server object to be used with [L4::Server](#) and [L4::Basic\\_registry](#).

```
#include <ipc_server>
```

Inheritance diagram for L4::Server\_object:



Collaboration diagram for L4::Server\_object:



### Public Member Functions

- virtual int `dispatch` (unsigned long rights, `lpc::lostream` &ios)=0  
*The abstract handler for client requests to the object.*
- `l4_msgtag_t` `dispatch` (`l4_msgtag_t` tag, unsigned rights, `l4_utcb_t` \*utcb) override  
*The abstract handler for client requests to the object.*

### Public Member Functions inherited from `L4::Epiface`

- `Epiface` ()  
*Make a server object.*
- virtual `Demand` `get_buffer_demand` () const =0  
*Get the server-side receive buffer demand for this object.*
- virtual `~Epiface` ()=0  
*Destroy the object.*
- Stored\_cap `obj_cap` () const  
*Get the capability to the kernel object belonging to this object.*
- `Server_iface` \* `server_iface` () const  
*Get pointer to server interface at which the object is currently registered.*
- int `set_server` (`Server_iface` \*srv, `Cap`< void > cap, bool managed=false)  
*Set server registration info for the object.*
- void `set_obj_cap` (`Cap`< void > const &cap)  
*Deprecated server registration function.*

## Additional Inherited Members

## Public Types inherited from L4::Epiface

- typedef [lpc\\_svr::Server\\_iface](#) **Server\_iface**  
*Type for abstract server interface.*
- typedef [lpc\\_svr::Server\\_iface::Demand](#) **Demand**  
*Type for server-side receive buffer demand.*

### 15.196.1 Detailed Description

Abstract server object to be used with [L4::Server](#) and [L4::Basic\\_registry](#).

#### Note

Usually [L4::Server\\_object\\_t](#) is used as a base class when writing server objects.

This server object provides an abstract interface that is used by the [L4::Registry\\_dispatcher](#) model. You can derive subclasses from this interface and implement application specific server objects.

Definition at line 49 of file [ipc\\_server](#).

### 15.196.2 Member Function Documentation

#### 15.196.2.1 dispatch() [1/2]

```
l4_msgtag_t L4::Server_object::dispatch (
    l4_msgtag_t tag,
    unsigned rights,
    l4_utcb_t * utcb ) [inline], [override], [virtual]
```

The abstract handler for client requests to the object.

#### Parameters

<i>tag</i>	The message tag for this invocation.
<i>rights</i>	The rights bits in the invoked capability.
<i>utcb</i>	The UTCB used for the invocation.

#### Return values

<code>-L4_ENOREPLY</code>	No reply message is send.
<code>&lt;0</code>	Error, reply with error code.
<code>&gt;=0</code>	Success, reply with return value.

This function must be implemented by application specific server objects.

Implements [L4::Epiface](#).

Definition at line 71 of file [ipc\\_server](#).

References [dispatch\(\)](#).

Here is the call graph for this function:



### 15.196.2.2 `dispatch()` [2/2]

```
virtual int L4::Server_object::dispatch (
    unsigned long rights,
    Ipc::Iostream & ios ) [pure virtual]
```

The abstract handler for client requests to the object.

#### Parameters

<i>rights</i>	The rights bits in the invoked capability.
<i>ios</i>	The <a href="#">ipc::iostream</a> for reading the request and writing the reply.

#### Return values

<code>-L4_ENOREPLY</code>	Instructs the server loop to not send a reply.
<code>&lt; 0</code>	Error, reply with error code.
<code>&gt;= 0</code>	Success, reply with return value.

This function must be implemented by application specific server objects. The implementation must unmarshall data from the stream (`ios`) and create a reply by marshalling to the stream (`ios`). For details about the IPC stream see [IPC stream operators](#).

#### Note

You need to extract the complete message from the `ios` stream before inserting any reply data or before doing any function call that may use the UTCB. Otherwise, the incoming message may get lost.

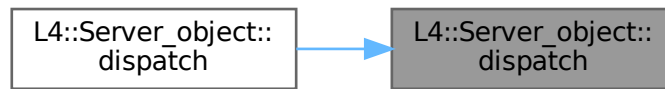
Implemented in [L4::Server\\_object\\_x< Derived, IFACE, BASE >](#).

#### Examples

[examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Referenced by [dispatch\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

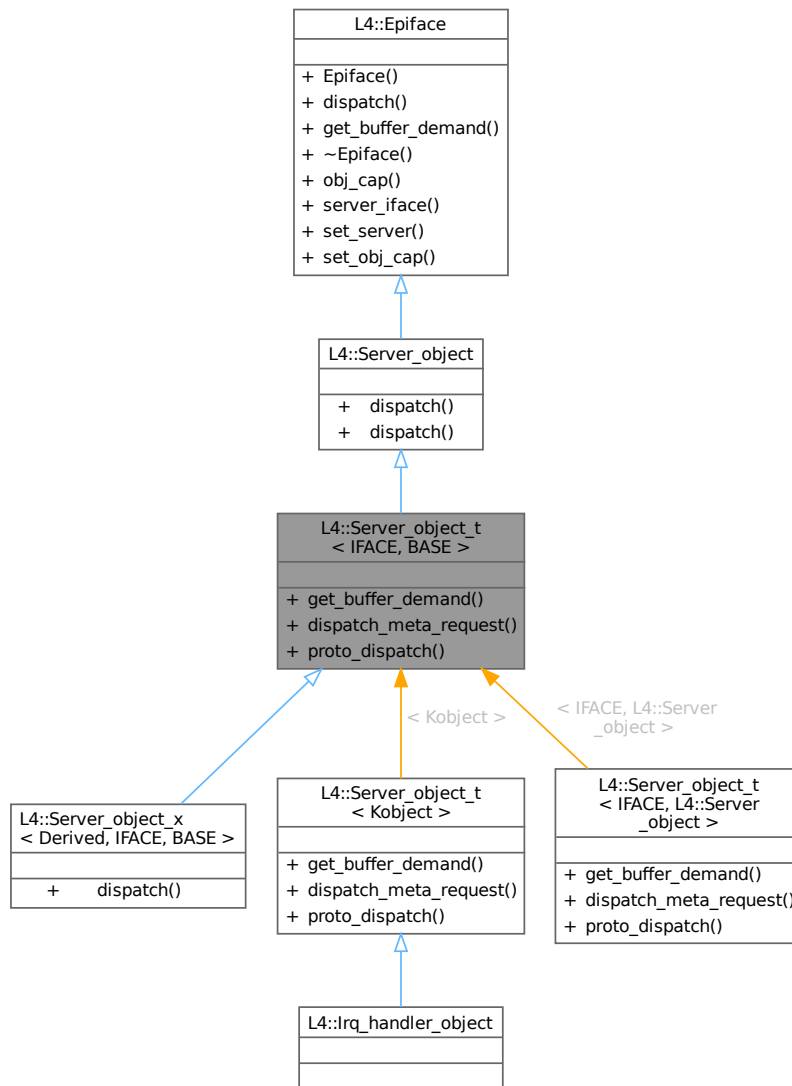
- [l4/cxx/ipc\\_server](#)

## 15.197 L4::Server\_object\_t< IFACE, BASE > Struct Template Reference

Base class (template) for server implementing server objects.

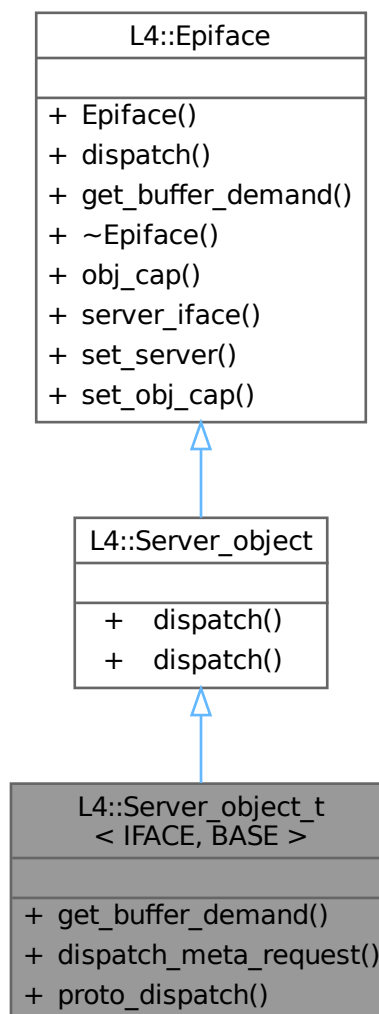
```
#include <ipc_server>
```

Inheritance diagram for L4::Server\_object\_t< IFACE, BASE >:





Collaboration diagram for L4::Server\_object\_t< IFACE, BASE >:



## Public Types

- typedef IFACE **Interface**  
*Data type of the IPC interface definition.*

## Public Types inherited from L4::Epiface

- typedef [lpc\\_svr::Server\\_iface](#) **Server\_iface**  
*Type for abstract server interface.*
- typedef [lpc\\_svr::Server\\_iface::Demand](#) **Demand**  
*Type for server-side receive buffer demand.*

## Public Member Functions

- `BASE::Demand` `get_buffer_demand` () const override
- int `dispatch_meta_request` (`L4::lpc::lostream` &ios)

*Implementation of the meta protocol based on IFACE.*

## Public Member Functions inherited from `L4::Server_object`

- virtual int `dispatch` (unsigned long rights, `lpc::lostream` &ios)=0
- `l4_msgtag_t` `dispatch` (`l4_msgtag_t` tag, unsigned rights, `l4_utcb_t` \*utcb) override

*The abstract handler for client requests to the object.*

*The abstract handler for client requests to the object.*

## Public Member Functions inherited from `L4::Epiface`

- `Epiface` ()
- virtual `~Epiface` ()=0
- Stored\_cap `obj_cap` () const
- `Server_iface` \* `server_iface` () const
- int `set_server` (`Server_iface` \*srv, `Cap`< void > cap, bool managed=false)
- void `set_obj_cap` (`Cap`< void > const &cap)

*Make a server object.*

*Destroy the object.*

*Get the capability to the kernel object belonging to this object.*

*Get pointer to server interface at which the object is currently registered.*

*Set server registration info for the object.*

*Deprecated server registration function.*

## Static Public Member Functions

- template<typename THIS >  
static int `proto_dispatch` (THIS \*self, `l4_umword_t` rights, `L4::lpc::lostream` &ios)

*Implementation of protocol-based dispatch for this server object.*

### 15.197.1 Detailed Description

```
template<typename IFACE, typename BASE = L4::Server_object>
struct L4::Server_object_t< IFACE, BASE >
```

Base class (template) for server implementing server objects.

#### Template Parameters

<code>IFACE</code>	The IPC interface class that defines the interface that shall be implemented.
<code>BASE</code>	The server object base class (usually <code>L4::Server_object</code> ).

## Examples

[examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 91 of file [ipc\\_server](#).

## 15.197.2 Member Function Documentation

### 15.197.2.1 dispatch\_meta\_request()

```
template<typename IFACE , typename BASE = L4::Server_object>
int L4::Server_object_t< IFACE, BASE >::dispatch_meta_request (
    L4::Ipc::Iostream & ios ) [inline]
```

Implementation of the meta protocol based on *IFACE*.

## Parameters

<i>ios</i>	The IO stream used for receiving the message.
------------	---

This function can be used to handle incoming [L4\\_PROTO\\_META](#) protocol requests. The implementation uses the [L4::Type\\_info](#) of *IFACE* to handle the requests. Call this function in the implementation of [Server\\_object::dispatch\(\)](#) when the received message tag has protocol [L4\\_PROTO\\_META](#) ([L4::Meta::Protocol](#)).

Definition at line 110 of file [ipc\\_server](#).

### 15.197.2.2 get\_buffer\_demand()

```
template<typename IFACE , typename BASE = L4::Server_object>
BASE::Demand L4::Server_object_t< IFACE, BASE >::get_buffer_demand ( ) const [inline], [override],
[virtual]
```

## Returns

the server-side buffer demand based in *IFACE*.

Implements [L4::Epiface](#).

Definition at line 97 of file [ipc\\_server](#).

### 15.197.2.3 proto\_dispatch()

```
template<typename IFACE , typename BASE = L4::Server_object>
template<typename THIS >
static int L4::Server_object_t< IFACE, BASE >::proto_dispatch (
    THIS * self,
    l4_umword_t rights,
    L4::Ipc::Iostream & ios ) [inline], [static]
```

Implementation of protocol-based dispatch for this server object.

## Parameters

<i>self</i>	The this pointer for the object (inherits from <a href="#">Server_object_t</a> ).
<i>rights</i>	The rights from the received IPC (forwarded to <code>p_dispatch()</code> ).
<i>ios</i>	The message stream for the incoming and the reply message.

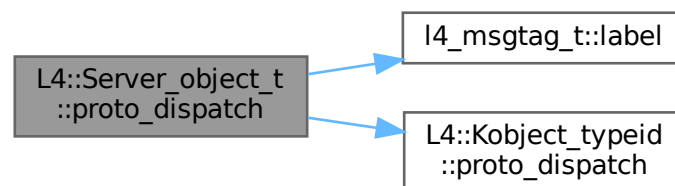
[Server](#) objects may call this function from their [dispatch\(\)](#) function. This function reads the protocol ID from the message tag and uses the `p_dispatch` code to dispatch to overloaded `p_dispatch` functions of self.

Definition at line 125 of file [ipc\\_server](#).

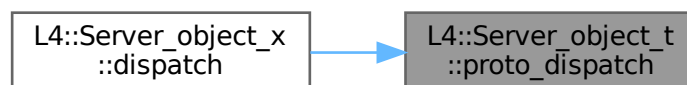
References [l4\\_msgtag\\_t::label\(\)](#), and [L4::Kobject\\_typeid< T >::proto\\_dispatch\(\)](#).

Referenced by [L4::Server\\_object\\_x< Derived, IFACE, BASE >::dispatch\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

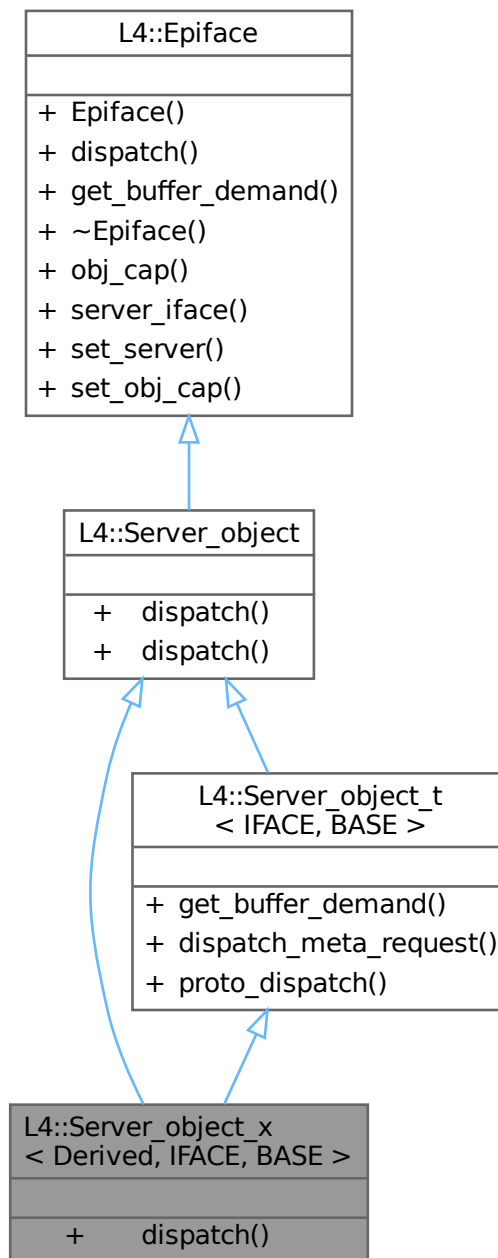
- [l4/cxx/ipc\\_server](#)

## 15.198 L4::Server\_object\_x< Derived, IFACE, BASE > Struct Template Reference

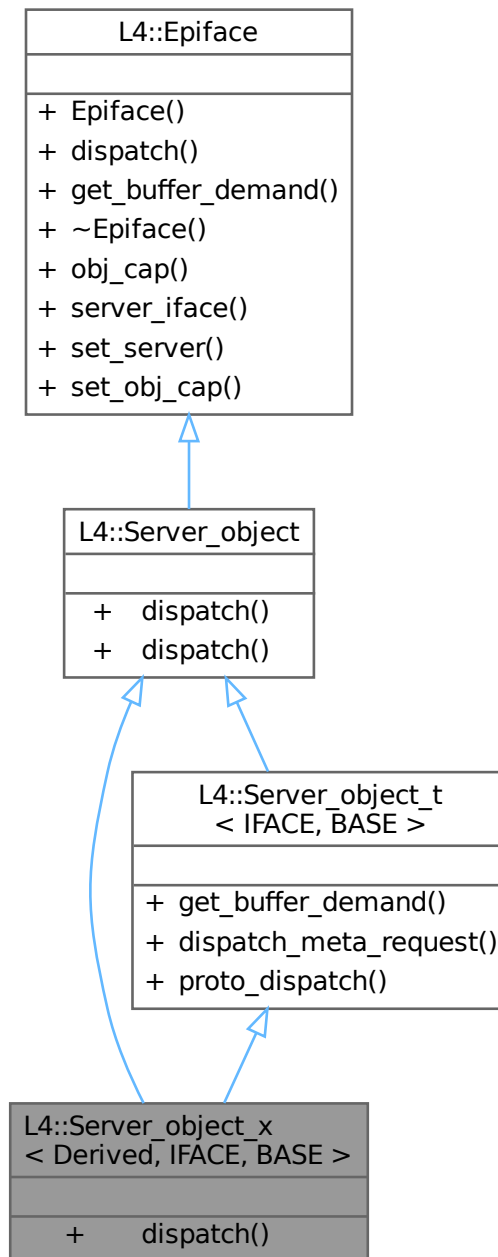
Helper class to implement p\_dispatch based server objects.

```
#include <ipc_server>
```

Inheritance diagram for L4::Server\_object\_x< Derived, IFACE, BASE >:



Collaboration diagram for L4::Server\_object\_x< Derived, IFACE, BASE >:



### Public Member Functions

- `int dispatch (l4_umword_t r, L4::lpc::lostream &ios)`  
*Implementation forwarding to `p_dispatch()`.*

### Public Member Functions inherited from **L4::Server\_object**

- `l4_msgtag_t dispatch (l4_msgtag_t tag, unsigned rights, l4_utcb_t *utcb)` override

*The abstract handler for client requests to the object.*

## Public Member Functions inherited from L4::Epiface

- **Epiface** ()  
*Make a server object.*
- virtual **~Epiface** ()=0  
*Destroy the object.*
- Stored\_cap **obj\_cap** () const  
*Get the capability to the kernel object belonging to this object.*
- **Server\_iface** \* **server\_iface** () const  
*Get pointer to server interface at which the object is currently registered.*
- int **set\_server** (**Server\_iface** \*srv, **Cap**< void > cap, bool managed=false)  
*Set server registration info for the object.*
- void **set\_obj\_cap** (**Cap**< void > const &cap)  
*Deprecated server registration function.*

## Public Member Functions inherited from L4::Server\_object\_t< IFACE, BASE >

- BASE::Demand **get\_buffer\_demand** () const override
- int **dispatch\_meta\_request** (L4::lpc::lostream &ios)  
*Implementation of the meta protocol based on IFACE.*

## Additional Inherited Members

## Public Types inherited from L4::Epiface

- typedef **lpc\_svr::Server\_iface** **Server\_iface**  
*Type for abstract server interface.*
- typedef **lpc\_svr::Server\_iface::Demand** **Demand**  
*Type for server-side receive buffer demand.*

## Public Types inherited from L4::Server\_object\_t< IFACE, BASE >

- typedef IFACE **Interface**  
*Data type of the IPC interface definition.*

## Static Public Member Functions inherited from L4::Server\_object\_t< IFACE, BASE >

- template<typename THIS >  
static int **proto\_dispatch** (THIS \*self, **l4\_umword\_t** rights, L4::lpc::lostream &ios)  
*Implementation of protocol-based dispatch for this server object.*

### 15.198.1 Detailed Description

```
template<typename Derived, typename IFACE, typename BASE = L4::Server_object>
struct L4::Server_object_x< Derived, IFACE, BASE >
```

Helper class to implement p\_dispatch based server objects.

## Template Parameters

<i>Derived</i>	The data type of your server object class.
<i>IFACE</i>	The data type providing the interface definition for the object.
<i>BASE</i>	Optional data-type of the base server object (usually <a href="#">L4::Server_object</a> )

This class implements the standard [dispatch\(\)](#) function of [L4::Server\\_object](#) and forwards incoming messages to a set of overloaded `p_dispatch()` functions. There must be a `p_dispatch()` function in `Derived` for each interface provided by `IFACE` with the signature

```
int p_dispatch(Iface *, unsigned rights, L4::Ipc::Iostream &)
```

that is called for messages with `protocol == Iface::Protocol`.

Example signature for [L4Re::Dataspace](#) is:

```
int p_dispatch(L4Re::Dataspace *, unsigned, L4::Ipc::Iostream &)
```

Definition at line [154](#) of file [ipc\\_server](#).

The documentation for this struct was generated from the following file:

- [l4/cxx/ipc\\_server](#)

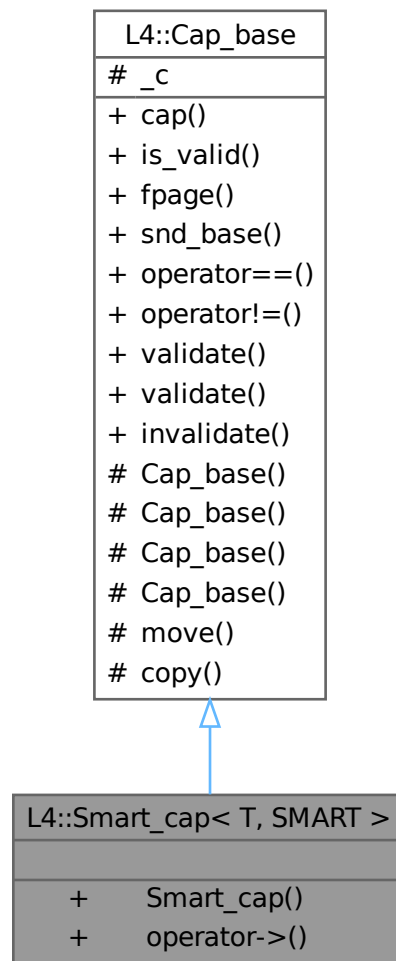
## 15.199 L4::Smart\_cap< T, SMART > Class Template Reference

Smart capability class.

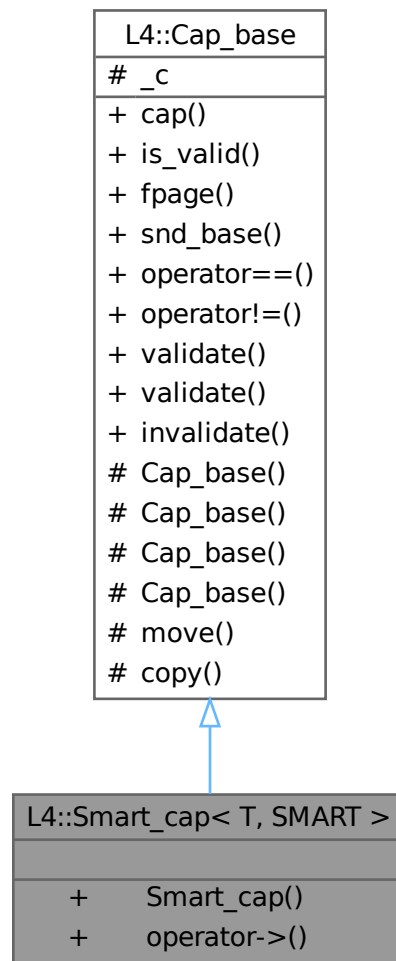
```
#include <smart_capability>
```



Inheritance diagram for L4::Smart\_cap< T, SMART >:



Collaboration diagram for L4::Smart\_cap< T, SMART >:



## Public Member Functions

- `template<typename O >`  
`Smart_cap (Cap< O > const &p) noexcept`  
*Internal constructor, use to generate a capability from a `this` pointer.*
- `Cap< T > operator-> ( ) const noexcept`  
*Member access of a `T`.*

## Public Member Functions inherited from L4::Cap\_base

- `l4_cap_idx_t cap ( ) const noexcept`  
*Return capability selector.*
- `bool is_valid ( ) const noexcept`  
*Test whether the capability is a valid capability index (i.e., not `L4_INVALID_CAP`).*

- [l4\\_fpage\\_t fpage](#) (unsigned rights=[L4\\_CAP\\_FPAGE\\_RWS](#)) const noexcept  
*Return flex-page for the capability.*
- [l4\\_umword\\_t snd\\_base](#) (unsigned grant=[L4\\_MAP\\_ITEM\\_MAP](#), [l4\\_cap\\_idx\\_t](#) base=[L4\\_INVALID\\_CAP](#)) const noexcept  
*Return send base.*
- bool **operator==** ([Cap\\_base](#) const &o) const noexcept  
*Test if two capabilities are equal.*
- bool **operator!=** ([Cap\\_base](#) const &o) const noexcept  
*Test if two capabilities are not equal.*
- [l4\\_msgtag\\_t validate](#) ([l4\\_utcb\\_t](#) \*u=[l4\\_utcb\(\)](#)) const noexcept  
*Check whether a capability is present (refers to an object).*
- [l4\\_msgtag\\_t validate](#) ([Cap](#)< [Task](#) > task, [l4\\_utcb\\_t](#) \*u=[l4\\_utcb\(\)](#)) const noexcept  
*Check whether a capability is present (refers to an object).*
- void **invalidate** () noexcept  
*Set this capability to invalid ([L4\\_INVALID\\_CAP](#)).*

### Additional Inherited Members

### Public Types inherited from [L4::Cap\\_base](#)

- enum [No\\_init\\_type](#) { [No\\_init](#) }  
*Special value for uninitialized capability objects.*
- enum [Cap\\_type](#) { [Invalid](#) = [L4\\_INVALID\\_CAP](#) }  
*Invalid capability type.*

### Protected Member Functions inherited from [L4::Cap\\_base](#)

- [Cap\\_base](#) ([l4\\_cap\\_idx\\_t](#) c) noexcept  
*Generate a capability from its C representation.*
- **Cap\_base** ([Cap\\_type](#) cap) noexcept  
*Constructor to create an invalid capability.*
- [Cap\\_base](#) ([l4\\_default\\_caps\\_t](#) cap) noexcept  
*Initialize capability with one of the default capabilities.*
- **Cap\_base** () noexcept  
*Create an uninitialized instance.*
- void [move](#) ([Cap\\_base](#) const &src) const  
*Replace this capability with the contents of `src`.*
- void [copy](#) ([Cap\\_base](#) const &src) const  
*Copy a capability.*

### Protected Attributes inherited from [L4::Cap\\_base](#)

- [l4\\_cap\\_idx\\_t \\_c](#)  
*The C representation of a capability selector.*

### 15.199.1 Detailed Description

```
template<typename T, typename SMART>
class L4::Smart_cap< T, SMART >
```

Smart capability class.

Definition at line 36 of file [smart\\_capability](#).

### 15.199.2 Constructor & Destructor Documentation

#### 15.199.2.1 Smart\_cap()

```
template<typename T , typename SMART >
template<typename O >
L4::Smart_cap< T, SMART >::Smart_cap (
    Cap< O > const & p ) [inline], [noexcept]
```

Internal constructor, use to generate a capability from a `this` pointer.

#### Attention

This constructor is only useful to generate a capability from the `this` pointer of an objected that is an [L4::Kobject](#). Do `never` use this constructor for something else!

#### Parameters

<i>p</i>	The <code>this</code> pointer of the <a href="#">Kobject</a> or derived object
----------	--

Definition at line 73 of file [smart\\_capability](#).

The documentation for this class was generated from the following file:

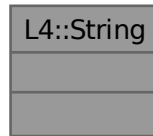
- [l4/sys/smart\\_capability](#)

## 15.200 L4::String Class Reference

A null-terminated string container class.

```
#include <string.h>
```

Collaboration diagram for L4::String:



### 15.200.1 Detailed Description

A null-terminated string container class.

Definition at line 33 of file [string.h](#).

The documentation for this class was generated from the following file:

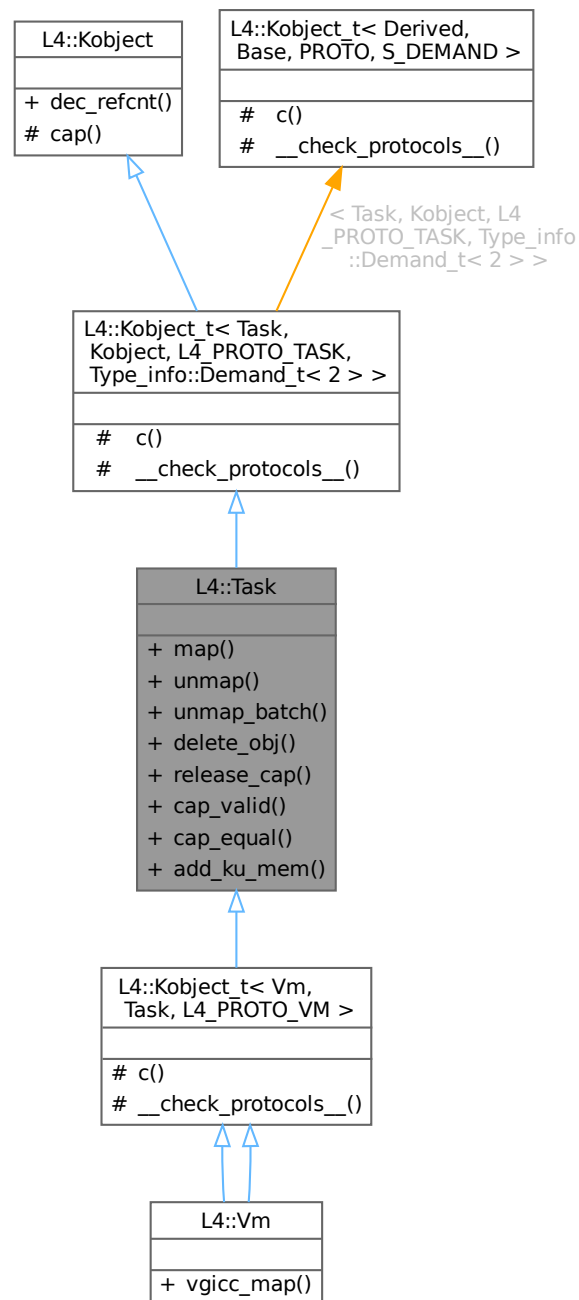
- [l4/cxx/string.h](#)

## 15.201 L4::Task Class Reference

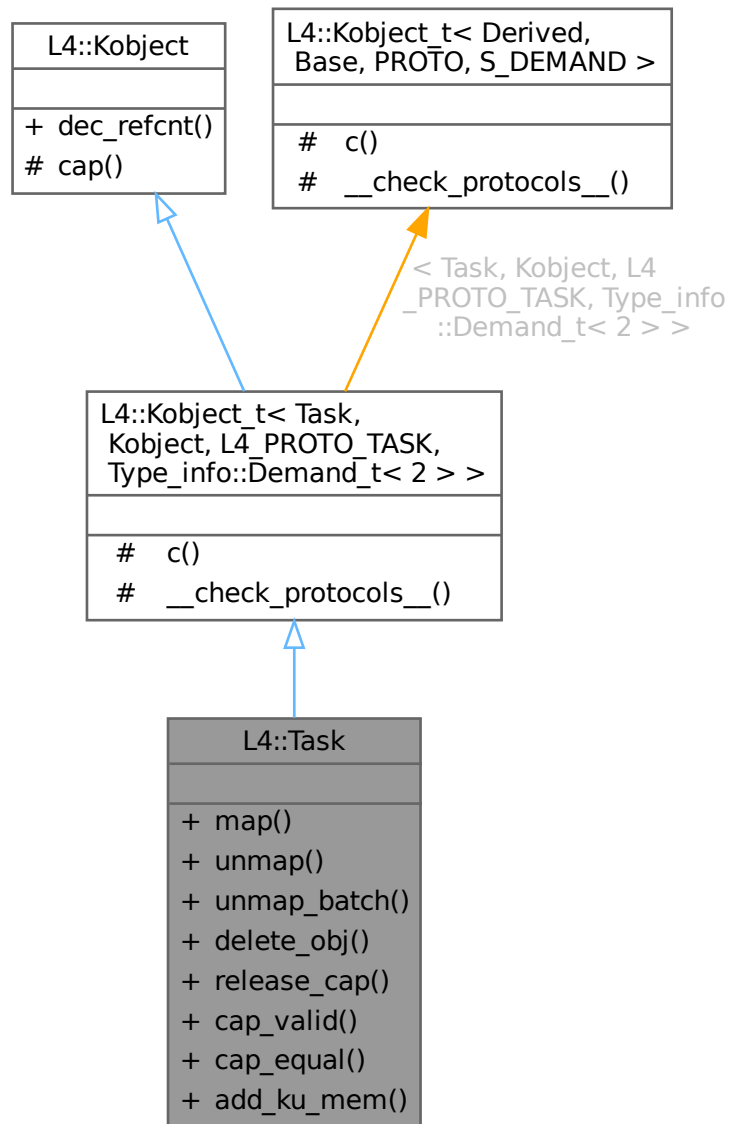
C++ interface of the [Task](#) kernel object, see [Task](#) for the C interface.

```
#include <task>
```

Inheritance diagram for L4::Task:



Collaboration diagram for L4::Task:



## Public Member Functions

- `l4_msgtag_t map (Cap< Task > const &src_task, l4_fpage_t const &snd_fpage, l4_umword_t snd_base, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Map resources available in the source task to a destination task.*
- `l4_msgtag_t unmap (l4_fpage_t const &fpag, l4_umword_t map_mask, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Revoke rights from the task.*
- `l4_msgtag_t unmap_batch (l4_fpage_t const *fpages, unsigned num_fpages, l4_umword_t map_mask, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Revoke rights from a task.*

- `l4_msgtag_t delete_obj (L4::Cap< void > obj, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Release capability and delete object.*
- `l4_msgtag_t release_cap (L4::Cap< void > cap, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Release object capability.*
- `l4_msgtag_t cap_valid (Cap< void > const &cap, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Check whether a capability is present (refers to an object).*
- `l4_msgtag_t cap_equal (Cap< void > const &cap_a, Cap< void > const &cap_b, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Test whether two capabilities point to the same object with the same rights.*
- `l4_msgtag_t add_ku_mem (l4_fpage_t *fpage, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Add kernel-user memory.*

## Public Member Functions inherited from `L4::Kobject`

- `l4_msgtag_t dec_refcnt (l4_mword_t diff, l4_utcb_t *utcb=l4_utcb())`  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

## Protected Types inherited from

`L4::Kobject_t< Task, Kobject, L4_PROTO_TASK, Type_info::Demand_t< 2 > >`

- typedef `Task Class`  
*The target interface type (inheriting from `Kobject_t`)*
- typedef `Typeid::Iface< PROTO, Task > __iface`  
*The interface description for the derived class.*
- typedef `Typeid::Merge_list< Typeid::Iface_list< __iface >, typename Base::__iface_list > __iface_list`  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from

`L4::Kobject_t< Task, Kobject, L4_PROTO_TASK, Type_info::Demand_t< 2 > >`

- `L4::Cap< Class > c () const noexcept`  
*Get the capability to ourselves.*

## Protected Member Functions inherited from `L4::Kobject`

- `l4_cap_idx_t cap () const noexcept`  
*Return capability selector.*

## Static Protected Member Functions inherited from

`L4::Kobject_t< Task, Kobject, L4_PROTO_TASK, Type_info::Demand_t< 2 > >`

- static void `__check_protocols__ () noexcept`  
*Helper to check for protocol conflicts.*



## 15.201.1 Detailed Description

C++ interface of the [Task](#) kernel object, see [Task](#) for the C interface.

The [L4::Task](#) class represents a combination of the address spaces provided by the [L4Re](#) micro kernel. A task consists of at least a memory address space and an object address space. On IA32 there is also an IO-port address space associated with an [L4::Task](#).

[L4::Task](#) objects are created using the [L4::Factory](#) interface.

### Include File

```
#include <l4/sys/task>
```

Definition at line 44 of file [task](#).

## 15.201.2 Member Function Documentation

### 15.201.2.1 add\_ku\_mem()

```
l4_msgtag_t L4::Task::add_ku_mem (
    l4_fpage_t * fpage,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

Add kernel-user memory.

#### Parameters

<i>in, out</i>	<i>fpage</i>	Flexpage describing the virtual area the memory goes to. On systems without MMU, the flexpage is adjusted to reflect the actually allocated physical address.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

#### Returns

Syscall return tag

Kernel-user memory (ku\_mem) is memory that is shared between the kernel and user-space. It is needed for the UTCB area of threads (see [L4::Thread::Attr::bind\(\)](#)) and for (extended) vCPU state. Note that existing kernel-user memory cannot be unmapped or mapped somewhere else.

#### Note

The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page ([L4\\_PAGE\\_SIZE](#)). A portable implementation should not depend on allocations greater than 16KiB to succeed.

This function is only guaranteed to work on [L4::Task](#) objects. It might or might not work on [L4::Vm](#) objects or on [L4Re::Dma\\_space](#) objects but there is no practical use for adding kernel-user memory to [L4::Vm](#) objects or to [L4Re::Dma\\_space](#) objects.

Definition at line 255 of file [task](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 15.201.2.2 cap\_equal()

```

l4_msgtag_t L4::Task::cap_equal (
    Cap< void > const & cap_a,
    Cap< void > const & cap_b,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Test whether two capabilities point to the same object with the same rights.

#### Parameters

<i>cap<sub>a</sub></i>	First capability selector to compare.
<i>cap<sub>b</sub></i>	Second capability selector to compare.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

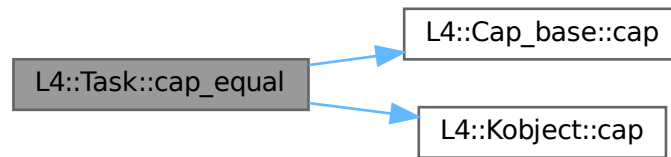
#### Return values

<i>l4_msgtag_t::label()</i> = 1	<i>cap<sub>a</sub></i> and <i>cap<sub>b</sub></i> point to the same object.
<i>l4_msgtag_t::label()</i> = 0	The two caps do <b>not</b> point to the same object.

Definition at line 225 of file [task](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 15.201.2.3 cap\_valid()

```

l4_msgtag_t L4::Task::cap_valid (
    Cap< void > const & cap,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Check whether a capability is present (refers to an object).

#### Parameters

<i>cap</i>	Valid capability to check for presence.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

#### Return values

<i>l4_msgtag_t::label()</i> > 0	Capability is present (refers to an object).
<i>l4_msgtag_t::label()</i> == 0	No capability present (void object).

A capability is considered present when it refers to an existing kernel object.

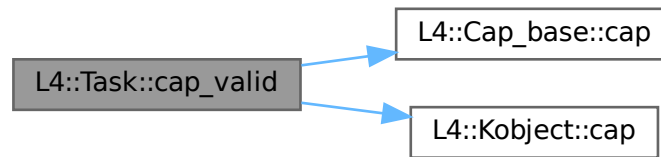
#### Precondition

`cap` must be a valid capability (i.e. `cap.is_valid() == true`). If you are unsure about the validity of your capability use [L4::Cap.validate\(\)](#) instead.

Definition at line 208 of file [task](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



#### 15.201.2.4 delete\_obj()

```

l4_msgtag_t L4::Task::delete_obj (
    L4::Cap< void > obj,
    l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
  
```

Release capability and delete object.

##### Parameters

<i>obj</i>	Capability index of the object to delete.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

##### Returns

Syscall return tag

If `obj` has the delete permission, initiates the deletion of the object. This implies that all capabilities for that object are gone afterwards. However, kernel-internally, objects are not destroyed until all other kernel objects holding a reference to it drop the reference. Hence, quota used by that object might not be freed immediately.

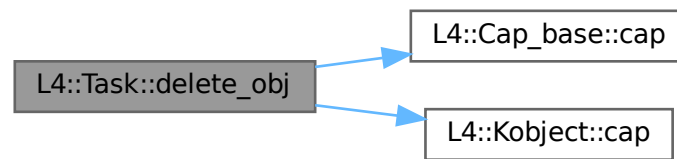
If `obj` does not have the delete permission, no error will be reported and only the capability `obj` is removed. (Note that, depending on the object's reference counter, this might still imply initiation of deletion.)

This operation is equivalent to [unmap\(\)](#) with `L4_FP_DELETE_OBJ` flag.

Definition at line 168 of file [task](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 15.201.2.5 map()

```

l4_msgtag_t L4::Task::map (
    Cap< Task > const & src_task,
    l4_fpage_t const & snd_fpage,
    l4_umword_t snd_base,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Map resources available in the source task to a destination task.

#### Parameters

<i>src_task</i>	Capability selector of the source task.
<i>snd_fpage</i>	Send flexpage that describes an area in the address space or object space of the source task.
<i>snd_base</i>	Send base that describes an offset in the receive window of the destination task. The lower bits contain additional map control flags (see <a href="#">l4_fpage_cacheability_opt_t</a> for memory mappings, <a href="#">L4_obj_fpage_ctl</a> for object mappings, and <a href="#">L4_MAP_ITEM_GRANT</a> ; also see <a href="#">l4_map_control()</a> and <a href="#">l4_map_obj_control()</a> ).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

#### Returns

Syscall return tag. The function [l4\\_error\(\)](#) shall be used to test if the map operation was successful.

#### Return values

<a href="#">L4_EOK</a>	Operation successful (but see notes below).
<a href="#">-L4_EPERM</a>	No <a href="#">L4_CAP_FPAGE_W</a> right on capability used to invoke this operation.
<a href="#">-L4_EINVAL</a>	Invalid source task capability.
<a href="#">-L4_IPC_SEMAPFAILED</a>	The map operation failed due to limited quota.

This method allows for asynchronous transfer of capabilities, memory mappings, and IO-port mappings (on IA32) from one task to another. The destination task is the task referenced by the capability on which the map is invoked, and the receive window is the whole address space of that task. By specifying proper rights in the `snd_fpage` and `snd_base`, it is possible to remove rights during transfer.

**Note**

If the send flex page is of type [L4\\_FPAGE\\_OBJ](#), the [L4\\_CAP\\_FPAGE\\_S](#) right is removed from the transferred capability unless both the source and destination task capabilities possess the [L4\\_CAP\\_FPAGE\\_S](#) right themselves.

Even with [l4\\_error\(\)](#) returning [L4\\_EOK](#) there might be cases where not all pages of the send flexpage were mapped respectively granted to the destination task, for instance, if the corresponding mapping in the destination task does already exist.

For more information on spaces and mappings, see [Spaces and Mappings](#). The flexpage API is described in more detail at [Flex pages](#).

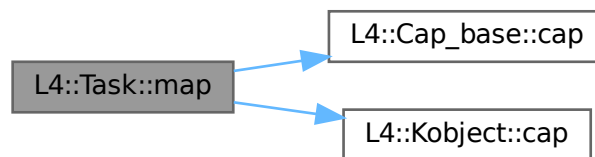
**Note**

For peculiarities when using grant, see [L4\\_MAP\\_ITEM\\_GRANT](#).

Definition at line 95 of file [task](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**15.201.2.6 release\_cap()**

```

l4_msgtag_t L4::Task::release_cap (
    L4::Cap< void > cap,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Release object capability.

**Parameters**

<i>cap</i>	Capability selector of the object to release.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

**Returns**

Syscall return tag.

This operation unmaps the capability from `this` task.

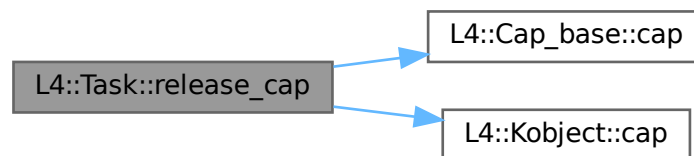
**Note**

If the reference counter of the kernel object referenced by `cap` goes down to zero, the deletion of the object is initiated. Objects are not destroyed until all other kernel objects holding a reference to it drop the reference.

Definition at line 187 of file [task](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**15.201.2.7 unmap()**

```

l4_msgtag_t L4::Task::unmap (
    l4_fpage_t const & fpage,
    l4_umword_t map_mask,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Revoke rights from the task.

**Parameters**

<i>fpage</i>	Flexpage that describes an area in one capability space of <code>this</code> task
<i>map_mask</i>	Unmap mask, see <a href="#">l4_unmap_flags_t</a>
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

**Returns**

Syscall return tag

This method allows to revoke rights from the destination task. For a flex page describing IO ports or memory, it also revokes rights from all the tasks that got the rights delegated from the destination task (i.e., this operation does a recursive rights revocation). If the set of rights is empty after the revocation, the capability is unmapped. It is guaranteed that the rights revocation is completed before this function returns.

**Note**

If the reference counter of a kernel object referenced in `fpage` goes down to zero (as a result of deleting capabilities), the deletion of the object is initiated. Objects are not destroyed until all other kernel objects holding a reference to it drop the reference.

Definition at line 123 of file [task](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 15.201.2.8 unmap\_batch()

```

l4_msgtag_t L4::Task::unmap_batch (
    l4_fpage_t const * fpages,
    unsigned num_fpages,
    l4_umword_t map_mask,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Revoke rights from a task.

#### Parameters

<i>fpages</i>	An array of flexpages. Each item describes an area in one capability space of <code>this</code> task.
<i>num_fpages</i>	Number of fpages in the <i>fpages</i> array.
<i>map_mask</i>	Unmap mask, see <a href="#">l4_unmap_flags_t</a> .
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

Revoke rights for an array of flexpages, see [unmap](#) for details.

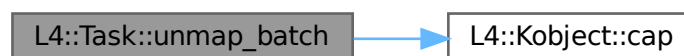
#### Precondition

The caller needs to take care that `num_fpages` is not bigger than `L4_UTCB_GENERIC_DATA_SIZE - 2`.

Definition at line 142 of file [task](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:





The documentation for this class was generated from the following file:

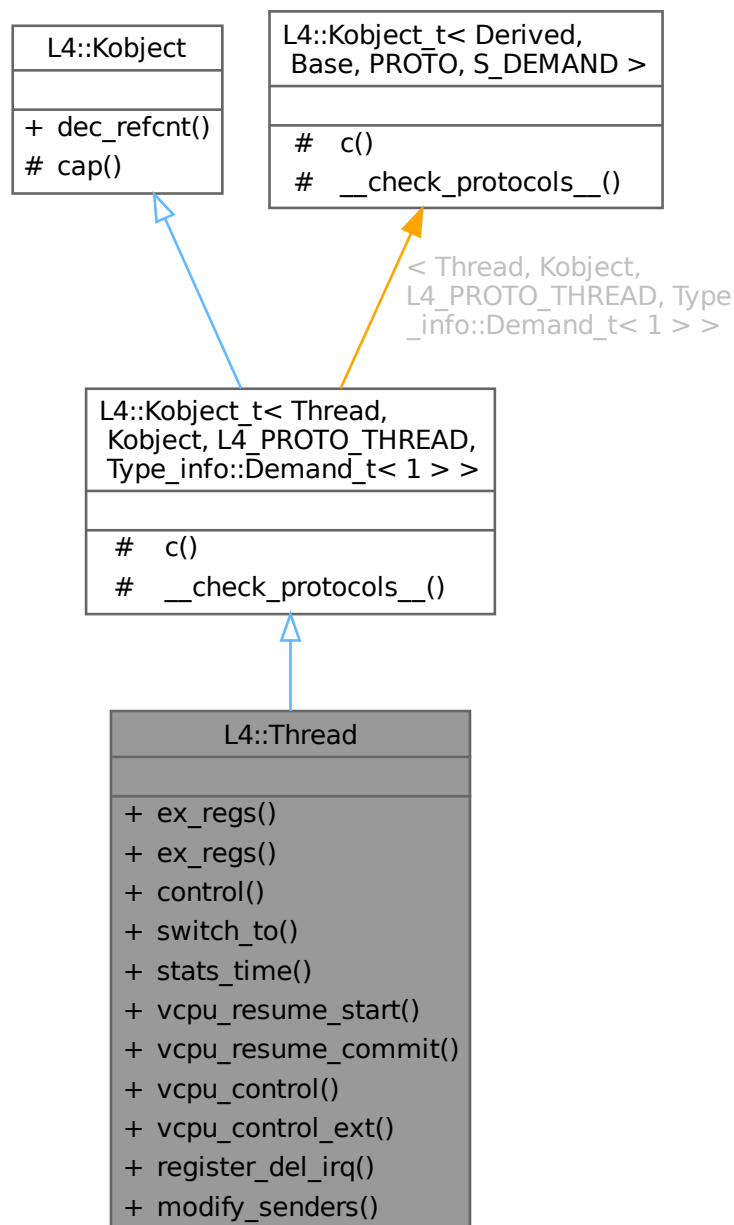
- [l4/sys/task](#)

## 15.202 L4::Thread Class Reference

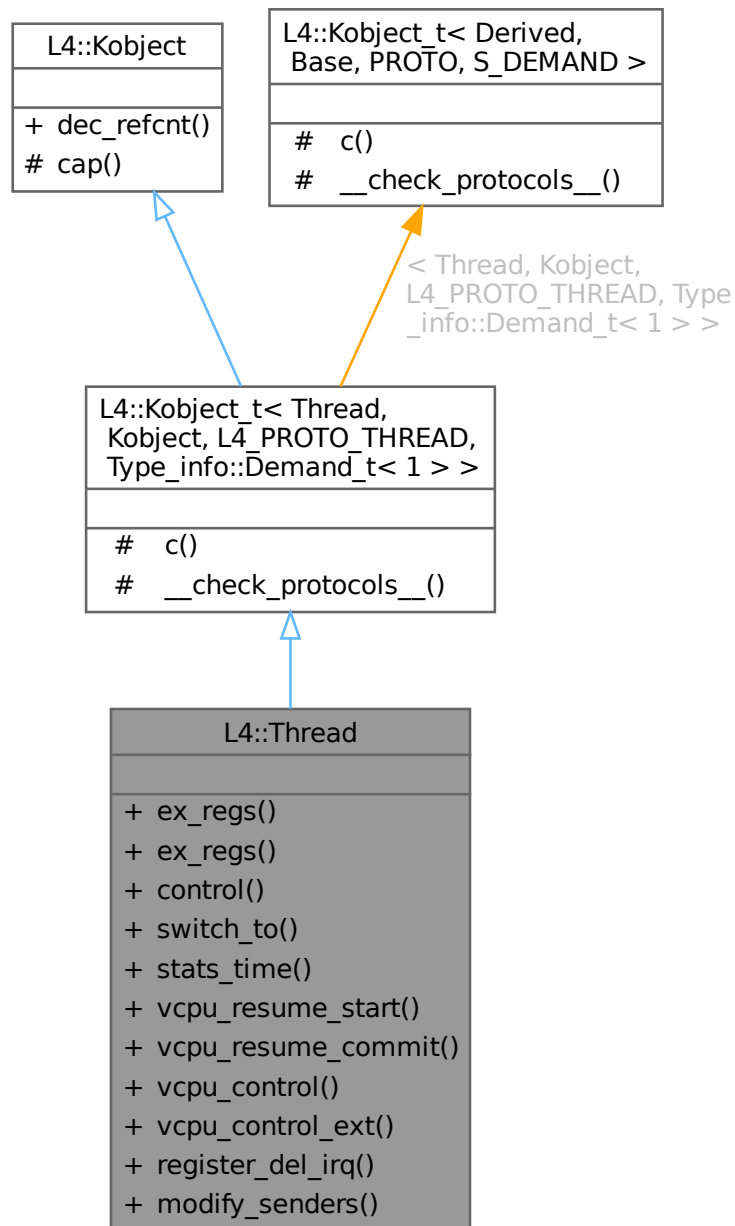
C++ [L4](#) kernel thread interface, see [Thread](#) for the C interface.

```
#include <thread>
```

Inheritance diagram for L4::Thread:



Collaboration diagram for L4::Thread:



## Data Structures

- class [Attr](#)

*Thread* attributes used for *control()*.

- class [Modify\\_senders](#)

*Class wrapping a list of rules which modify the sender label of IPC messages inbound to this thread.*

## Public Member Functions

- [l4\\_msgtag\\_t ex\\_regs](#) ([l4\\_addr\\_t](#) ip, [l4\\_addr\\_t](#) sp, [l4\\_umword\\_t](#) flags, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Exchange basic thread registers.*
- [l4\\_msgtag\\_t ex\\_regs](#) ([l4\\_addr\\_t](#) \*ip, [l4\\_addr\\_t](#) \*sp, [l4\\_umword\\_t](#) \*flags, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Exchange basic thread registers and return previous values.*
- [l4\\_msgtag\\_t control](#) ([Attr](#) const &attr) noexcept  
*Commit the given thread-attributes object.*
- [l4\\_msgtag\\_t switch\\_to](#) ([l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Switch execution to this thread.*
- [l4\\_msgtag\\_t stats\\_time](#) ([l4\\_kernel\\_clock\\_t](#) \*us, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Get consumed time of thread in us.*
- [l4\\_msgtag\\_t vcpu\\_resume\\_start](#) ([l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Resume from vCPU asynchronous IPC handler, start.*
- [l4\\_msgtag\\_t vcpu\\_resume\\_commit](#) ([l4\\_msgtag\\_t](#) tag, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Resume from vCPU asynchronous IPC handler, commit.*
- [l4\\_msgtag\\_t vcpu\\_control](#) ([l4\\_addr\\_t](#) vcpu\_state, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Enable the vCPU feature for the thread.*
- [l4\\_msgtag\\_t vcpu\\_control\\_ext](#) ([l4\\_addr\\_t](#) ext\_vcpu\_state, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Enable the extended vCPU feature for the thread.*
- [l4\\_msgtag\\_t register\\_del\\_irq](#) ([Cap](#) < [l4\\_irq](#) > irq, [l4\\_utcb\\_t](#) \*u=[l4\\_utcb](#)()) noexcept  
*Register an IRQ that will trigger upon deletion events.*
- [l4\\_msgtag\\_t modify\\_senders](#) ([Modify\\_senders](#) const &todo) noexcept  
*Apply sender modification rules.*

## Public Member Functions inherited from [L4::Kobject](#)

- [l4\\_msgtag\\_t dec\\_refcnt](#) ([l4\\_mword\\_t](#) diff, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)())  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

## Protected Types inherited from

[L4::Kobject\\_t](#) < [Thread](#), [Kobject](#), [L4\\_PROTO\\_THREAD](#), [Type\\_info::Demand\\_t](#) < 1 > >

- typedef [Thread](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef [Typeid::Iface](#) < [PROTO](#), [Thread](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef [Typeid::Merge\\_list](#) < [Typeid::Iface\\_list](#) < **\_\_Iface** >, typename [Base::\\_\\_Iface\\_list](#) > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from

[L4::Kobject\\_t](#) < [Thread](#), [Kobject](#), [L4\\_PROTO\\_THREAD](#), [Type\\_info::Demand\\_t](#) < 1 > >

- [L4::Cap](#) < [Class](#) > **c** () const noexcept  
*Get the capability to ourselves.*

## Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t](#) [cap](#) () const noexcept  
*Return capability selector.*

## Static Protected Member Functions inherited from [L4::Kobject\\_t< Thread, Kobject, L4\\_PROTO\\_THREAD, Type\\_info::Demand\\_t< 1 > >](#)

- static void [\\_\\_check\\_protocols\\_\\_](#) () noexcept  
*Helper to check for protocol conflicts.*

### 15.202.1 Detailed Description

C++ [L4](#) kernel thread interface, see [Thread](#) for the C interface.

The [Thread](#) class defines a thread of execution in the [L4](#) context. Usually user-level and kernel threads are mapped 1:1 to each other. [Thread](#) kernel objects are created using a factory, see the [L4::Factory](#) API ([L4::Factory::create\(\)](#)).

Amongst other things an [L4::Thread](#) encapsulates:

- CPU state
  - General-purpose registers
  - Program counter
  - Stack pointer
- FPU state
- Scheduling parameters, see the [L4::Scheduler](#) API
- Execution state
  - Blocked, Runnable, Running

[Thread](#) objects provide an API for

- [Thread](#) configuration and manipulation
- [Thread](#) switching.

#### Include File

```
#include <l4/sys/thread>
```

For the C interface see the [Thread](#) API. For more elaborated documentation on the vCPU feature see [vCPU API](#).

Definition at line 59 of file [thread](#).

### 15.202.2 Member Function Documentation

#### 15.202.2.1 [control\(\)](#)

```
l4\_msgtag\_t L4::Thread::control (  
    Attr const & attr ) [inline], [noexcept]
```

Commit the given thread-attributes object.

## Parameters

<i>attr</i>	the attribute object to commit to the thread.
-------------	---

## Returns

Syscall return tag containing one of the following return codes.

## Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EPERM</i>	No <a href="#">L4_CAP_FPAGE_S</a> right on the capability used to invoke this operation or the task capability set in <a href="#">Attr::bind()</a> .
<i>-L4_EINVAL</i>	Malformed thread-attributes.

Definition at line 249 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

15.202.2.2 `ex_regs()` [1/2]

```

14_msgtag_t L4::Thread::ex_regs (
    14_addr_t * ip,
    14_addr_t * sp,
    14_umword_t * flags,
    14_utcb_t * utcb = 14_utcb() ) [inline], [noexcept]
  
```

Exchange basic thread registers and return previous values.

## Parameters

in, out	<i>ip</i>	New instruction pointer, use ~0UL to leave the instruction pointer unchanged, return previous instruction pointer.
in, out	<i>sp</i>	New stack pointer, use ~0UL to leave the stack pointer unchanged, returns previous stack pointer.
in, out	<i>flags</i>	Ex-regs flags, see <a href="#">L4_thread_ex_regs_flags</a> , return previous CPU flags of the thread.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">14_utcb</a> .

## Returns

System call return tag. [out] parameters are only valid if the function returns successfully. Use [l4\\_error\(\)](#) to check.

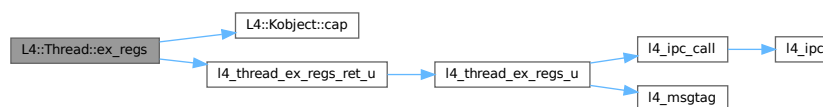
This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see [flags](#)). If the thread is in an IPC operation or if [L4\\_THREAD\\_EX\\_REGS\\_TRIGGER\\_EXCEPTION](#) forces an IPC then changes in IP and SP take effect directly after returning from this IPC.

The thread is started using [L4::Scheduler::run\\_thread\(\)](#). However, if at the time [L4::Scheduler::run\\_thread\(\)](#) is called, the instruction pointer of the thread is invalid, a later call to [ex\\_regs\(\)](#) with a valid instruction pointer might start the thread.

Definition at line 123 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4\\_thread\\_ex\\_regs\\_ret\\_u\(\)](#).

Here is the call graph for this function:



### 15.202.2.3 ex\_regs() [2/2]

```

l4_msgtag_t L4::Thread::ex_regs (
    l4_addr_t ip,
    l4_addr_t sp,
    l4_umword_t flags,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Exchange basic thread registers.

## Parameters

<i>ip</i>	New instruction pointer, use ~0UL to leave the instruction pointer unchanged.
<i>sp</i>	New stack pointer, use ~0UL to leave the stack pointer unchanged.
<i>flags</i>	Ex-regs flags, see <a href="#">L4_thread_ex_regs_flags</a> .
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

## Returns

System call return tag.

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and

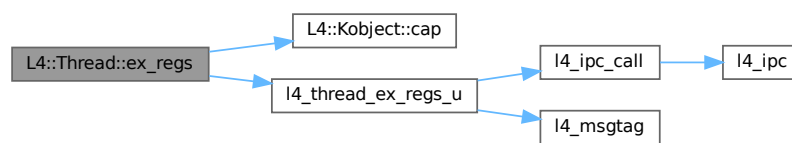
to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if `L4_THREAD_EX_REGS_TRIGGER_EXCEPTION` forces an IPC then changes in IP and SP take effect directly after returning from this IPC.

The thread is started using `L4::Scheduler::run_thread()`. However, if at the time `L4::Scheduler::run_thread()` is called, the instruction pointer of the thread is invalid, a later call to `ex_regs()` with a valid instruction pointer might start the thread.

Definition at line 90 of file `thread`.

References `L4::Kobject::cap()`, and `l4_thread_ex_regs_u()`.

Here is the call graph for this function:



#### 15.202.2.4 modify\_senders()

```
l4_msgtag_t L4::Thread::modify_senders (
    Modify_senders const & todo ) [inline], [noexcept]
```

Apply sender modification rules.

##### Parameters

<i>todo</i>	Prepared sender modification rules.
-------------	-------------------------------------

##### Returns

System call return tag.

The modification rules are applied to all IPCs to the thread (whether directly or by IPC gate) that are already in flight, that is that the sender is already blocking on.

See `Modify_senders` for a detailed description when applying sender modification rules is required.

##### Note

Modifying the senders of a thread running on a different CPU core is not supported.

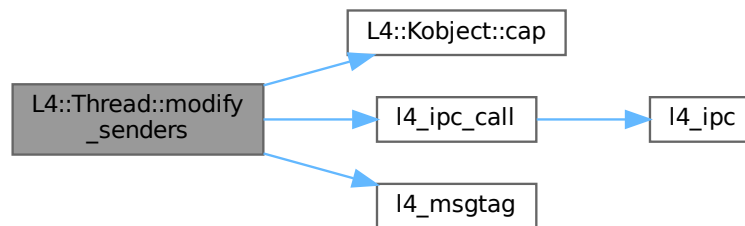
See also

[l4\\_thread\\_modify\\_sender\\_commit\(\)](#)

Definition at line 501 of file [thread](#).

References [L4::Kobject::cap\(\)](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), and [L4\\_PROTO\\_THREAD](#).

Here is the call graph for this function:



### 15.202.2.5 register\_del\_irq()

```

l4_msgtag_t L4::Thread::register_del_irq (
    Cap< Irq > irq,
    l4_utcb_t * u = l4_utcb() ) [inline], [noexcept]
  
```

Register an IRQ that will trigger upon deletion events.

#### Parameters

<i>irq</i>	Capability selector for the IRQ object to be triggered.
<i>u</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

#### Returns

System call return tag containing the return code.

#### Return values

<code>-L4_BUSY</code>	A deletion IRQ is already bound to this thread.
<code>-L4_EPERM</code>	<a href="#">L4_CAP_FPAGE_W</a> missing on <code>irq</code>

In case the `irq` is already bound to an interrupt source, it is unbound first. When `irq` is deleted, it will be deregistered first. A registered deletion `irq` can only be deregistered by deleting the `irq` or the thread.

List of deletion events:



- deletion of one or several IPC gates bound to this thread.

When the deletion event is delivered, there is no indication about which IPC gate was deleted.

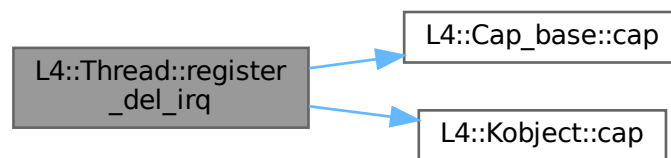
See also

[l4\\_thread\\_register\\_del\\_irq](#)

Definition at line 414 of file [thread](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 15.202.2.6 stats\_time()

```
l4_msgtag_t L4::Thread::stats_time (
    l4_kernel_clock_t * us,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

Get consumed time of thread in us.

Parameters

out	<i>us</i>	Consumed time in $\mu$ s.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

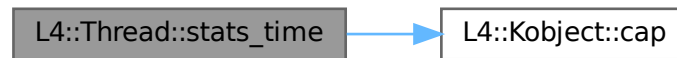
Returns

Syscall return tag.

Definition at line 270 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 15.202.2.7 switch\_to()

```

l4_msgtag_t L4::Thread::switch_to (
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Switch execution to this thread.

#### Parameters

<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .
-------------	--

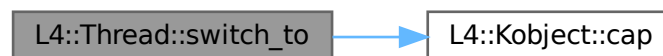
#### Note

The current time slice is inherited to this thread.

Definition at line 259 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 15.202.2.8 vcpu\_control()

```

l4_msgtag_t L4::Thread::vcpu_control (
    l4_addr_t vcpu_state,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Enable the vCPU feature for the thread.

## Parameters

<i>vcpu_state</i>	A virtual address pointing to a <a href="#">l4_vcpu_state_t</a> . It must be a valid kernel-user-memory address (see <a href="#">L4::Task::add_ku_mem()</a> ).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

## Returns

Syscall return tag.

This function enables the vCPU feature of `this` thread

The kernel-user memory starting at `vcpu_state` must be at least 128-byte aligned and must cover the size of [l4\\_vcpu\\_state\\_t](#).

The asynchronous IPC handling is described at [vCPU API](#).

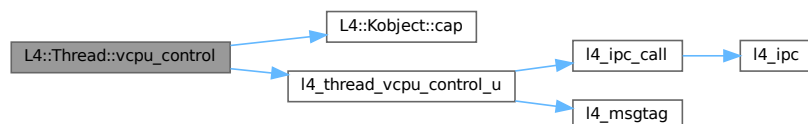
## Note

Disabling of the vCPU feature is optional and currently not supported.

Definition at line 354 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4\\_thread\\_vcpu\\_control\\_u\(\)](#).

Here is the call graph for this function:



## 15.202.2.9 vcpu\_control\_ext()

```

l4_msgtag_t L4::Thread::vcpu_control_ext (
    l4_addr_t ext_vcpu_state,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Enable the extended vCPU feature for the thread.

## Parameters

<i>ext_vcpu_state</i>	The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see <a href="#">L4::Task::add_ku_mem()</a> ).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

**Returns**

Syscall return tag.

The extended vCPU feature allows the use of hardware-virtualization features such as Intel's VT or AMD's SVM.

This function enables the extended vCPU feature of `this` thread. Enabling the extended vCPU feature also enables the vCPU feature.

The kernel-user memory area starting at `ext_vcpu_state` must be at least 4 KiB aligned and must cover a size of `L4_PAGESIZE`. It includes the data of `l4_vcpu_state_t` at offset 0, the extended vCPU state at offset `L4_VCPU_OFFSET_EXT_STATE`, and, on some platforms, the extended vCPU information at offset `L4_VCPU_OFFSET_EXT_INFOS`.

**Note**

Enabling the extended vCPU feature for a thread running on a different CPU core is currently not supported.

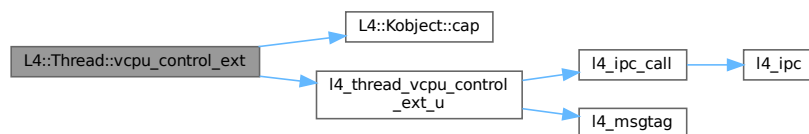
Disabling of the extended vCPU feature is currently not supported.

Upgrading from non-extended vCPU feature to extended vCPU feature is currently not supported.

Definition at line 387 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4\\_thread\\_vcpu\\_control\\_ext\\_u\(\)](#).

Here is the call graph for this function:

**15.202.2.10 vcpu\_resume\_commit()**

```

l4_msgtag_t L4::Thread::vcpu_resume_commit (
    l4_msgtag_t tag,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Resume from vCPU asynchronous IPC handler, commit.

**Parameters**

<i>tag</i>	Tag to use, returned by <a href="#">l4_thread_vcpu_resume_start()</a> .
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

**Returns**

Syscall return tag containing one of the following return codes.

## Return values

0	Indicates a VM exit, provided that <code>thread</code> is in extended vCPU mode with virtual interrupts cleared.
1	Indicates an incoming IPC message, provided that the <code>thread</code> is in extended vCPU mode with virtual interrupts cleared.
-L4_EPERM	The user task capability set in the vCPU state is missing the <a href="#">L4_CAP_FPAGE_S</a> right.
-L4_ENOENT	The user task capability set in the vCPU state is invalid.
-L4_EINVAL	<code>thread</code> is not the current running thread, or does not have the vCPU feature enabled.
<0	A supplied mapping failed.

All flex pages in the UTCB (added with [l4\\_sndfpage\\_add\(\)](#) after [l4\\_thread\\_vcpu\\_resume\\_start\(\)](#)) are unconditionally mapped into the user task configured in the vCPU state.

To resume into another address space, the capability to the target [Task](#) (or [L4::Vm](#)) must be set in [l4\\_vcpu\\_state\\_t::user\\_task](#) together with [L4\\_VCPU\\_F\\_USER\\_MODE](#). The capability selector must have all lower bits clear (see [L4\\_CAP\\_MASK](#)). The kernel adds the [L4\\_SYSF\\_SEND](#) flag there to indicate that the capability has been referenced in the kernel. Consecutive resumes will not reference the task capability again until all lower bits are cleared again. To release a task use a different task capability or use an invalid capability with the [L4\\_SYSF\\_REPLY](#) flag set.

The asynchronous IPC handling is described at [vCPU API](#).

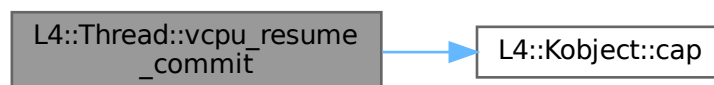
## See also

[l4\\_thread\\_vcpu\\_resume\\_commit](#)

Definition at line 330 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



## 15.202.2.11 vcpu\_resume\_start()

```

l4_msgtag_t L4::Thread::vcpu_resume_start (
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Resume from vCPU asynchronous IPC handler, start.

## Parameters

<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .
-------------	--

## Returns

Message tag to be used for [l4\\_sndfpage\\_add\(\)](#) and [l4\\_thread\\_vcpu\\_resume\\_commit\(\)](#)

The vCPU resume functionality is split in multiple functions to allow the specification of additional send-flex-pages using [l4\\_sndfpage\\_add\(\)](#).

The asynchronous IPC handling is described at [vCPU API](#).

## See also

[l4\\_thread\\_vcpu\\_resume\\_start](#)

Definition at line [289](#) of file [thread](#).

The documentation for this class was generated from the following file:

- [l4/sys/thread](#)

## 15.203 L4::Thread::Attr Class Reference

[Thread](#) attributes used for [control\(\)](#).

```
#include <thread>
```

Collaboration diagram for L4::Thread::Attr:



## Public Member Functions

- [Attr](#) ([l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Create a thread-attribute object with the given UTCB.*
- void [pager](#) ([Cap](#)< void > const &pager) noexcept  
*Set the pager capability selector.*
- [Cap](#)< void > [pager](#) () noexcept  
*Get the capability selector used for page-fault messages.*
- void [exc\\_handler](#) ([Cap](#)< void > const &exc\_handler) noexcept  
*Set the exception-handler capability selector.*
- [Cap](#)< void > [exc\\_handler](#) () noexcept  
*Get the capability selector used for exception messages.*
- void [bind](#) ([l4\\_utcb\\_t](#) \*thread\_utcb, [Cap](#)< [Task](#) > const &task) noexcept  
*Bind the thread to a task.*
- void [alien](#) (int on) noexcept  
*Enable alien mode.*
- void [ux\\_host\\_syscall](#) (int on) noexcept  
*Allow host system calls on Fiasco-UX.*

## Friends

- class [L4::Thread](#)

## 15.203.1 Detailed Description

[Thread](#) attributes used for [control\(\)](#).

This class is responsible for initializing various attributes of a thread in a UTCB for the [control\(\)](#) method.

### Note

Instantiation of this class starts the preparation of the UTCB. Do not invoke any non-Attr functions between the instantiation and the call to [L4::Thread::control\(\)](#).

### See also

[Thread control](#) for some more details.

Definition at line [141](#) of file [thread](#).

## 15.203.2 Constructor & Destructor Documentation

### 15.203.2.1 Attr()

```
L4::Thread::Attr::Attr (
    l4\_utcb\_t * utcb = l4\_utcb() ) [inline], [explicit], [noexcept]
```

Create a thread-attribute object with the given UTCB.

## Parameters

<i>utcb</i>	The UTCB to use for the later <a href="#">L4::Thread::control()</a> function. Usually this is the UTCB of the calling thread. See <a href="#">l4_utcb()</a> .
-------------	---

Definition at line [155](#) of file [thread](#).

### 15.203.3 Member Function Documentation

#### 15.203.3.1 [alien\(\)](#)

```
void L4::Thread::Attr::alien (
    int on ) [inline], [noexcept]
```

Enable alien mode.

## Parameters

<i>on</i>	Boolean value defining the state of the feature.
-----------	--

For a thread in alien mode the kernel produces just an exception IPC for each IPC and exception caused by the alien thread instead of handling these events regularly. (Page faults of alien threads and interrupts occurring while the alien thread is running are always handled regularly.) While the alien thread is blocking, the exception handler can inspect and modify the state of the alien thread and potentially also the syscall arguments. If the exception handler replies with [L4\\_PROTO\\_ALLOW\\_SYSCALL](#) as message tag, the kernel handles the next IPC or exception of the alien thread in a regular way. If the exception handler leaves certain thread state unchanged (in particular the instruction pointer), this will be the IPC or exception that caused the call of the exception handler. For a regularly processed IPC or exception of the alien thread the kernel also performs an exception IPC on kernel exit.

This feature can be used to attach a debugger to a thread and trace all object invocations and their results. It could also be used to handle other systems that use the same syscall instruction as [L4Re](#).

Definition at line [223](#) of file [thread](#).

#### 15.203.3.2 [bind\(\)](#)

```
void L4::Thread::Attr::bind (
    l4_utcb_t * thread_utcb,
    Cap< Task > const & task ) [inline], [noexcept]
```

Bind the thread to a task.

## Parameters

<i>thread_utcb</i>	The thread's UTCB address within the task it shall be bound to. The address must be aligned (architecture dependent; at least word aligned) and it must point to at least <a href="#">L4_UTCB_OFFSET</a> bytes of kernel-user memory.
<i>task</i>	The task the thread shall be bound to.



**Precondition**

The thread must not be bound to a task yet.

A thread may execute code in the context of a task if and only if the thread is bound to the task. To actually start execution, `L4::Thread::ex_regs()` needs to be used. Execution in the context of the task means that the code has access to all the task's resources (and only those). The executed code itself must be one of those resources. A thread can be bound at most once to a task.

**Note**

The UTCBs of different threads in the same task should not overlap in order to prevent data corruption.

Definition at line 217 of file `thread`.

**15.203.3.3 `exc_handler()` [1/2]**

```
Cap< void > L4::Thread::Attr::exc_handler ( ) [inline], [noexcept]
```

Get the capability selector used for exception messages.

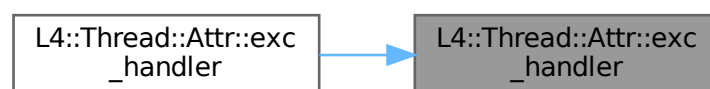
**Returns**

The capability selector used to send exception messages. The selector is valid in the task the thread is bound to.

Definition at line 193 of file `thread`.

Referenced by `exc_handler()`.

Here is the caller graph for this function:

**15.203.3.4 `exc_handler()` [2/2]**

```
void L4::Thread::Attr::exc_handler (
    Cap< void > const & exc_handler ) [inline], [noexcept]
```

Set the exception-handler capability selector.

## Parameters

<i>exc_handler</i>	The capability selector that shall be used for exception messages. This capability selector must be valid within the task the thread is bound to.
--------------------	---

Definition at line 184 of file [thread](#).

References [exc\\_handler\(\)](#).

Here is the call graph for this function:



### 15.203.3.5 pager() [1/2]

```
Cap< void > L4::Thread::Attr::pager ( ) [inline], [noexcept]
```

Get the capability selector used for page-fault messages.

## Returns

The capability selector used to send page-fault messages. The selector is valid in the task the thread is bound to.

Definition at line 174 of file [thread](#).

Referenced by [pager\(\)](#).

Here is the caller graph for this function:



### 15.203.3.6 pager() [2/2]

```
void L4::Thread::Attr::pager (
    Cap< void > const & pager ) [inline], [noexcept]
```

Set the pager capability selector.

## Parameters

<i>pager</i>	The capability selector that shall be used for page-fault messages. This capability selector must be valid within the task the thread is bound to.
--------------	--

Definition at line 165 of file [thread](#).

References [pager\(\)](#).

Here is the call graph for this function:



## 15.203.3.7 ux\_host\_syscall()

```
void L4::Thread::Attr::ux_host_syscall (
    int on ) [inline], [noexcept]
```

Allow host system calls on Fiasco-UX.

## Precondition

Running on Fiasco-UX.

Definition at line 231 of file [thread](#).

The documentation for this class was generated from the following file:

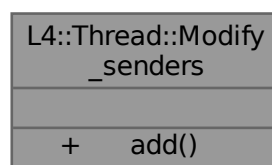
- [l4/sys/thread](#)

## 15.204 L4::Thread::Modify\_senders Class Reference

Class wrapping a list of rules which modify the sender label of IPC messages inbound to this thread.

```
#include <thread>
```

Collaboration diagram for L4::Thread::Modify\_senders:



## Public Member Functions

- `int add (l4_umword_t match_mask, l4_umword_t match, l4_umword_t del_bits, l4_umword_t add_bits) noexcept`

*Add a rule.*

### 15.204.1 Detailed Description

Class wrapping a list of rules which modify the sender label of IPC messages inbound to this thread.

Use the `add()` function to add modification rules, and use `modify_senders()` to commit. Do not use the UTCTB in between as it is used by `add()` and `modify_senders()`.

This mechanism shall be used to change the source object labels of every pending IPC of an IPC gate or an IRQ if the labels in such pending IPC become invalid for the receiving thread, potentially because:

- a thread was unbound from an IPC gate / IRQ, or
- an IPC gate / IRQ was removed, or
- the label of an IPC gate / IRQ bound to a thread was changed.

It is not required to perform the `modify_sender` mechanism after an IPC gate or an IRQ was bound to a thread for the first time.

Definition at line 435 of file `thread`.

### 15.204.2 Member Function Documentation

#### 15.204.2.1 add()

```
int L4::Thread::Modify_senders::add (
    l4_umword_t match_mask,
    l4_umword_t match,
    l4_umword_t del_bits,
    l4_umword_t add_bits ) [inline], [noexcept]
```

Add a rule.

#### Parameters

<i>match_mask</i>	Bitmask of bits to match the label.
<i>match</i>	Bitmask that must be equal to the label after applying <i>match_mask</i> .
<i>del_bits</i>	Bits to be deleted from the label.
<i>add_bits</i>	Bits to be added to the label.

#### Returns

0 on success, <0 on error

In pseudo code: if `((sender_label & match_mask) == match) { sender_label = (sender_label & ~del_bits) | add_bits; }`

Only the first match is applied.

See also

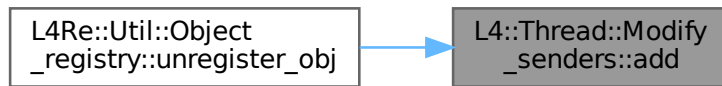
[l4\\_thread\\_modify\\_sender\\_add\(\)](#)

Definition at line [468](#) of file [thread](#).

References [L4\\_ENOMEM](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [L4Re::Util::Object\\_registry::unregister\\_obj\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

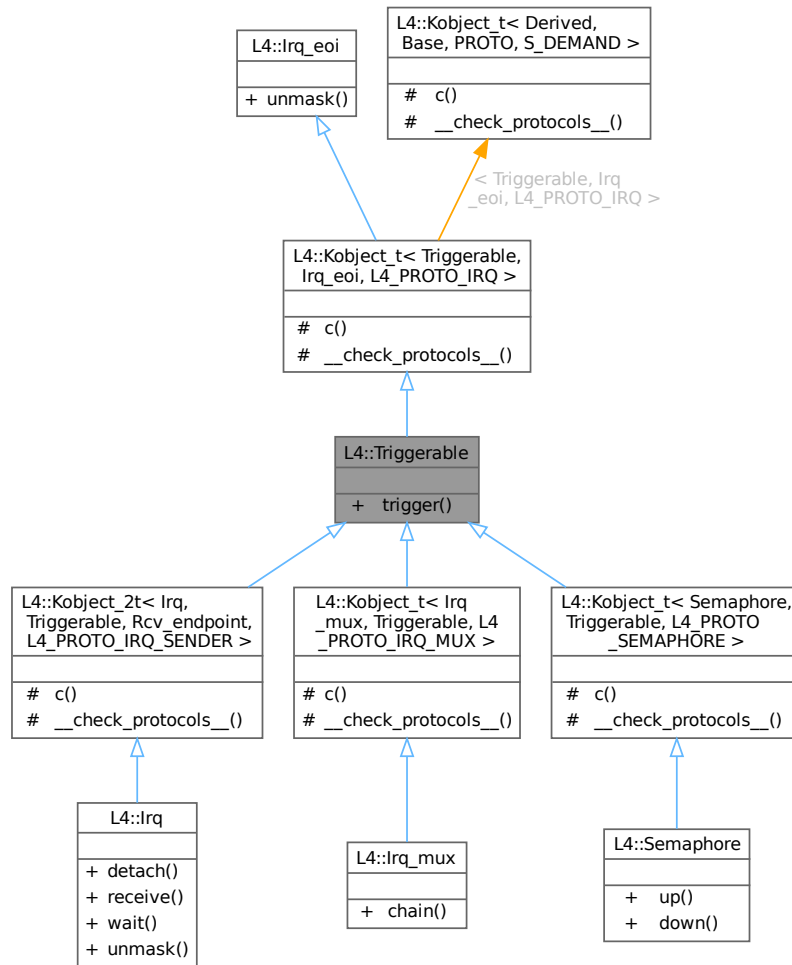
- [l4/sys/thread](#)

## 15.205 L4::Triggerable Struct Reference

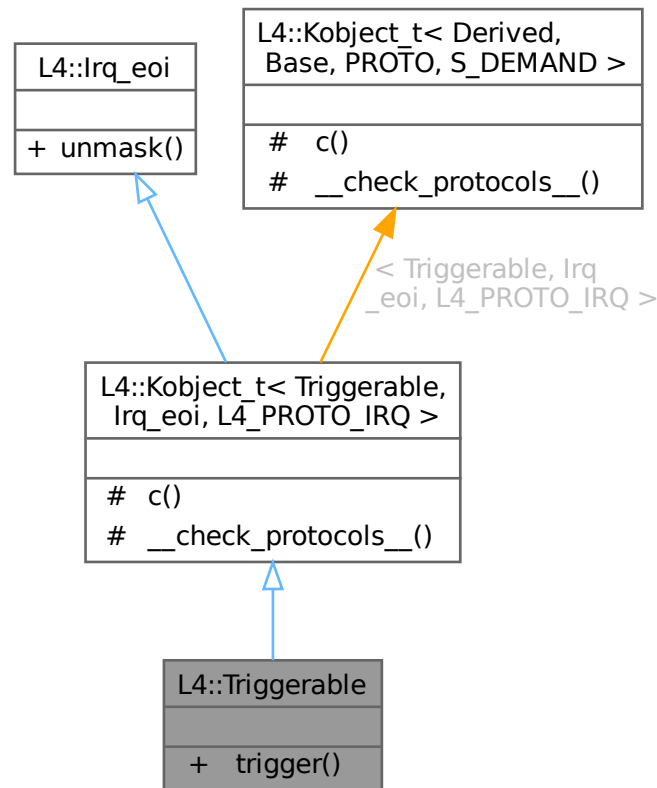
Interface that allows an object to be triggered by some source.

```
#include <irq>
```

Inheritance diagram for L4::Triggerable:



Collaboration diagram for L4::Triggerable:



### Public Member Functions

- [l4\\_msgtag\\_t trigger](#) ([l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Trigger the object.*

### Public Member Functions inherited from [L4::Irq\\_eoi](#)

- [l4\\_msgtag\\_t unmask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Unmask the given interrupt line.*

### Additional Inherited Members

### Protected Types inherited from [L4::Kobject\\_t< Triggerable, Irq\\_eoi, L4\\_PROTO\\_IRQ >](#)

- typedef [Triggerable](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, [Triggerable](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_Iface](#) >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from [L4::Kobject\\_t< Triggerable, Irq\\_eoi, L4\\_PROTO\\_IRQ >](#)

- [L4::Cap< Class > c\(\)](#) const noexcept  
*Get the capability to ourselves.*

## Static Protected Member Functions inherited from [L4::Kobject\\_t< Triggerable, Irq\\_eoi, L4\\_PROTO\\_IRQ >](#)

- static void [\\_\\_check\\_protocols\\_\\_\(\)](#) noexcept  
*Helper to check for protocol conflicts.*

### 15.205.1 Detailed Description

Interface that allows an object to be triggered by some source.

The interface specifies no semantics for the trigger operation, this is defined by derived objects.

This interface is usually used in conjunction with [L4::lcu](#).

Definition at line 90 of file [irq](#).

### 15.205.2 Member Function Documentation

#### 15.205.2.1 trigger()

```
l4_msgtag_t L4::Triggerable::trigger (
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

Trigger the object.

#### Parameters

<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
-------------	--

#### Returns

Syscall return tag for a send-only operation, this means there is no return value except [L4\\_MSGTAG\\_ERROR](#) indicating success or failure of the send operation. Use [l4\\_ipc\\_error\(\)](#) to check for errors and **do not** use [l4\\_error\(\)](#).

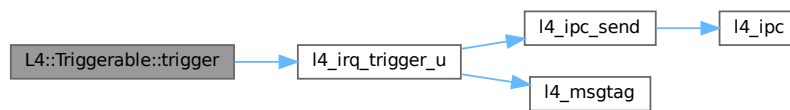
Definition at line 102 of file [irq](#).

References [l4\\_irq\\_trigger\\_u\(\)](#).

Referenced by [L4::Semaphore::up\(\)](#).



Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

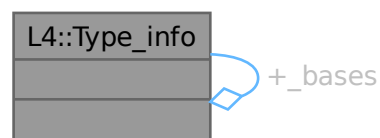
- [l4/sys/irq](#)

## 15.206 L4::Type\_info Struct Reference

Dynamic Type Information for [L4Re](#) Interfaces.

```
#include <l4/sys/capability>
```

Collaboration diagram for `L4::Type_info`:



## Data Structures

- class [Demand](#)  
*Data type for expressing the needed receive buffers at the server-side of an interface.*
- struct [Demand\\_t](#)  
*Template type statically describing demand of receive buffers.*
- struct [Demand\\_union\\_t](#)  
*Template type statically describing the combination of two [Demand](#) object.*

### 15.206.1 Detailed Description

Dynamic Type Information for [L4Re](#) Interfaces.

This class represents the runtime-dynamic type information for [L4Re](#) interfaces, and is not intended to be used directly by applications.

#### Note

The interface of is subject to changes.

The main use for this info is to be used by the implementation of the [L4::cap\\_dynamic\\_cast\(\)](#) function.

Definition at line [510](#) of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

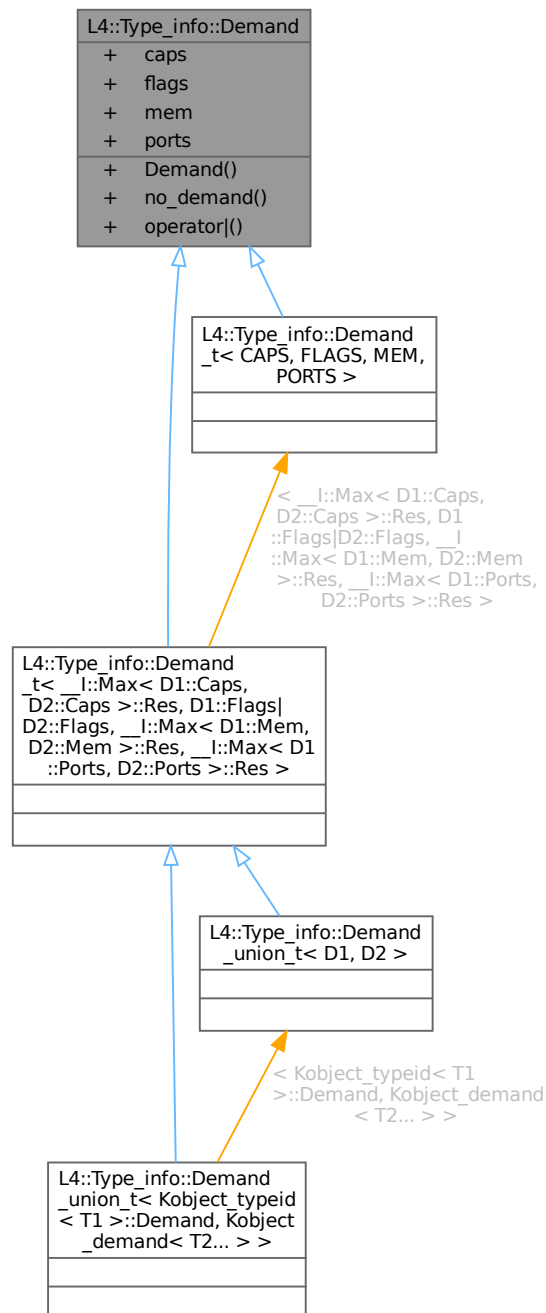
- [l4/sys/\\_\\_typeinfo.h](#)

## 15.207 L4::Type\_info::Demand Class Reference

Data type for expressing the needed receive buffers at the server-side of an interface.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Type\_info::Demand:



Collaboration diagram for L4::Type\_info::Demand:

L4::Type_info::Demand	
+	caps
+	flags
+	mem
+	ports
+	Demand()
+	no_demand()
+	operator ()

### Public Member Functions

- **Demand** (unsigned char **caps**=0, unsigned char **flags**=0, unsigned char **mem**=0, unsigned char **ports**=0) noexcept  
*Make **Demand** object.*
- bool **no\_demand** () const noexcept
- **Demand operator|** (**Demand** const &rhs) const noexcept  
*get the combined demand of this and rhs*

### Data Fields

- unsigned char **caps**  
*number of capability receive buffers.*
- unsigned char **flags**  
*flags, such as the need for timeouts (TBD).*
- unsigned char **mem**  
*number of memory receive buffers.*
- unsigned char **ports**  
*number of IO-port receive buffers.*

## 15.207.1 Detailed Description

Data type for expressing the needed receive buffers at the server-side of an interface.

Definition at line 517 of file `__typeinfo.h`.

## 15.207.2 Constructor & Destructor Documentation

### 15.207.2.1 Demand()

```
L4::Type_info::Demand::Demand (
    unsigned char caps = 0,
    unsigned char flags = 0,
    unsigned char mem = 0,
    unsigned char ports = 0 ) [inline], [explicit], [noexcept]
```

Make [Demand](#) object.

#### Parameters

<i>caps</i>	number of capability receive buffers
<i>flags</i>	flags, such as the need for timeouts (TBD).
<i>mem</i>	number of memory receive windows.
<i>ports</i>	number of IO-port receive windows.

Definition at line 538 of file [\\_\\_typeinfo.h](#).

## 15.207.3 Member Function Documentation

### 15.207.3.1 no\_demand()

```
bool L4::Type_info::Demand::no_demand ( ) const [inline], [noexcept]
```

#### Returns

true if there is no demand at all

Definition at line 543 of file [\\_\\_typeinfo.h](#).

Referenced by [L4::lpc\\_svr::Br\\_manager\\_no\\_buffers::alloc\\_buffer\\_demand\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

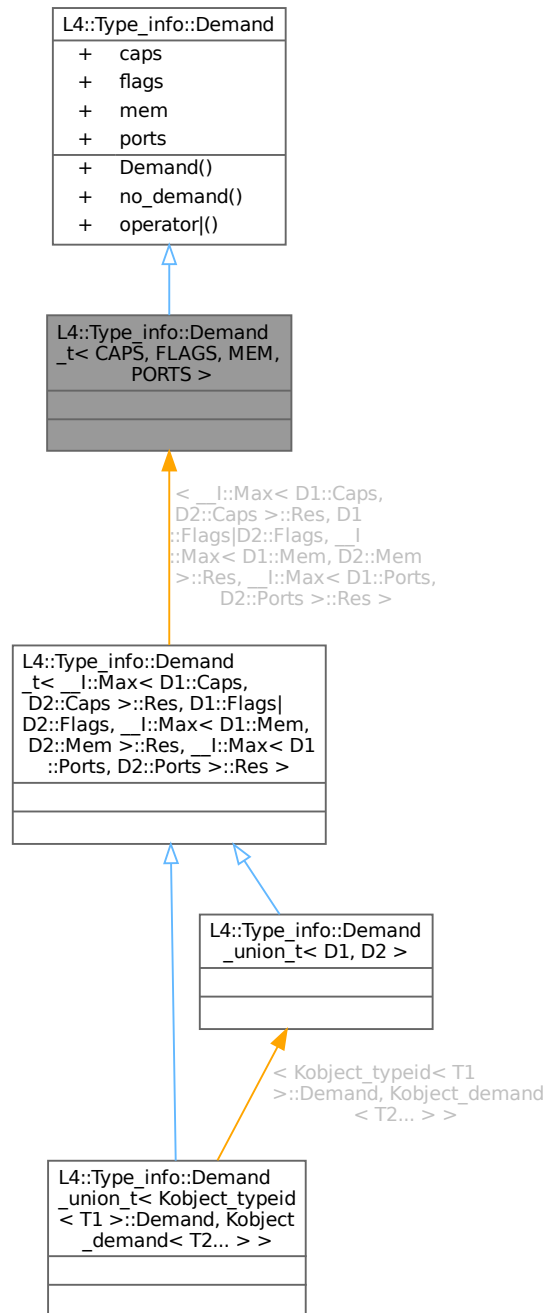
- [l4/sys/\\_\\_typeinfo.h](#)

## 15.208 L4::Type\_info::Demand\_t< CAPS, FLAGS, MEM, PORTS > Struct Template Reference

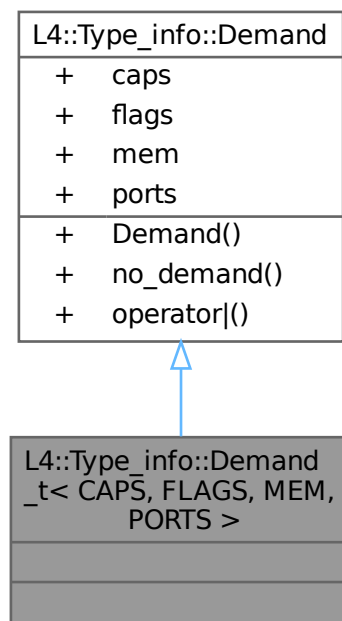
Template type statically describing demand of receive buffers.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Type\_info::Demand\_t< CAPS, FLAGS, MEM, PORTS >:



Collaboration diagram for L4::Type\_info::Demand\_t< CAPS, FLAGS, MEM, PORTS >:



## Public Types

- enum { `Caps` = CAPS , `Flags` = FLAGS , `Mem` = MEM , `Ports` = PORTS }

## Additional Inherited Members

## Public Member Functions inherited from L4::Type\_info::Demand

- `Demand` (unsigned char `caps`=0, unsigned char `flags`=0, unsigned char `mem`=0, unsigned char `ports`=0) noexcept  
*Make `Demand` object.*
- bool `no_demand` () const noexcept
- `Demand operator|` (`Demand` const &rhs) const noexcept  
*get the combined demand of this and rhs*

## Data Fields inherited from L4::Type\_info::Demand

- unsigned char `caps`  
*number of capability receive buffers.*
- unsigned char `flags`  
*flags, such as the need for timeouts (TBD).*
- unsigned char `mem`  
*number of memory receive buffers.*
- unsigned char `ports`  
*number of IO-port receive buffers.*

### 15.208.1 Detailed Description

```
template<unsigned char CAPS = 0, unsigned char FLAGS = 0, unsigned char MEM = 0, unsigned char
PORTS = 0>
struct L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >
```

Template type statically describing demand of receive buffers.

#### Template Parameters

<i>CAPS</i>	number of capability receive buffers needed.
<i>FLAGS</i>	flags, such as the need for timeouts (TBD).
<i>MEM</i>	number of memory receive windows needed.
<i>PORTS</i>	number of IO-port receive windows needed.

Definition at line 564 of file [\\_\\_typeinfo.h](#).

### 15.208.2 Member Enumeration Documentation

#### 15.208.2.1 anonymous enum

```
template<unsigned char CAPS = 0, unsigned char FLAGS = 0, unsigned char MEM = 0, unsigned char
PORTS = 0>
anonymous enum
```

#### Enumerator

Caps	number of capability receive buffers.
Flags	flags, such as the need for timeouts.
Mem	number of memory receive windows.
Ports	number of IO-port receive windows.

Definition at line 566 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

- [l4/sys/\\_\\_typeinfo.h](#)

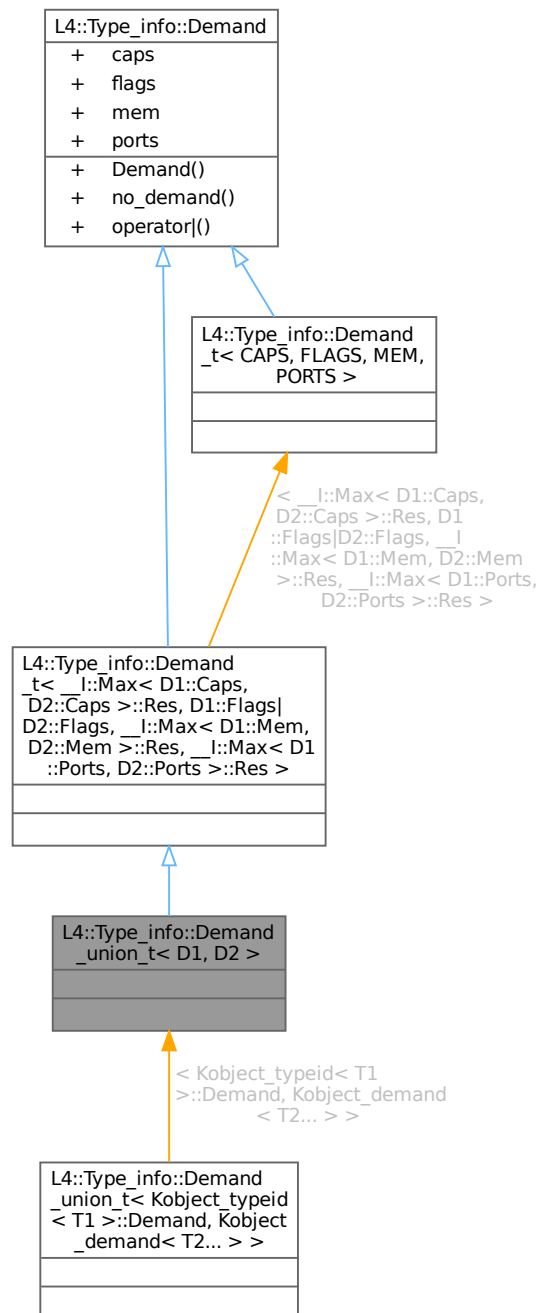
## 15.209 L4::Type\_info::Demand\_union\_t< D1, D2 > Struct Template Reference

Template type statically describing the combination of two [Demand](#) object.

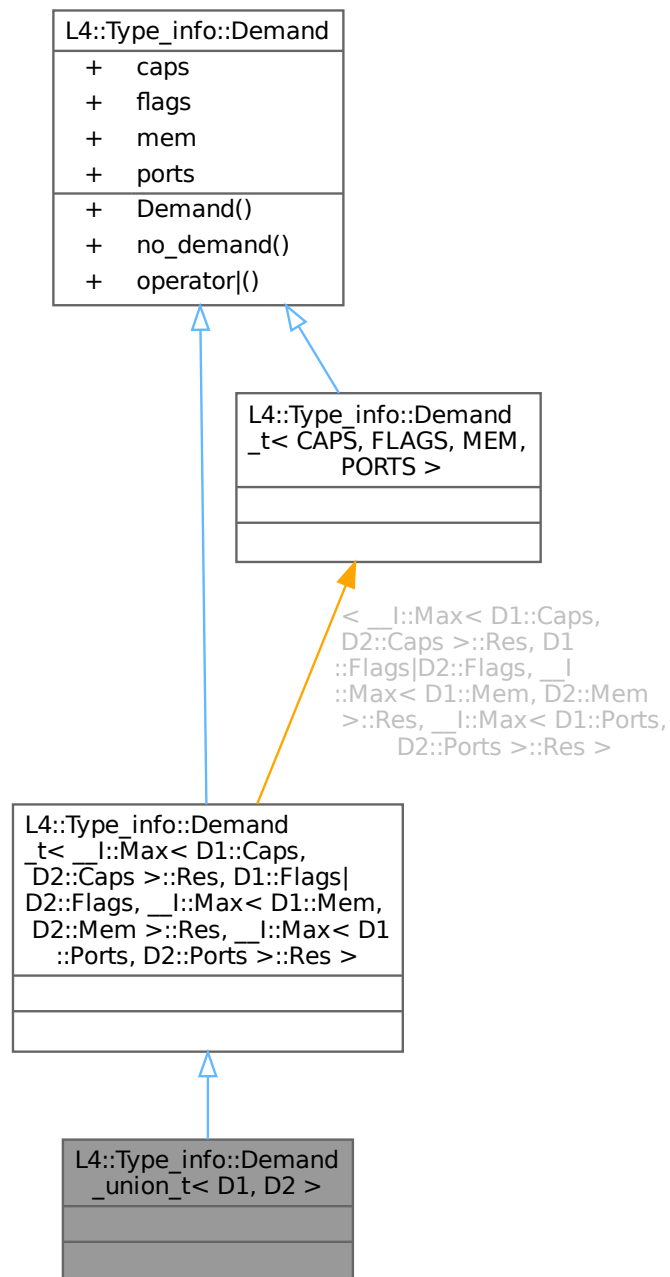
```
#include <l4/sys/capability>
```



Inheritance diagram for L4::Type\_info::Demand\_union\_t< D1, D2 >:



Collaboration diagram for L4::Type\_info::Demand\_union\_t< D1, D2 >:



#### Additional Inherited Members

#### Public Member Functions inherited from L4::Type\_info::Demand

- `Demand` (unsigned char `caps`=0, unsigned char `flags`=0, unsigned char `mem`=0, unsigned char `ports`=0) noexcept

Make [Demand](#) object.

- bool [no\\_demand](#) () const noexcept
- [Demand operator](#)| ([Demand](#) const &rhs) const noexcept

*get the combined demand of this and rhs*

## Data Fields inherited from [L4::Type\\_info::Demand](#)

- unsigned char **caps**  
*number of capability receive buffers.*
- unsigned char **flags**  
*flags, such as the need for timeouts (TBD).*
- unsigned char **mem**  
*number of memory receive buffers.*
- unsigned char **ports**  
*number of IO-port receive buffers.*

### 15.209.1 Detailed Description

```
template<typename D1, typename D2>
struct L4::Type_info::Demand_union_t< D1, D2 >
```

Template type statically describing the combination of two [Demand](#) object.

Template Parameters

<i>D1</i>	first demand object.
<i>D2</i>	second demand object.

Definition at line [584](#) of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

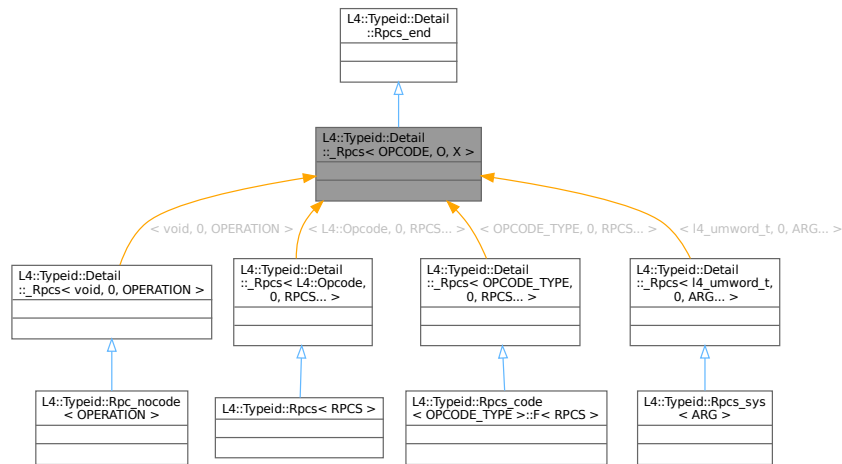
- [l4/sys/\\_\\_typeinfo.h](#)

## 15.210 L4::Typeid::Detail::\_Rpc< OPCODE, O, X > Struct Template Reference

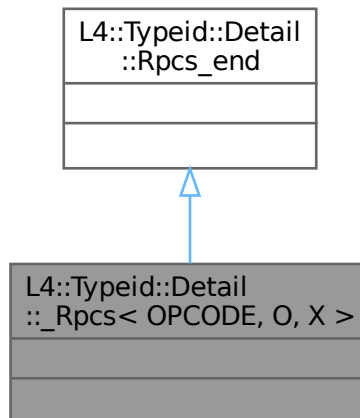
Empty list of RPCs.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Detail::\_Rpc< OPCODE, O, X >:



Collaboration diagram for L4::Typeid::Detail::\_Rpc< OPCODE, O, X >:



### 15.210.1 Detailed Description

```
template<typename OPCODE, unsigned O, typename ... X>
struct L4::Typeid::Detail::_Rpc< OPCODE, O, X >
```

Empty list of RPCs.

Definition at line 376 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

- [l4/sys/\\_\\_typeinfo.h](#)

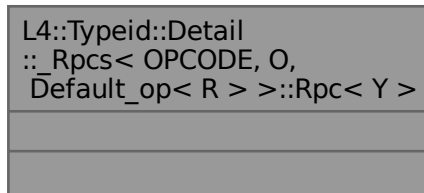
## 15.211 L4::Typeid::Detail::\_Rpc< OPCODE, O, Default\_op< R > >::Rpc< Y > Struct Template Reference

Find the given RPC in the list.

```
#include <__typeinfo.h>
```

Inherits L4::Typeid::Detail::\_Rpc< OP, RPCS >.

Collaboration diagram for L4::Typeid::Detail::\_Rpc< OPCODE, O, Default\_op< R > >::Rpc< Y >:



### 15.211.1 Detailed Description

```
template<typename OPCODE, unsigned O, typename R>
template<typename Y>
struct L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y >
```

Find the given RPC in the list.

Definition at line 410 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

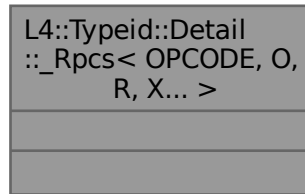
- [l4/sys/\\_\\_typeinfo.h](#)

## 15.212 L4::Typeid::Detail::\_Rpc< OPCODE, O, R, X... > Struct Template Reference

Non-empty list of RPCs.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Typeid::Detail::\_Rpc< OPCODE, O, R, X... >:



## Data Structures

- struct [Rpc](#)

*Find the given RPC in the list.*

## Public Types

- enum  
*The opcode value to use for this RPC, may be bogus if the opcode\_type is void.*
- typedef [\\_Rpc](#) **type**  
*The list element itself.*
- typedef OPCODE **opcode\_type**  
*The data type for the opcode.*
- typedef R **rpc**  
*The RPC type L4::lpc::Msg::Rpc\_call or L4::lpc::Msg::Rpc\_inline\_call.*
- typedef [\\_Rpc](#)< OPCODE, \_Get\_opcode< R, O >::value+1, X... >::type **next**  
*The next RPC in the list or [Rpc\\_end](#) if this is the last.*

### 15.212.1 Detailed Description

```
template<typename OPCODE, unsigned O, typename R, typename ... X>
struct L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >
```

Non-empty list of RPCs.

Definition at line 380 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

- l4/sys/[\\_\\_typeinfo.h](#)

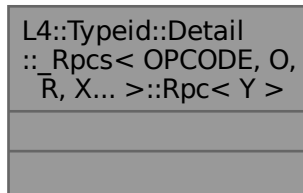
## 15.213 L4::Typeid::Detail::\_Rpc< OPCODE, O, R, X... >::Rpc< Y > Struct Template Reference

Find the given RPC in the list.

```
#include <__typeinfo.h>
```

Inherits L4::Typeid::Detail::\_Rpc< OP, RPCS >.

Collaboration diagram for L4::Typeid::Detail::\_Rpc< OPCODE, O, R, X... >::Rpc< Y >:



### 15.213.1 Detailed Description

```
template<typename OPCODE, unsigned O, typename R, typename ... X>
template<typename Y>
struct L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y >
```

Find the given RPC in the list.

Definition at line 393 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

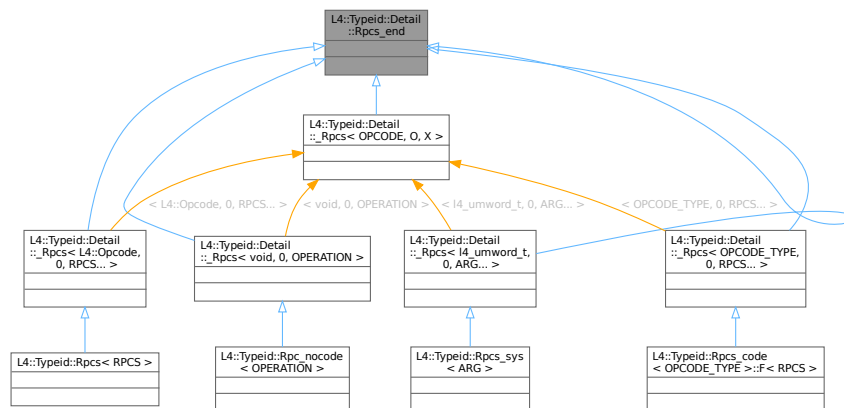
- [l4/sys/\\_\\_typeinfo.h](#)

## 15.214 L4::Typeid::Detail::\_Rpc\_end Struct Reference

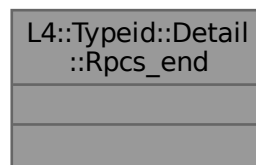
Internal end-of-list marker.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Detail::Rpc\_end:



Collaboration diagram for L4::Typeid::Detail::Rpc\_end:



### 15.214.1 Detailed Description

Internal end-of-list marker.

Definition at line 328 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

- [l4/sys/\\_\\_typeinfo.h](#)

## 15.215 L4::Typeid::P\_dispatch< LIST > Struct Template Reference

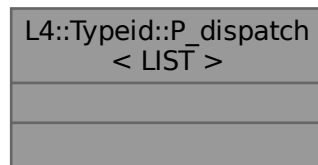
Use for protocol based dispatch stage.

```
#include <__typeinfo.h>
```

Inherits L4::Typeid::P\_dispatch< LIST >.



Collaboration diagram for L4::Typeid::P\_dispatch< LIST >:



### 15.215.1 Detailed Description

```
template<typename LIST>
struct L4::Typeid::P_dispatch< LIST >
```

Use for protocol based dispatch stage.

Definition at line 319 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

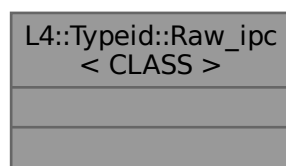
- [l4/sys/\\_\\_typeinfo.h](#)

## 15.216 L4::Typeid::Raw\_ipc< CLASS > Struct Template Reference

RPCs list for passing raw incoming IPC to the server object.

```
#include <l4/sys/capability>
```

Collaboration diagram for L4::Typeid::Raw\_ipc< CLASS >:



### 15.216.1 Detailed Description

```
template<typename CLASS>
struct L4::Typeid::Raw_ipc< CLASS >
```

RPCs list for passing raw incoming IPC to the server object.

## Template Parameters

<i>CLASS</i>	The type of the interface (e.g., <a href="#">L4::lcu</a> )
--------------	--

This template allows to have fully handcrafted IPC protocols.

Definition at line 423 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

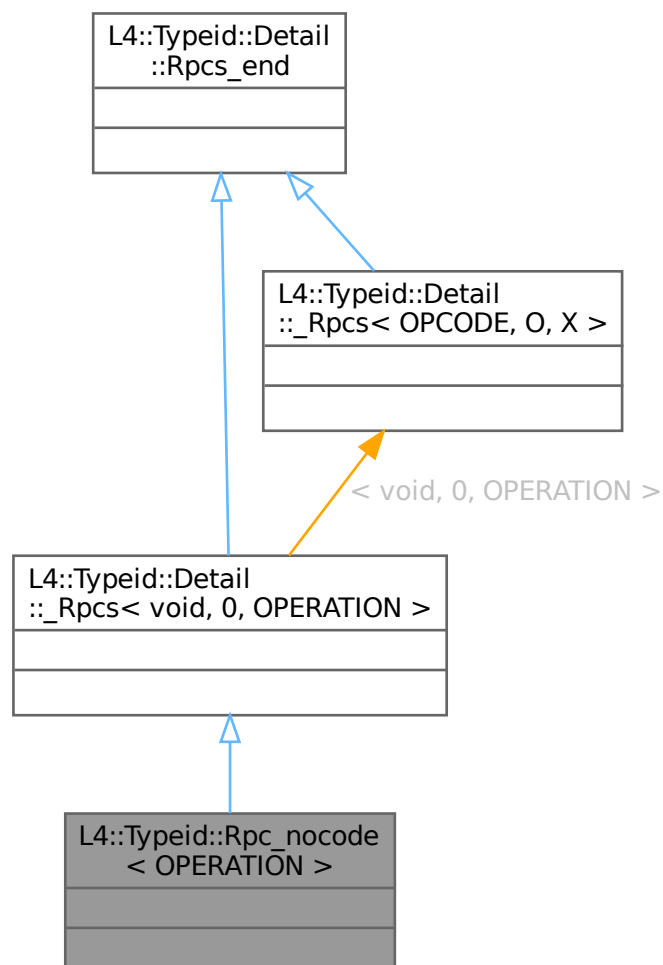
- [l4/sys/\\_\\_typeinfo.h](#)

## 15.217 L4::Typeid::Rpc\_nocode< OPERATION > Struct Template Reference

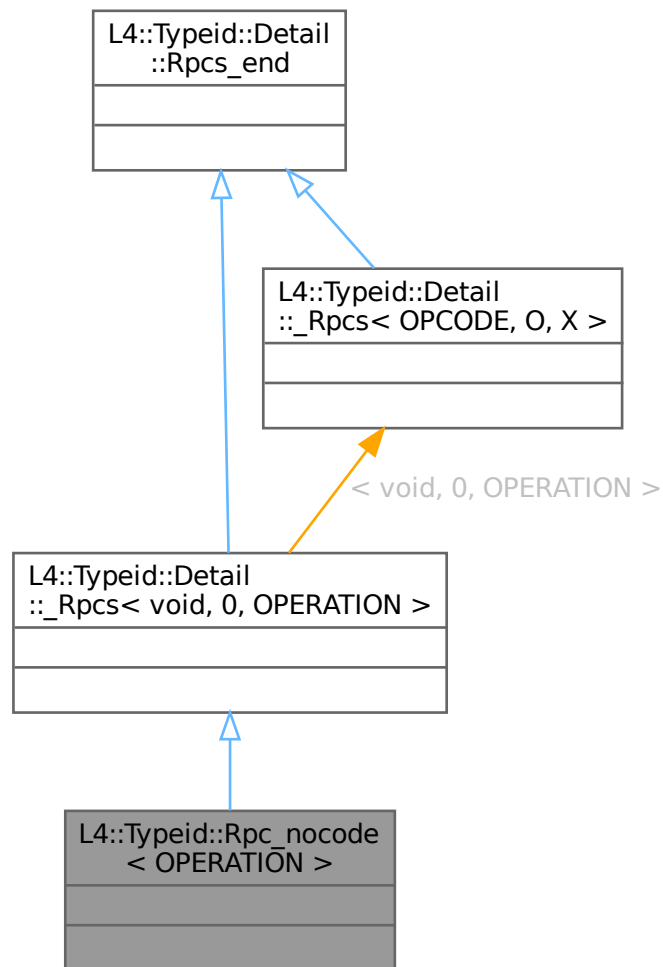
List of RPCs of an interface using a single operation without an opcode.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpc\_nocode< OPERATION >:



Collaboration diagram for L4::Typeid::Rpc\_nocode< OPERATION >:



### 15.217.1 Detailed Description

```
template<typename OPERATION>
struct L4::Typeid::Rpc_nocode< OPERATION >
```

List of RPCs of an interface using a single operation without an opcode.

Template Parameters

<i>OPERATION</i>	The RPC operation as defined by L4_RPC etc.
------------------	---

Definition at line 465 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

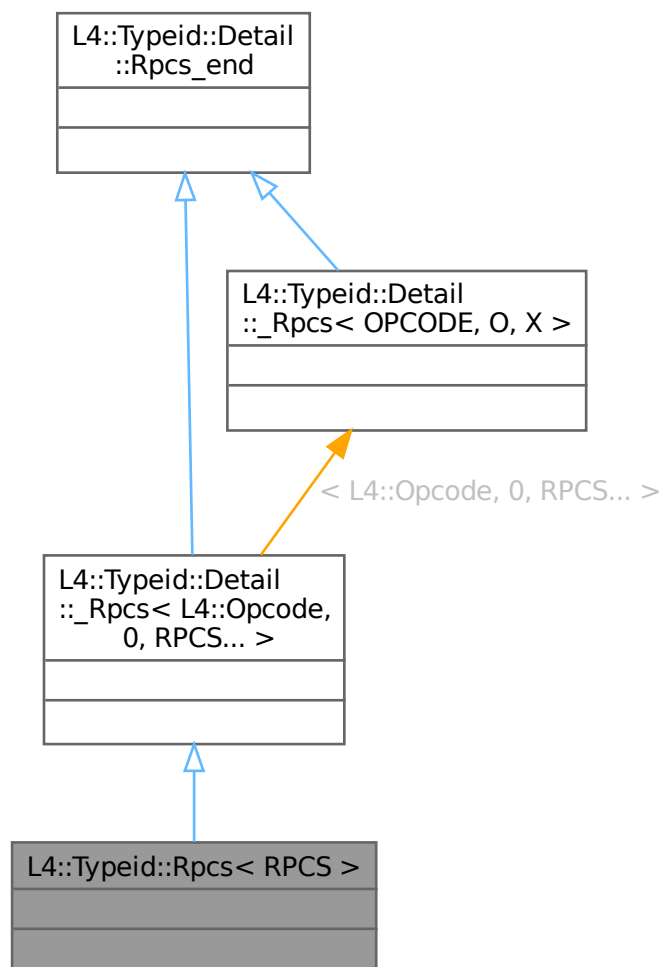
- [l4/sys/\\_\\_typeinfo.h](#)

## 15.218 L4::Typeid::Rpc< RPCS > Struct Template Reference

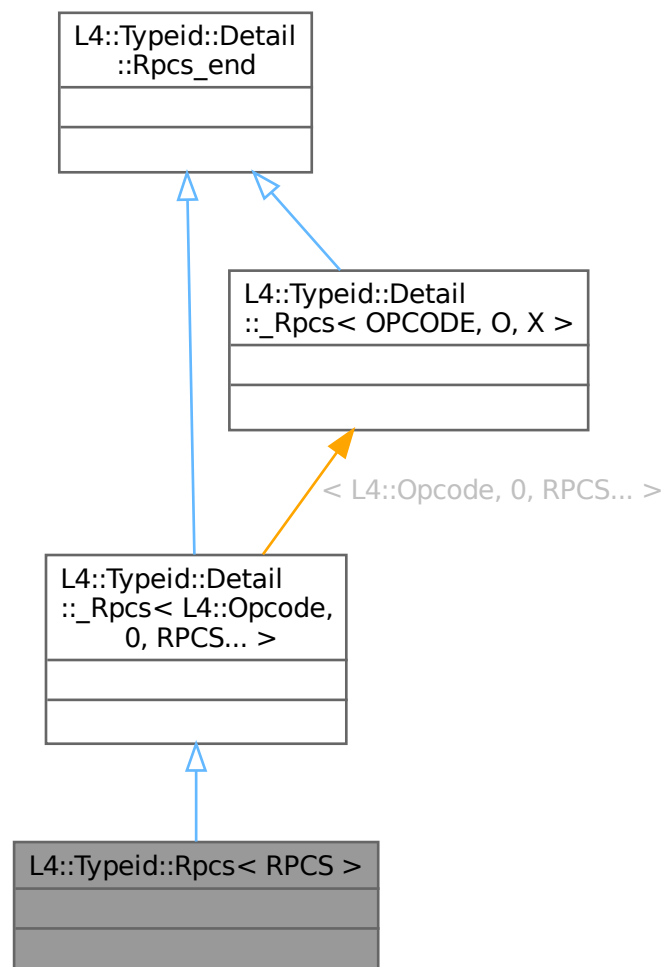
Standard list of RPCs of an interface.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpc< RPCS >:



Collaboration diagram for L4::Typeid::Rpc< RPCS >:



### 15.218.1 Detailed Description

```
template<typename ... RPCS>
struct L4::Typeid::Rpc< RPCS >
```

Standard list of RPCs of an interface.

#### Template Parameters

<i>RPCS</i>	list of RPC types as defined by L4_RPC etc.
-------------	---

This is the default list for RPC functions of an interface, it uses [L4::Opcode](#) as opcode type and uses opcodes starting from 0.

Definition at line 439 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

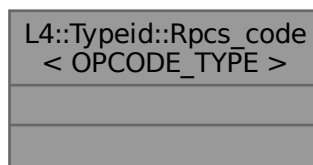
- [l4/sys/\\_\\_typeinfo.h](#)

## 15.219 L4::Typeid::Rpc\_code< OPCODE\_TYPE > Struct Template Reference

List of RPCs of an interface using a special opcode type.

```
#include <l4/sys/capability>
```

Collaboration diagram for L4::Typeid::Rpc\_code< OPCODE\_TYPE >:



### Data Structures

- struct [F](#)

### 15.219.1 Detailed Description

```
template<typename OPCODE_TYPE>
struct L4::Typeid::Rpc_code< OPCODE_TYPE >
```

List of RPCs of an interface using a special opcode type.

#### Template Parameters

<i>OPCODE_TYPE</i>	The data type of the opcode.
--------------------	------------------------------

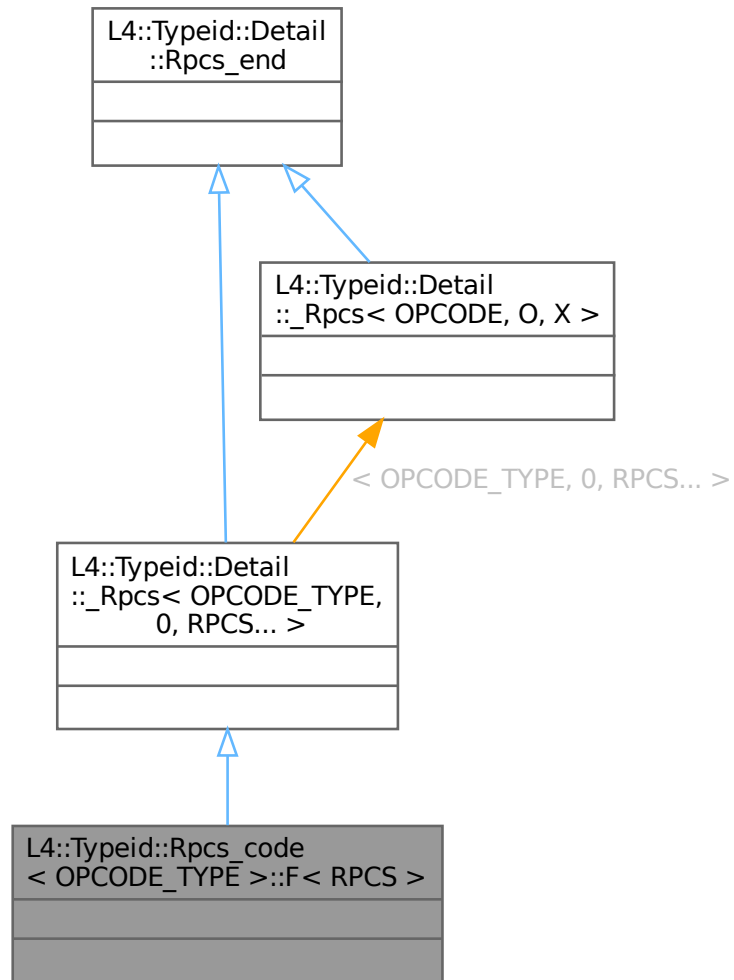
List for RPC functions of an interface, using OPCODE\_TYPE as data type for the opcode, opcodes starting from 0.

Definition at line 450 of file [\\_\\_typeinfo.h](#).

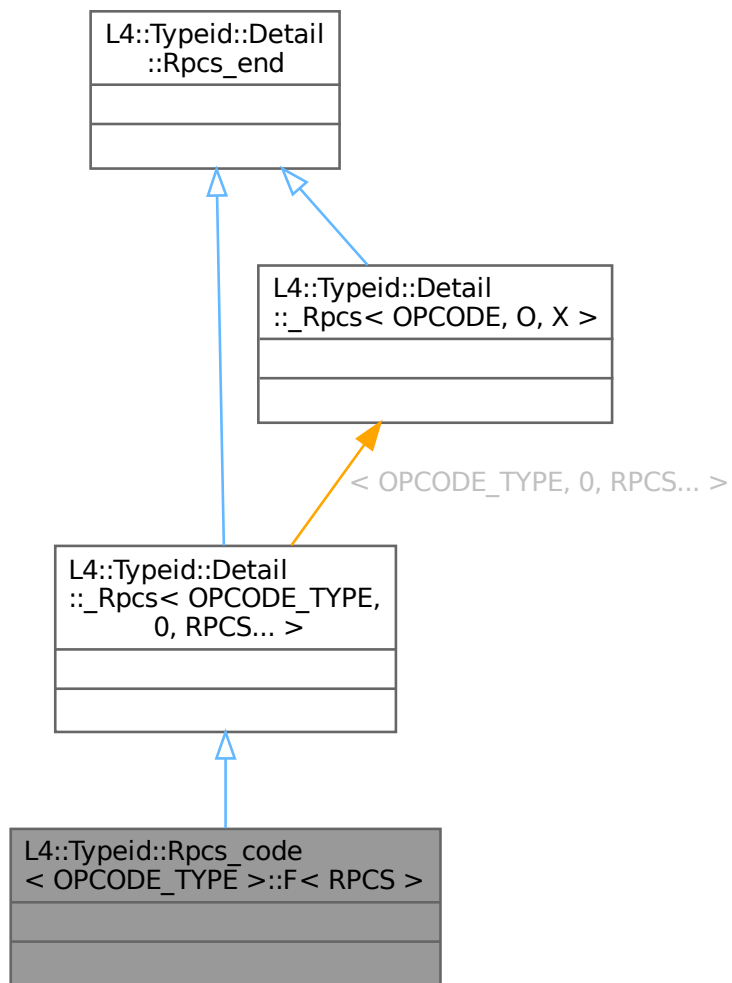
The documentation for this struct was generated from the following file:

- [l4/sys/\\_\\_typeinfo.h](#)

## 15.220 L4::Typeid::Rpc\_code< OPCODE\_TYPE >::F< RPCS > Struct Template Reference



Collaboration diagram for L4::Typeid::Rpc\_code< OPCODE\_TYPE >::F< RPCS >:



### 15.220.1 Detailed Description

```

template<typename OPCODE_TYPE>
template<typename ... RPCS>
struct L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >

```

#### Template Parameters

<i>RPCS</i>	list of RPC types as defined by L4_RPC etc.
-------------	---

Definition at line 456 of file `__typeinfo.h`.

The documentation for this struct was generated from the following file:

- `l4/sys/__typeinfo.h`

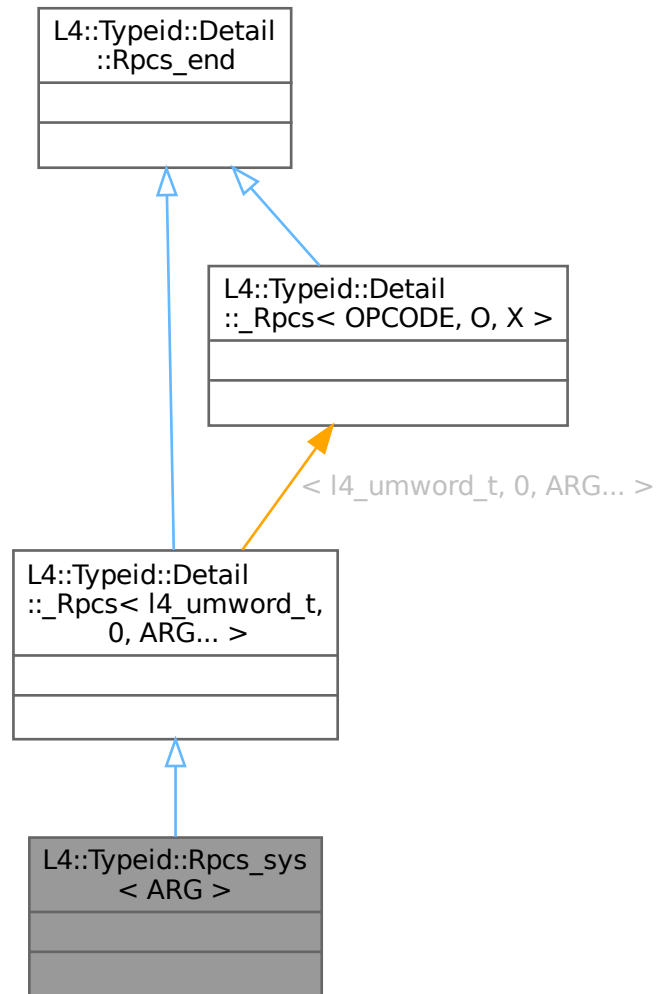


## 15.221 L4::Typeid::Rpcsys< ARG > Struct Template Reference

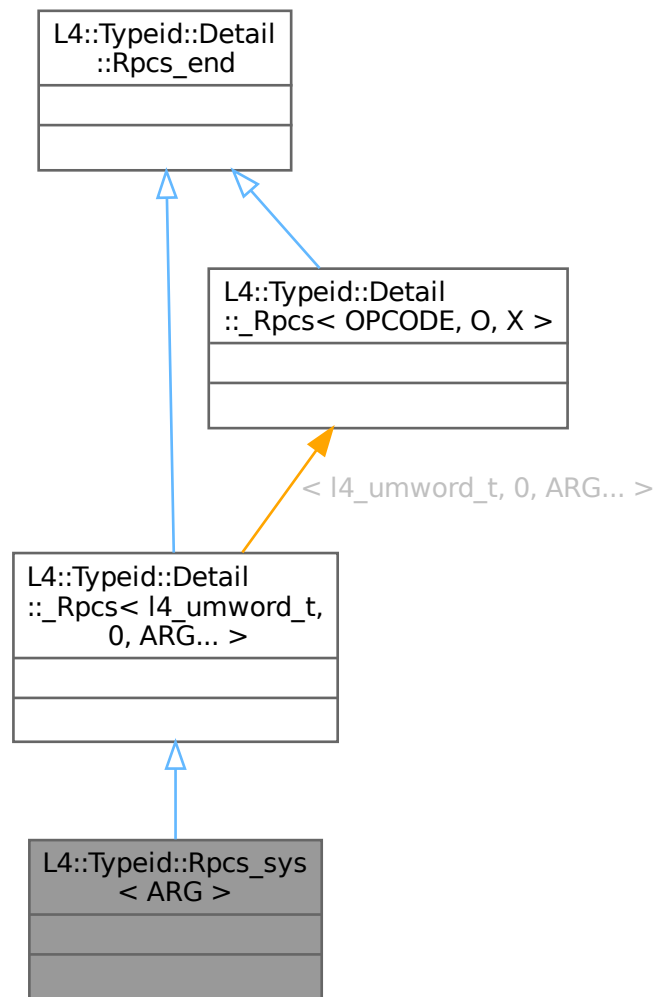
List of RPCs typically used for kernel interfaces.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpcsys< ARG >:



Collaboration diagram for L4::Typeid::Rpcsys< ARG >:



### 15.221.1 Detailed Description

```
template<typename ... ARG>
struct L4::Typeid::Rpcsys< ARG >
```

List of RPCs typically used for kernel interfaces.

Template Parameters

<i><b>RPCS</b></i>	list of RPC types as defined by L4_RPC etc.
--------------------	---

This list of RPC functions uses `I4_umword_t` as type for the opcode as most kernel protocol do.

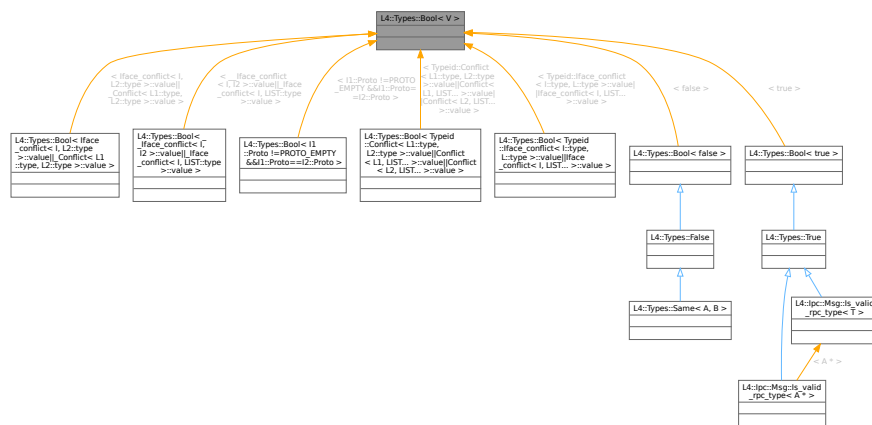
The documentation for this struct was generated from the following file:

- `l4/sys/__typeinfo.h`

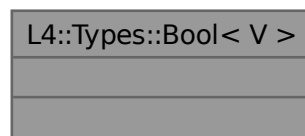
Boolean meta type.

```
#include <types>
```

Inheritance diagram for L4::Types::Bool< V >:



Collaboration diagram for L4::Types::Bool< V >:



## Public Types

- `typedef Bool < V > type`  
*The meta type itself.*

### 15.222.1 Detailed Description

**template<bool V>**

```
struct L4::Types::Bool< V >
```

Boolean meta type.

## Template Parameters

V	The boolean value
---	-------------------

Definition at line 299 of file [types](#).

The documentation for this struct was generated from the following file:

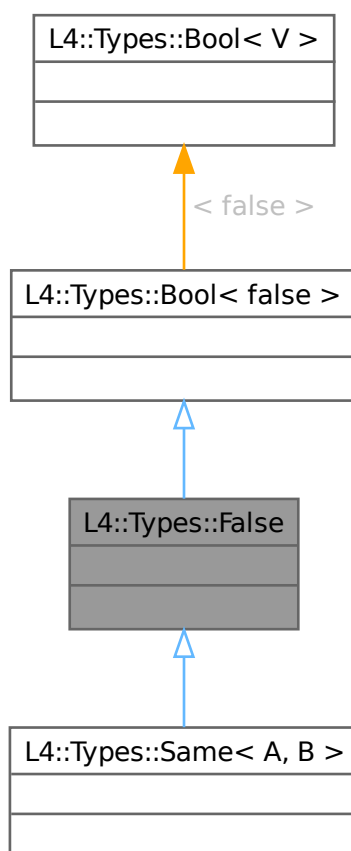
- [l4/sys/cxx/types](#)

## 15.223 L4::Types::False Struct Reference

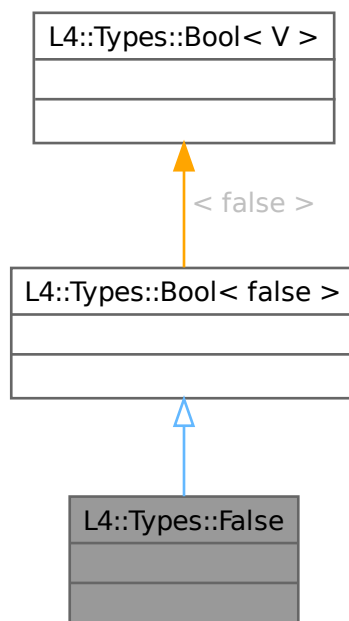
[False](#) meta value.

```
#include <types>
```

Inheritance diagram for L4::Types::False:



Collaboration diagram for L4::Types::False:



#### Additional Inherited Members

#### Public Types inherited from **L4::Types::Bool< false >**

- typedef **Bool< V >** **type**  
*The meta type itself.*

#### 15.223.1 Detailed Description

**False** meta value.

Definition at line 307 of file [types](#).

The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

## 15.224 L4::Types::Flags< BITS\_ENUM, UNDERLYING > Class Template Reference

Template for defining typical [Flags](#) bitmaps.

```
#include <types>
```

Collaboration diagram for L4::Types::Flags< BITS\_ENUM, UNDERLYING >:

L4::Types::Flags< BITS\_ENUM, UNDERLYING >

```
+ Flags()
+ Flags()
+ Flags()
+ operator bool()
+ operator!()
+ operator|=()
+ operator|=()
+ operator&=()
+ operator&=()
+ operator~()
+ clear()
+ as_value()
+ from_raw()
```

### Public Types

- enum [None\\_type](#) { [None](#) }  
*The none type to get an empty bitmap.*
- typedef UNDERLYING **value\_type**  
*type of the underlying value*
- typedef BITS\_ENUM **bits\_enum\_type**  
*enum type defining a name for each bit*
- typedef [Flags](#)< BITS\_ENUM, UNDERLYING > **type**  
*the Flags<> type itself*

## Public Member Functions

- [Flags](#) ([None\\_type](#))  
*Make an empty bitmap.*
- **Flags** ()  
*Make default [Flags](#).*
- [Flags](#) ([BITS\\_ENUM](#) e)  
*Make flags from bit name.*
- **operator bool** () const  
*Support for `if (flags)` syntax (test for non-empty flags).*
- **bool operator!** () const  
*Support for `if (!flags)` syntax (test for empty flags).*
- [type](#) & **operator|=** ([type](#) rhs)  
*Support |= of two compatible [Flags](#) types.*
- [type](#) & **operator|=** ([bits\\_enum\\_type](#) rhs)  
*Support |= of [Flags](#) type and bit name.*
- [type](#) & **operator&=** ([type](#) rhs)  
*Support &= of two compatible [Flags](#) types.*
- [type](#) & **operator&=** ([bits\\_enum\\_type](#) rhs)  
*Support &= of [Flags](#) type and bit name.*
- [type](#) **operator~** () const  
*Support ~ for [Flags](#) types.*
- [type](#) & **clear** ([bits\\_enum\\_type](#) flag)  
*Clear the given flag.*
- [value\\_type](#) **as\_value** () const  
*Get the underlying value.*

## Static Public Member Functions

- static [type from\\_raw](#) ([value\\_type](#) v)  
*Make flags from a raw value of [value\\_type](#).*

## Friends

- [type operator|](#) ([type](#) lhs, [type](#) rhs)  
*Support | of two compatible [Flags](#) types.*
- [type operator|](#) ([type](#) lhs, [bits\\_enum\\_type](#) rhs)  
*Support | of [Flags](#) type and bit name.*
- [type operator&](#) ([type](#) lhs, [type](#) rhs)  
*Support & of two compatible [Flags](#) types.*
- [type operator&](#) ([type](#) lhs, [bits\\_enum\\_type](#) rhs)  
*Support & of [Flags](#) type and bit name.*

## 15.224.1 Detailed Description

```
template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
class L4::Types::Flags< BITS_ENUM, UNDERLYING >
```

Template for defining typical [Flags](#) bitmaps.

## Template Parameters

<i>BITS_ENUM</i>	enum type that defines a name for each bit in the bitmap. The values of the enum members must be the number of the bit ( <i>not</i> a mask).
<i>UNDERLYING</i>	The underlying data type used to represent the bitmap.

The resulting data type provides a type-safe version that allows bitwise `and` and `or` operations with the `BITS_ENUM` members. As well as, test for 0 or !0.

## Example:

```
enum Test_flag
{
    Do_weak_tests,
    Do_strong_tests
};

typedef L4::Types::Flags<Test_flag> Test_flags;

Test_flags x = Do_weak_tests;

if (x & Do_strong_tests) { ... }
x |= Do_strong_tests;
if (x & Do_strong_tests) { ... }
```

Definition at line 63 of file [types](#).

## 15.224.2 Member Enumeration Documentation

### 15.224.2.1 None\_type

```
template<typename BITS_ENUM , typename UNDERLYING = unsigned long>
enum L4::Types::Flags::None_type
```

The none type to get an empty bitmap.

## Enumerator

None	Use this to get an empty bitmap.
------	----------------------------------

Definition at line 79 of file [types](#).

## 15.224.3 Constructor & Destructor Documentation

### 15.224.3.1 Flags() [1/2]

```
template<typename BITS_ENUM , typename UNDERLYING = unsigned long>
L4::Types::Flags< BITS_ENUM, UNDERLYING >::Flags (
    None_type ) [inline]
```

Make an empty bitmap.

Usually used for implicit conversion from `Flags::None`.

```
Flags x = Flags::None;
```

Definition at line 89 of file [types](#).



### 15.224.3.2 Flags() [2/2]

```
template<typename BITS_ENUM , typename UNDERLYING = unsigned long>
L4::Types::Flags< BITS_ENUM, UNDERLYING >::Flags (
    BITS_ENUM e ) [inline]
```

Make flags from bit name.

Usually used for implicit conversion for a bit name.

```
Test_flags f = Do_strong_tests;
```

Definition at line 102 of file [types](#).

## 15.224.4 Member Function Documentation

### 15.224.4.1 clear()

```
template<typename BITS_ENUM , typename UNDERLYING = unsigned long>
type & L4::Types::Flags< BITS_ENUM, UNDERLYING >::clear (
    bits_enum_type flag ) [inline]
```

Clear the given flag.

#### Parameters

<i>flag</i>	The flag that shall be cleared.
-------------	---------------------------------

`flags.clear(The_flag)` is a shortcut for `flags &= ~Flags(The_flag)`.

Definition at line 153 of file [types](#).

References [L4::Types::Flags< BITS\\_ENUM, UNDERLYING >::operator&=\(\)](#).

Here is the call graph for this function:



### 15.224.4.2 from\_raw()

```
template<typename BITS_ENUM , typename UNDERLYING = unsigned long>
static type L4::Types::Flags< BITS_ENUM, UNDERLYING >::from_raw (
    value_type v ) [inline], [static]
```

Make flags from a raw value of *value\_type*.

This function may be used for example in C wrapper code.

Definition at line 109 of file [types](#).

The documentation for this class was generated from the following file:

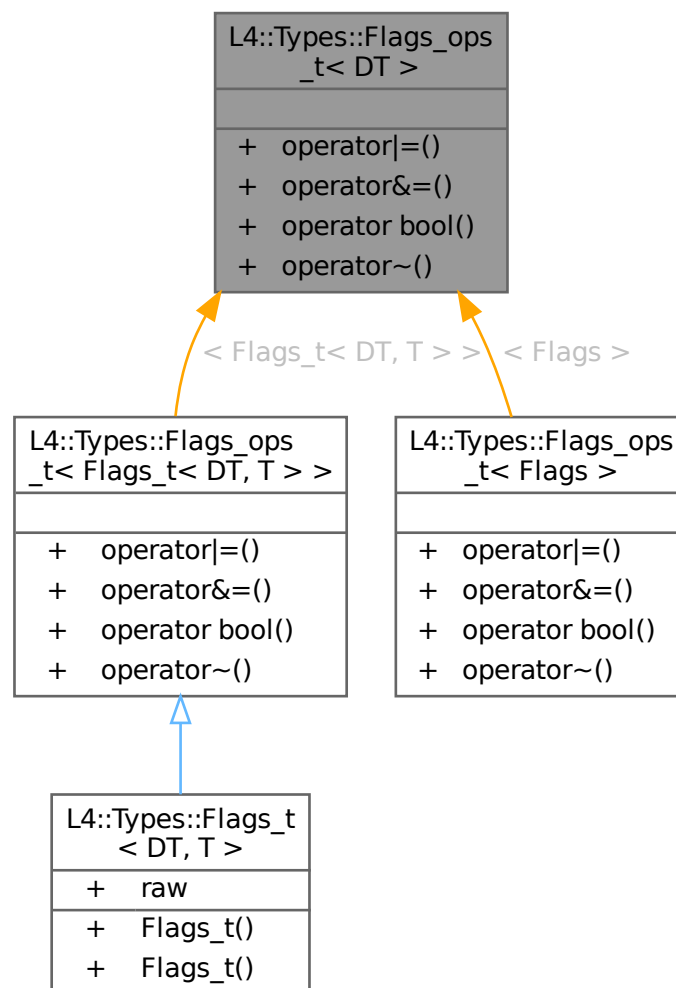
- [l4/sys/cxx/types](#)

## 15.225 L4::Types::Flags\_ops\_t< DT > Struct Template Reference

Mixin class to define a set of friend bitwise operators on DT.

```
#include <types>
```

Inheritance diagram for L4::Types::Flags\_ops\_t< DT >:



Collaboration diagram for L4::Types::Flags\_ops\_t< DT >:

L4::Types::Flags_ops_t< DT >
<ul style="list-style-type: none"> <li>+ operator =()</li> <li>+ operator&amp;=()</li> <li>+ operator bool()</li> <li>+ operator~()</li> </ul>

### Public Member Functions

- DT **operator|=** (DT r)  
*bitwise or assignment for DT*
- DT **operator&=** (DT r)  
*bitwise and assignment for DT*
- constexpr **operator bool** () const  
*explicit conversion to bool for tests*
- constexpr DT **operator~** () const  
*bitwise negation for DT*

### Friends

- constexpr DT **operator|** (DT l, DT r)  
*bitwise or for DT*
- constexpr DT **operator&** (DT l, DT r)  
*bitwise and for DT*
- constexpr bool **operator==** (DT l, DT r)  
*equality for DT*
- constexpr bool **operator!=** (DT l, DT r)  
*inequality for DT*

## 15.225.1 Detailed Description

```
template<typename DT>
struct L4::Types::Flags_ops_t< DT >
```

Mixin class to define a set of friend bitwise operators on DT.

## Template Parameters

<i>DT</i>	The type usually inheriting from <a href="#">Flags_ops_t</a> with a member <i>raw</i> of enum or integral type.
-----------	---

Definition at line 231 of file [types](#).

The documentation for this struct was generated from the following file:

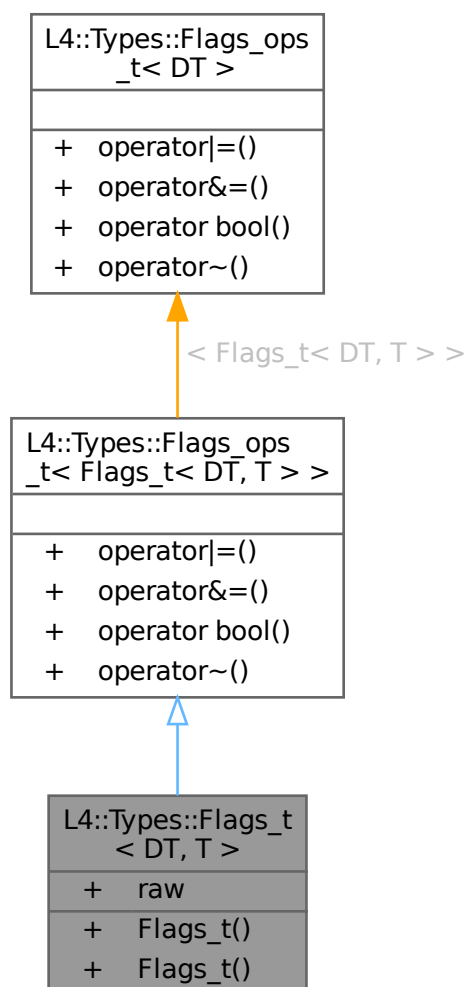
- [l4/sys/cxx/types](#)

## 15.226 L4::Types::Flags\_t< DT, T > Struct Template Reference

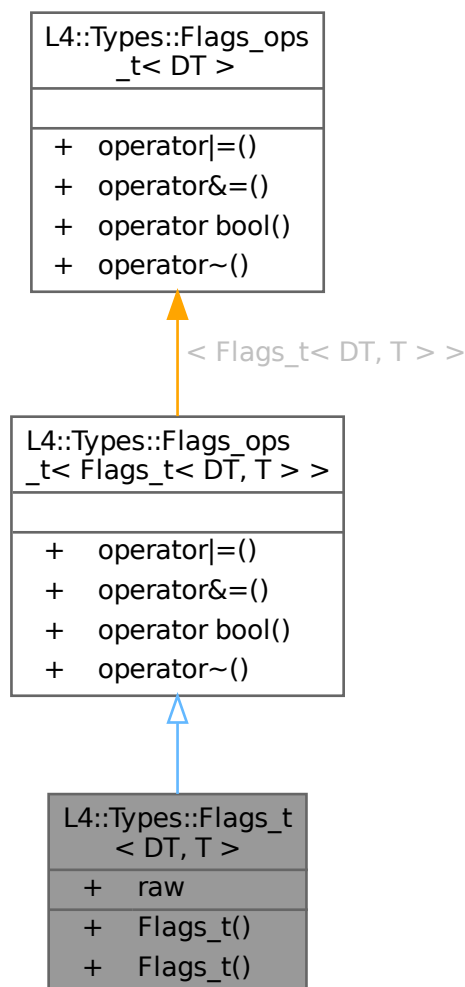
Template type to define a flags type with bitwise operations.

```
#include <types>
```

Inheritance diagram for L4::Types::Flags\_t< DT, T >:



Collaboration diagram for L4::Types::Flags\_t< DT, T >:



## Public Member Functions

- **Flags\_t**( )=default  
*Default (uninitializing) constructor.*
- constexpr **Flags\_t**( T f)  
*Explicit initialization from the underlying type.*

## Public Member Functions inherited from [L4::Types::Flags\\_ops\\_t< Flags\\_t< DT, T > >](#)

- [Flags\\_t](#)< DT, T > **operator|=**( [Flags\\_t](#)< DT, T > r)  
*bitwise or assignment for DT*
- [Flags\\_t](#)< DT, T > **operator&=**( [Flags\\_t](#)< DT, T > r)  
*bitwise and assignment for DT*

- constexpr **operator bool** () const  
*explicit conversion to bool for tests*
- constexpr **Flags\_t**< DT, T > **operator~** () const  
*bitwise negation for DT*

## Data Fields

- T raw  
*Raw integral value.*

### 15.226.1 Detailed Description

**template**<typename DT, typename T>  
**struct** L4::Types::Flags\_t< DT, T >

Template type to define a flags type with bitwise operations.

#### Template Parameters

<i>DT</i>	determinator type to make the resulting type unique (unused).
<i>T</i>	underlying type used to store the bits, usually an integral type.

Definition at line 283 of file [types](#).

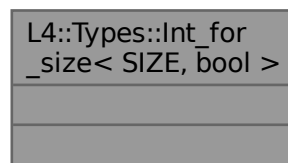
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

### 15.227 L4::Types::Int\_for\_size< SIZE, bool > Struct Template Reference

Metafunction to get an unsigned integral type for the given size.

Collaboration diagram for L4::Types::Int\_for\_size< SIZE, bool >:



### 15.227.1 Detailed Description

```
template<unsigned SIZE, bool = true>
struct L4::Types::Int_for_size< SIZE, bool >
```

Metafunction to get an unsigned integral type for the given size.

Template Parameters

<i>SIZE</i>	The size of the integer in bytes.
-------------	-----------------------------------

Definition at line 164 of file [types](#).

The documentation for this struct was generated from the following file:

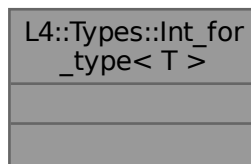
- [l4/sys/cxx/types](#)

## 15.228 L4::Types::Int\_for\_type< T > Struct Template Reference

Metafunction to get an integral type of the same size as T.

```
#include <types>
```

Collaboration diagram for L4::Types::Int\_for\_type< T >:



### Public Types

- typedef [Int\\_for\\_size](#)< sizeof(T)>::type **type**  
*The resulting unsigned integer type with the size like T.*

### 15.228.1 Detailed Description

```
template<typename T>
struct L4::Types::Int_for_type< T >
```

Metafunction to get an integral type of the same size as T.

## Template Parameters

<i>T</i>	The type for which an unsigned integral type with the same size is needed.
----------	--

Definition at line 191 of file [types](#).

The documentation for this struct was generated from the following file:

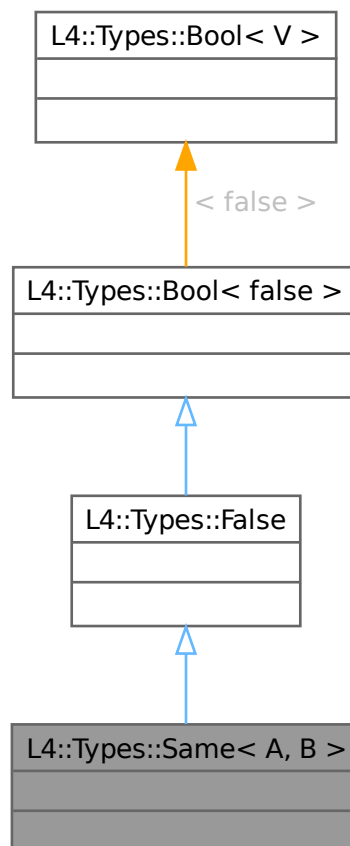
- [l4/sys/cxx/types](#)

## 15.229 L4::Types::Same< A, B > Struct Template Reference

Compare two data types for equality.

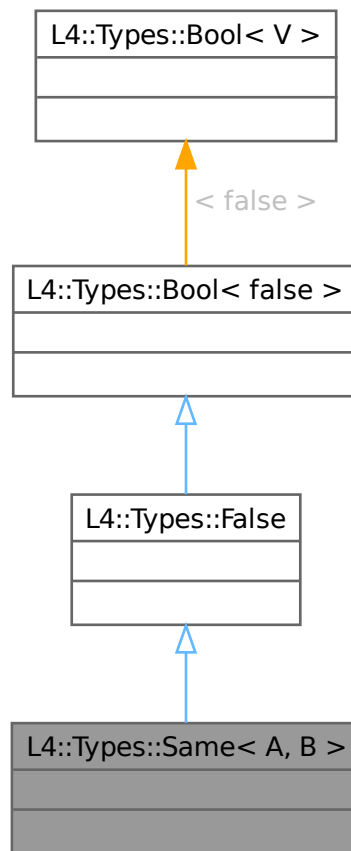
```
#include <types>
```

Inheritance diagram for L4::Types::Same< A, B >:





Collaboration diagram for L4::Types::Same< A, B >:



#### Additional Inherited Members

#### Public Types inherited from [L4::Types::Bool< false >](#)

- typedef [Bool< V >](#) **type**  
The meta type itself.

#### 15.229.1 Detailed Description

```
template<typename A, typename B>
struct L4::Types::Same< A, B >
```

Compare two data types for equality.

##### Template Parameters

<i>A</i>	The first data type
<i>B</i>	The second data type

The result is the boolean `True` if A and B are the same types.

Definition at line 323 of file `types`.

The documentation for this struct was generated from the following file:

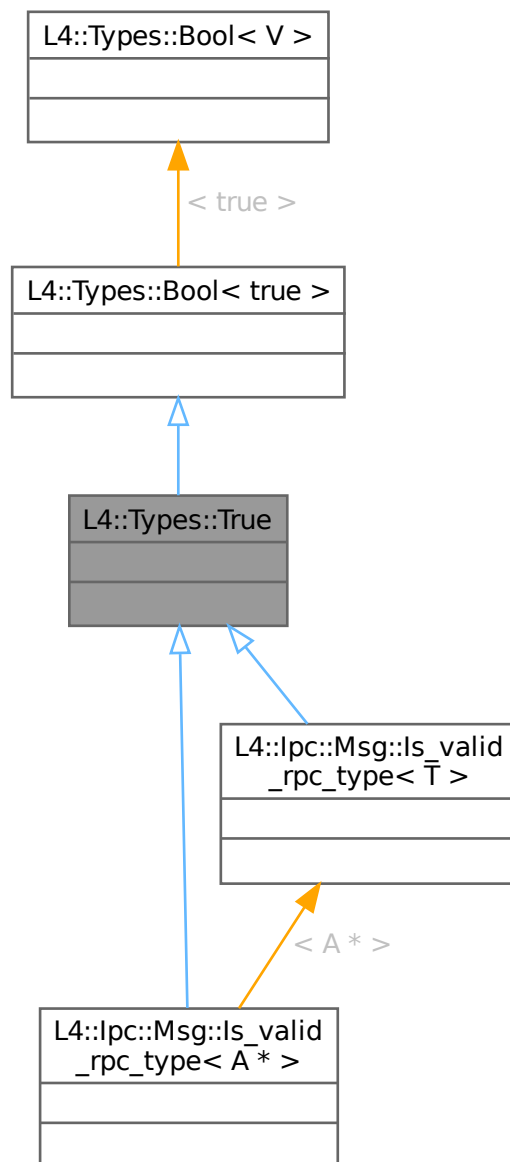
- `l4/sys/cxx/types`

## 15.230 L4::Types::True Struct Reference

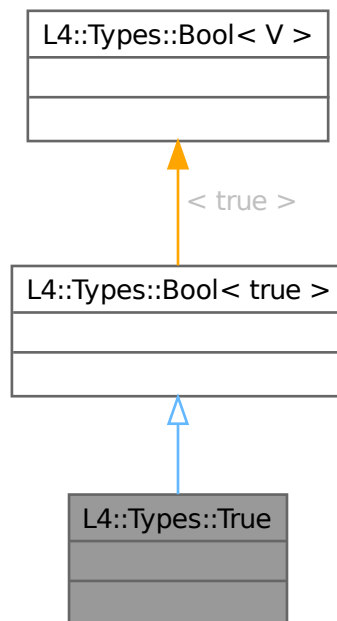
`True` meta value.

```
#include <types>
```

Inheritance diagram for `L4::Types::True`:



Collaboration diagram for L4::Types::True:



#### Additional Inherited Members

#### Public Types inherited from `L4::Types::Bool< true >`

- typedef `Bool< V >` **type**  
*The meta type itself.*

#### 15.230.1 Detailed Description

`True` meta value.

Definition at line 311 of file `types`.

The documentation for this struct was generated from the following file:

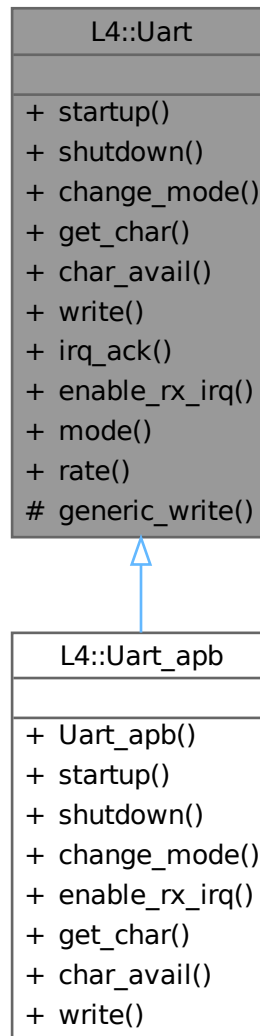
- `l4/sys/cxx/types`

## 15.231 L4::Uart Class Reference

[Uart](#) driver abstraction.

```
#include <uart_base.h>
```

Inheritance diagram for L4::Uart:



Collaboration diagram for L4::Uart:

L4::Uart
<ul style="list-style-type: none"> <li>+ startup()</li> <li>+ shutdown()</li> <li>+ change_mode()</li> <li>+ get_char()</li> <li>+ char_avail()</li> <li>+ write()</li> <li>+ irq_ack()</li> <li>+ enable_rx_irq()</li> <li>+ mode()</li> <li>+ rate()</li> <li># generic_write()</li> </ul>

### Public Member Functions

- virtual bool [startup](#) (Io\_register\_block const \*regs)=0  
*Start the UART driver.*
- virtual void [shutdown](#) ()=0  
*Terminate the UART driver.*
- virtual bool [change\\_mode](#) (Transfer\_mode m, Baud\_rate r)=0  
*Set certain parameters of the UART.*
- virtual int [get\\_char](#) (bool blocking=true) const =0  
*Read a character from the UART.*
- virtual int [char\\_avail](#) () const =0  
*Check if there is at least one character available for reading from the UART.*
- virtual int [write](#) (char const \*s, unsigned long count, bool blocking=true) const =0  
*Transmit a number of characters.*
- virtual void [irq\\_ack](#) ()  
*Acknowledge a received interrupt.*
- virtual bool [enable\\_rx\\_irq](#) (bool=true)  
*Enable the receive IRQ.*
- Transfer\_mode [mode](#) () const  
*Return the transfer mode.*
- Baud\_rate [rate](#) () const  
*Return the baud rate.*

### Protected Member Functions

- template<typename Uart\_driver >  
int [generic\\_write](#) (char const \*s, unsigned long count, bool blocking=true) const  
*Internal function transmitting each character one-after-another and finally waiting that the transmission did actually finish.*

### 15.231.1 Detailed Description

[Uart](#) driver abstraction.

Definition at line 25 of file [uart\\_base.h](#).

### 15.231.2 Member Function Documentation

#### 15.231.2.1 `change_mode()`

```
virtual bool L4::Uart::change_mode (
    Transfer_mode m,
    Baud_rate r ) [pure virtual]
```

Set certain parameters of the UART.

##### Parameters

<i>m</i>	UART mode. Depends on the hardware.
<i>r</i>	Baud rate.

##### Return values

<i>true</i>	Mode setting succeeded (or was not performed at all).
<i>false</i>	Mode setting failed for some reason.

##### Note

Some drivers don't perform any mode setting at all and just return true.

Implemented in [L4::Uart\\_apb](#).

#### 15.231.2.2 `char_avail()`

```
virtual int L4::Uart::char_avail ( ) const [pure virtual]
```

Check if there is at least one character available for reading from the UART.

##### Returns

0 if there is no character available for reading, !=0 otherwise.

Implemented in [L4::Uart\\_apb](#).

#### 15.231.2.3 `enable_rx_irq()`

```
virtual bool L4::Uart::enable_rx_irq (
    bool = true ) [inline], [virtual]
```

Enable the receive IRQ.

## Return values

<i>true</i>	The RX IRQ was successfully enabled / disabled.
<i>false</i>	The RX IRQ couldn't be enabled / disabled. The driver does not support this operation.

Reimplemented in [L4::Uart\\_apb](#).

Definition at line 116 of file [uart\\_base.h](#).

#### 15.231.2.4 generic\_write()

```
template<typename Uart_driver >
int L4::Uart::generic_write (
    char const * s,
    unsigned long count,
    bool blocking = true ) const [inline], [protected]
```

Internal function transmitting each character one-after-another and finally waiting that the transmission did actually finish.

## Parameters

<i>s</i>	Buffer containing the characters.
<i>count</i>	The number of characters to transmit.
<i>blocking</i>	If true, wait until there is space in the transmit buffer and also wait until every character was successful transmitted. Otherwise do not wait.

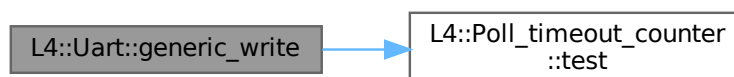
## Returns

The number of successful written characters.

Definition at line 145 of file [uart\\_base.h](#).

References [L4::Poll\\_timeout\\_counter::test\(\)](#).

Here is the call graph for this function:



#### 15.231.2.5 get\_char()

```
virtual int L4::Uart::get_char (
    bool blocking = true ) const [pure virtual]
```

Read a character from the UART.

**Parameters**

<i>blocking</i>	If true, wait until a character is available for reading. Otherwise do not wait and just return -1 if no character is available.
-----------------	--

**Returns**

The actual character read from the UART.

Implemented in [L4::Uart\\_apb](#).

**15.231.2.6 mode()**

```
Transfer_mode L4::Uart::mode ( ) const [inline]
```

Return the transfer mode.

**Returns**

The transfer mode.

Definition at line [123](#) of file [uart\\_base.h](#).

**15.231.2.7 rate()**

```
Baud_rate L4::Uart::rate ( ) const [inline]
```

Return the baud rate.

**Returns**

The baud rate.

Definition at line [130](#) of file [uart\\_base.h](#).

**15.231.2.8 shutdown()**

```
virtual void L4::Uart::shutdown ( ) [pure virtual]
```

Terminate the UART driver.

This includes disabling of interrupts.

Implemented in [L4::Uart\\_apb](#).

**15.231.2.9 startup()**

```
virtual bool L4::Uart::startup (
    Io_register_block const * regs ) [pure virtual]
```

Start the UART driver.



## Parameters

<i>regs</i>	IO register block of the UART.
-------------	--------------------------------

## Return values

<i>true</i>	Startup succeeded.
<i>false</i>	Startup failed.

Implemented in [L4::Uart\\_apb](#).

**15.231.2.10 write()**

```
virtual int L4::Uart::write (
    char const * s,
    unsigned long count,
    bool blocking = true ) const [pure virtual]
```

Transmit a number of characters.

## Parameters

<i>s</i>	Buffer containing the characters.
<i>count</i>	Number of characters to transmit.
<i>blocking</i>	If true, wait until there is space in the transmit buffer and also wait until every character was successful transmitted. Otherwise do not wait.

## Returns

The number of successfully written characters.

Implemented in [L4::Uart\\_apb](#).

The documentation for this class was generated from the following file:

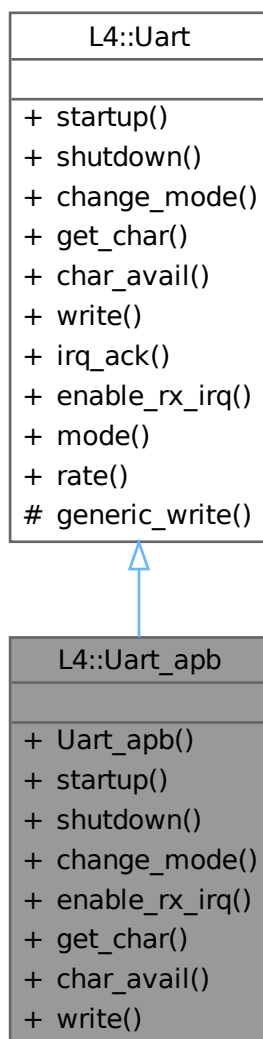
- pkg/drivers-frst/uart/include/uart\_base.h

**15.232 L4::Uart\_apb Class Reference**

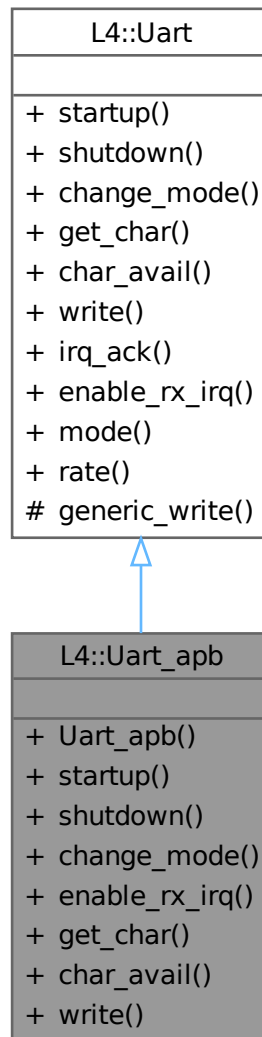
Driver for the Advanced Peripheral Bus (APB) UART from the Cortex-M System Design Kit (apb).

```
#include <uart_apb.h>
```

Inheritance diagram for L4::Uart\_apb:



Collaboration diagram for L4::Uart\_apb:



## Public Member Functions

- **Uart\_apb** (unsigned freq)  
*freq == 0 means unknown and don't change baud rate*
- bool **startup** (lo\_register\_block const \*) override  
*Start the UART driver.*
- void **shutdown** () override  
*Terminate the UART driver.*
- bool **change\_mode** (Transfer\_mode m, Baud\_rate r) override  
*Set certain parameters of the UART.*
- bool **enable\_rx\_irq** (bool enable) override  
*Enable the receive IRQ.*

- int [get\\_char](#) (bool blocking=true) const override  
*Read a character from the UART.*
- int [char\\_aval](#) () const override  
*Check if there is at least one character available for reading from the UART.*
- int [write](#) (char const \*s, unsigned long count, bool blocking=true) const override  
*Transmit a number of characters.*

## Public Member Functions inherited from [L4::Uart](#)

- virtual void [irq\\_ack](#) ()  
*Acknowledge a received interrupt.*
- Transfer\_mode [mode](#) () const  
*Return the transfer mode.*
- Baud\_rate [rate](#) () const  
*Return the baud rate.*

## Additional Inherited Members

## Protected Member Functions inherited from [L4::Uart](#)

- template<typename Uart\_driver >  
int [generic\\_write](#) (char const \*s, unsigned long count, bool blocking=true) const  
*Internal function transmitting each character one-after-another and finally waiting that the transmission did actually finish.*

### 15.232.1 Detailed Description

Driver for the Advanced Peripheral Bus (APB) UART from the Cortex-M System Design Kit (apb).

Definition at line 18 of file [uart\\_apb.h](#).

### 15.232.2 Member Function Documentation

#### 15.232.2.1 [change\\_mode\(\)](#)

```
bool L4::Uart_apb::change_mode (
    Transfer_mode m,
    Baud_rate r ) [override], [virtual]
```

Set certain parameters of the UART.

#### Parameters

<i>m</i>	UART mode. Depends on the hardware.
<i>r</i>	Baud rate.

## Return values

<i>true</i>	Mode setting succeeded (or was not performed at all).
<i>false</i>	Mode setting failed for some reason.

## Note

Some drivers don't perform any mode setting at all and just return true.

Implements [L4::Uart](#).

**15.232.2.2 char\_avail()**

```
int L4::Uart_apb::char_avail ( ) const [override], [virtual]
```

Check if there is at least one character available for reading from the UART.

## Returns

0 if there is no character available for reading, !=0 otherwise.

Implements [L4::Uart](#).

**15.232.2.3 enable\_rx\_irq()**

```
bool L4::Uart_apb::enable_rx_irq (
    bool ) [override], [virtual]
```

Enable the receive IRQ.

## Return values

<i>true</i>	The RX IRQ was successfully enabled / disabled.
<i>false</i>	The RX IRQ couldn't be enabled / disabled. The driver does not support this operation.

Reimplemented from [L4::Uart](#).

**15.232.2.4 get\_char()**

```
int L4::Uart_apb::get_char (
    bool blocking = true ) const [override], [virtual]
```

Read a character from the UART.

## Parameters

<i>blocking</i>	If true, wait until a character is available for reading. Otherwise do not wait and just return -1 if no character is available.
-----------------	--

**Returns**

The actual character read from the UART.

Implements [L4::Uart](#).

**15.232.2.5 shutdown()**

```
void L4::Uart_apb::shutdown ( ) [override], [virtual]
```

Terminate the UART driver.

This includes disabling of interrupts.

Implements [L4::Uart](#).

**15.232.2.6 startup()**

```
bool L4::Uart_apb::startup (
    Io_register_block const * regs ) [override], [virtual]
```

Start the UART driver.

**Parameters**

<i>regs</i>	IO register block of the UART.
-------------	--------------------------------

**Return values**

<i>true</i>	Startup succeeded.
<i>false</i>	Startup failed.

Implements [L4::Uart](#).

**15.232.2.7 write()**

```
int L4::Uart_apb::write (
    char const * s,
    unsigned long count,
    bool blocking = true ) const [override], [virtual]
```

Transmit a number of characters.

**Parameters**

<i>s</i>	Buffer containing the characters.
<i>count</i>	Number of characters to transmit.
<i>blocking</i>	If true, wait until there is space in the transmit buffer and also wait until every character was successful transmitted. Otherwise do not wait.

**Returns**

The number of successfully written characters.

Implements [L4::Uart](#).

The documentation for this class was generated from the following file:

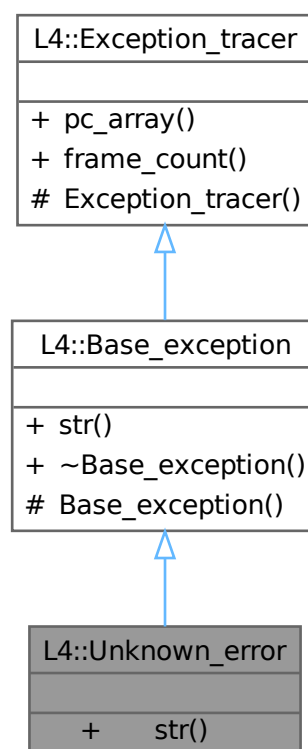
- pkg/drivers-frst/uart/include/uart\_apb.h

## 15.233 L4::Unknown\_error Class Reference

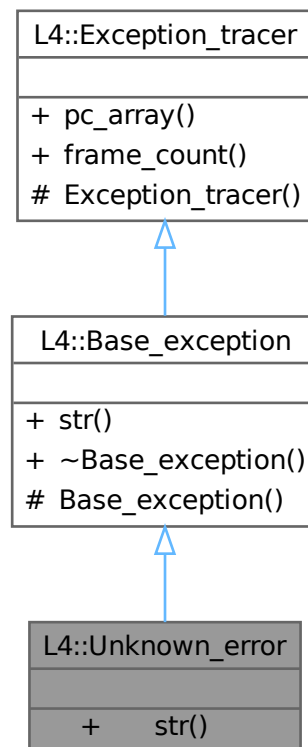
[Exception](#) for an unknown condition.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Unknown\_error:



Collaboration diagram for L4::Unknown\_error:



### Public Member Functions

- `char const * str ()` `const noexcept` override  
*Return a human readable string for the exception.*

### Public Member Functions inherited from [L4::Base\\_exception](#)

- `virtual ~Base_exception ()` `noexcept`  
*Destruction.*

### Public Member Functions inherited from [L4::Exception\\_tracer](#)

- `void const *const * pc_array ()` `const noexcept`  
*Get the array containing the call trace.*
- `int frame_count ()` `const noexcept`  
*Get the number of entries that are valid in the call trace.*



## Additional Inherited Members

### Protected Member Functions inherited from [L4::Base\\_exception](#)

- **Base\_exception** () noexcept  
*Create a base exception.*

### Protected Member Functions inherited from [L4::Exception\\_tracer](#)

- **Exception\_tracer** () noexcept  
*Create a back trace.*

## 15.233.1 Detailed Description

[Exception](#) for an unknown condition.

This error is usually used when a server returns an unknown return state to the client, this may indicate incompatible messages used by the client and the server.

Definition at line 219 of file [exceptions](#).

The documentation for this class was generated from the following file:

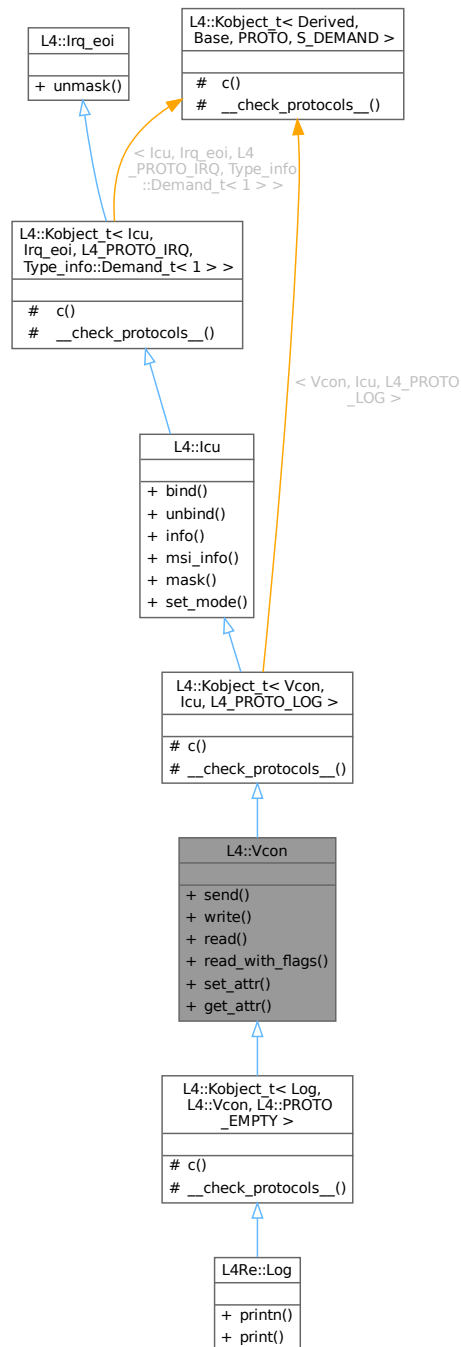
- [l4/cxx/exceptions](#)

## 15.234 L4::Vcon Class Reference

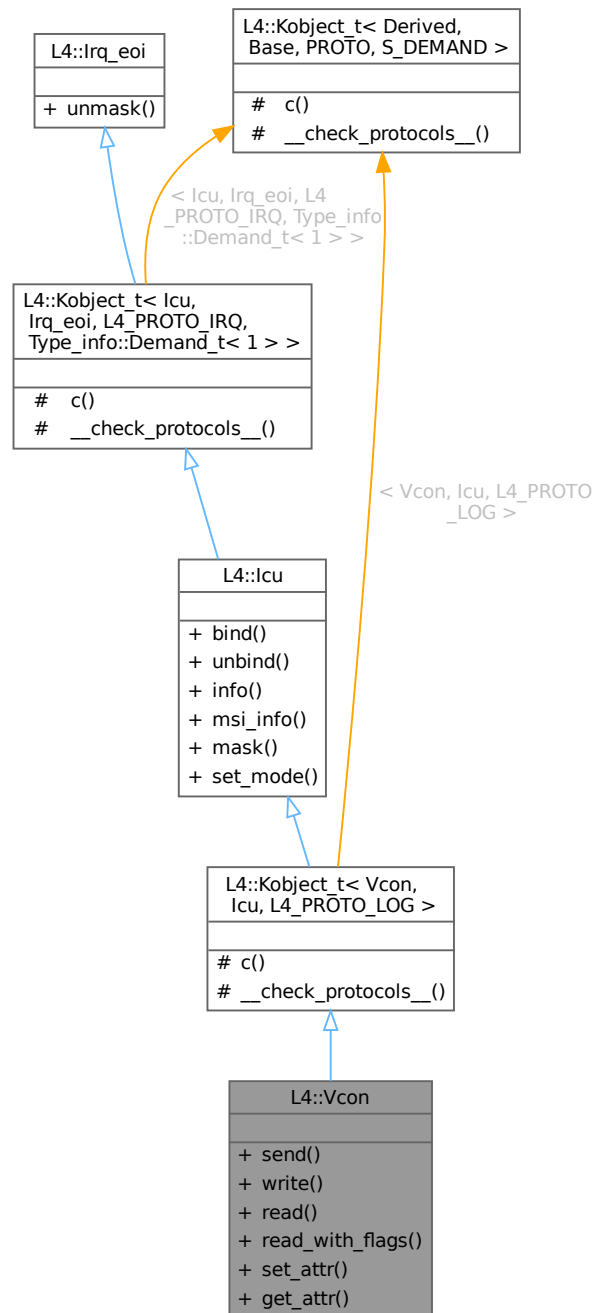
C++ [L4 Vcon](#) interface, see [Virtual Console](#) for the C interface.

```
#include <vcon>
```

Inheritance diagram for L4::Vcon:



Collaboration diagram for L4::Vcon:



## Public Member Functions

- `l4_msgtag_t send` (char const \*buf, unsigned size, `l4_utcb_t *utcb=l4_utcb()`) const noexcept  
Send data to *this* virtual console.
- `long write` (char const \*buf, unsigned size, `l4_utcb_t *utcb=l4_utcb()`) const noexcept  
Write data to *this* virtual console.
- `int read` (char \*buf, unsigned size, `l4_utcb_t *utcb=l4_utcb()`) const noexcept

Read data from *this* virtual console.

- `int read_with_flags (char *buf, unsigned size, l4\_utcb\_t *utcb=l4\_utcb\(\)) const noexcept`

Read data from *this* virtual console which also returns flags.

- `l4\_msgtag\_t set_attr (l4\_vcon\_attr\_t const *attr, l4\_utcb\_t *utcb=l4\_utcb\(\)) const noexcept`

Set the attributes of *this* virtual console.

- `l4\_msgtag\_t get_attr (l4\_vcon\_attr\_t *attr, l4\_utcb\_t *utcb=l4\_utcb\(\)) const noexcept`

Get attributes of *this* virtual console.

## Public Member Functions inherited from [L4::Icu](#)

- `l4\_msgtag\_t bind (unsigned irqnum, L4::Cap< Triggerable > irq, l4\_utcb\_t *utcb=l4\_utcb\(\)) noexcept`

Bind an interrupt line of an interrupt controller to an interrupt object.

- `l4\_msgtag\_t unbind (unsigned irqnum, L4::Cap< Triggerable > irq, l4\_utcb\_t *utcb=l4\_utcb\(\)) noexcept`

Remove binding of an interrupt line from the interrupt controller object.

- `l4\_msgtag\_t info (l4\_icu\_info\_t *info, l4\_utcb\_t *utcb=l4\_utcb\(\)) noexcept`

Get information about the ICU features.

- `l4\_msgtag\_t msi_info (l4\_umword\_t irqnum, l4\_uint64\_t source, l4\_icu\_msi\_info\_t *msi_info)`

Get MSI info about IRQ.

- `l4\_msgtag\_t mask (unsigned irqnum, l4\_umword\_t *label=0, l4\_timeout\_t to=L4\_IPC\_NEVER, l4\_utcb\_t *utcb=l4\_utcb\(\)) noexcept`

Mask an IRQ line.

- `l4\_msgtag\_t set_mode (unsigned irqnum, l4\_umword\_t mode, l4\_utcb\_t *utcb=l4\_utcb\(\)) noexcept`

Set interrupt mode.

## Public Member Functions inherited from [L4::Irq\\_eoi](#)

- `l4\_msgtag\_t unmask (unsigned irqnum, l4\_umword\_t *label=0, l4\_timeout\_t to=L4\_IPC\_NEVER, l4\_utcb\_t *utcb=l4\_utcb\(\)) noexcept`

Unmask the given interrupt line.

## Additional Inherited Members

## Protected Types inherited from [L4::Kobject\\_t](#)< [Vcon](#), [Icu](#), [L4\\_PROTO\\_LOG](#) >

- `typedef Vcon Class`

The target interface type (inheriting from [Kobject\\_t](#))

- `typedef Typeid::Iface< PROTO, Vcon > __Iface`

The interface description for the derived class.

- `typedef Typeid::Merge_list< Typeid::Iface_list< __Iface >, typename Base::__Iface_list > __Iface_list`

The list of all RPC interfaces provided directly or through inheritance.

## Protected Types inherited from

## [L4::Kobject\\_t](#)< [Icu](#), [Irq\\_eoi](#), [L4\\_PROTO\\_IRQ](#), [Type\\_info::Demand\\_t](#)< 1 > >

- `typedef Icu Class`

The target interface type (inheriting from [Kobject\\_t](#))

- `typedef Typeid::Iface< PROTO, Icu > __Iface`

The interface description for the derived class.

- `typedef Typeid::Merge_list< Typeid::Iface_list< __Iface >, typename Base::__Iface_list > __Iface_list`

The list of all RPC interfaces provided directly or through inheritance.

### Protected Member Functions inherited from [L4::Kobject\\_t< Vcon, Icu, L4\\_PROTO\\_LOG >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject\\_t< Icu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< Vcon, Icu, L4\\_PROTO\\_LOG >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< Icu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

## 15.234.1 Detailed Description

C++ [L4 Vcon](#) interface, see [Virtual Console](#) for the C interface.

[L4::Vcon](#) is a virtual console for simple character-based input and output. The interrupt for read events is provided by the virtual key interrupt.

The [Vcon](#) interface inherits from [L4::Icu](#) and [L4::Irq\\_eoi](#) for managing the virtual key interrupt which, in contrast to hardware IRQs, implements a limited functionality:

- Only IRQ line 0 is supported, no MSI vectors.
- The IRQ is edge-triggered and the IRQ mode cannot be changed.
- As the IRQ is edge-triggered, it does not have to be explicitly unmasked.

A server implementing the virtual console protocol has a queue for input events. When the first input event is added to the empty queue, the virtual key interrupt is triggered. Further events are added to the queue without generating further interrupts. The queue is emptied when a client reads all queued input events.

#### Include File

```
#include <l4/sys/vcon>
```

See the [Virtual Console](#) for the C interface.

Definition at line 56 of file [vcon](#).

## 15.234.2 Member Function Documentation

### 15.234.2.1 [get\\_attr\(\)](#)

```
l4\_msgtag\_t L4::Vcon::get_attr (
    l4\_vcon\_attr\_t * attr,
    l4\_utcb\_t * utcb = l4\_utcb\(\) ) const [inline], [noexcept]
```

Get attributes of `this` virtual console.

## Parameters

out	attr	Attribute structure. Contains the attributes after a successful call of this function.
	utcb	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

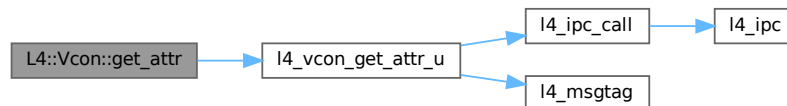
## Returns

Syscall return tag.

Definition at line 162 of file [vcon](#).

References [l4\\_vcon\\_get\\_attr\\_u\(\)](#).

Here is the call graph for this function:



## 15.234.2.2 read()

```

int L4::Vcon::read (
    char * buf,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb() ) const [inline], [noexcept]

```

Read data from this virtual console.

## Parameters

out	buf	Pointer to data buffer.
	size	Size of the data buffer in bytes.
	utcb	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

## Return values

-L4_EPERM	The <a href="#">Vcon</a> instance requires the <a href="#">L4_CAP_FPAGE_W</a> right on the capability used to invoke this operation and this right is not present.
>size	More bytes to read, size bytes are in the buffer buf.
<=size	Number of bytes read.

## Note

Size must not exceed [L4\\_VCON\\_READ\\_SIZE](#).

Definition at line 109 of file [vcon](#).

References [l4\\_vcon\\_read\\_u\(\)](#).

Here is the call graph for this function:



### 15.234.2.3 read\_with\_flags()

```
int L4::Vcon::read_with_flags (
    char * buf,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb() ) const [inline], [noexcept]
```

Read data from this virtual console which also returns flags.

#### Parameters

out	<i>buf</i>	Pointer to data buffer.
	<i>size</i>	Size of the data buffer in bytes.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

#### Return values

<code>-L4_EPERM</code>	The <a href="#">Vcon</a> instance requires the <a href="#">L4_CAP_FPAGE_W</a> right on the capability used to invoke this operation and this right is not present.
<code>&gt;size</code>	More bytes to read, <i>size</i> bytes are in the buffer <i>buf</i> .
<code>&lt;=size</code>	Number of bytes read.

If this function returns a positive value the caller can check the [L4\\_VCON\\_READ\\_STAT\\_BREAK](#) flag bit for a break condition. The bytes read can be obtained by masking the return value with [L4\\_VCON\\_READ\\_SIZE\\_MASK](#).

If a break condition is signaled, it is always the first event in the transmitted content, i.e. all characters supplied by this read call follow the break condition.

#### Note

Size must not exceed [L4\\_VCON\\_READ\\_SIZE](#).

Definition at line 136 of file [vcon](#).

#### 15.234.2.4 send()

```
l4_msgtag_t L4::Vcon::send (
    char const * buf,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb() ) const [inline], [noexcept]
```

Send data to this virtual console.

##### Parameters

<i>buf</i>	Pointer to the data buffer.
<i>size</i>	Size of the data buffer in bytes.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

##### Returns

Syscall return tag

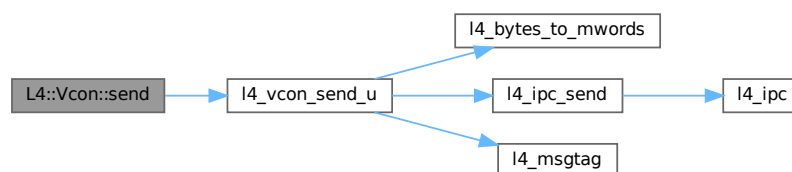
##### Note

Size must not exceed [L4\\_VCON\\_WRITE\\_SIZE](#), a proper value of the `size` parameter is NOT checked. Also, this function is a send only operation, this means there is no return value except for a failed send operation. Use [l4\\_ipc\\_error\(\)](#) to check for send errors, do not use [l4\\_error\(\)](#), as [l4\\_error\(\)](#) will always return an error.

Definition at line 76 of file [vcon](#).

References [l4\\_vcon\\_send\\_u\(\)](#).

Here is the call graph for this function:



#### 15.234.2.5 set\_attr()

```
l4_msgtag_t L4::Vcon::set_attr (
    l4_vcon_attr_t const * attr,
    l4_utcb_t * utcb = l4_utcb() ) const [inline], [noexcept]
```

Set the attributes of this virtual console.



## Parameters

<i>attr</i>	Attribute structure with the attributes for the virtual console.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

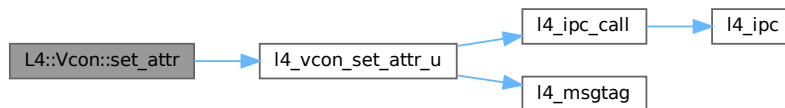
## Returns

Syscall return tag.

Definition at line 149 of file [vcon](#).

References [l4\\_vcon\\_set\\_attr\\_u\(\)](#).

Here is the call graph for this function:



## 15.234.2.6 write()

```

long L4::Vcon::write (
    char const * buf,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb() ) const [inline], [noexcept]

```

Write data to this virtual console.

## Parameters

<i>buf</i>	Pointer to the data buffer.
<i>size</i>	Size of the data buffer in bytes.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

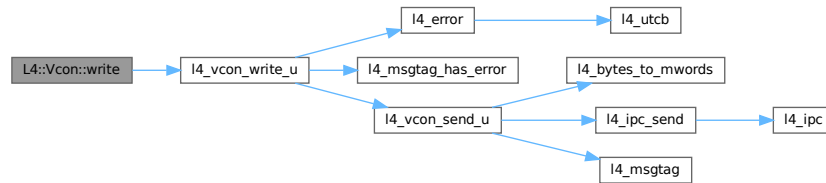
## Return values

<0	Error.
>=0	Number of bytes written to the virtual console.

Definition at line 90 of file [vcon](#).

References [l4\\_vcon\\_write\\_u\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

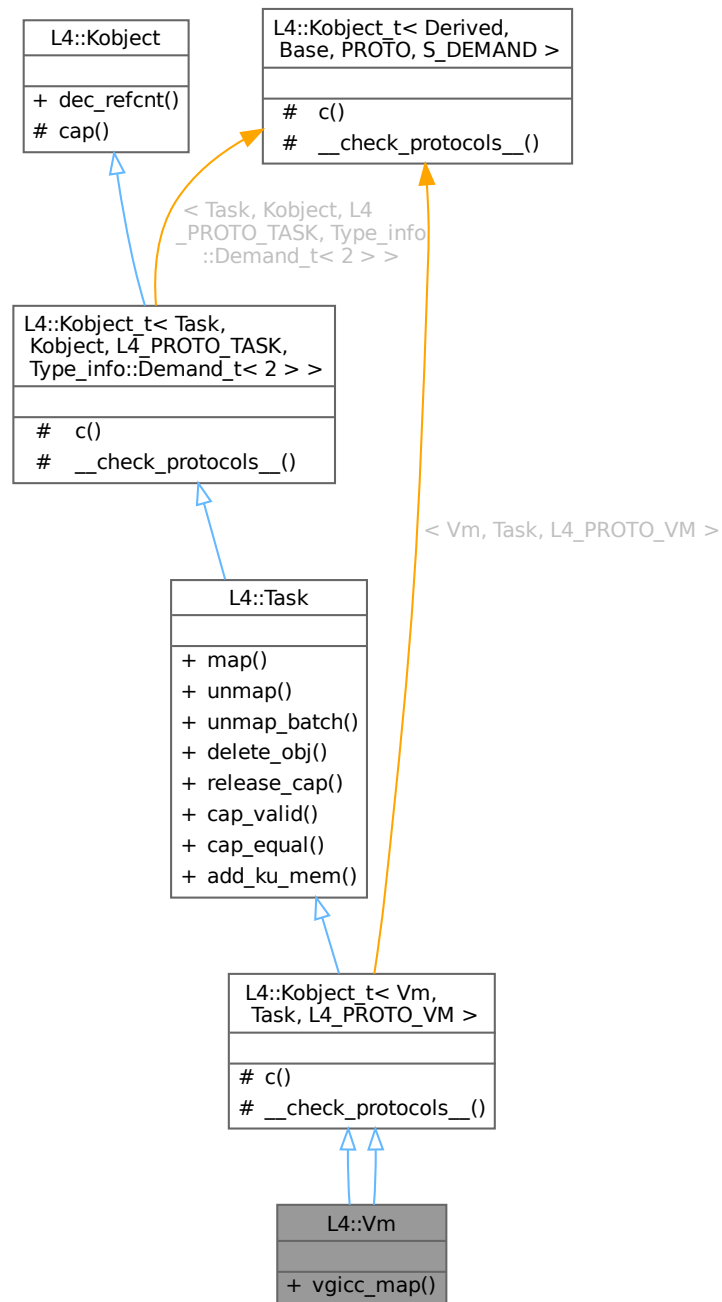
- `l4/sys/vcon`

## 15.235 L4::Vm Class Reference

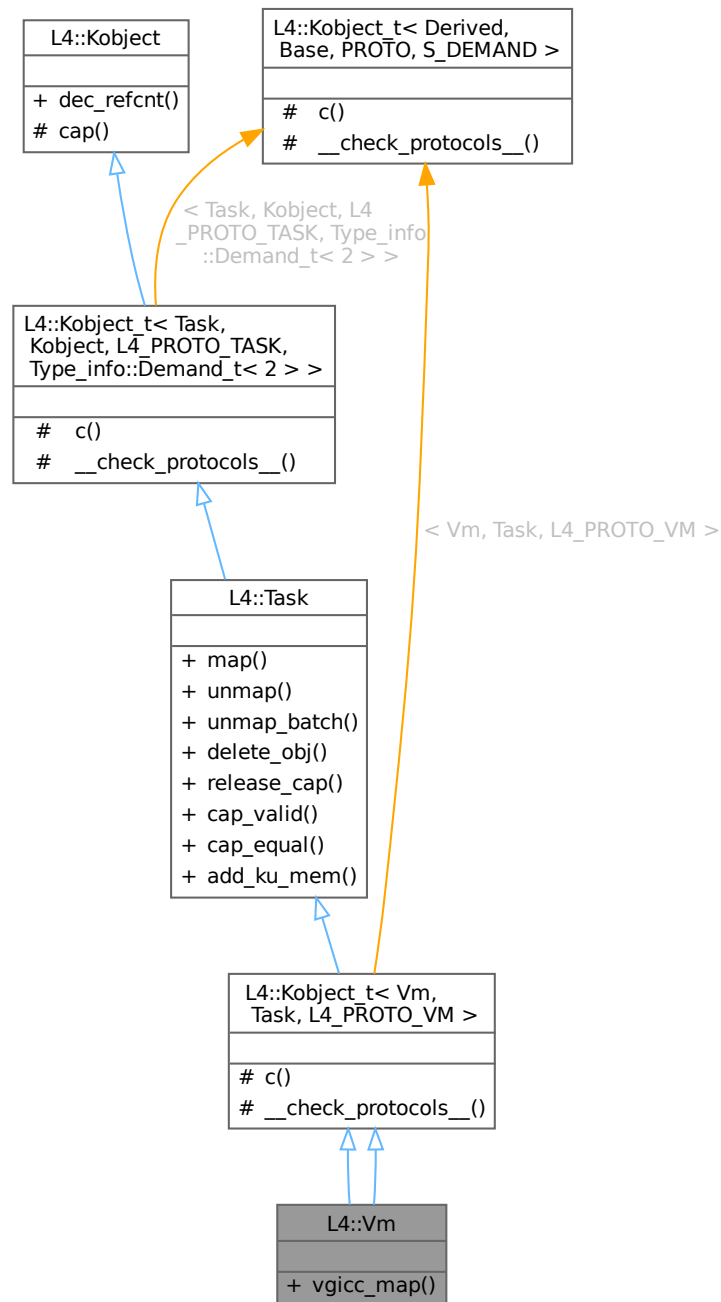
Virtual machine host address space.

```
#include <vm>
```

Inheritance diagram for L4::Vm:



Collaboration diagram for L4::Vm:



## Public Member Functions

- [l4\\_msgtag\\_t vgicc\\_map](#) ([l4\\_fpage\\_t](#) const [vgicc\\_fpage](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
Map the GIC virtual CPU interface page to the task in case Fiasco detected a GIC version 2.

## Public Member Functions inherited from L4::Task

- [l4\\_msgtag\\_t map](#) ([Cap](#)< [Task](#) > const &src\_task, [l4\\_fpage\\_t](#) const &snd\_fpage, [l4\\_umword\\_t](#) snd\_base, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Map resources available in the source task to a destination task.*
- [l4\\_msgtag\\_t unmap](#) ([l4\\_fpage\\_t](#) const &fpage, [l4\\_umword\\_t](#) map\_mask, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Revoke rights from the task.*
- [l4\\_msgtag\\_t unmap\\_batch](#) ([l4\\_fpage\\_t](#) const \*fpages, unsigned num\_fpages, [l4\\_umword\\_t](#) map\_mask, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Revoke rights from a task.*
- [l4\\_msgtag\\_t delete\\_obj](#) ([L4::Cap](#)< void > obj, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Release capability and delete object.*
- [l4\\_msgtag\\_t release\\_cap](#) ([L4::Cap](#)< void > cap, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Release object capability.*
- [l4\\_msgtag\\_t cap\\_valid](#) ([Cap](#)< void > const &cap, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Check whether a capability is present (refers to an object).*
- [l4\\_msgtag\\_t cap\\_equal](#) ([Cap](#)< void > const &cap\_a, [Cap](#)< void > const &cap\_b, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Test whether two capabilities point to the same object with the same rights.*
- [l4\\_msgtag\\_t add\\_ku\\_mem](#) ([l4\\_fpage\\_t](#) \*fpage, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Add kernel-user memory.*

## Public Member Functions inherited from L4::Kobject

- [l4\\_msgtag\\_t dec\\_refcnt](#) ([l4\\_mword\\_t](#) diff, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)())  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

## Protected Types inherited from L4::Kobject\_t< Vm, Task, L4\_PROTO\_VM >

- typedef [Vm Class](#)  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< [PROTO](#), [Vm](#) > [\\_\\_Iface](#)  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_Iface](#) >, typename Base::\_\_Iface\_list > [\\_\\_Iface\\_list](#)  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Types inherited from

## [L4::Kobject\\_t< Task, Kobject, L4\\_PROTO\\_TASK, Type\\_info::Demand\\_t< 2 > >](#)

- typedef [Task Class](#)  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< [PROTO](#), [Task](#) > [\\_\\_Iface](#)  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_Iface](#) >, typename Base::\_\_Iface\_list > [\\_\\_Iface\\_list](#)  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from [L4::Kobject\\_t< Vm, Task, L4\\_PROTO\\_VM >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject\\_t< Task, Kobject, L4\\_PROTO\\_TASK, Type\\_info::Demand\\_t< 2 > >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t cap \(\)](#) const noexcept  
*Return capability selector.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< Vm, Task, L4\\_PROTO\\_VM >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< Task, Kobject, L4\\_PROTO\\_TASK, Type\\_info::Demand\\_t< 2 > >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

## 15.235.1 Detailed Description

Virtual machine host address space.

[L4::Vm](#) is a specialisation of [L4::Task](#), used for virtual machines. The microkernel employs an appropriate page-table format for hosting VMs, such as ePT on VT-x. On Arm, it offers a call to make the virtual GICC area available to the VM.

Definition at line 28 of file [\\_\\_vm-arm.h](#).

## 15.235.2 Member Function Documentation

### 15.235.2.1 vgicc\_map()

```
l4_msgtag_t L4::Vm::vgicc_map (
    l4_fpage_t const vgicc_fpage,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

Map the GIC virtual CPU interface page to the task in case Fiasco detected a GIC version 2.

## Parameters

<i>vgicc_fpage</i>	Flexpage that describes an area in the address space of the destination task to map the vGICC page to.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

## Returns

Syscall return tag.

Definition at line 41 of file [\\_\\_vm-arm.h](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

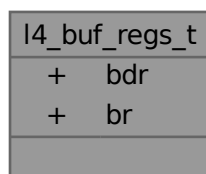
- [l4/sys/\\_\\_vm-arm.h](#)
- [l4/sys/vm](#)

## 15.236 l4\_buf\_regs\_t Struct Reference

Encapsulation of the buffer-registers block in the UTCB.

```
#include <utcb.h>
```

Collaboration diagram for `l4_buf_regs_t`:



## Data Fields

- [l4\\_umword\\_t](#) **bdr**  
*Buffer descriptor.*
- [l4\\_umword\\_t](#) **br** [L4\_UTCB\_GENERIC\_BUFFERS\_SIZE]  
*Buffer registers.*

### 15.236.1 Detailed Description

Encapsulation of the buffer-registers block in the UTCB.

Definition at line 93 of file [utcb.h](#).

The documentation for this struct was generated from the following file:

- [l4/sys/utcb.h](#)

### 15.237 l4\_exc\_regs\_t Struct Reference

UTCB structure for exceptions.

```
#include <utcb.h>
```

Collaboration diagram for `l4_exc_regs_t`:

l4_exc_regs_t
+ pfa
+ err
+ r
+ sp
+ ulr
+ _dummy1
+ pc
+ cpsr
+ tpidruo
+ tpidrurw
and 31 more...



## Data Fields

- [l4\\_umword\\_t pfa](#)  
*page fault address*
- [l4\\_umword\\_t err](#)  
*error code*
- [l4\\_umword\\_t r \[13\]](#)  
*registers*
- [l4\\_umword\\_t sp](#)  
*stack pointer*
- [l4\\_umword\\_t ulr](#)  
*ulr*
- [l4\\_umword\\_t \\_dummy1](#)  
*dummy*
- [l4\\_umword\\_t pc](#)  
*pc*
- [l4\\_umword\\_t cpsr](#)  
*cpsr*
- [l4\\_umword\\_t tpidruro](#)  
*Thread-ID register.*
- [l4\\_umword\\_t tpidrurw](#)  
*Thread-ID register.*
- [l4\\_umword\\_t r15](#)  
*r15*
- [l4\\_umword\\_t r14](#)  
*r14*
- [l4\\_umword\\_t r13](#)  
*r13*
- [l4\\_umword\\_t r12](#)  
*r12*
- [l4\\_umword\\_t r11](#)  
*r11*
- [l4\\_umword\\_t r10](#)  
*r10*
- [l4\\_umword\\_t r9](#)  
*r9*
- [l4\\_umword\\_t r8](#)  
*r8*
- [l4\\_umword\\_t rdi](#)  
*rdi*
- [l4\\_umword\\_t rsi](#)  
*rsi*
- [l4\\_umword\\_t rbp](#)  
*rbp*
- [l4\\_umword\\_t rbx](#)  
*rbx*
- [l4\\_umword\\_t rdx](#)  
*rdx*
- [l4\\_umword\\_t rcx](#)  
*rcx*
- [l4\\_umword\\_t rax](#)

- rax*
- [l4\\_umword\\_t trapno](#)  
*trap number*
- [l4\\_umword\\_t ip](#)  
*instruction pointer*
- [l4\\_umword\\_t dummy1](#)  
*dummy*
- [l4\\_umword\\_t flags](#)  
*rflags*
- [l4\\_umword\\_t ss](#)  
*stack segment register*
- [l4\\_umword\\_t es](#)  
*es register*
- [l4\\_umword\\_t ds](#)  
*ds register*
- [l4\\_umword\\_t gs](#)  
*gs register*
- [l4\\_umword\\_t fs](#)  
*fs register*
- [l4\\_umword\\_t edi](#)  
*edi register*
- [l4\\_umword\\_t esi](#)  
*esi register*
- [l4\\_umword\\_t ebp](#)  
*ebp register*
- [l4\\_umword\\_t ebx](#)  
*ebx register*
- [l4\\_umword\\_t edx](#)  
*edx register*
- [l4\\_umword\\_t ecx](#)  
*ecx register*
- [l4\\_umword\\_t eax](#)  
*eax register*

### 15.237.1 Detailed Description

UTCB structure for exceptions.

#### Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 38 of file [utcb.h](#).

### 15.237.2 Field Documentation

#### 15.237.2.1 flags

```
l4\_umword\_t l4_exc_regs_t::flags
```

rflags

eflags

Definition at line 80 of file [utcb.h](#).

### 15.237.2.2 ss

`l4_umword_t l4_exc_regs_t::ss`

stack segment register

ss register

Definition at line 82 of file [utcb.h](#).

The documentation for this struct was generated from the following files:

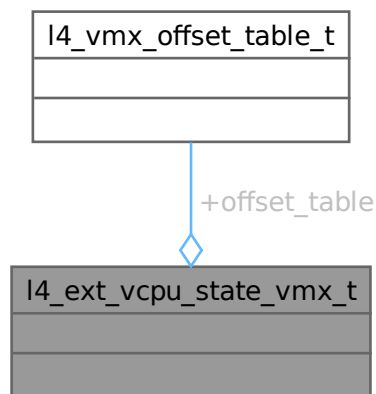
- [arm/l4/sys/utcb.h](#)
- [amd64/l4/sys/utcb.h](#)
- [x86/l4/sys/utcb.h](#)

## 15.238 l4\_ext\_vcpu\_state\_vmx\_t Struct Reference

VMX extended vCPU state.

```
#include <__vm-vmx.h>
```

Collaboration diagram for `l4_ext_vcpu_state_vmx_t`:



### 15.238.1 Detailed Description

VMX extended vCPU state.

For completeness, this is the overall memory layout of the vCPU:

0x000 - 0x1ff: Standard vCPU state [l4\\_vcpu\\_state\\_t](#) (with padding). 0x200 - 0x3ff: VMX capabilities (with padding). 0x400 - 0xffff: VMX extended vCPU state.

The memory layout of the VMX extended vCPU state is as follows:

0x000 - 0x007: Reserved (ignored by the kernel). In the hardware VMCS, the revision identifier and the abort indicator are stored in this area. Hereby we simply ignore these two entries. 0x008 - 0x00f: User space data (ignored by the kernel). This currently stores the pointer to a different VMX extended vCPU state that has been loaded into the given state. 0x010 - 0x013: VMCS field index of the software-defined CR2 field in the software VMCS. 0x014 - 0x017: Reserved. 0x018 - 0x01f: Capability of the hardware VMCS object (with padding). 0x020 - 0x047: Software VMCS field offset table [l4\\_vmx\\_offset\\_table\\_t](#). 0x048 - 0x0bf: Reserved. 0x0c0 - 0xabf: Software VMCS fields (with padding). 0xac0 - 0xbff: Software VMCS fields dirty bitmap (with padding).

Definition at line 231 of file [\\_\\_vm-vmx.h](#).

The documentation for this struct was generated from the following file:

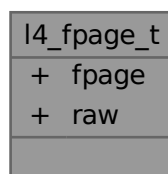
- [l4/sys/\\_\\_vm-vmx.h](#)

## 15.239 l4\_fpage\_t Union Reference

[L4](#) flexpage type.

```
#include <__l4_fpage.h>
```

Collaboration diagram for [l4\\_fpage\\_t](#):



### Data Fields

- [l4\\_umword\\_t](#) **fpage**  
*Raw value.*
- [l4\\_umword\\_t](#) **raw**  
*Raw value.*

### 15.239.1 Detailed Description

L4 flexpage type.

Definition at line 85 of file [\\_\\_l4\\_fpage.h](#).

The documentation for this union was generated from the following file:

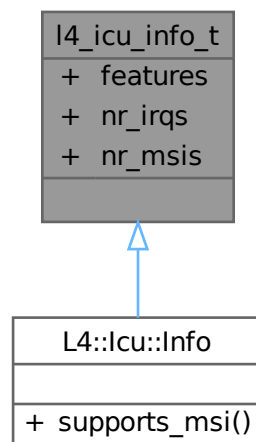
- l4/sys/\_\_l4\_fpage.h

## 15.240 l4\_icu\_info\_t Struct Reference

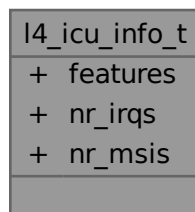
Info structure for an ICU.

```
#include <icu.h>
```

Inheritance diagram for l4\_icu\_info\_t:



Collaboration diagram for l4\_icu\_info\_t:



## Data Fields

- unsigned [features](#)  
*Feature flags.*
- unsigned [nr\\_irqs](#)  
*The number of IRQ lines supported by the ICU,.*
- unsigned [nr\\_msis](#)  
*The number of MSI vectors supported by the ICU,.*

### 15.240.1 Detailed Description

Info structure for an ICU.

This structure contains information about the features of an ICU.

See also

[l4\\_icu\\_info\(\)](#).

Definition at line [172](#) of file [icu.h](#).

### 15.240.2 Field Documentation

#### 15.240.2.1 features

```
unsigned l4_icu_info_t::features
```

Feature flags.

If [L4\\_ICU\\_FLAG\\_MSI](#) is set the ICU supports MSIs.

Definition at line [179](#) of file [icu.h](#).

Referenced by [L4::Icu::Info::supports\\_msi\(\)](#).

The documentation for this struct was generated from the following file:

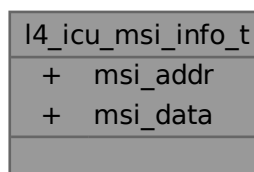
- [l4/sys/icu.h](#)

## 15.241 l4\_icu\_msi\_info\_t Struct Reference

Info to use for a specific MSI.

```
#include <icu.h>
```

Collaboration diagram for [l4\\_icu\\_msi\\_info\\_t](#):



**Data Fields**

- [l4\\_uint64\\_t msi\\_addr](#)  
*Value to use as address when sending this MSI.*
- [l4\\_uint32\\_t msi\\_data](#)  
*Value to use as data written to msi\_addr, when sending this MSI.*

**15.241.1 Detailed Description**

Info to use for a specific MSI.

Definition at line 193 of file [icu.h](#).

The documentation for this struct was generated from the following file:

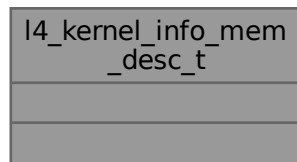
- [l4/sys/icu.h](#)

**15.242 l4\_kernel\_info\_mem\_desc\_t Struct Reference**

Memory descriptor data structure.

```
#include <memdesc.h>
```

Collaboration diagram for l4\_kernel\_info\_mem\_desc\_t:

**15.242.1 Detailed Description**

Memory descriptor data structure.

**Note**

This data type is opaque, and must be accessed by the accessor functions defined in this module.

Definition at line 84 of file [memdesc.h](#).

The documentation for this struct was generated from the following file:

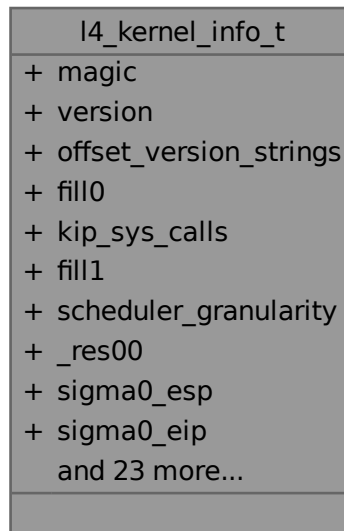
- [l4/sys/memdesc.h](#)

## 15.243 l4\_kernel\_info\_t Struct Reference

[L4 Kernel Interface Page](#).

```
#include <__kip-32bit.h>
```

Collaboration diagram for l4\_kernel\_info\_t:



### Data Fields

- [l4\\_uint32\\_t](#) **magic**  
*Kernel Info Page identifier ("L4μK").*
- [l4\\_uint32\\_t](#) **version**  
*Kernel version.*
- [l4\\_uint8\\_t](#) **offset\_version\_strings**  
*offset to version string*
- [l4\\_uint8\\_t](#) **fill0** [3]  
*reserved*
- [l4\\_uint8\\_t](#) **kip\_sys\_calls**  
*pointer to system calls*
- [l4\\_uint8\\_t](#) **fill1** [2]  
*reserved*
- [l4\\_umword\\_t](#) **scheduler\_granularity**  
*for rounding time slices*
- [l4\\_umword\\_t](#) **\_res00** [3]  
*default\_kdebug\_end*
- [l4\\_umword\\_t](#) **sigma0\_esp**  
*Sigma0 start stack pointer.*



- [l4\\_umword\\_t sigma0\\_eip](#)  
*Sigma0 instruction pointer.*
- [l4\\_umword\\_t \\_res01](#) [2]  
*reserved*
- [l4\\_umword\\_t sigma1\\_esp](#)  
*Sigma1 start stack pointer.*
- [l4\\_umword\\_t sigma1\\_eip](#)  
*Sigma1 instruction pointer.*
- [l4\\_umword\\_t \\_res02](#) [2]  
*reserved*
- [l4\\_umword\\_t root\\_esp](#)  
*Root task stack pointer.*
- [l4\\_umword\\_t root\\_eip](#)  
*Root task instruction pointer.*
- [l4\\_umword\\_t \\_res03](#) [2]  
*reserved*
- [l4\\_umword\\_t \\_res50](#) [1]  
*reserved*
- [l4\\_umword\\_t mem\\_info](#)  
*memory information*
- [l4\\_umword\\_t \\_res58](#) [2]  
*reserved*
- [l4\\_umword\\_t \\_res04](#) [16]  
*reserved*
- [l4\\_umword\\_t \\_res05](#) [2]  
*reserved*
- [l4\\_umword\\_t frequency\\_cpu](#)  
*CPU frequency in kHz.*
- [l4\\_umword\\_t frequency\\_bus](#)  
*Bus frequency.*
- [l4\\_umword\\_t \\_res06](#) [10]  
*reserved*
- [l4\\_umword\\_t user\\_ptr](#)  
*user\_ptr*
- [l4\\_umword\\_t vhw\\_offset](#)  
*offset to vhw structure*
- [l4\\_uint64\\_t magic](#)  
*Kernel Info Page identifier ("L4μK").*
- [l4\\_uint64\\_t version](#)  
*Kernel version.*
- [l4\\_uint8\\_t fill2](#) [7]  
*reserved*
- [l4\\_uint8\\_t fill3](#) [6]  
*reserved*
- [l4\\_umword\\_t \\_res\\_a0](#) [1]  
*reserved*
- [l4\\_umword\\_t \\_res\\_b0](#) [2]  
*reserver*

### 15.243.1 Detailed Description

[L4 Kernel Interface Page](#).

#### Examples

[examples/sys/ux-vhw/main.c](#).

Definition at line 38 of file [\\_\\_kip-32bit.h](#).

The documentation for this struct was generated from the following files:

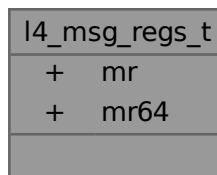
- [l4/sys/\\_\\_kip-32bit.h](#)
- [l4/sys/\\_\\_kip-64bit.h](#)

## 15.244 l4\_msg\_regs\_t Union Reference

Encapsulation of the message-register block in the UTCB.

```
#include <utcb.h>
```

Collaboration diagram for `l4_msg_regs_t`:



#### Data Fields

- [l4\\_umword\\_t](#) **mr** [`L4_UTCB_GENERIC_DATA_SIZE`]  
*Message registers.*
- [l4\\_uint64\\_t](#) **mr64** [`L4_UTCB_GENERIC_DATA_SIZE/(sizeof(l4_uint64_t)/sizeof(l4_umword_t))`]  
*Message registers 64bit alias.*

### 15.244.1 Detailed Description

Encapsulation of the message-register block in the UTCB.

#### Examples

[examples/sys/utcb-ipc/main.c](#).

Definition at line 78 of file [utcb.h](#).

The documentation for this union was generated from the following file:

- [l4/sys/utcb.h](#)

## 15.245 l4\_msgtag\_t Struct Reference

Message tag data structure.

```
#include <types.h>
```

Collaboration diagram for l4\_msgtag\_t:

l4_msgtag_t
+ raw
+ label()
+ label()
+ words()
+ items()
+ flags()
+ is_page_fault()
+ is_preemption()
+ is_sys_exception()
+ is_exception()
+ is_sigma0()
+ is_io_page_fault()
+ has_error()

### Public Member Functions

- long **label** () const [L4\\_NOTHROW](#)  
*Get the protocol value.*
- void **label** (long v) [L4\\_NOTHROW](#)  
*Set the protocol value.*
- unsigned **words** () const [L4\\_NOTHROW](#)  
*Get the number of untyped words.*
- unsigned **items** () const [L4\\_NOTHROW](#)  
*Get the number of typed items.*
- unsigned **flags** () const [L4\\_NOTHROW](#)  
*Get the flags value.*
- bool **is\_page\_fault** () const [L4\\_NOTHROW](#)  
*Test if protocol indicates page-fault protocol.*
- bool **is\_preemption** () const [L4\\_NOTHROW](#)  
*Test if protocol indicates preemption protocol.*
- bool **is\_sys\_exception** () const [L4\\_NOTHROW](#)  
*Test if protocol indicates system-exception protocol.*
- bool **is\_exception** () const [L4\\_NOTHROW](#)

- *Test if protocol indicates exception protocol.*  
bool **is\_sigma0** () const [L4\\_NOTHROW](#)
- *Test if protocol indicates sigma0 protocol.*  
bool **is\_io\_page\_fault** () const [L4\\_NOTHROW](#)
- *Test if protocol indicates IO-page-fault protocol.*  
unsigned **has\_error** () const [L4\\_NOTHROW](#)
- *Test if flags indicate an error.*

## Data Fields

- [l4\\_mword\\_t](#) **raw**  
*raw value*

## 15.245.1 Detailed Description

Message tag data structure.

### Include File

```
#include <l4/sys/types.h>
```

Describes the details of an IPC operation, in particular which parts of the UTCB have to be transmitted, and also flags to enable real-time and FPU extensions.

The message tag also contains a user-defined label that could be used to specify a protocol ID. Some negative values are reserved for kernel protocols such as page faults and exceptions.

The type must be treated completely opaque.

### Examples

[examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#), [examples/libs/l4re/streammap/server.cc](#), [examples/sys/aliens/main.c](#), [examples/sys/ipc/ipc\\_example.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line [163](#) of file [types.h](#).

## 15.245.2 Member Function Documentation

### 15.245.2.1 flags()

```
unsigned l4_msgtag_t::flags ( ) const [inline]
```

Get the flags value.

The flags are a combination of the flags defined by [L4\\_msgtag\\_flags](#).

Definition at line [181](#) of file [types.h](#).

References [raw](#).

The documentation for this struct was generated from the following file:

- [l4/sys/types.h](#)

## 15.246 l4\_sched\_cpu\_set\_t Struct Reference

CPU sets.

```
#include <scheduler.h>
```

Collaboration diagram for l4\_sched\_cpu\_set\_t:

l4_sched_cpu_set_t
+ gran_offset
+ map
+ granularity()
+ offset()
+ set()

### Public Member Functions

- unsigned char [granularity](#) () const
- unsigned [offset](#) () const
- void [set](#) (unsigned char [granularity](#), unsigned [offset](#))

*Set offset and granularity.*

### Data Fields

- [l4\\_umword\\_t](#) [gran\\_offset](#)  
*Combination of granularity and offset.*
- [l4\\_umword\\_t](#) [map](#)  
*Bitmap of CPUs.*

### 15.246.1 Detailed Description

CPU sets.

#### Examples

[examples/sys/migrate/thread\\_migrate.cc](#).

Definition at line 69 of file [scheduler.h](#).

## 15.246.2 Member Function Documentation

### 15.246.2.1 granularity()

```
unsigned char l4_sched_cpu_set_t::granularity ( ) const [inline]
```

#### Returns

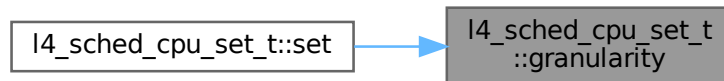
Get granularity value

Definition at line 92 of file [scheduler.h](#).

References [gran\\_offset](#).

Referenced by [set\(\)](#).

Here is the caller graph for this function:



### 15.246.2.2 offset()

```
unsigned l4_sched_cpu_set_t::offset ( ) const [inline]
```

#### Returns

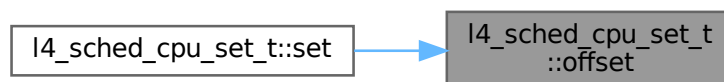
Get offset value

Definition at line 94 of file [scheduler.h](#).

References [gran\\_offset](#).

Referenced by [set\(\)](#).

Here is the caller graph for this function:



### 15.246.2.3 set()

```
void l4_sched_cpu_set_t::set (
    unsigned char granularity,
    unsigned offset ) [inline]
```

Set offset and granularity.

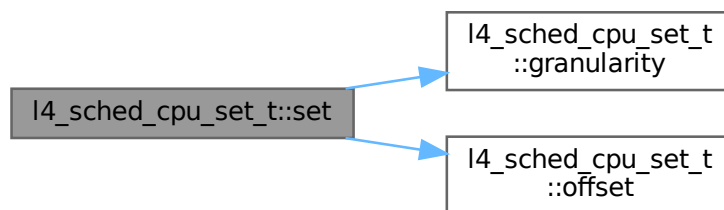
## Parameters

<i>granularity</i>	Granularity in log2 notation.
<i>offset</i>	Offset. Must be a multiple of $2^{\text{granularity}}$ .

Definition at line 101 of file [scheduler.h](#).

References [gran\\_offset](#), [granularity\(\)](#), and [offset\(\)](#).

Here is the call graph for this function:



### 15.246.3 Field Documentation

#### 15.246.3.1 gran\_offset

`l4_umword_t l4_sched_cpu_set_t::gran_offset`

Combination of granularity and offset.

The granularity defines how many CPUs each bit in map describes. And the offset is the number of the first CPU described by the first bit in the bitmap.

#### Precondition

offset must be a multiple of  $2^{\text{granularity}}$ .

MSB	LSB
8bit granularity	24bit offset ..

Definition at line 83 of file [scheduler.h](#).

Referenced by [granularity\(\)](#), [L4::Scheduler::info\(\)](#), [l4\\_sched\\_cpu\\_set\(\)](#), [offset\(\)](#), and [set\(\)](#).

The documentation for this struct was generated from the following file:

- [l4/sys/scheduler.h](#)

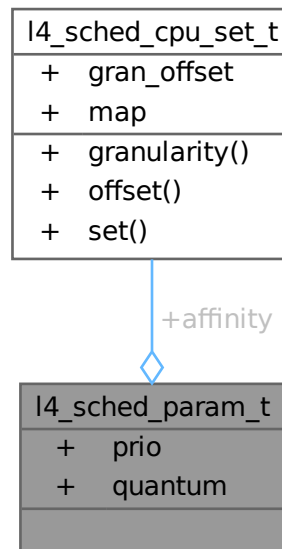


## 15.247 l4\_sched\_param\_t Struct Reference

Scheduler parameter set.

```
#include <scheduler.h>
```

Collaboration diagram for l4\_sched\_param\_t:



### Data Fields

- [l4\\_sched\\_cpu\\_set\\_t](#) **affinity**  
*CPU affinity.*
- [l4\\_umword\\_t](#) **prio**  
*Priority for scheduling.*
- [l4\\_umword\\_t](#) **quantum**  
*Timeslice in micro seconds.*

### 15.247.1 Detailed Description

Scheduler parameter set.

#### Examples

[examples/sys/aliens/main.c](#), [examples/sys/migrate/thread\\_migrate.cc](#), [examples/sys/singlestep/main.c](#),  
[examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line [184](#) of file [scheduler.h](#).

The documentation for this struct was generated from the following file:

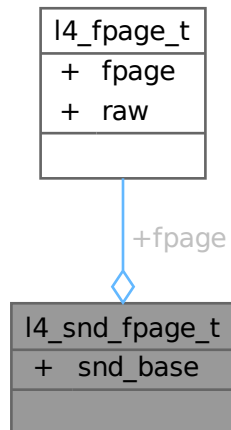
- [l4/sys/scheduler.h](#)

## 15.248 l4\_snd\_fpage\_t Struct Reference

Send-flex-page types.

```
#include <__l4_fpage.h>
```

Collaboration diagram for l4\_snd\_fpage\_t:



### Data Fields

- `l4_umword_t snd_base`  
*Offset in receive window (send base)*
- `l4_fpage_t fpage`  
*Source flex-page descriptor.*

### 15.248.1 Detailed Description

Send-flex-page types.

Definition at line 108 of file `__l4_fpage.h`.

The documentation for this struct was generated from the following file:

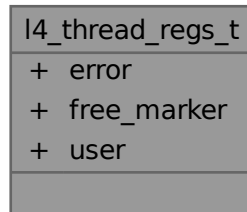
- `l4/sys/__l4_fpage.h`

## 15.249 l4\_thread\_regs\_t Struct Reference

Encapsulation of the thread-control-register block of the UTCB.

```
#include <utcb.h>
```

Collaboration diagram for l4\_thread\_regs\_t:



### Data Fields

- [l4\\_umword\\_t](#) **error**  
*System call error codes.*
- [l4\\_umword\\_t](#) **free\_marker**  
*Kernel free marker.*
- [l4\\_umword\\_t](#) **user** [3]  
*User values (ignored and preserved by the kernel)*

### 15.249.1 Detailed Description

Encapsulation of the thread-control-register block of the UTCB.

Definition at line 110 of file [utcb.h](#).

The documentation for this struct was generated from the following file:

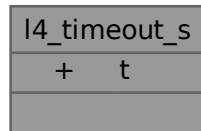
- l4/sys/[utcb.h](#)

## 15.250 l4\_timeout\_s Struct Reference

Basic timeout specification.

```
#include <__timeout.h>
```

Collaboration diagram for l4\_timeout\_s:



### Data Fields

- [l4\\_uint16\\_t](#) *t*  
*timeout value*

### 15.250.1 Detailed Description

Basic timeout specification.

Basically a floating point number with 10 bits mantissa and 5 bits exponent ( $t = m \cdot 2^e$ ).

If bit 15 == 1 the timeout is absolute and the lower 6 bits encode the index of the UTCB buffer register(s) holding the absolute 64-bit timeout value. On 32-bit systems, two consecutive UTCB buffer registers are used.

Definition at line 48 of file [\\_\\_timeout.h](#).

The documentation for this struct was generated from the following file:

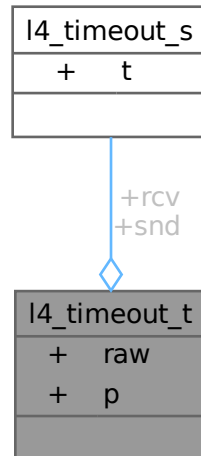
- [l4/sys/\\_\\_timeout.h](#)

## 15.251 l4\_timeout\_t Union Reference

Timeout pair.

```
#include <__timeout.h>
```

Collaboration diagram for l4\_timeout\_t:



### Data Fields

- `l4_uint32_t raw`  
*raw value*
- struct {  
    `l4_timeout_s rcv`  
        *receive timeout*  
    `l4_timeout_s snd`  
        *send timeout*  
} `p`  
  
*combined timeout*

### 15.251.1 Detailed Description

Timeout pair.

For IPC there are usually a send and a receive timeout. So this structure contains a pair of timeouts.

Definition at line 60 of file `__timeout.h`.

The documentation for this union was generated from the following file:

- `l4/sys/__timeout.h`

## 15.252 l4\_vcon\_attr\_t Struct Reference

Vcon attribute structure.

```
#include <vcon.h>
```

Collaboration diagram for l4\_vcon\_attr\_t:

l4_vcon_attr_t
+ i_flags
+ o_flags
+ l_flags
+ set_raw()

### Public Member Functions

- void [set\\_raw](#) ()  
*Set terminal attributes to disable all special processing.*

### Data Fields

- [l4\\_umword\\_t i\\_flags](#)  
*input flags*
- [l4\\_umword\\_t o\\_flags](#)  
*output flags*
- [l4\\_umword\\_t l\\_flags](#)  
*local flags*

### 15.252.1 Detailed Description

Vcon attribute structure.

The flags members can be a combination of their respective enums.

See also

[L4\\_vcon\\_i\\_flags](#)  
[L4\\_vcon\\_o\\_flags](#)  
[L4\\_vcon\\_l\\_flags](#)

Examples

[examples/sys/isr/main.c](#).

Definition at line 196 of file [vcon.h](#).

## 15.252.2 Member Function Documentation

### 15.252.2.1 set\_raw()

```
void l4_vcon_attr_t::set_raw ( ) [inline]
```

Set terminal attributes to disable all special processing.

Removes all flags that would mangle the read or written characters. Also disables echoing and any special processing of characters.

Definition at line 459 of file [vcon.h](#).

References [l4\\_vcon\\_set\\_attr\\_raw\(\)](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

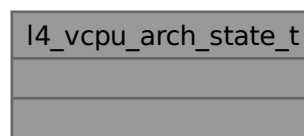
- [l4/sys/vcon.h](#)

## 15.253 l4\_vcpu\_arch\_state\_t Struct Reference

Architecture-specific vCPU state.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for `l4_vcpu_arch_state_t`:



### 15.253.1 Detailed Description

Architecture-specific vCPU state.

Definition at line 85 of file [\\_\\_vcpu-arch.h](#).

The documentation for this struct was generated from the following files:

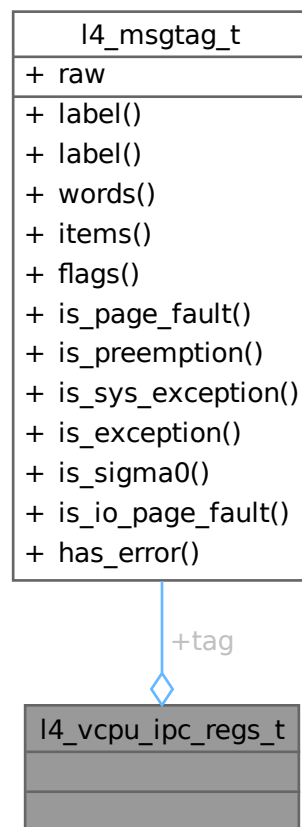
- [arm/l4/sys/\\_\\_vcpu-arch.h](#)
- [amd64/l4/sys/\\_\\_vcpu-arch.h](#)

## 15.254 l4\_vcpu\_ipc\_regs\_t Struct Reference

vCPU message registers.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for l4\_vcpu\_ipc\_regs\_t:





### 15.254.1 Detailed Description

vCPU message registers.

Definition at line 94 of file [\\_\\_vcpu-arch.h](#).

The documentation for this struct was generated from the following files:

- [arm/l4/sys/\\_\\_vcpu-arch.h](#)
- [amd64/l4/sys/\\_\\_vcpu-arch.h](#)
- [x86/l4/sys/\\_\\_vcpu-arch.h](#)

## 15.255 l4\_vcpu\_regs\_t Struct Reference

vCPU registers.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for l4\_vcpu\_regs\_t:

l4_vcpu_regs_t
+ pfa
+ err
+ sp
+ ip
+ flags
+ tpidruro
+ tpidrurw
+ r15
+ r14
+ r13
and 20 more...

### Data Fields

- [l4\\_umword\\_t](#) **pfa**  
*page fault address*
- [l4\\_umword\\_t](#) **err**  
*error code*
- [l4\\_umword\\_t](#) **sp**

- stack pointer*
- [l4\\_umword\\_t ip](#)
- instruction pointer*
- [l4\\_umword\\_t flags](#)
- eflags*
- [l4\\_umword\\_t tpidrur0](#)
- Thread-ID register.*
- [l4\\_umword\\_t tpidrurw](#)
- Thread-ID register.*
- [l4\\_umword\\_t r15](#)
- r15 register*
- [l4\\_umword\\_t r14](#)
- r14 register*
- [l4\\_umword\\_t r13](#)
- r13 register*
- [l4\\_umword\\_t r12](#)
- r12 register*
- [l4\\_umword\\_t r11](#)
- r11 register*
- [l4\\_umword\\_t r10](#)
- r10 register*
- [l4\\_umword\\_t r9](#)
- r9 register*
- [l4\\_umword\\_t r8](#)
- r8 register*
- [l4\\_umword\\_t di](#)
- rdi register*
- [l4\\_umword\\_t si](#)
- rsi register*
- [l4\\_umword\\_t bp](#)
- rbp register*
- [l4\\_umword\\_t bx](#)
- rbx register*
- [l4\\_umword\\_t dx](#)
- rdx register*
- [l4\\_umword\\_t cx](#)
- rcx register*
- [l4\\_umword\\_t ax](#)
- rax register*
- [l4\\_umword\\_t trapno](#)
- trap number*
- [l4\\_umword\\_t cs](#)
- dummy*
- [l4\\_umword\\_t ss](#)
- ss register*
- [l4\\_umword\\_t es](#)
- es register*
- [l4\\_umword\\_t ds](#)
- ds register*
- [l4\\_umword\\_t gs](#)
- gs register*

- [l4\\_umword\\_t fs](#)  
*fs register*
- [l4\\_umword\\_t dummy1](#)  
*dummy*

### 15.255.1 Detailed Description

vCPU registers.

Definition at line 66 of file [\\_\\_vcpu-arch.h](#).

### 15.255.2 Field Documentation

#### 15.255.2.1 ax

[l4\\_umword\\_t](#) l4\_vcpu\_regs\_t::ax

rax register

eax register

Definition at line 88 of file [\\_\\_vcpu-arch.h](#).

#### 15.255.2.2 bp

[l4\\_umword\\_t](#) l4\_vcpu\_regs\_t::bp

rbp register

ebp register

Definition at line 83 of file [\\_\\_vcpu-arch.h](#).

#### 15.255.2.3 bx

[l4\\_umword\\_t](#) l4\_vcpu\_regs\_t::bx

rbx register

ebx register

Definition at line 85 of file [\\_\\_vcpu-arch.h](#).

#### 15.255.2.4 cx

[l4\\_umword\\_t](#) l4\_vcpu\_regs\_t::cx

rcx register

ecx register

Definition at line 87 of file [\\_\\_vcpu-arch.h](#).

#### 15.255.2.5 di

`14_umword_t` `l4_vcpu_regs_t::di`

rdi register

edi register

Definition at line 81 of file [\\_\\_vcpu-arch.h](#).

#### 15.255.2.6 dx

`14_umword_t` `l4_vcpu_regs_t::dx`

rdx register

edx register

Definition at line 86 of file [\\_\\_vcpu-arch.h](#).

#### 15.255.2.7 si

`14_umword_t` `l4_vcpu_regs_t::si`

rsi register

esi register

Definition at line 82 of file [\\_\\_vcpu-arch.h](#).

The documentation for this struct was generated from the following files:

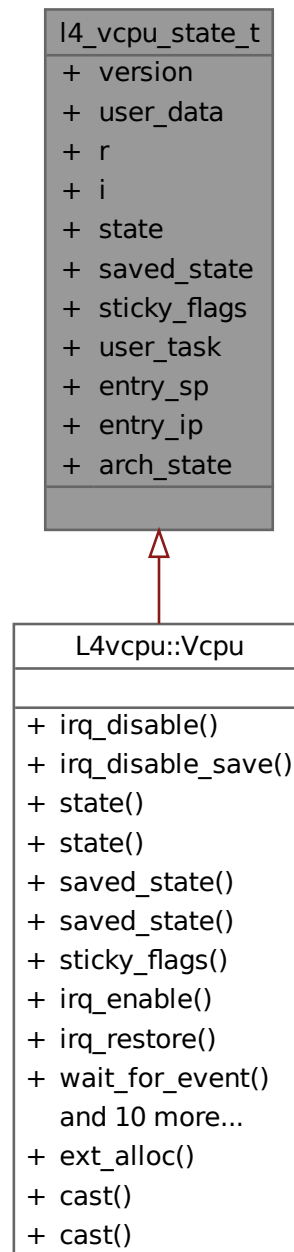
- [arm/l4/sys/\\_\\_vcpu-arch.h](#)
- [amd64/l4/sys/\\_\\_vcpu-arch.h](#)
- [x86/l4/sys/\\_\\_vcpu-arch.h](#)

## 15.256 l4\_vcpu\_state\_t Struct Reference

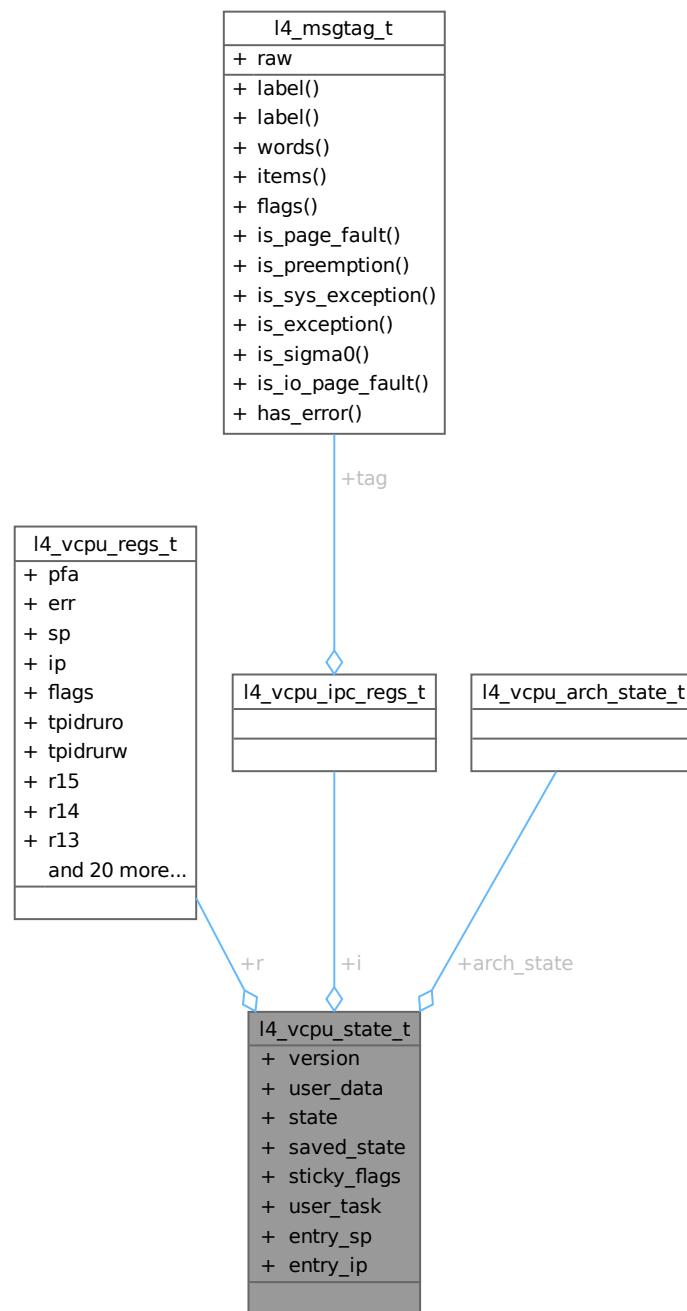
State of a vCPU.

```
#include <vcpu.h>
```

Inheritance diagram for l4\_vcpu\_state\_t:



Collaboration diagram for `l4_vcpu_state_t`:



## Data Fields

- [l4\\_umword\\_t version](#)  
*vCPU ABI version.*
- [l4\\_umword\\_t user\\_data](#) [7]  
*User-specific data.*
- [l4\\_vcpu\\_regs\\_t r](#)

- Register state.*
- [l4\\_vcpu\\_ipc\\_regs\\_t](#) **i**  
*IPC state.*
- [l4\\_uint16\\_t](#) **state**  
*Current vCPU state. See [L4\\_vcpu\\_state\\_flags](#).*
- [l4\\_uint16\\_t](#) **saved\_state**  
*Saved vCPU state. See [L4\\_vcpu\\_state\\_flags](#).*
- [l4\\_uint16\\_t](#) **sticky\_flags**  
*Pending flags. See [L4\\_vcpu\\_sticky\\_flags](#).*
- [l4\\_cap\\_idx\\_t](#) **user\_task**  
*User task to use.*
- [l4\\_umword\\_t](#) **entry\_sp**  
*Stack pointer for entry (when coming from user task)*
- [l4\\_umword\\_t](#) **entry\_ip**  
*IP for entry.*
- [l4\\_vcpu\\_arch\\_state\\_t](#) **arch\_state**  
*Architecture-specific state.*

### 15.256.1 Detailed Description

State of a vCPU.

Definition at line 86 of file [vcpu.h](#).

### 15.256.2 Field Documentation

#### 15.256.2.1 version

```
l4\_umword\_t l4_vcpu_state_t::version
```

vCPU ABI version.

Set by the kernel and must be checked by the user for equality with [L4\\_VCPU\\_STATE\\_VERSION](#).

Definition at line 88 of file [vcpu.h](#).

Referenced by [l4\\_vcpu\\_check\\_version\(\)](#).

The documentation for this struct was generated from the following file:

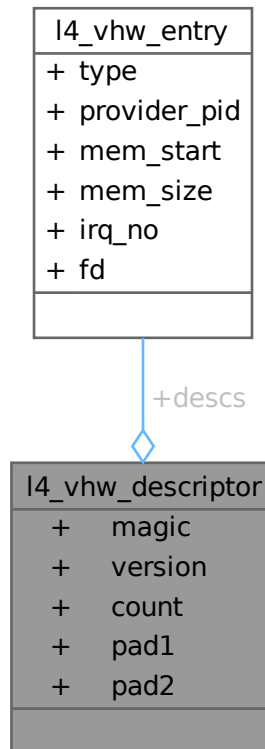
- [l4/sys/vcpu.h](#)

## 15.257 I4\_vhw\_descriptor Struct Reference

Virtual hardware devices description.

```
#include <vhw.h>
```

Collaboration diagram for I4\_vhw\_descriptor:



### Data Fields

- [I4\\_uint32\\_t](#) **magic**  
*Magic.*
- [I4\\_uint8\\_t](#) **version**  
*Version of the descriptor.*
- [I4\\_uint8\\_t](#) **count**  
*Number of entries.*
- [I4\\_uint8\\_t](#) **pad1**  
*padding*
- [I4\\_uint8\\_t](#) **pad2**  
*padding*
- struct [I4\\_vhw\\_entry](#) **descs** []  
*Array of device descriptions.*



### 15.257.1 Detailed Description

Virtual hardware devices description.

#### Examples

[examples/sys/ux-vhw/main.c](#).

Definition at line 70 of file [vhw.h](#).

The documentation for this struct was generated from the following file:

- [l4/sys/vhw.h](#)

## 15.258 l4\_vhw\_entry Struct Reference

Description of a device.

```
#include <vhw.h>
```

Collaboration diagram for l4\_vhw\_entry:

l4_vhw_entry
+ type
+ provider_pid
+ mem_start
+ mem_size
+ irq_no
+ fd

#### Data Fields

- enum [l4\\_vhw\\_entry\\_type](#) **type**  
*Type of virtual hardware.*
- [l4\\_uint32\\_t](#) **provider\_pid**  
*Host PID of the VHW provider.*
- [l4\\_addr\\_t](#) **mem\_start**  
*Start of memory region.*
- [l4\\_addr\\_t](#) **mem\_size**  
*Size of memory region.*
- [l4\\_uint32\\_t](#) **irq\_no**  
*IRQ number.*
- [l4\\_uint32\\_t](#) **fd**  
*File descriptor.*

### 15.258.1 Detailed Description

Description of a device.

#### Examples

[examples/sys/ux-vhw/main.c](#).

Definition at line 55 of file [vhw.h](#).

The documentation for this struct was generated from the following file:

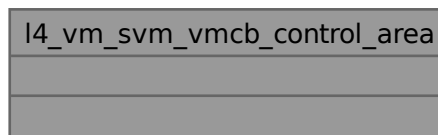
- [l4/sys/vhw.h](#)

## 15.259 l4\_vm\_svm\_vmcb\_control\_area Struct Reference

VMCB structure for SVM VMs.

```
#include <__vm-svm.h>
```

Collaboration diagram for `l4_vm_svm_vmcb_control_area`:



### 15.259.1 Detailed Description

VMCB structure for SVM VMs.

Definition at line 39 of file [\\_\\_vm-svm.h](#).

The documentation for this struct was generated from the following file:

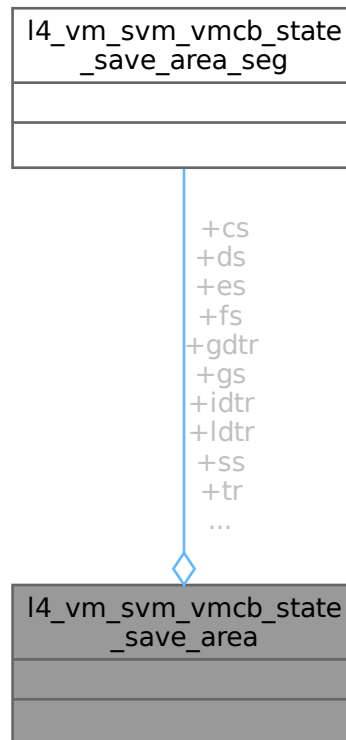
- [l4/sys/\\_\\_vm-svm.h](#)

## 15.260 l4\_vm\_svm\_vmcb\_state\_save\_area Struct Reference

State save area structure for SVM VMs.

```
#include <__vm-svm.h>
```

Collaboration diagram for l4\_vm\_svm\_vmcb\_state\_save\_area:



### 15.260.1 Detailed Description

State save area structure for SVM VMs.

Definition at line 96 of file [\\_\\_vm-svm.h](#).

The documentation for this struct was generated from the following file:

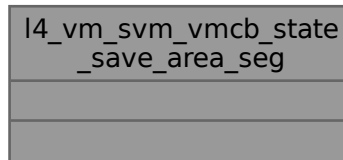
- `l4/sys/__vm-svm.h`

## 15.261 l4\_vm\_svm\_vmcb\_state\_save\_area\_seg Struct Reference

State save area segment selector struct.

```
#include <__vm-svm.h>
```

Collaboration diagram for l4\_vm\_svm\_vmcb\_state\_save\_area\_seg:



### 15.261.1 Detailed Description

State save area segment selector struct.

Definition at line 84 of file [\\_\\_vm-svm.h](#).

The documentation for this struct was generated from the following file:

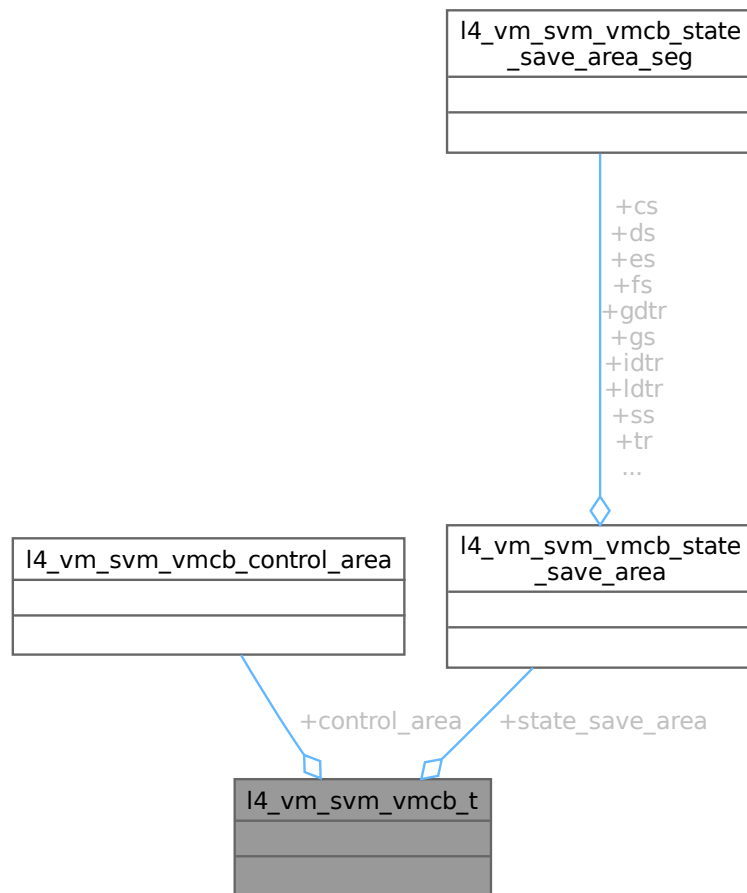
- l4/sys/\_\_vm-svm.h

## 15.262 l4\_vm\_svm\_vmcb\_t Struct Reference

Control structure for SVM VMs.

```
#include <__vm-svm.h>
```

Collaboration diagram for l4\_vm\_svm\_vmcb\_t:



### 15.262.1 Detailed Description

Control structure for SVM VMs.

Definition at line 165 of file `__vm-svm.h`.

The documentation for this struct was generated from the following file:

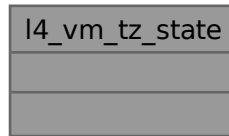
- `l4/sys/__vm-svm.h`

## 15.263 l4\_vm\_tz\_state Struct Reference

state structure for TrustZone VMs

```
#include <vm.h>
```

Collaboration diagram for l4\_vm\_tz\_state:



### 15.263.1 Detailed Description

state structure for TrustZone VMs

Definition at line 52 of file [vm.h](#).

The documentation for this struct was generated from the following file:

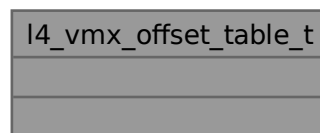
- [arm/l4/sys/vm.h](#)

## 15.264 l4\_vmx\_offset\_table\_t Struct Reference

Software VMCS field offset table.

```
#include <__vm-vmx.h>
```

Collaboration diagram for l4\_vmx\_offset\_table\_t:



### 15.264.1 Detailed Description

Software VMCS field offset table.

The memory layout is as follows:

0x00 - 0x02: 3 offsets for 16-bit fields. 0x03: Reserved. 0x04 - 0x06: 3 offsets for 64-bit fields. 0x07: Reserved. 0x08 - 0x0a: 3 offsets for 32-bit fields. 0x0b: Reserved. 0x0c - 0x0e: 3 offsets for natural-width fields. 0x0f: Reserved. 0x10 - 0x12: 3 limits for 16-bit fields. 0x13: Reserved. 0x14 - 0x16: 3 limits for 64-bit fields. 0x17: Reserved. 0x18 - 0x1a: 3 limits for 32-bit fields. 0x1b: Reserved. 0x1c - 0x1e: 3 limits for natural-width fields. 0x1f: Reserved. 0x20 - 0x23: 4 index shifts. 0x24: Offset of the first software VMCS field. 0x25: Size of the software VMCS fields. 0x26 - 0x27: Reserved.

The offsets/limits in each size category are in the following order:

- Control fields.
- Read-only fields.
- Guest fields.

The index shifts are in the following order:

- 16-bit.
- 64-bit.
- 32-bit.
- Natural-width.

All offsets/limits/sizes are represented in a 64-byte granule.

The offsets (after being multiplied by 64) are indexes in the values array in [l4\\_ext\\_vcpu\\_state\\_vmx\\_t](#) and bit indexes in the dirty\_bitmap array in [l4\\_ext\\_vcpu\\_state\\_vmx\\_t](#).

The limits (after being multiplied by 64) represent the range of the available indexes.

Definition at line 186 of file [\\_\\_vm-vmx.h](#).

The documentation for this struct was generated from the following file:

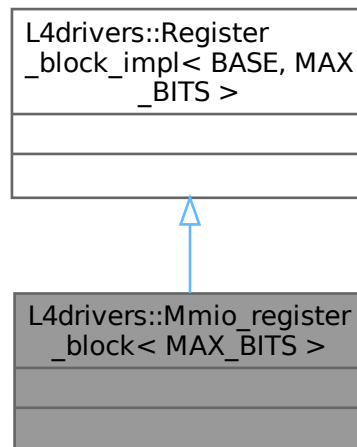
- l4/sys/\_\_vm-vmx.h

## 15.265 L4drivers::Mmio\_register\_block< MAX\_BITS > Struct Template Reference

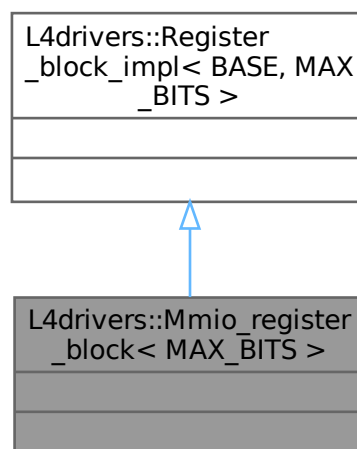
An MMIO block with up to 64-bit register access (32-bit default) and little endian byte order.

```
#include <hw_mmio_register_block>
```

Inheritance diagram for L4drivers::Mmio\_register\_block< MAX\_BITS >:



Collaboration diagram for L4drivers::Mmio\_register\_block< MAX\_BITS >:





### 15.265.1 Detailed Description

```
template<unsigned MAX_BITS = 32>
struct L4drivers::Mmio_register_block< MAX_BITS >
```

An MMIO block with up to 64-bit register access (32-bit default) and little endian byte order.

Definition at line 42 of file [hw\\_mmio\\_register\\_block](#).

The documentation for this struct was generated from the following file:

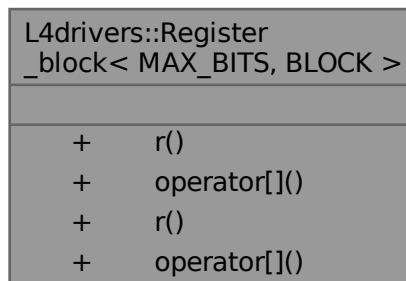
- pkg/drivers-frst/include/hw\_mmio\_register\_block

## 15.266 L4drivers::Register\_block< MAX\_BITS, BLOCK > Class Template Reference

Handles a reference to a register block of the given maximum access width.

```
#include <hw_register_block>
```

Collaboration diagram for L4drivers::Register\_block< MAX\_BITS, BLOCK >:



### Public Member Functions

- template<unsigned BITS>  
[Ro\\_register\\_tmpl](#)< BITS, Block > [r](#) (unsigned offset) const  
*Read only access to register at offset offset.*
- Ro\_register [operator\[\]](#) (unsigned offset) const  
*Read only access to register at offset offset.*
- template<unsigned BITS>  
[Register\\_tmpl](#)< BITS, Block > [r](#) (unsigned offset)  
*Read/write access to register at offset offset.*
- Register [operator\[\]](#) (unsigned offset)  
*Read/write access to register at offset offset.*

### 15.266.1 Detailed Description

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_↵
BITS> >>
class L4drivers::Register_block< MAX_BITS, BLOCK >
```

Handles a reference to a register block of the given maximum access width.

#### Template Parameters

<i>MAX_BITS</i>	Maximum access width for the registers in this block.
<i>BLOCK</i>	Type implementing the register accesses (read<>(), write<>(), modify<>(), set<>(), and clear<>()).

Provides access to registers in this block via [r<WIDTH>\(\)](#) and [operator\[\]\(\)](#).

#### Example usage:

```
void test()
{
    // create a register block reference for max. 16bit accesses, using a
    // MMIO register block implementation (at address 0x1000).
    Hw::Register_block<16> regs = new Hw::Mmio_register_block<16>(0x1000);

    // Alternatively it is allowed to use an implementation that allows
    // wider access than actually needed.
    Hw::Register_block<16> regs = new Hw::Mmio_register_block<32>(0x1000);

    // read a 16bit register at offset 8byte
    unsigned short x = regs.r<16>(8);
    unsigned short x1 = regs[8];          // alternative

    // read an 8bit register at offset 0byte
    unsigned v = regs.r<8>(0);

    // do a 16bit write to register at offset 2byte (four variants)
    regs[2] = 22;
    regs.r<16>(2) = 22;
    regs[2].write(22);
    regs.r<16>().write(22);

    // do an 8bit write (two variants)
    regs.r<8>(0) = 9;
    regs.r<8>(0).write(9);

    // do 16bit read-modify-write (two variants)
    regs[4].modify(0xf, 3); // clear 4 lowest bits and set them to 3
    regs.r<16>(4).modify(0xf, 3);

    // do 8bit read-modify-write
    regs.r<8>(0).modify(0xf, 3);

    // fails to compile, because of too wide access
    // (32 bit access but regs is Hw::Register_block<16>)
    unsigned long v = regs.r<32>(4)
}
```

Definition at line 329 of file [hw\\_register\\_block](#).

### 15.266.2 Member Function Documentation

#### 15.266.2.1 operator[]() [1/2]

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_↵
BITS> >>
Register L4drivers::Register_block< MAX_BITS, BLOCK >::operator[] (
    unsigned offset ) [inline]
```

Read/write access to register at offset *offset*.

## Parameters

<i>offset</i>	The offset of the register within the register file.
---------------	--

## Returns

register object allowing read and write access with width *MAX\_BITS*.

Definition at line 384 of file [hw\\_register\\_block](#).

**15.266.2.2 operator[]()** [2/2]

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> >>
Ro_register L4drivers::Register_block< MAX_BITS, BLOCK >::operator[] (
    unsigned offset ) const [inline]
```

Read only access to register at offset *offset*.

## Parameters

<i>offset</i>	The offset of the register within the register file.
---------------	--

## Returns

register object allowing read only access with width *MAX\_BITS*.

Definition at line 364 of file [hw\\_register\\_block](#).

**15.266.2.3 r()** [1/2]

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> >>
template<unsigned BITS>
Register_tmpl< BITS, Block > L4drivers::Register_block< MAX_BITS, BLOCK >::r (
    unsigned offset ) [inline]
```

Read/write access to register at offset *offset*.

## Template Parameters

<i>BITS</i>	the access width in bits for the register.
-------------	--

## Parameters

<i>offset</i>	The offset of the register within the register file.
---------------	--

**Returns**

register object allowing read and write access with width *BITS*.

Definition at line 375 of file [hw\\_register\\_block](#).

**15.266.2.4 r() [2/2]**

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> >>
template<unsigned BITS>
Ro_register_tmpl< BITS, Block > L4drivers::Register_block< MAX_BITS, BLOCK >::r (
    unsigned offset ) const [inline]
```

Read only access to register at offset *offset*.

**Template Parameters**

<i>BITS</i>	the access width in bits for the register.
-------------	--

**Parameters**

<i>offset</i>	The offset of the register within the register file.
---------------	--

**Returns**

register object allowing read only access with width *BITS*.

Definition at line 356 of file [hw\\_register\\_block](#).

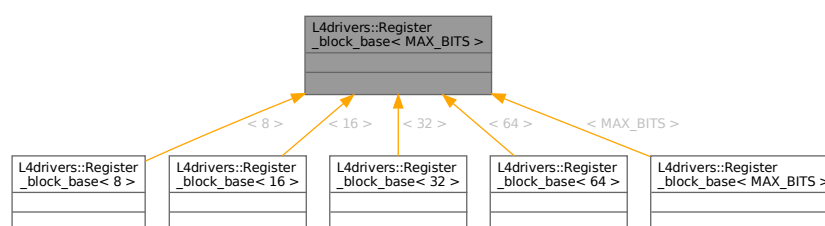
The documentation for this class was generated from the following file:

- pkg/drivers-frst/include/hw\_register\_block

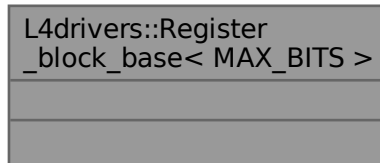
## 15.267 L4drivers::Register\_block\_base< MAX\_BITS > Struct Template Reference

Abstract register block interface.

Inheritance diagram for L4drivers::Register\_block\_base< MAX\_BITS >:



Collaboration diagram for L4drivers::Register\_block\_base< MAX\_BITS >:



### 15.267.1 Detailed Description

```
template<unsigned MAX_BITS = 32>
struct L4drivers::Register_block_base< MAX_BITS >
```

Abstract register block interface.

#### Template Parameters

<i>MAX_BITS</i>	The maximum access width for the registers.
-----------------	---

This interfaces is based on virtual `do_read_<xx>` and `do_write_<xx>` methods that have to be implemented up to the maximum access width.

Definition at line 71 of file [hw\\_register\\_block](#).

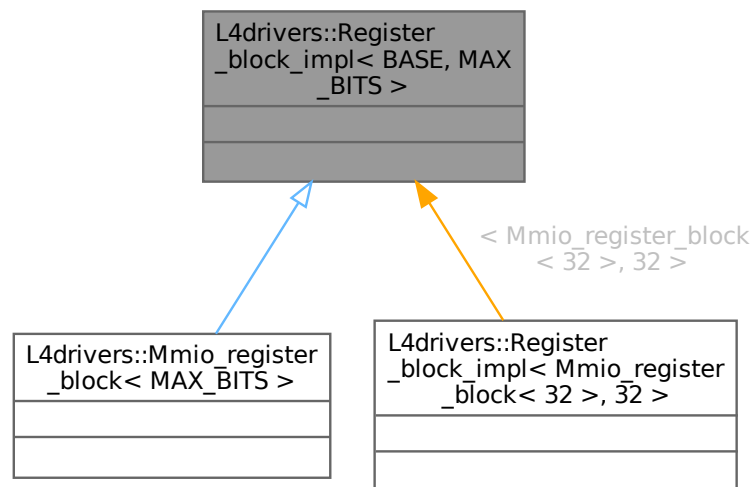
The documentation for this struct was generated from the following file:

- `pkg/drivers-frst/include/hw_register_block`

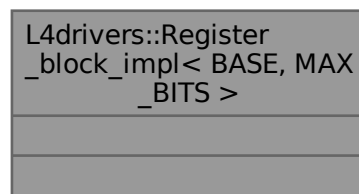
## 15.268 L4drivers::Register\_block\_impl< BASE, MAX\_BITS > Struct Template Reference

Implementation helper for register blocks.

Inheritance diagram for L4drivers::Register\_block\_impl< BASE, MAX\_BITS >:



Collaboration diagram for L4drivers::Register\_block\_impl< BASE, MAX\_BITS >:



### 15.268.1 Detailed Description

```

template<typename BASE, unsigned MAX_BITS = 32>
struct L4drivers::Register_block_impl< BASE, MAX_BITS >

```

Implementation helper for register blocks.

#### Parameters

<i>BASE</i>	The class implementing read<> and write<> template functions for accessing the registers. This class must inherit from <a href="#">Register_block_impl</a> .
<i>MAX_BITS</i>	The maximum access width for the register file. Supported values are 8, 16, 32, or 64.

This template allows easy implementation of register files by providing read<> and write<> template functions, see [Mmio\\_register\\_block](#) as an example.

Definition at line 454 of file [hw\\_register\\_block](#).

The documentation for this struct was generated from the following file:

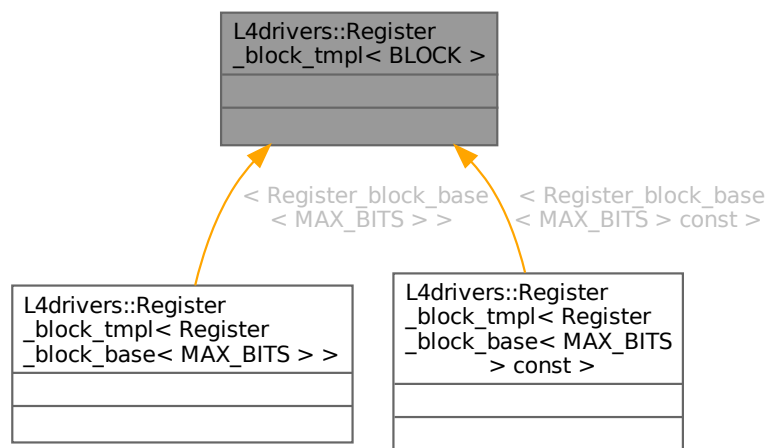
- pkg/drivers-frst/include/hw\_register\_block

## 15.269 L4drivers::Register\_block\_tmpl< BLOCK > Class Template Reference

Helper template that translates to the [Register\\_block\\_base](#) interface.

```
#include <hw_register_block>
```

Inheritance diagram for L4drivers::Register\_block\_tmpl< BLOCK >:



Collaboration diagram for L4drivers::Register\_block\_tmpl< BLOCK >:



### 15.269.1 Detailed Description

```
template<typename BLOCK>
class L4drivers::Register_block_tmpl< BLOCK >
```

Helper template that translates to the [Register\\_block\\_base](#) interface.

#### Template Parameters

<i>BLOCK</i>	The type of the <a href="#">Register_block_base</a> interface to use.
--------------	---

This helper translates `read<T>()`, `write<T>()`, `set<T>()`, `clear<T>()`, and `modify<T>()` calls to `BLOCK::do_read_<xx>` and `BLOCK::do_write_<xx>`.

Definition at line 155 of file [hw\\_register\\_block](#).

The documentation for this class was generated from the following file:

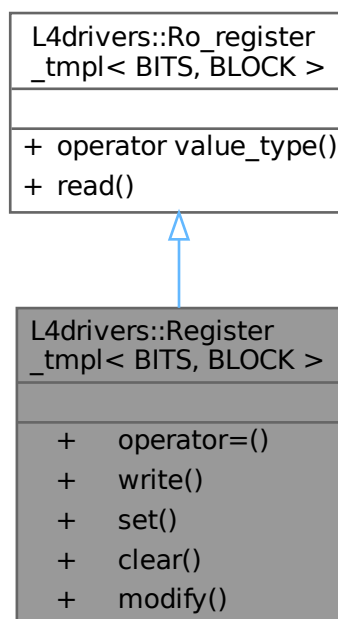
- `pkg/drivers-frst/include/hw_register_block`

## 15.270 L4drivers::Register\_tmpl< BITS, BLOCK > Class Template Reference

Single hardware register inside a [Register\\_block\\_base](#) interface.

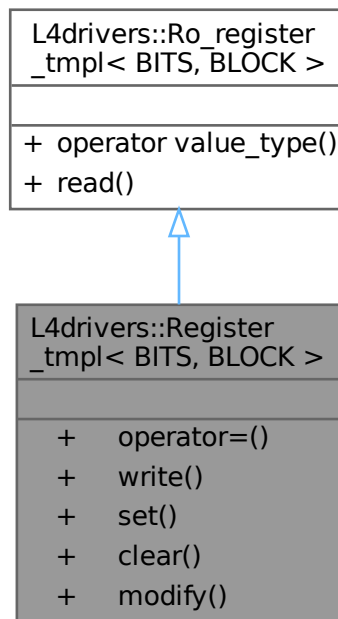
```
#include <hw_register_block>
```

Inheritance diagram for `L4drivers::Register_tmpl< BITS, BLOCK >`:





Collaboration diagram for L4drivers::Register\_tmpl< BITS, BLOCK >:



### Public Member Functions

- [Register\\_tmpl](#) & `operator=` (value\_type val)  
*write val into the hardware register.*
- void [write](#) (value\_type val)  
*write val into the hardware register.*
- value\_type [set](#) (value\_type set\_bits)  
*set bits in set\_bits in the hardware register.*
- value\_type [clear](#) (value\_type clear\_bits)  
*clears bits in clear\_bits in the hardware register.*
- value\_type [modify](#) (value\_type clear\_bits, value\_type set\_bits)  
*clears bits in clear\_bits and sets bits in set\_bits in the hardware register.*

### Public Member Functions inherited from [L4drivers::Ro\\_register\\_tmpl< BITS, BLOCK >](#)

- [operator value\\_type](#) () const  
*read the value from the hardware register.*
- value\_type [read](#) () const  
*read the value from the hardware register.*

#### 15.270.1 Detailed Description

```
template<unsigned BITS, typename BLOCK>
class L4drivers::Register_tmpl< BITS, BLOCK >
```

Single hardware register inside a [Register\\_block\\_base](#) interface.

## Template Parameters

<i>BITS</i>	The access width for the register in bits.
<i>BLOCK</i>	the type of the <a href="#">Register_block_base</a> interface.

## Note

Objects of this type must be used only in temporary contexts not in global, class, or object scope.

Definition at line 236 of file [hw\\_register\\_block](#).

## 15.270.2 Member Function Documentation

### 15.270.2.1 clear()

```
template<unsigned BITS, typename BLOCK >
value_type L4drivers::Register\_tmpl< BITS, BLOCK >::clear (
    value_type clear_bits ) [inline]
```

clears bits in *clear\_bits* in the hardware register.

## Parameters

<i>clear_bits</i>	bits to be cleared within the hardware register.
-------------------	--

This is a read-modify-write function that does a logical and of the old value from the register with the negated value of *clear\_bits*.

```
unsigned old_value = read();
write(old_value & ~clear_bits);
```

Definition at line 289 of file [hw\\_register\\_block](#).

### 15.270.2.2 modify()

```
template<unsigned BITS, typename BLOCK >
value_type L4drivers::Register\_tmpl< BITS, BLOCK >::modify (
    value_type clear_bits,
    value_type set_bits ) [inline]
```

clears bits in *clear\_bits* and sets bits in *set\_bits* in the hardware register.

## Parameters

<i>clear_bits</i>	bits to be cleared within the hardware register.
<i>set_bits</i>	bits to set in the hardware register.

This is a read-modify-write function that first does a logical and of the old value from the register with the negated value of *clear\_bits* and then does a logical or with *set\_bits*.

```
unsigned old_value = read();
```

```
write((old_value & ~clear_bits) | set_bits);
```

Definition at line 307 of file [hw\\_register\\_block](#).

### 15.270.2.3 operator=()

```
template<unsigned BITS, typename BLOCK >
Register_tmpl & L4drivers::Register_tmpl< BITS, BLOCK >::operator= (
    value_type val ) [inline]
```

write *val* into the hardware register.

#### Parameters

<i>val</i>	the value to write into the hardware register.
------------	--

Definition at line 251 of file [hw\\_register\\_block](#).

### 15.270.2.4 set()

```
template<unsigned BITS, typename BLOCK >
value_type L4drivers::Register_tmpl< BITS, BLOCK >::set (
    value_type set_bits ) [inline]
```

set bits in *set\_bits* in the hardware register.

#### Parameters

<i>set_bits</i>	bits to be set within the hardware register.
-----------------	--

This is a read-modify-write function that does a logical or of the old value from the register with *set\_bits*.

```
unsigned old_value = read();
write(old_value | set_bits);
```

Definition at line 273 of file [hw\\_register\\_block](#).

### 15.270.2.5 write()

```
template<unsigned BITS, typename BLOCK >
void L4drivers::Register_tmpl< BITS, BLOCK >::write (
    value_type val ) [inline]
```

write *val* into the hardware register.

#### Parameters

<i>val</i>	the value to write into the hardware register.
------------	--

Definition at line 258 of file [hw\\_register\\_block](#).

The documentation for this class was generated from the following file:

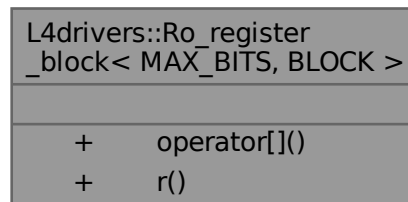
- pkg/drivers-frst/include/hw\_register\_block

## 15.271 L4drivers::Ro\_register\_block< MAX\_BITS, BLOCK > Class Template Reference

Handles a reference to a read only register block of the given maximum access width.

```
#include <hw_register_block>
```

Collaboration diagram for L4drivers::Ro\_register\_block< MAX\_BITS, BLOCK >:



### Public Member Functions

- Ro\_register [operator\[\]](#) (unsigned offset) const  
*Read only access to register at offset offset.*
- template<unsigned BITS>  
[Ro\\_register\\_tmpl](#)< BITS, Block > [r](#) (unsigned offset) const  
*Read only access to register at offset offset.*

### 15.271.1 Detailed Description

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> const >>
class L4drivers::Ro_register_block< MAX_BITS, BLOCK >
```

Handles a reference to a read only register block of the given maximum access width.

#### Template Parameters

<i>MAX_BITS</i>	Maximum access width for the registers in this block.
<i>BLOCK</i>	Type implementing the register accesses (read<>()),

Provides read only access to registers in this block via `r<WIDTH>()` and `operator[]()`.

Definition at line 403 of file [hw\\_register\\_block](#).

## 15.271.2 Member Function Documentation

### 15.271.2.1 operator[]()

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> const >>
Ro_register L4drivers::Ro_register_block< MAX_BITS, BLOCK >::operator[] (
    unsigned offset ) const [inline]
```

Read only access to register at offset *offset*.

#### Parameters

<i>offset</i>	The offset of the register within the register file.
---------------	--

#### Returns

register object allowing read only access with width *MAX\_BITS*.

Definition at line 425 of file [hw\\_register\\_block](#).

### 15.271.2.2 r()

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> const >>
template<unsigned BITS>
Ro_register_tmpl< BITS, Block > L4drivers::Ro_register_block< MAX_BITS, BLOCK >::r (
    unsigned offset ) const [inline]
```

Read only access to register at offset *offset*.

#### Template Parameters

<i>BITS</i>	the access width in bits for the register.
-------------	--

#### Parameters

<i>offset</i>	The offset of the register within the register file.
---------------	--

#### Returns

register object allowing read only access with width *BITS*.

Definition at line 435 of file [hw\\_register\\_block](#).

The documentation for this class was generated from the following file:

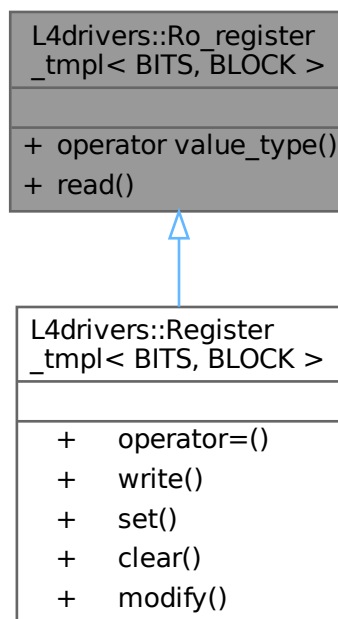
- pkg/drivers-frst/include/hw\_register\_block

## 15.272 L4drivers::Ro\_register\_tmpl< BITS, BLOCK > Class Template Reference

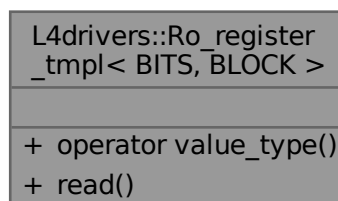
Single read only register inside a [Register\\_block\\_base](#) interface.

```
#include <hw_register_block>
```

Inheritance diagram for L4drivers::Ro\_register\_tmpl< BITS, BLOCK >:



Collaboration diagram for L4drivers::Ro\_register\_tmpl< BITS, BLOCK >:



## Public Member Functions

- [operator value\\_type](#) () const  
*read the value from the hardware register.*
- [value\\_type read](#) () const  
*read the value from the hardware register.*

### 15.272.1 Detailed Description

**template<unsigned BITS, typename BLOCK>**  
**class L4drivers::Ro\_register\_tmpl< BITS, BLOCK >**

Single read only register inside a [Register\\_block\\_base](#) interface.

#### Template Parameters

<i>BITS</i>	The access with of the register in bits.
<i>BLOCK</i>	The type for the <a href="#">Register_block_base</a> interface.

#### Note

Objects of this type must be used only in temporary contexts not in global, class, or object scope.

Allows simple read only access to a hardware register.

Definition at line 200 of file [hw\\_register\\_block](#).

### 15.272.2 Member Function Documentation

#### 15.272.2.1 operator value\_type()

```
template<unsigned BITS, typename BLOCK >  
L4drivers::Ro\_register\_tmpl< BITS, BLOCK >::operator value_type ( ) const [inline]
```

read the value from the hardware register.

#### Returns

value read from the hardware register.

Definition at line 216 of file [hw\\_register\\_block](#).

### 15.272.2.2 read()

```
template<unsigned BITS, typename BLOCK >
value_type L4drivers::Ro_register_tmpl< BITS, BLOCK >::read ( ) const [inline]
```

read the value from the hardware register.

#### Returns

value from the hardware register.

Definition at line 223 of file [hw\\_register\\_block](#).

The documentation for this class was generated from the following file:

- [pkg/drivers-frst/include/hw\\_register\\_block](#)

## 15.273 L4Re::Cap\_alloc Class Reference

Capability allocator interface.

```
#include <cap_alloc>
```

Inherited by `L4Re::Cap_alloc_t< ALLOC >`.

Collaboration diagram for `L4Re::Cap_alloc`:

L4Re::Cap_alloc
<ul style="list-style-type: none"> <li>+ alloc()</li> <li>+ alloc()</li> <li>+ free()</li> <li>+ ~Cap_alloc()</li> <li>+ get_cap_alloc()</li> </ul>

### Public Member Functions

- virtual `L4::Cap< void > alloc ()` noexcept=0  
*Allocate a capability.*
- template<typename T >  
`L4::Cap< T > alloc ()` noexcept  
*Allocate a capability.*
- virtual void `free (L4::Cap< void > cap, l4_cap_idx_t task=L4_INVALID_CAP, unsigned unmap_↵ flags=L4_FP_ALL_SPACES)` noexcept=0  
*Free a capability.*
- virtual `~Cap_alloc ()=0`  
*Destructor.*



## Static Public Member Functions

- `template<typename CAP_ALLOC >`  
`static L4Re::Cap_alloc * get\_cap\_alloc (CAP_ALLOC &ca)`  
*Construct an instance of a capability allocator.*

## 15.273.1 Detailed Description

Capability allocator interface.

Definition at line 41 of file [cap\\_alloc](#).

## 15.273.2 Member Function Documentation

### 15.273.2.1 `alloc()` [1/2]

```
template<typename T >
L4::Cap< T > L4Re::Cap_alloc::alloc ( ) [inline], [noexcept]
```

Allocate a capability.

#### Returns

Capability of type T

Definition at line 64 of file [cap\\_alloc](#).

References [alloc\(\)](#).

Here is the call graph for this function:



### 15.273.2.2 alloc() [2/2]

```
virtual L4::Cap< void > L4Re::Cap_alloc::alloc ( ) [pure virtual], [noexcept]
```

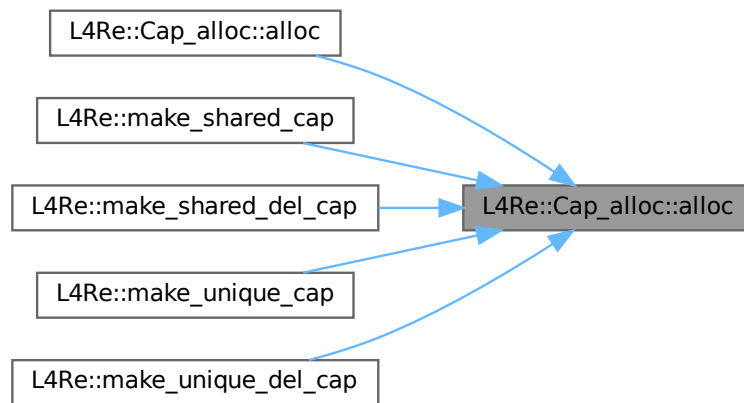
Allocate a capability.

#### Returns

Capability of type void

Referenced by [alloc\(\)](#), [L4Re::make\\_shared\\_cap\(\)](#), [L4Re::make\\_shared\\_del\\_cap\(\)](#), [L4Re::make\\_unique\\_cap\(\)](#), and [L4Re::make\\_unique\\_del\\_cap\(\)](#).

Here is the caller graph for this function:



### 15.273.2.3 free()

```
virtual void L4Re::Cap_alloc::free (
    L4::Cap< void > cap,
    l4_cap_idx_t task = L4_INVALID_CAP,
    unsigned unmap_flags = L4_FP_ALL_SPACES ) [pure virtual], [noexcept]
```

Free a capability.

#### Parameters

<i>cap</i>	Capability to free.
<i>task</i>	If set, task to unmap the capability from.
<i>unmap_flags</i>	Flags for unmap, see <code>l4_unmap_flags_t</code> .

#### 15.273.2.4 get\_cap\_alloc()

```
template<typename CAP_ALLOC >
static L4Re::Cap\_alloc * L4Re::Cap_alloc::get_cap_alloc (
    CAP_ALLOC & ca ) [inline], [static]
```

Construct an instance of a capability allocator.

##### Parameters

<i>ca</i>	Capability allocator
-----------	----------------------

##### Returns

Instance of a capability allocator.

Definition at line 90 of file [cap\\_alloc](#).

References [L4\\_FP\\_ALL\\_SPACES](#), and [L4\\_INVALID\\_CAP](#).

The documentation for this class was generated from the following file:

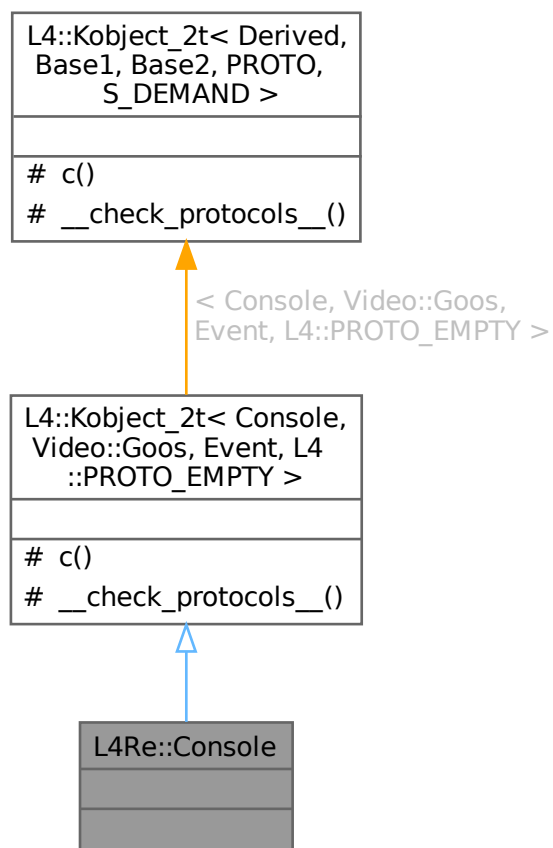
- [l4/re/cap\\_alloc](#)

## 15.274 L4Re::Console Class Reference

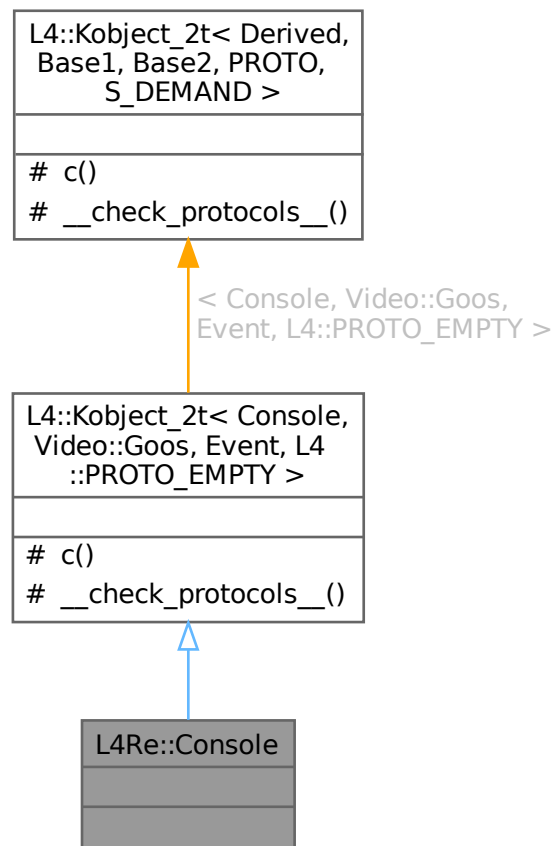
[Console](#) class.

```
#include <console>
```

Inheritance diagram for L4Re::Console:



Collaboration diagram for L4Re::Console:



### Additional Inherited Members

### Protected Types inherited from

**L4::Kobject\_2t< Console, Video::Goos, Event, L4::PROTO\_EMPTY >**

- typedef Console [Class](#)  
The target interface type (inheriting from [Kobject\\_t](#))
- typedef Typeid::Iface< PROTO, Console > [\\_\\_lfacer](#)  
The interface description for the derived class.
- typedef Typeid::Merge\_list< Typeid::Ifacer\_list< [\\_\\_lfacer](#) >, Typeid::Merge\_list< typename Base1::\_\_lfacer\_list, typename Base2::\_\_lfacer\_list > > [\\_\\_lfacer\\_list](#)  
The list of all RPC interfaces provided directly or through inheritance.

### Protected Member Functions inherited from

**L4::Kobject\_2t< Console, Video::Goos, Event, L4::PROTO\_EMPTY >**

- [L4::Cap< Class > c\(\)](#) const noexcept  
Get the capability to ourselves.

**Static Protected Member Functions inherited from****L4::Kobject\_2t< Console, Video::Goos, Event, L4::PROTO\_EMPTY >**

- static void `__check_protocols__()` noexcept  
*Helper to check for protocol conflicts.*

**15.274.1 Detailed Description**

`Console` class.

Definition at line 39 of file `console`.

The documentation for this class was generated from the following file:

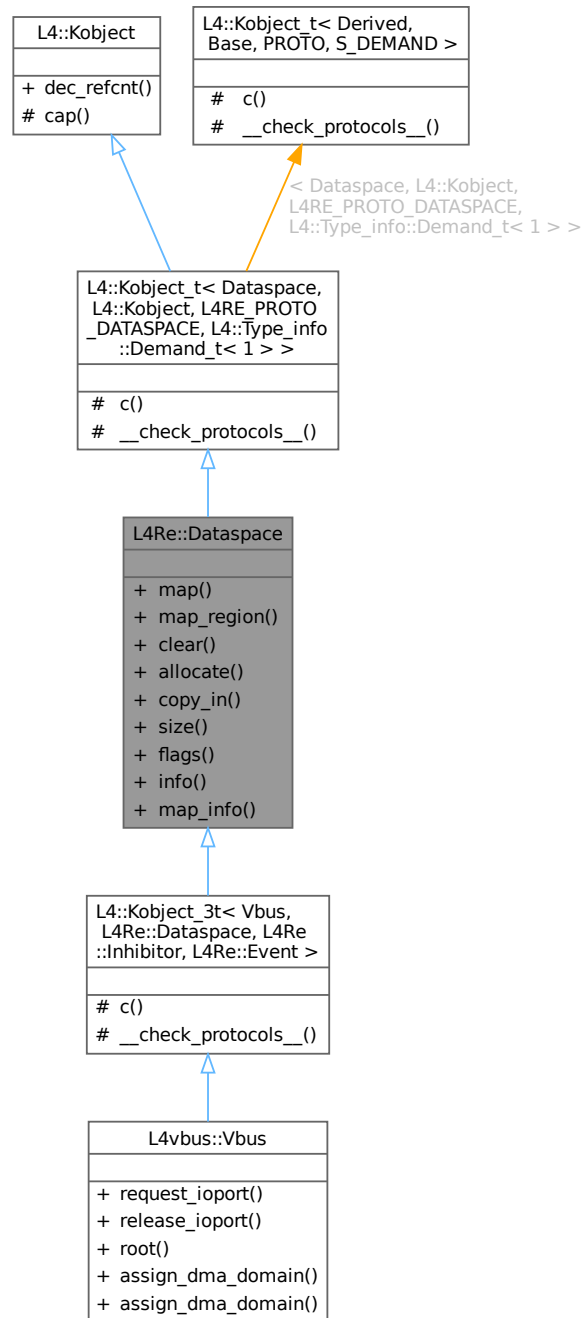
- `l4/re/console`

**15.275 L4Re::Dataspace Class Reference**

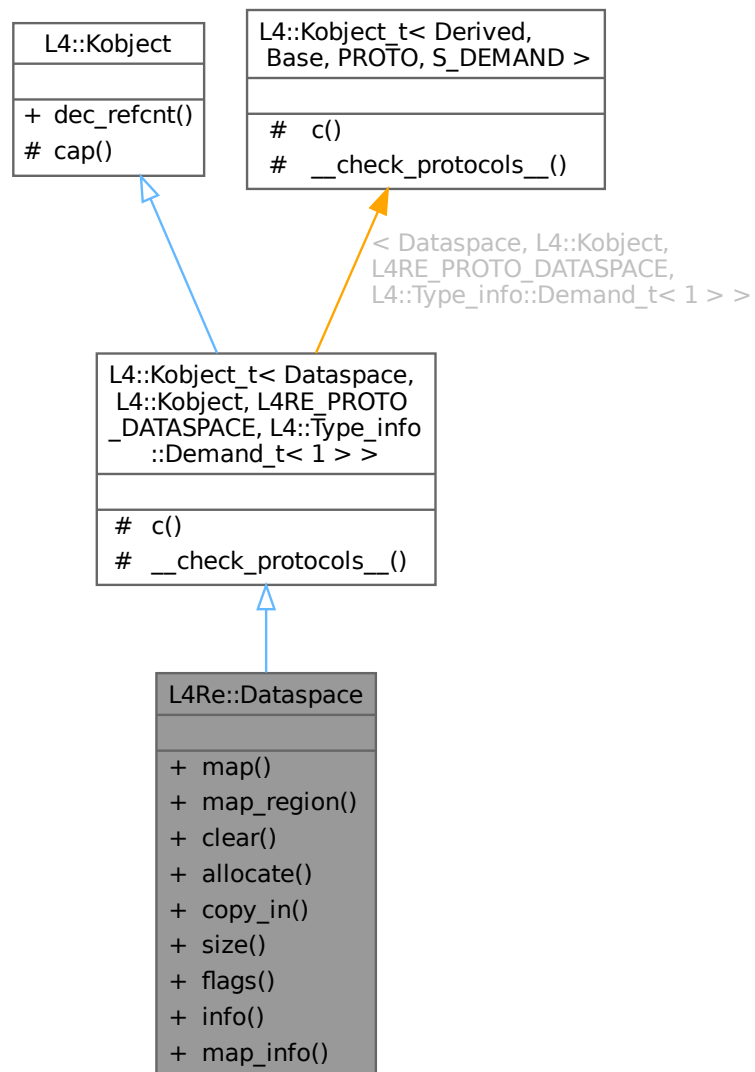
Interface for memory-like objects.

```
#include <dataspace>
```

Inheritance diagram for L4Re::Dataspace:



Collaboration diagram for L4Re::Dataspace:



## Data Structures

- struct [F](#)  
*Dataspace flags definitions.*
- struct [Stats](#)  
*Information about the dataspace.*

## Public Member Functions

- long [map](#) (Offset offset, Flags [flags](#), Map\_addr local\_addr, Map\_addr min\_addr, Map\_addr max\_addr, [L4::Cap](#)< [L4::Task](#) > dst=[L4::Cap](#)< [L4::Task](#) >::Invalid) const noexcept  
*Request a flex-page mapping from the dataspace.*



- long [map\\_region](#) (Offset offset, Flags [flags](#), Map\_addr min\_addr, Map\_addr max\_addr, [L4::Cap](#)< [L4::Task](#) > dst=[L4::Cap](#)< [L4::Task](#) >::Invalid) const noexcept  
*Map a part of a dataspace into a local memory area.*
- long [clear](#) (Offset offset, Size [size](#))  
*Clear parts of a dataspace.*
- long [allocate](#) (Offset offset, Size [size](#))  
*Allocate a range in the dataspace.*
- long [copy\\_in](#) (Offset dst\_offs, [L4::lpc::Cap](#)< [Dataspace](#) > src, Offset src\_offs, Size [size](#))  
*Copy contents from another dataspace.*
- Size [size](#) () const noexcept  
*Get size of a dataspace.*
- Flags [flags](#) () const noexcept  
*Get flags of the dataspace.*
- long [info](#) ([Stats](#) \*stats)  
*Get information on the dataspace.*
- long [map\\_info](#) ([l4\\_addr\\_t](#) \*, [l4\\_addr\\_t](#) \*)  
*Get mapping range of dataspace.*

### Public Member Functions inherited from [L4::Kobject](#)

- [l4\\_msgtag\\_t dec\\_refcnt](#) ([l4\\_mword\\_t](#) diff, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)())  
*Decrement the in kernel reference counter for the object.*

### Additional Inherited Members

### Protected Types inherited from

[L4::Kobject\\_t](#)< [Dataspace](#), [L4::Kobject](#), [L4RE\\_PROTO\\_DATASPACE](#), [L4::Type\\_info::Demand\\_t](#)< 1 > >

- typedef Dataspace **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< [PROTO](#), Dataspace > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_Iface](#) >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Dataspace](#), [L4::Kobject](#), [L4RE\\_PROTO\\_DATASPACE](#), [L4::Type\\_info::Demand\\_t](#)< 1 > >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t cap](#) () const noexcept  
*Return capability selector.*

## Static Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Dataspace](#), [L4::Kobject](#), [L4RE\\_PROTO\\_DATASPACE](#), [L4::Type\\_info::Demand\\_t](#)< 1 > >

- static void `__check_protocols__()` noexcept

*Helper to check for protocol conflicts.*

### 15.275.1 Detailed Description

Interface for memory-like objects.

Dataspaces are a central abstraction provided by [L4Re](#). A dataspace is an abstraction for any thing that is available via usual memory access instructions. A dataspace can be a file, as well as the memory-mapped registers of a device, or anonymous memory, such as a heap.

The dataspace interface defines a set of methods that allow any kind of dataspace to be attached (mapped) to the virtual address space of an [L4](#) task and then be accessed via memory-access instructions. The [L4Re::Rm](#) interface can be used to attach a dataspace to a virtual address space of a task paged by a certain instance of a region map.

#### Include File

```
#include <l4/re/dataspace>
```

#### Examples

[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), and [examples/libs/l4re/c++/shared\\_](#)

Definition at line 61 of file [dataspace](#).

### 15.275.2 Member Function Documentation

#### 15.275.2.1 allocate()

```
long L4Re::Dataspace::allocate (
    Offset offset,
    Size size )
```

Allocate a range in the dataspace.

#### Parameters

<i>offset</i>	Offset in the dataspace, in bytes.
<i>size</i>	Size of the range, in bytes.

#### Return values

<i>L4_EOK</i>	Success
<i>-L4_ERANGE</i>	Given range is outside the dataspace. (A dataspace provider may also silently ignore areas outside the dataspace.)
<i>-L4_ENOMEM</i>	Not enough memory available.
<0	IPC errors

On success, at least the given range is guaranteed to be allocated. The dataspace manager may also allocate more memory due to page granularity.

The memory is allocated with the same rights as the dataspace capability.

### 15.275.2.2 clear()

```
long L4Re::Dataspace::clear (
    Offset offset,
    Size size )
```

Clear parts of a dataspace.

#### Parameters

<i>offset</i>	Offset within dataspace (in bytes).
<i>size</i>	Size of region to clear (in bytes).

#### Return values

$\geq 0$	Success.
<code>-L4_ERANGE</code>	Given range is outside the dataspace. (A dataspace provider may also silently ignore areas outside the dataspace.)
<code>-L4_EACCESS</code>	No <code>L4_CAP_FPAGE_W</code> right on dataspace capability.
$< 0$	IPC errors

Zeros out the memory. Depending on the type of memory the memory could also be deallocated and replaced by a shared zero-page.

### 15.275.2.3 copy\_in()

```
long L4Re::Dataspace::copy_in (
    Offset dst_offs,
    L4::Ipc::Cap< Dataspace > src,
    Offset src_offs,
    Size size )
```

Copy contents from another dataspace.

#### Parameters

<i>dst_offs</i>	Offset in destination dataspace.
<i>src</i>	Source dataspace to copy from.
<i>src_offs</i>	Offset in the source dataspace.
<i>size</i>	Size to copy (in bytes).

#### Return values

<code>L4_EOK</code>	Success
---------------------	---------

## Return values

<code>-L4_EACCESS</code>	No <a href="#">L4_CAP_FPAGE_W</a> right on the destination dataspace.
<code>-L4_EINVAL</code>	Invalid parameter supplied.
<code>&lt;0</code>	IPC errors

The copy operation may use copy-on-write mechanisms. The operation may also fail if both dataspaces are not from the same dataspace manager or the dataspace managers do not cooperate.

**15.275.2.4 flags()**

```
Dataspace::Flags L4Re::Dataspace::flags ( ) const [noexcept]
```

Get flags of the dataspace.

## Return values

<code>&gt;=0</code>	Flags of the dataspace
<code>&lt;0</code>	IPC errors

## See also

[L4Re::Dataspace::F::Flags](#)

Definition at line 122 of file [dataspace\\_impl.h](#).

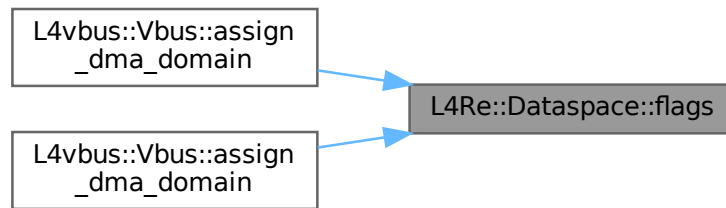
References [L4Re::Dataspace::Stats::flags](#), and [info\(\)](#).

Referenced by [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#), and [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.275.2.5 info()

```
long L4Re::Dataspace::info (
    Stats * stats )
```

Get information on the dataspace.

#### Parameters

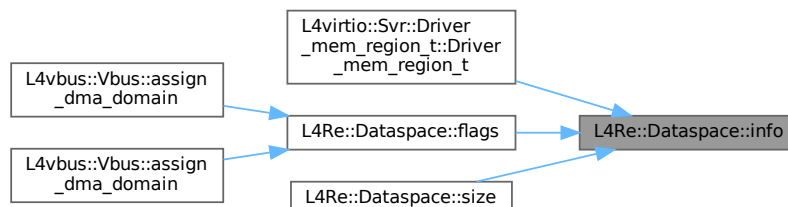
out	stats	Dataspace information
-----	-------	-----------------------

#### Return values

0	Success
<0	IPC errors

Referenced by [L4virtio::Svr::Driver\\_mem\\_region\\_t< DATA >::Driver\\_mem\\_region\\_t\(\)](#), [flags\(\)](#), and [size\(\)](#).

Here is the caller graph for this function:



### 15.275.2.6 map()

```
long L4Re::Dataspace::map (
    Dataspace::Offset offset,
    Dataspace::Flags flags,
    Dataspace::Map_addr local_addr,
    Dataspace::Map_addr min_addr,
    Dataspace::Map_addr max_addr,
    L4::Cap< L4::Task > dst = L4::Cap<L4::Task>::Invalid ) const [noexcept]
```

Request a flex-page mapping from the dataspace.

#### Parameters

<i>offset</i>	Offset to start within dataspace
<i>flags</i>	<a href="#">Dataspace</a> flags, see <a href="#">L4Re::Dataspace::F::Flags</a> .
<i>local_addr</i>	Local address to map to.
<i>min_addr</i>	Defines start of receive window. (Rounded down to page size.)
<i>max_addr</i>	Defines end of receive window. (Rounded up to page size.)
<i>dst</i>	Optional destination task of the mapping. If invalid, the callers task is implicitly the destination.

#### Return values

<i>L4_EOK</i>	Success
<i>-L4_ERANGE</i>	Invalid offset.
<i>-L4_EPERM</i>	Insufficient permission to map with requested rights.
<i>&lt;0</i>	IPC errors

The map call will attempt to map the largest possible flexpage that covers the given local address and still fits into the region defined by *min\_addr* and *max\_addr*. If the given region is invalid or does not overlap the local address, the smallest valid page size is used.

Definition at line 96 of file [dataspace\\_impl.h](#).

References [L4\\_LOG2\\_PAGESIZE](#).

### 15.275.2.7 map\_info()

```
long L4Re::Dataspace::map_info (
    l4_addr_t * ,
    l4_addr_t * ) [inline]
```

Get mapping range of dataspace.

In case of a MMU-less system, the dataspace must be mapped at the correct address in the task because virtual and physical address must match. This method returns the start and end address of the physically contiguous buffer backing the dataspace.

On MMU-enabled system any page aligned address is permissible. On such systems the method is just a stub.

## Parameters

out	<i>start_addr</i>	Start address of dataspace.
out	<i>end_addr</i>	End address (inclusive) of dataspace.

## Return values

$>0$	Start/end address have been set and need to be obeyed.
$0$	No constraint of mapping address.
$-L4\_EPMR$	Cannot infer mapping address. <a href="#">Dataspace</a> not mappable.
$<0$	IPC errors.

Definition at line 315 of file [dataspace](#).

## 15.275.2.8 map\_region()

```
long L4Re::Dataspace::map_region (
    Dataspace::Offset offset,
    Dataspace::Flags flags,
    Dataspace::Map_addr min_addr,
    Dataspace::Map_addr max_addr,
    L4::Cap< L4::Task > dst = L4::Cap<L4::Task>::Invalid ) const [noexcept]
```

Map a part of a dataspace into a local memory area.

## Parameters

<i>offset</i>	Offset to start within dataspace.
<i>flags</i>	<a href="#">Dataspace</a> flags, see <a href="#">L4Re::Dataspace::F::Flags</a> .
<i>min_addr</i>	(Inclusive) start of the receive area.
<i>max_addr</i>	(Exclusive) end of receive area.
<i>dst</i>	Optional destination task of the mapping. If invalid, the callers task is implicitly the destination.

## Return values

<i>L4_EOK</i>	Success
$-L4\_ERANGE$	Invalid offset or receive area larger than the dataspace.
$-L4\_EPMR$	Insufficient permission to map with requested rights.
$<0$	IPC errors

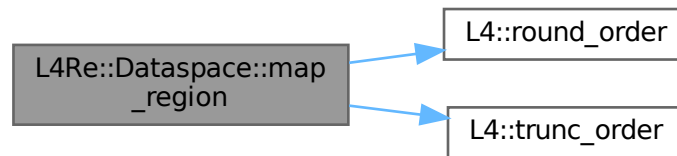
This is a convenience function which maps flex-pages consecutively into the given memory area in the local task. The area is expected to be filled completely. If the dataspace is not large enough to provide the mappings for the entire size of the area, then an error is returned. Mappings may or may not have been already established at that point.

*offset* and *min\_addr* are rounded down to the next `L4_PAGESIZE` boundary when necessary. *max\_addr* is rounded up to the page boundary. If the resulting maximum address is less or equal than the minimum address, then the function is a noop.

Definition at line 56 of file [dataspace\\_impl.h](#).

References [L4\\_LOG2\\_PAGESIZE](#), [L4\\_UNLIKELY](#), [L4::round\\_order\(\)](#), and [L4::trunc\\_order\(\)](#).

Here is the call graph for this function:



#### 15.275.2.9 size()

```
Dataspace::Size L4Re::Dataspace::size ( ) const [noexcept]
```

Get size of a dataspace.

##### Returns

Size of the dataspace in bytes.

Definition at line 112 of file [dataspace\\_impl.h](#).

References [info\(\)](#), and [L4Re::Dataspace::Stats::size](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- [l4/re/dataspace](#)
- [l4/re/impl/dataspace\\_impl.h](#)

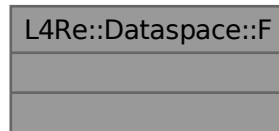


## 15.276 L4Re::Dataspace::F Struct Reference

[Dataspace](#) flags definitions.

```
#include <dataspace>
```

Collaboration diagram for L4Re::Dataspace::F:



### Public Types

- enum { [Caching\\_shift](#) = 4 }
- enum [Flags](#) {  
    [R](#) = L4\_FPAGE\_RO , [Ro](#) = L4\_FPAGE\_RO , [RW](#) = L4\_FPAGE\_RW , [W](#) = L4\_FPAGE\_W ,  
    [X](#) = L4\_FPAGE\_X , [RX](#) = L4\_FPAGE\_RX , [RWX](#) = L4\_FPAGE\_RWX , [Rights\\_mask](#) = 0x0f ,  
    [Normal](#) = 0x00 , [Cacheable](#) = Normal , [Bufferable](#) = 0x10 , [Uncacheable](#) = 0x20 ,  
    [Caching\\_mask](#) = 0x30 }  
    *Flags for map operations.*

### 15.276.1 Detailed Description

[Dataspace](#) flags definitions.

Definition at line 68 of file [dataspace](#).

### 15.276.2 Member Enumeration Documentation

#### 15.276.2.1 anonymous enum

anonymous enum

##### Enumerator

Caching_shift	shift value for caching flags
---------------	-------------------------------

Definition at line 70 of file [dataspace](#).

### 15.276.2.2 Flags

```
enum L4Re::Dataspace::F::Flags
```

Flags for map operations.

A dataspace implementation must check the requested flags during the map and other operations against the dataspace rights.

#### Enumerator

R	Request read-only mapping.
Ro	Request read-only mapping.
RW	Request read-write mapping.
W	Request write-only mapping.
X	Request execute-only mapping.
RX	Request read-execute mapping.
RWX	Request read-write-execute mapping.
Rights_mask	All rights bits available for mappings.
Normal	Request normal memory mapping.
Cacheable	Request normal memory mapping.
Bufferable	Request bufferable (write buffered) mappings.
Uncacheable	Request uncacheable memory mappings.
Caching_mask	Mask for caching flags.

Definition at line 81 of file [dataspace](#).

The documentation for this struct was generated from the following file:

- [l4/re/dataspace](#)

## 15.277 L4Re::Dataspace::Stats Struct Reference

Information about the dataspace.

```
#include <dataspace>
```

Collaboration diagram for L4Re::Dataspace::Stats:

L4Re::Dataspace::Stats	
+	size
+	flags

**Data Fields**

- Size **size**  
*size*
- Flags **flags**  
*flags*

**15.277.1 Detailed Description**

Information about the dataspace.

Definition at line 136 of file [dataspace](#).

The documentation for this struct was generated from the following file:

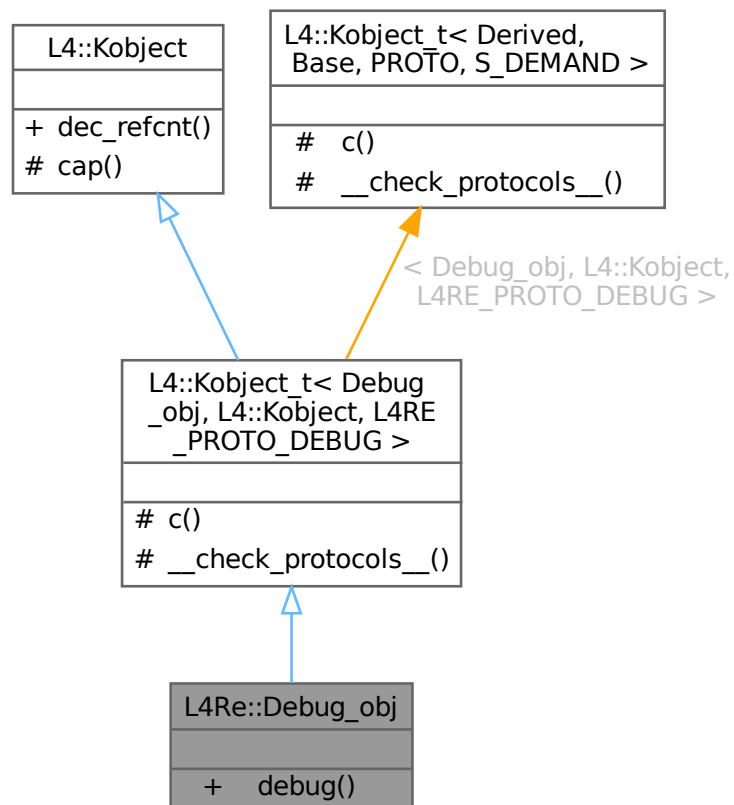
- [l4/re/dataspace](#)

**15.278 L4Re::Debug\_obj Class Reference**

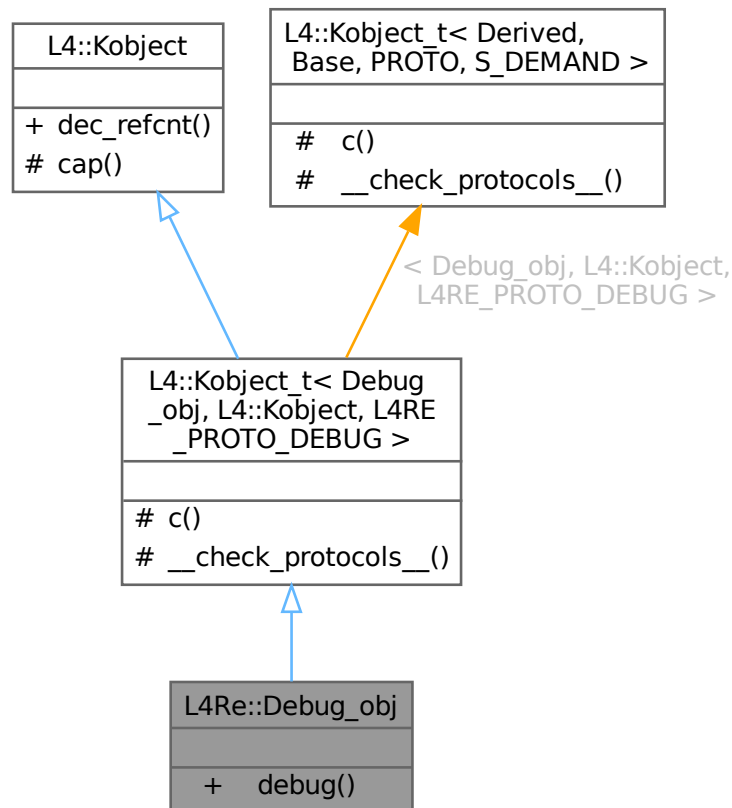
Debug interface.

```
#include <debug>
```

Inheritance diagram for L4Re::Debug\_obj:



Collaboration diagram for L4Re::Debug\_obj:



### Public Member Functions

- long [debug](#) (unsigned long function)  
*Debug call.*

### Public Member Functions inherited from [L4::Kobject](#)

- [l4\\_msgtag\\_t dec\\_refcnt](#) ([l4\\_mword\\_t](#) diff, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#))  
*Decrement the in kernel reference counter for the object.*

### Additional Inherited Members

### Protected Types inherited from

[L4::Kobject\\_t< Debug\\_obj, L4::Kobject, L4RE\\_PROTO\\_DEBUG >](#)

- typedef Debug\_obj **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, Debug\_obj > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_Iface](#) >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from [L4::Kobject\\_t< Debug\\_obj, L4::Kobject, L4RE\\_PROTO\\_DEBUG >](#)

- [L4::Cap< Class > c\(\)](#) const noexcept  
*Get the capability to ourselves.*

## Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t cap\(\)](#) const noexcept  
*Return capability selector.*

## Static Protected Member Functions inherited from [L4::Kobject\\_t< Debug\\_obj, L4::Kobject, L4RE\\_PROTO\\_DEBUG >](#)

- static void [\\_\\_check\\_protocols\\_\\_\(\)](#) noexcept  
*Helper to check for protocol conflicts.*

### 15.278.1 Detailed Description

Debug interface.

See also

[Debugging API](#).

Definition at line 51 of file [debug](#).

### 15.278.2 Member Function Documentation

#### 15.278.2.1 debug()

```
long L4Re::Debug_obj::debug (
    unsigned long function )
```

Debug call.

Parameters

<i>function</i>	Function to call.
-----------------	-------------------

Returns

- L4\_EOK
- IPC errors

An object can provide a number of debug functions, each identified by some integer. This method is used to fan out to these functions from a common entry point.

The documentation for this class was generated from the following file:

- [l4/re/debug](#)

## 15.279 L4Re::Default\_event\_payload Struct Reference

Default event stream payload.

```
#include <event>
```

Collaboration diagram for L4Re::Default\_event\_payload:

L4Re::Default_event_payload
+ type
+ code
+ value
+ stream_id

### Data Fields

- unsigned short **type**  
*Type of event.*
- unsigned short **code**  
*Code of event.*
- int **value**  
*Value of event.*
- [l4\\_umword\\_t](#) **stream\_id**  
*Stream ID.*

### 15.279.1 Detailed Description

Default event stream payload.

Definition at line [242](#) of file [event](#).

The documentation for this struct was generated from the following file:

- [l4/re/event](#)

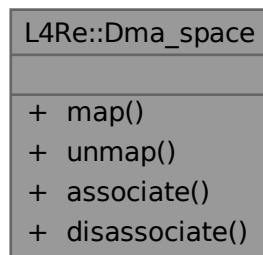
## 15.280 L4Re::Dma\_space Class Reference

Managed DMA Address Space.

```
#include <dma_space>
```

Inherits L4::Kobject\_0t< Derived, PROTO, S\_DEMAND >.

Collaboration diagram for L4Re::Dma\_space:



### Public Types

- enum [Direction](#) { [Bidirectional](#) , [To\\_device](#) , [From\\_device](#) , [None](#) }  
*Direction of the DMA transfers.*
- enum [Attribute](#) { [No\\_sync](#) }  
*Attributes used for the memory region during the transfer.*
- enum [Space\\_attrib](#) { [Coherent](#) , [Phys\\_space](#) }  
*Attributes assigned to the DMA space when associated with a specific device.*
- typedef [l4\\_uint64\\_t](#) **Dma\_addr**  
*Data type for DMA addresses.*
- typedef [L4::Types::Flags](#)< [Attribute](#) > **Attributes**  
*Attributes for DMA mappings.*
- typedef [L4::Types::Flags](#)< [Space\\_attrib](#) > **Space\_attribs**  
*Attributes used when configuring the DMA space.*

### Public Member Functions

- long [map](#) ([L4::lpc::Cap](#)< [L4Re::Dataspace](#) > src, [L4Re::Dataspace::Offset](#) offset, [L4::lpc::ln\\_out](#)< [l4\\_size\\_t](#) \* > size, [Attributes](#) attrs, [Direction](#) dir, [Dma\\_addr](#) \*dma\_addr)  
*Map the given part of this data space into the DMA address space.*
- long [unmap](#) ([Dma\\_addr](#) dma\_addr, [l4\\_size\\_t](#) size, [Attributes](#) attrs, [Direction](#) dir)  
*Unmap the given part of this data space from the DMA address space.*
- long [associate](#) ([L4::lpc::Opt](#)< [L4::lpc::Cap](#)< [L4::Task](#) > > dma\_task, [Space\\_attribs](#) attr)  
*Associate a (kernel) DMA space for a device to this Dma\_space.*
- long [disassociate](#) ()  
*Disassociate the (kernel) DMA space from this Dma\_space.*

## 15.280.1 Detailed Description

Managed DMA Address Space.

A managed [Dma\\_space](#) represents the [L4Re](#) abstraction of an DMA address space of one or several devices. Devices are assigned to a managed [Dma\\_space](#) by binding the [Dma\\_space](#) to the respective DMA domain (see [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#)), which might link the [Dma\\_space](#) with a kernel [DMA space](#). Note that several DMA domains can be bound to the same [Dma\\_space](#). Whenever a device needs direct access to parts of an [L4Re::Dataspace](#), that part of the data space must be mapped to the managed [Dma\\_space](#) that is assigned to that device. Binding to DMA domains must happen before mapping. After the DMA accesses to the memory are finished the memory must be unmapped from the device's DMA address space.

Mapping to a managed DMA address space, using `map()`, makes the given parts of the data space visible to the associated device at the returned DMA address. As long as the memory is mapped into a DMA space it is 'pinned' and cannot be subject to dynamic memory management such as swapping. Additionally, `map()` is responsible for the necessary syncing operations before the DMA.

`unmap()` is the reverse operation to `map()` and unmaps the given data-space part for the DMA address space. `unmap()` is responsible for the necessary sync operations after the DMA.

Definition at line 63 of file [dma\\_space](#).

## 15.280.2 Member Typedef Documentation

### 15.280.2.1 Attributes

```
typedef L4::Types::Flags<Attribute> L4Re::Dma_space::Attributes
```

Attributes for DMA mappings.

See also

[Attribute](#)

Definition at line 108 of file [dma\\_space](#).

## 15.280.3 Member Enumeration Documentation

### 15.280.3.1 Attribute

```
enum L4Re::Dma_space::Attribute
```

Attributes used for the memory region during the transfer.

See also

[Attributes](#)



## Enumerator

No_sync	Do not sync the memory hierarchy. When this flag is <i>not set</i> (default) the memory region shall be made coherent to the point-of-coherency of the device associated with this <a href="#">Dma_space</a> . When using this attribute the client is responsible for syncing the memory hierarchy for DMA. This can either be done using the cache API or by another <code>map()</code> or <code>unmap()</code> operation of the same part of the data space (without the <a href="#">No_sync</a> attribute).
---------	---

Definition at line 87 of file [dma\\_space](#).

## 15.280.3.2 Direction

```
enum L4Re::Dma_space::Direction
```

Direction of the DMA transfers.

## Enumerator

Bidirectional	device reads and writes to the memory
To_device	device reads the memory
From_device	device writes to the memory
None	device is coherently connected to the memory

Definition at line 75 of file [dma\\_space](#).

## 15.280.3.3 Space\_attrib

```
enum L4Re::Dma_space::Space_attrib
```

Attributes assigned to the DMA space when associated with a specific device.

## See also

[Space\\_attribs](#)

## Enumerator

Coherent	The device is connected coherently with the cache. This means that the <code>map()</code> and <code>unmap()</code> do not need to sync CPU caches before and after DMA.
Phys_space	The DMA space has no DMA task assigned and uses the CPUs physical memory.

Definition at line 115 of file [dma\\_space](#).

## 15.280.4 Member Function Documentation

## 15.280.4.1 associate()

```
long L4Re::Dma_space::associate (
```

```

L4::Ipc::Opt< L4::Ipc::Cap< L4::Task > > dma_task,
Space_attribs attr )

```

Associate a (kernel) [DMA space](#) for a device to this [Dma\\_space](#).

#### Parameters

in	<i>dma_task</i>	The (kernel) <a href="#">DMA space</a> used for the device that shall be associated with this DMA space. In case no IOMMU is present or configured, the <i>dma_task</i> might be an invalid capability when <a href="#">L4Re::Dma_space::Phys_space</a> is set in <i>attr</i> , in this case the CPUs physical memory is used as DMA address space.
in	<i>attr</i>	Attributes for this DMA space. See <a href="#">L4Re::Dma_space::Space_attrib</a> .

#### Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EPERM</i>	No <a href="#">L4_CAP_FPAGE_W</a> right on the <a href="#">Dma_space</a> capability.
<i>-L4_EINVAL</i>	
<i>-L4_ENOENT</i>	

#### Precondition

requires capability rights: {RW}

### 15.280.4.2 disassociate()

```
long L4Re::Dma_space::disassociate ( )
```

Disassociate the (kernel) [DMA space](#) from this [Dma\\_space](#).

#### Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EPERM</i>	No <a href="#">L4_CAP_FPAGE_W</a> right on the <a href="#">Dma_space</a> capability.
<i>-L4_ENOENT</i>	

#### Precondition

requires capability rights: {RW}

### 15.280.4.3 map()

```

long L4Re::Dma_space::map (
    L4::Ipc::Cap< L4Re::Dataspace > src,
    L4Re::Dataspace::Offset offset,
    L4::Ipc::In_out< l4_size_t * > size,
    Attributes attrs,

```

```
Direction dir,  
Dma_addr * dma_addr )
```

Map the given part of this data space into the DMA address space.

## Parameters

in	<i>src</i>	Source data space (that describes the memory). Caller needs write right to the data space.
in	<i>offset</i>	The offset (bytes) within <i>src</i> .
in, out	<i>size</i>	The size (bytes) of the region to be mapped for DMA, after successful mapping the size returned is the size mapped for DMA as a single block. This size might be smaller than the original input size, in this case the caller might call <code>map()</code> again with a new offset and the remaining size.
in	<i>attrs</i>	The attributes used for this DMA mapping (a combination of <a href="#">Dma_space::Attribute</a> values).
in	<i>dir</i>	The direction of the DMA transfer issued with this mapping. The same value must later be passed to <a href="#">unmap()</a> .
out	<i>dma_addr</i>	The DMA address to use for DMA with the associated device.

## Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EPERM</i>	No <a href="#">L4_CAP_FPAGE_W</a> right on <i>src</i> capability.
<i>-L4_EINVAL</i>	The <i>src</i> capability is invalid or does not refer to a valid dataspace.
<i>-L4_EEXIST</i>	The specified region overlaps an existing mapping.
<i>-L4_ENOMEM</i>	Not enough memory to allocate internal datastructures.
<i>-L4_ERANGE</i>	<i>offset</i> is larger than the size of the dataspace.

## Precondition

requires capability rights: {R}

## Note

[associate\(\)](#) must be called prior to mapping memory. Usually this is done implicitly when binding the managed [Dma\\_space](#) to a DMA domain (see [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#)).

## 15.280.4.4 unmap()

```
long L4Re::Dma_space::unmap (
    Dma_addr dma_addr,
    l4_size_t size,
    Attributes attrs,
    Direction dir )
```

Unmap the given part of this data space from the DMA address space.

## Parameters

<i>dma_addr</i>	The DMA address (returned by <a href="#">Dma_space::map()</a> ).
<i>size</i>	The size (bytes) of the memory region to unmap.
<i>attrs</i>	The attributes for the unmap (currently none).
<i>dir</i>	The direction of the finished DMA operation.

**Returns**

0 in the case of success, a negative error code otherwise.

**Precondition**

requires capability rights: {R}

The documentation for this class was generated from the following file:

- [l4/re/dma\\_space](#)

## 15.281 L4Re::Env Class Reference

C++ interface of the initial environment that is provided to an [L4](#) task.

```
#include <env>
```

Collaboration diagram for L4Re::Env:

L4Re::Env
<div>+ parent() + mem_alloc() + user_factory() + rm() + log() + main_thread() + task() + factory() + first_free_cap() + utcb_area() and 17 more... + env()</div>

**Public Types**

- typedef [l4re\\_env\\_cap\\_entry\\_t](#) **Cap\_entry**  
*C++ type for an entry in the initial objects array.*

## Public Member Functions

- [L4::Cap](#)< [Parent](#) > [parent](#) () const noexcept  
*Object-capability to the parent.*
- [L4::Cap](#)< [Mem\\_alloc](#) > [mem\\_alloc](#) () const noexcept  
*Object-capability to the memory allocator.*
- [L4::Cap](#)< [L4::Factory](#) > [user\\_factory](#) () const noexcept  
*Object-capability to the user-level object factory.*
- [L4::Cap](#)< [Rm](#) > [rm](#) () const noexcept  
*Object-capability to the region map.*
- [L4::Cap](#)< [Log](#) > [log](#) () const noexcept  
*Object-capability to the logging service.*
- [L4::Cap](#)< [L4::Thread](#) > [main\\_thread](#) () const noexcept  
*Object-capability of the first user thread.*
- [L4::Cap](#)< [L4::Task](#) > [task](#) () const noexcept  
*Object-capability of the user task.*
- [L4::Cap](#)< [L4::Factory](#) > [factory](#) () const noexcept  
*Object-capability to the factory object available to the task.*
- [l4\\_cap\\_idx\\_t](#) [first\\_free\\_cap](#) () const noexcept  
*First available capability selector.*
- [l4\\_fpage\\_t](#) [utcb\\_area](#) () const noexcept  
*UTCB area of the task.*
- [l4\\_addr\\_t](#) [first\\_free\\_utcb](#) () const noexcept  
*First free UTCB.*
- [Cap\\_entry](#) const \* [initial\\_caps](#) () const noexcept  
*Get a pointer to the first entry in the initial objects array.*
- [Cap\\_entry](#) const \* [get](#) (char const \*name, unsigned l) const noexcept  
*Get the Cap\_entry for the object named name.*
- template<typename T >  
[L4::Cap](#)< T > [get\\_cap](#) (char const \*name, unsigned l) const noexcept  
*Get the capability selector for the object named name.*
- template<typename T >  
[L4::Cap](#)< T > [get\\_cap](#) (char const \*name) const noexcept  
*Get the capability selector for the object named name.*
- void [parent](#) ([L4::Cap](#)< [Parent](#) > const &c) noexcept  
*Set parent object-capability.*
- void [mem\\_alloc](#) ([L4::Cap](#)< [Mem\\_alloc](#) > const &c) noexcept  
*Set memory allocator object-capability.*
- void [rm](#) ([L4::Cap](#)< [Rm](#) > const &c) noexcept  
*Set region map object-capability.*
- void [log](#) ([L4::Cap](#)< [Log](#) > const &c) noexcept  
*Set log object-capability.*
- void [main\\_thread](#) ([L4::Cap](#)< [L4::Thread](#) > const &c) noexcept  
*Set object-capability of first user thread.*
- void [factory](#) ([L4::Cap](#)< [L4::Factory](#) > const &c) noexcept  
*Set factory object-capability.*
- void [first\\_free\\_cap](#) ([l4\\_cap\\_idx\\_t](#) c) noexcept  
*Set first available capability selector.*
- void [utcb\\_area](#) ([l4\\_fpage\\_t](#) utcb) noexcept  
*Set UTCB area of the task.*
- void [first\\_free\\_utcb](#) ([l4\\_addr\\_t](#) u) noexcept

*Set first free UTCB.*

- [L4::Cap](#)< [L4::Scheduler](#) > [scheduler](#) () const noexcept

*Get the scheduler capability for the task.*

- void [scheduler](#) ([L4::Cap](#)< [L4::Scheduler](#) > const &c) noexcept

*Set the scheduler capability.*

- void [initial\\_caps](#) ([Cap\\_entry](#) \*first) noexcept

*Set the pointer to the first Cap\_entry in the initial objects array.*

## Static Public Member Functions

- static [Env](#) const \* [env](#) () noexcept

*Returns the initial environment for the current task.*

### 15.281.1 Detailed Description

C++ interface of the initial environment that is provided to an [L4](#) task.

The initial environment is provided to each [L4](#) task that is started by an [L4Re](#) conform loader, such as the Moe root task. The initial environment provides access to a set of initial capabilities and some additional information about the available resources, such as free UTCBs (see [Virtual Registers](#) ) and available entries in capability table (provided by the micro kernel).

Each of the initial capabilities is stored at a fixed index in the task's capability table and the [L4](#) runtime environment provides convenience functions to retrieve the capabilities. See the table below for an comprehensive overview.

Name	Object Type	Convenience Function
parent	<a href="#">L4Re::Parent</a>	<a href="#">L4Re::Env::parent()</a>
user_factory	<a href="#">L4::Factory</a>	<a href="#">L4Re::Env::user_factory()</a>
log	<a href="#">L4Re::Log</a>	<a href="#">L4Re::Env::log()</a>
main_thread	<a href="#">L4::Thread</a>	<a href="#">L4Re::Env::main_thread()</a>
rm	<a href="#">L4Re::Rm</a>	<a href="#">L4Re::Env::rm()</a>
factory	<a href="#">L4::Factory</a>	<a href="#">L4Re::Env::factory()</a>
task	<a href="#">L4::Task</a>	<a href="#">L4Re::Env::task()</a>
scheduler	<a href="#">L4::Scheduler</a>	<a href="#">L4Re::Env::scheduler()</a>

Additional information found in the initial environment is:

- First free entry in capability table
- The [UTCB](#) area (as flex page)
- First free UTCB (address in the UTCB area)

## Include File

```
#include <l4/re/env>
```

For an explanation of the default task capabilities see [l4\\_default\\_caps\\_t](#).

For the C interface refer to [Initial Environment](#).

Definition at line 85 of file [env](#).

## 15.281.2 Member Function Documentation

### 15.281.2.1 env()

```
static Env const * L4Re::Env::env ( ) [inline], [static], [noexcept]
```

Returns the initial environment for the current task.

#### Returns

Pointer to the initial environment class.

A typical use of this function is `L4Re::Env::env()-><member>()`

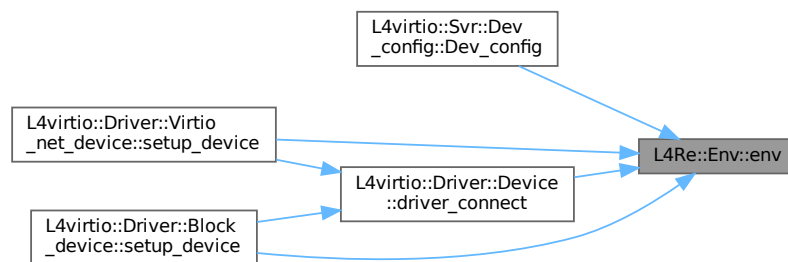
#### Examples

`examples/clntsrv/client.cc`, `examples/libs/l4re/c++/mem_alloc/ma+rm.cc`, `examples/libs/l4re/c++/shared_ds/ds_clnt.cc`, `examples/libs/l4re/c++/shared_ds/ds_srv.cc`, `examples/libs/l4re/streammap/client.cc`, and `examples/sys/migrate/thread_migrate`

Definition at line 103 of file `env`.

Referenced by `L4virtio::Svr::Dev_config::Dev_config()`, `L4virtio::Driver::Device::driver_connect()`, `L4virtio::Driver::Virtio_net_device::setup_device()` and `L4virtio::Driver::Block_device::setup_device()`.

Here is the caller graph for this function:



### 15.281.2.2 factory() [1/2]

```
L4::Cap< L4::Factory > L4Re::Env::factory ( ) const [inline], [noexcept]
```

Object-capability to the factory object available to the task.

#### Returns

Factory object-capability

Definition at line 151 of file `env`.

References `l4re_env_t::factory`.

### 15.281.2.3 factory() [2/2]

```
void L4Re::Env::factory (
    L4::Cap< L4::Factory > const & c ) [inline], [noexcept]
```

Set factory object-capability.



## Parameters

<code>c</code>	Factory object-capability
----------------	---------------------------

Definition at line 256 of file [env](#).

References [l4re\\_env\\_t::factory](#).

**15.281.2.4 first\_free\_cap()** [1/2]

```
l4_cap_idx_t L4Re::Env::first_free_cap ( ) const [inline], [noexcept]
```

First available capability selector.

## Returns

First capability selector.

First capability selector available for use for in the application.

Definition at line 159 of file [env](#).

References [l4re\\_env\\_t::first\\_free\\_cap](#).

**15.281.2.5 first\_free\_cap()** [2/2]

```
void L4Re::Env::first_free_cap (
    l4_cap_idx_t c ) [inline], [noexcept]
```

Set first available capability selector.

## Parameters

<code>c</code>	First capability selector available to the application.
----------------	---

Definition at line 262 of file [env](#).

References [l4re\\_env\\_t::first\\_free\\_cap](#).

**15.281.2.6 first\_free\_utcb()** [1/2]

```
l4_addr_t L4Re::Env::first_free_utcb ( ) const [inline], [noexcept]
```

First free UTCB.

## Returns

object-capability

First free UTCB within the UTCB area available for the application to use.

Definition at line 174 of file [env](#).

References [l4re\\_env\\_t::first\\_free\\_utcb](#).

**15.281.2.7 first\_free\_utcb()** [2/2]

```
void L4Re::Env::first_free_utcb (
    l4_addr_t u ) [inline], [noexcept]
```

Set first free UTCB.

**Parameters**

<i>u</i>	First UTCB available for the application to use.
----------	--

Definition at line 274 of file [env](#).

References [l4re\\_env\\_t::first\\_free\\_utcb](#).

**15.281.2.8 get()**

```
Cap_entry const * L4Re::Env::get (
    char const * name,
    unsigned l ) const [inline], [noexcept]
```

Get the Cap\_entry for the object named *name*.

**Parameters**

<i>name</i>	is the name of the object.
<i>l</i>	is the length of the name, thus <i>name</i> might not be zero terminated.

**Returns**

A pointer to the Cap\_entry for the object named *name*, or NULL if no such object was found.

Definition at line 192 of file [env](#).

References [l4re\\_env\\_get\\_cap\\_l\(\)](#).

Here is the call graph for this function:



### 15.281.2.9 get\_cap() [1/2]

```
template<typename T >  
L4::Cap< T > L4Re::Env::get_cap (   
    char const * name ) const [inline], [noexcept]
```

Get the capability selector for the object named *name*.

## Parameters

<i>name</i>	is the name of the object (zero terminated).
-------------	--

## Returns

A capability selector for the object named *name*, or an invalid capability selector if no such object was found.

Definition at line 219 of file [env](#).

**15.281.2.10 get\_cap()** [2/2]

```
template<typename T >
L4::Cap< T > L4Re::Env::get_cap (
    char const * name,
    unsigned l ) const [inline], [noexcept]
```

Get the capability selector for the object named *name*.

## Parameters

<i>name</i>	is the name of the object.
<i>l</i>	is the length of the name, thus <i>name</i> might not be zero terminated.

## Returns

A capability selector for the object named *name*, or an invalid capability selector if no such object was found.

## Examples

[examples/clntsrv/client.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), and [examples/libs/l4re/streammap/client.cc](#).

Definition at line 204 of file [env](#).

References [L4\\_ENOENT](#).

**15.281.2.11 initial\_caps()** [1/2]

```
Cap_entry const * L4Re::Env::initial_caps ( ) const [inline], [noexcept]
```

Get a pointer to the first entry in the initial objects array.

## Returns

A pointer to the first entry in the initial objects array.

Definition at line 181 of file [env](#).

References [l4re\\_env\\_t::caps](#).

**15.281.2.12 initial\_caps()** [2/2]

```
void L4Re::Env::initial_caps (
    Cap_entry * first ) [inline], [noexcept]
```

Set the pointer to the first *Cap\_entry* in the initial objects array.

## Parameters

<i>first</i>	is the first element in the array.
--------------	------------------------------------

Definition at line 296 of file [env](#).

References [l4re\\_env\\_t::caps](#).

**15.281.2.13 log()** [1/2]

```
L4::Cap< Log > L4Re::Env::log ( ) const [inline], [noexcept]
```

Object-capability to the logging service.

## Returns

[Log](#) object-capability

Definition at line 133 of file [env](#).

References [l4re\\_env\\_t::log](#).

**15.281.2.14 log()** [2/2]

```
void L4Re::Env::log (
    L4::Cap< Log > const & c ) [inline], [noexcept]
```

Set log object-capability.

## Parameters

<i>c</i>	<a href="#">Log</a> object-capability
----------	---------------------------------------

Definition at line 244 of file [env](#).

References [l4re\\_env\\_t::log](#).

**15.281.2.15 main\_thread()** [1/2]

```
L4::Cap< L4::Thread > L4Re::Env::main_thread ( ) const [inline], [noexcept]
```

Object-capability of the first user thread.

## Returns

Object-capability of the first user thread.

Definition at line 139 of file [env](#).

References [l4re\\_env\\_t::main\\_thread](#).

**15.281.2.16 main\_thread()** [2/2]

```
void L4Re::Env::main_thread (
    L4::Cap< L4::Thread > const & c ) [inline], [noexcept]
```

Set object-capability of first user thread.

**Parameters**

<b>c</b>	First thread's object-capability
----------	----------------------------------

Definition at line 250 of file [env](#).

References [l4re\\_env\\_t::main\\_thread](#).

**15.281.2.17 mem\_alloc()** [1/2]

```
L4::Cap< Mem_alloc > L4Re::Env::mem_alloc ( ) const [inline], [noexcept]
```

Object-capability to the memory allocator.

**Returns**

Memory allocator object-capability

**Examples**

[examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#).

Definition at line 116 of file [env](#).

References [l4re\\_env\\_t::mem\\_alloc](#).

**15.281.2.18 mem\_alloc()** [2/2]

```
void L4Re::Env::mem_alloc (
    L4::Cap< Mem_alloc > const & c ) [inline], [noexcept]
```

Set memory allocator object-capability.

**Parameters**

<b>c</b>	Memory allocator object-capability
----------	------------------------------------

Definition at line 232 of file [env](#).

References [l4re\\_env\\_t::mem\\_alloc](#).

**15.281.2.19 parent()** [1/2]

```
L4::Cap< Parent > L4Re::Env::parent ( ) const [inline], [noexcept]
```

Object-capability to the parent.

**Returns**

[Parent](#) object-capability

Definition at line 110 of file [env](#).

References [l4re\\_env\\_t::parent](#).

**15.281.2.20 parent()** [2/2]

```
void L4Re::Env::parent (
    L4::Cap< Parent > const & c ) [inline], [noexcept]
```

Set parent object-capability.

**Parameters**

c	<a href="#">Parent</a> object-capability
---	--

Definition at line 226 of file [env](#).

References [l4re\\_env\\_t::parent](#).

**15.281.2.21 rm()** [1/2]

```
L4::Cap< Rm > L4Re::Env::rm ( ) const [inline], [noexcept]
```

Object-capability to the region map.

**Returns**

Region map object-capability

**Examples**

[examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), and [examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#).

Definition at line 127 of file [env](#).

References [l4re\\_env\\_t::rm](#).

**15.281.2.22 rm()** [2/2]

```
void L4Re::Env::rm (
    L4::Cap< Rm > const & c ) [inline], [noexcept]
```

Set region map object-capability.

**Parameters**

<code>c</code>	Region map object-capability
----------------	------------------------------

Definition at line 238 of file [env](#).

References [l4re\\_env\\_t::rm](#).

**15.281.2.23 scheduler()** [1/2]

```
L4::Cap< L4::Scheduler > L4Re::Env::scheduler ( ) const [inline], [noexcept]
```

Get the scheduler capability for the task.

**Returns**

The capability selector for the default scheduler used for this task.

**Examples**

[examples/sys/migrate/thread\\_migrate.cc](#).

Definition at line 282 of file [env](#).

References [l4re\\_env\\_t::scheduler](#).

**15.281.2.24 scheduler()** [2/2]

```
void L4Re::Env::scheduler (
    L4::Cap< L4::Scheduler > const & c ) [inline], [noexcept]
```

Set the scheduler capability.

**Parameters**

<code>c</code>	is the capability to be set as scheduler.
----------------	---

Definition at line 289 of file [env](#).

References [l4re\\_env\\_t::scheduler](#).

**15.281.2.25 task()**

```
L4::Cap< L4::Task > L4Re::Env::task ( ) const [inline], [noexcept]
```

Object-capability of the user task.

**Returns**

Object-capability of the user task.

Definition at line 145 of file [env](#).



**15.281.2.26 utcb\_area()** [1/2]

```
l4_fpage_t L4Re::Env::utcb_area ( ) const [inline], [noexcept]
```

UTCB area of the task.

**Returns**

UTCB area

Definition at line 165 of file [env](#).

References [l4re\\_env\\_t::utcb\\_area](#).

**15.281.2.27 utcb\_area()** [2/2]

```
void L4Re::Env::utcb_area (
    l4_fpage_t utcbs ) [inline], [noexcept]
```

Set UTCB area of the task.

**Parameters**

<i>utcbs</i>	UTCB area
--------------	-----------

Definition at line 268 of file [env](#).

References [l4re\\_env\\_t::utcb\\_area](#).

The documentation for this class was generated from the following file:

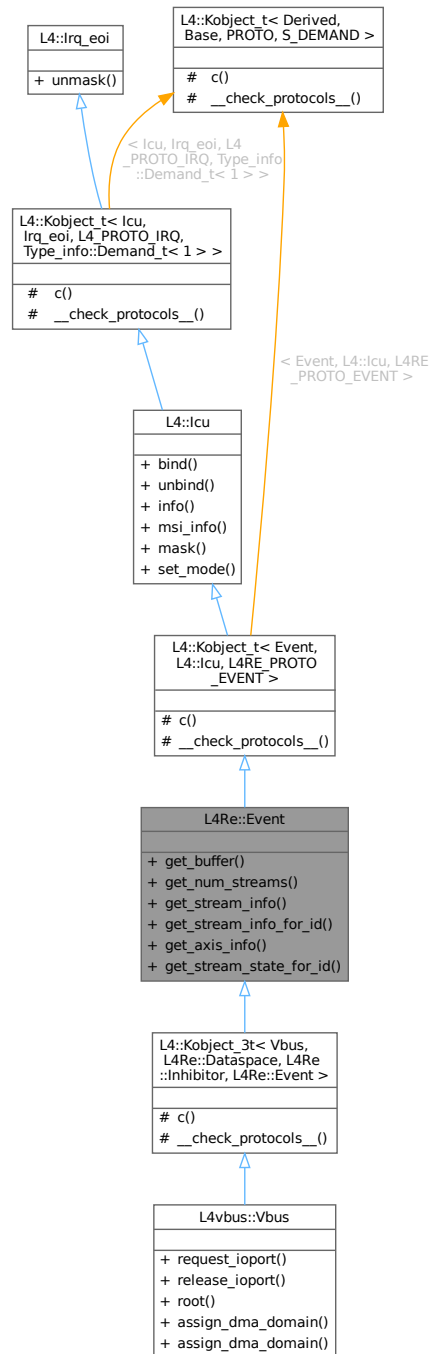
- [l4re/env](#)

**15.282 L4Re::Event Class Reference**

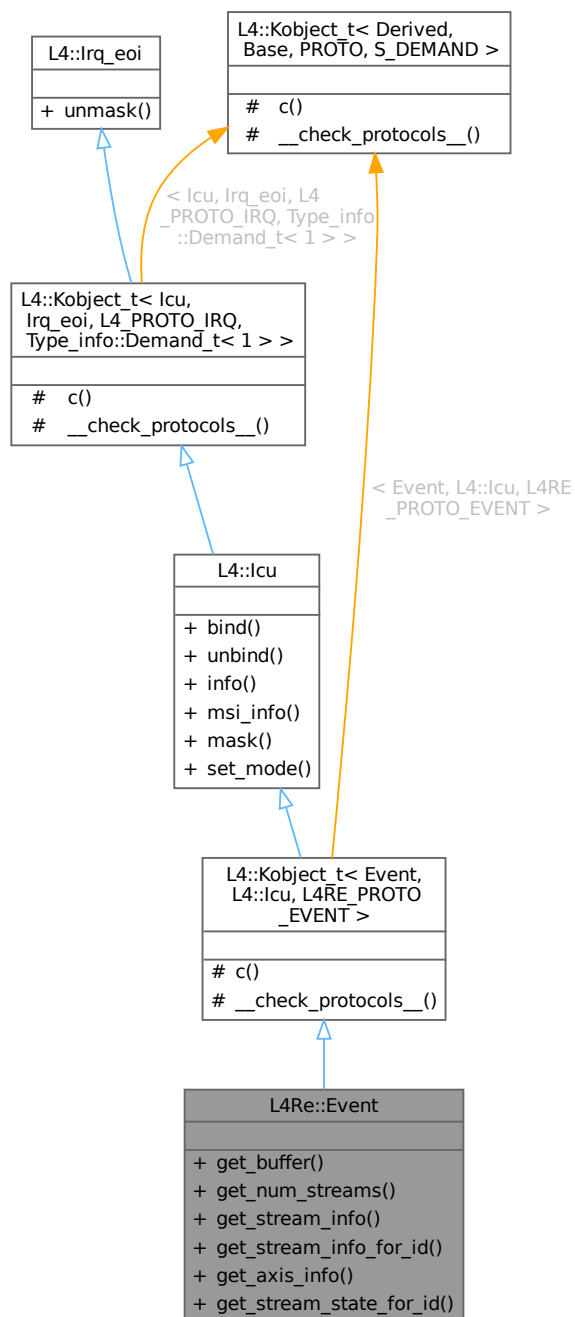
[Event](#) class.

```
#include <event>
```

Inheritance diagram for L4Re::Event:



Collaboration diagram for L4Re::Event:



## Public Member Functions

- long `get_buffer` (L4::ipc::Out< L4::Cap< Dataspace > > ds)  
*Get event signal buffer.*
- long `get_num_streams` ()  
*Get number of event streams.*
- long `get_stream_info` (int idx, Event\_stream\_info \*info)

*Get event stream infos.*

- long [get\\_stream\\_info\\_for\\_id](#) ([l4\\_umword\\_t](#) stream\_id, [Event\\_stream\\_info](#) \*info)

*Get event stream infos.*

- long [get\\_axis\\_info](#) ([l4\\_umword\\_t](#) stream\_id, unsigned naxes, unsigned const \*axis, [Event\\_absinfo](#) \*info) const noexcept

*Get event stream axis infos.*

- long [get\\_stream\\_state\\_for\\_id](#) ([l4\\_umword\\_t](#) stream\_id, [Event\\_stream\\_state](#) \*state)

*Get event stream state.*

## Public Member Functions inherited from [L4::Icu](#)

- [l4\\_msgtag\\_t](#) bind (unsigned irqnum, [L4::Cap](#)< [Triggerable](#) > irq, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept

*Bind an interrupt line of an interrupt controller to an interrupt object.*

- [l4\\_msgtag\\_t](#) unbind (unsigned irqnum, [L4::Cap](#)< [Triggerable](#) > irq, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept

*Remove binding of an interrupt line from the interrupt controller object.*

- [l4\\_msgtag\\_t](#) info ([l4\\_icu\\_info\\_t](#) \*info, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept

*Get information about the ICU features.*

- [l4\\_msgtag\\_t](#) msi\_info ([l4\\_umword\\_t](#) irqnum, [l4\\_uint64\\_t](#) source, [l4\\_icu\\_msi\\_info\\_t](#) \*msi\_info)

*Get MSI info about IRQ.*

- [l4\\_msgtag\\_t](#) mask (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept

*Mask an IRQ line.*

- [l4\\_msgtag\\_t](#) set\_mode (unsigned irqnum, [l4\\_umword\\_t](#) mode, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept

*Set interrupt mode.*

## Public Member Functions inherited from [L4::Irq\\_eoi](#)

- [l4\\_msgtag\\_t](#) unmask (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept

*Unmask the given interrupt line.*

## Additional Inherited Members

## Protected Types inherited from [L4::Kobject\\_t](#)< [Event](#), [L4::Icu](#), [L4RE\\_PROTO\\_EVENT](#) >

- typedef [Event](#) **Class**

*The target interface type (inheriting from [Kobject\\_t](#))*

- typedef [Typeid::Iface](#)< [PROTO](#), [Event](#) > **\_\_Iface**

*The interface description for the derived class.*

- typedef [Typeid::Merge\\_list](#)< [Typeid::Iface\\_list](#)< **\_\_Iface** >, typename [Base::\\_\\_Iface\\_list](#) > **\_\_Iface\_list**

*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Types inherited from

## [L4::Kobject\\_t](#)< [Icu](#), [Irq\\_eoi](#), [L4\\_PROTO\\_IRQ](#), [Type\\_info::Demand\\_t](#)< 1 > >

- typedef [Icu](#) **Class**

*The target interface type (inheriting from [Kobject\\_t](#))*

- typedef [Typeid::Iface](#)< [PROTO](#), [Icu](#) > **\_\_Iface**

*The interface description for the derived class.*

- typedef [Typeid::Merge\\_list](#)< [Typeid::Iface\\_list](#)< **\_\_Iface** >, typename [Base::\\_\\_Iface\\_list](#) > **\_\_Iface\_list**

*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from [L4::Kobject\\_t< Event, L4::lcu, L4RE\\_PROTO\\_EVENT >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject\\_t< lcu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< Event, L4::lcu, L4RE\\_PROTO\\_EVENT >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< lcu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

## 15.282.1 Detailed Description

[Event](#) class.

See also

[L4Re Event API](#)

Definition at line 148 of file [event](#).

## 15.282.2 Member Function Documentation

### 15.282.2.1 [get\\_axis\\_info\(\)](#)

```
long L4Re::Event::get_axis_info (
    l4_umword_t stream_id,
    unsigned naxes,
    unsigned const * axis,
    Event_absinfo * info ) const [inline], [noexcept]
```

Get event stream axis infos.

**Parameters**

	<i>stream↔ _id</i>	ID of the event stream.
in	<i>axes</i>	Array of axis IDs.
out	<i>info</i>	Array of axis infos.

**Return values**

$\geq 0$	Number of returned axes infos.
$< 0$	Error code.

Definition at line 208 of file [event](#).

**15.282.2.2 get\_buffer()**

```
long L4Re::Event::get_buffer (
    L4::Ipc::Out< L4::Cap< Dataspace > > ds )
```

Get event signal buffer.

**Parameters**

out	<i>ds</i>	<a href="#">Event</a> buffer.
-----	-----------	-------------------------------

**Return values**

0	Success
$< 0$	Error

**15.282.2.3 get\_num\_streams()**

```
long L4Re::Event::get_num_streams ( )
```

Get number of event streams.

**Return values**

$\geq 0$	Number of streams.
$< 0$	Error code.

**15.282.2.4 get\_stream\_info()**

```
long L4Re::Event::get_stream_info (
    int idx,
    Event_stream_info * info )
```

Get event stream infos.

Deprecated. Use [get\\_stream\\_info\\_for\\_id\(\)](#).

#### Parameters

	<i>idx</i>	ID of the event stream.
out	<i>info</i>	<a href="#">Event</a> stream info.

#### Return values

0	Success
<0	Error

### 15.282.2.5 [get\\_stream\\_info\\_for\\_id\(\)](#)

```
long L4Re::Event::get_stream_info_for_id (
    l4_umword_t stream_id,
    Event_stream_info * info )
```

Get event stream infos.

#### Parameters

	<i>stream↔ _id</i>	ID of the event stream.
out	<i>info</i>	<a href="#">Event</a> stream info.

#### Return values

0	Success
<0	Error

### 15.282.2.6 [get\\_stream\\_state\\_for\\_id\(\)](#)

```
long L4Re::Event::get_stream_state_for_id (
    l4_umword_t stream_id,
    Event_stream_state * state )
```

Get event stream state.

#### Parameters

	<i>stream↔ _id</i>	ID of the event stream.
out	<i>state</i>	<a href="#">Event</a> stream state.

## Return values

0	Success
<0	Error

The documentation for this class was generated from the following file:

- l4/re/event

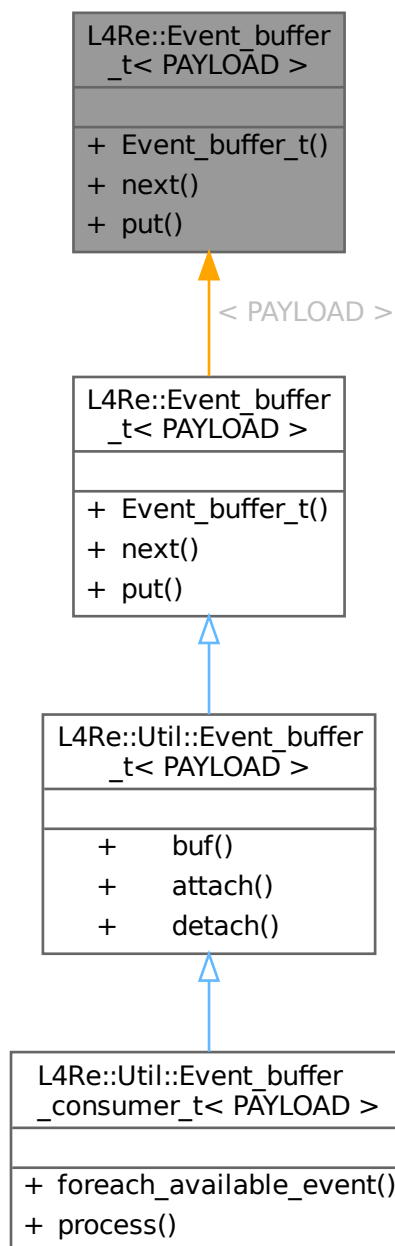
## 15.283 L4Re::Event\_buffer\_t< PAYLOAD > Class Template Reference

[Event](#) buffer class.

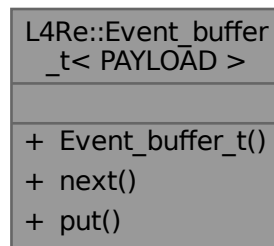
```
#include <event>
```



Inheritance diagram for L4Re::Event\_buffer\_t< PAYLOAD >:



Collaboration diagram for L4Re::Event\_buffer\_t< PAYLOAD >:



## Data Structures

- struct [Event](#)  
*Event structure used in buffer.*

## Public Member Functions

- [Event\\_buffer\\_t](#) (void \*buffer, [l4\\_addr\\_t](#) size)  
*Initialize event buffer.*
- [Event](#) \* [next](#) () noexcept  
*Next event in buffer.*
- bool [put](#) ([Event](#) const &ev) noexcept  
*Put event into buffer at current position.*

### 15.283.1 Detailed Description

```
template<typename PAYLOAD = Default_event_payload>
class L4Re::Event_buffer_t< PAYLOAD >
```

[Event](#) buffer class.

Definition at line 256 of file [event](#).

### 15.283.2 Constructor & Destructor Documentation

#### 15.283.2.1 Event\_buffer\_t()

```
template<typename PAYLOAD = Default_event_payload>
L4Re::Event_buffer_t< PAYLOAD >::Event_buffer_t (
    void * buffer,
    l4_addr_t size ) [inline]
```

Initialize event buffer.

#### Parameters

<i>buffer</i>	Pointer to buffer.
<i>size</i>	Size of buffer in bytes.

Definition at line 303 of file [event](#).

### 15.283.3 Member Function Documentation

#### 15.283.3.1 next()

```
template<typename PAYLOAD = Default_event_payload>
Event * L4Re::Event_buffer_t< PAYLOAD >::next ( ) [inline], [noexcept]
```

Next event in buffer.

#### Returns

0 if no event available, event otherwise.

Definition at line 313 of file [event](#).

References [L4Re::Event\\_buffer\\_t< PAYLOAD >::Event::time](#).

#### 15.283.3.2 put()

```
template<typename PAYLOAD = Default_event_payload>
bool L4Re::Event_buffer_t< PAYLOAD >::put (
    Event const & ev ) [inline], [noexcept]
```

Put event into buffer at current position.

#### Parameters

<i>ev</i>	<a href="#">Event</a> to put into the buffer.
-----------	---

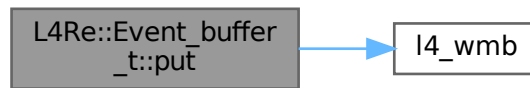
#### Returns

false if buffer is full and entry could not be added.

Definition at line 330 of file [event](#).

References [l4\\_wmb\(\)](#), and [L4Re::Event\\_buffer\\_t< PAYLOAD >::Event::time](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

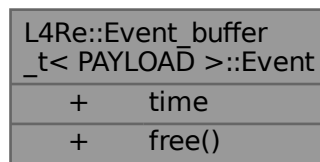
- l4/re/event

## 15.284 L4Re::Event\_buffer\_t< PAYLOAD >::Event Struct Reference

[Event](#) structure used in buffer.

```
#include <event>
```

Collaboration diagram for L4Re::Event\_buffer\_t< PAYLOAD >::Event:



### Public Member Functions

- void **free** () noexcept  
*Free the entry.*

### Data Fields

- long long **time**  
[Event](#) time stamp.

### 15.284.1 Detailed Description

```
template<typename PAYLOAD = Default_event_payload>
struct L4Re::Event_buffer_t< PAYLOAD >::Event
```

[Event](#) structure used in buffer.

Definition at line 263 of file [event](#).

The documentation for this struct was generated from the following file:

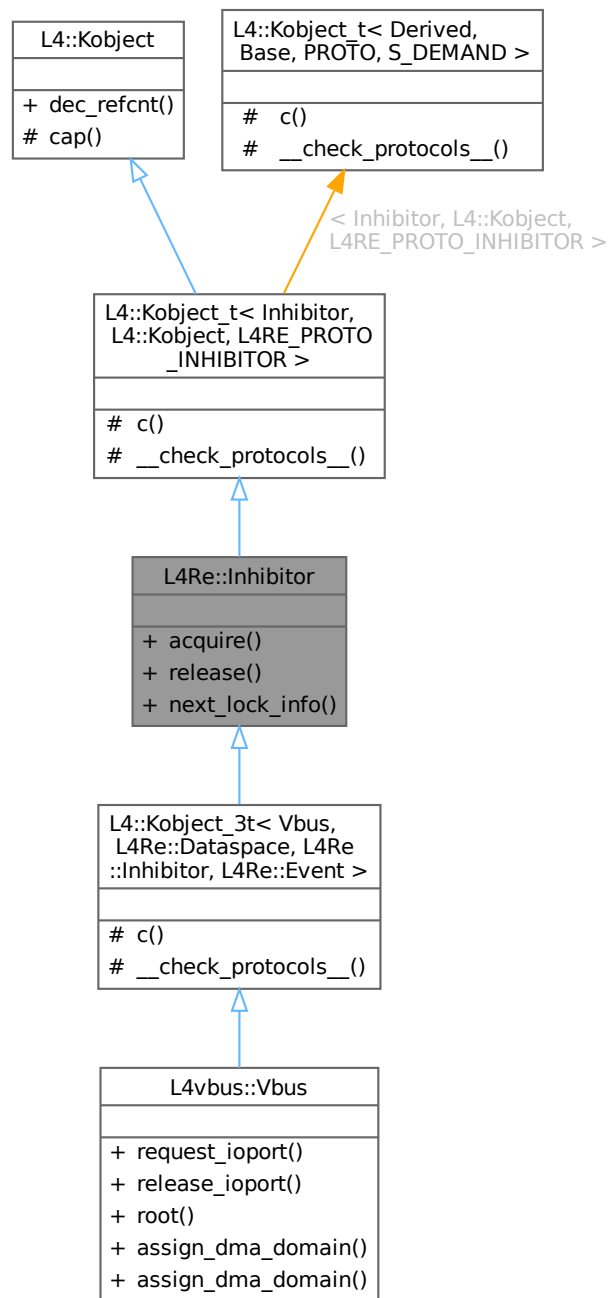
- [l4/re/event](#)

## 15.285 L4Re::Inhibitor Class Reference

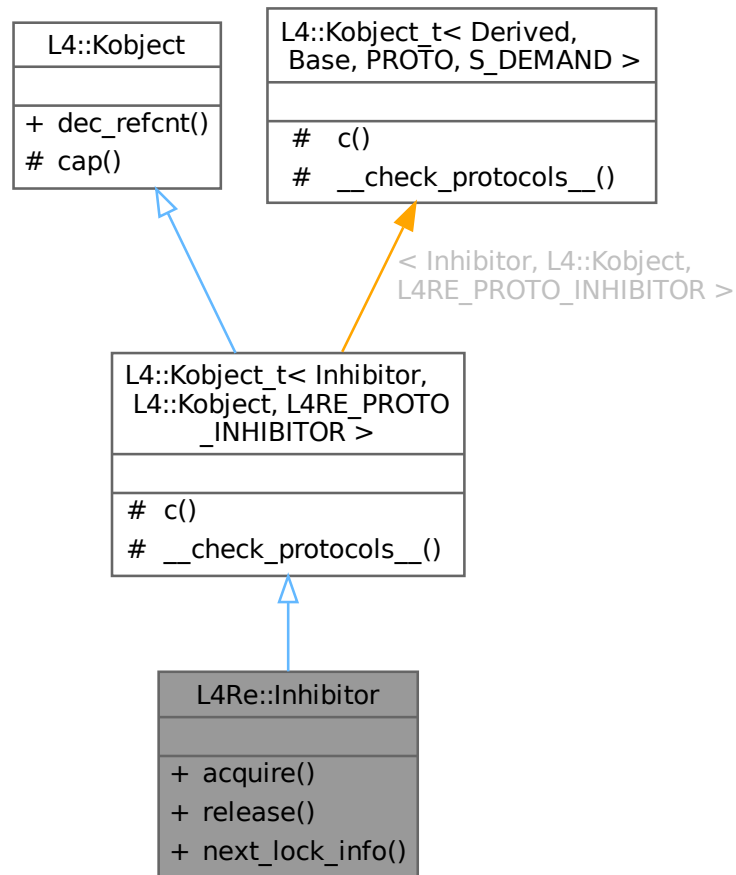
Set of inhibitor locks, which inhibit specific actions when held.

```
#include <inhibitor>
```

Inheritance diagram for L4Re::Inhibitor:



Collaboration diagram for L4Re::Inhibitor:



## Public Types

- enum { `Name_max` = 20 }

## Public Member Functions

- long `acquire` (`l4_umword_t` id, `L4::lpc::String<>` reason)  
*Acquire a specific inhibitor lock.*
- long `release` (`l4_umword_t` id)  
*Release a specific inhibitor lock.*
- long `next_lock_info` (char \*name, unsigned len, `l4_mword_t` current\_id=-1, `l4_utcb_t` \*utcb=`l4_utcb()`)  
*Get information for the next available inhibitor lock.*

## Public Member Functions inherited from `L4::Kobject`

- `l4_msgtag_t` `dec_refcnt` (`l4_mword_t` diff, `l4_utcb_t` \*utcb=`l4_utcb()`)  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

### Protected Types inherited from

[L4::Kobject\\_t< Inhibitor, L4::Kobject, L4RE\\_PROTO\\_INHIBITOR >](#)

- typedef Inhibitor **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, Inhibitor > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_Iface](#) >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from

[L4::Kobject\\_t< Inhibitor, L4::Kobject, L4RE\\_PROTO\\_INHIBITOR >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t cap \(\)](#) const noexcept  
*Return capability selector.*

### Static Protected Member Functions inherited from

[L4::Kobject\\_t< Inhibitor, L4::Kobject, L4RE\\_PROTO\\_INHIBITOR >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

## 15.285.1 Detailed Description

Set of inhibitor locks, which inhibit specific actions when held.

This interface provides access to a set of inhibitor locks, each determined by an ID that is specific to the [Inhibitor](#) object. Each individual lock shall prevent, a specific (implementation defined) action to be executed, as long as the lock is held.

For example there can be an inhibitor lock to prevent a transition to suspend-to-RAM state and a different one to prevent shutdown.

A client shall take an inhibitor lock if it needs to execute code before the action is taken. For example a lock-screen application shall grab an inhibitor lock for the suspend action to be able to lock the screen before the system goes to sleep.

[Inhibitor](#) locks are usually closely related to specific events. Usually a server automatically subscribes a client holding a lock to the corresponding event. The server shall send the event to inform the client that an action is pending. Upon reception of the event, the client is supposed to release the corresponding inhibitor lock.

Definition at line 40 of file [inhibitor](#).

## 15.285.2 Member Enumeration Documentation

### 15.285.2.1 anonymous enum

anonymous enum



## Enumerator

Name_max	The maximum length of a lock's name.
----------	--------------------------------------

Definition at line 44 of file [inhibitor](#).

## 15.285.3 Member Function Documentation

### 15.285.3.1 acquire()

```
long L4Re::Inhibitor::acquire (
    l4_umword_t id,
    L4::Ipc::String<> reason )
```

Acquire a specific inhibitor lock.

## Parameters

<i>id</i>	ID of the inhibitor lock that the client intends to acquire
<i>reason</i>	The reason why you need the lock. Used for informing the user or debugging.

## Return values

0	Success
-L4_ENODEV	The specified <i>id</i> does not exist.

### 15.285.3.2 next\_lock\_info()

```
long L4Re::Inhibitor::next_lock_info (
    char * name,
    unsigned len,
    l4_mword_t current_id = -1,
    l4_utcb_t * utcb = l4_utcb() ) [inline]
```

Get information for the next available inhibitor lock.

## Parameters

<i>name</i>	A pointer to a buffer for the name of the lock.
<i>len</i>	The length of the available buffer (usually <a href="#">Name_max</a> is used).
<i>current_id</i>	The ID of the last available lock, use -1 to get the first lock.
<i>utcb</i>	The UTCB to use for the message.

## Return values

>0	The ID of the next available lock if there is one (in this case <i>name</i> shall contain the name of the inhibitor lock).
----	--

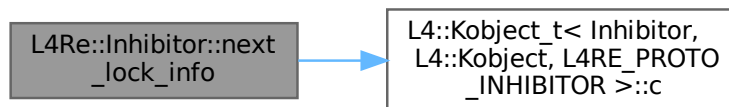
## Return values

<code>-L4_ENODEV</code>	There are no more locks.
-------------------------	--------------------------

Definition at line 86 of file [inhibitor](#).

References [L4::Kobject\\_t< Inhibitor, L4::Kobject, L4RE\\_PROTO\\_INHIBITOR >::c\(\)](#).

Here is the call graph for this function:



### 15.285.3.3 release()

```
long L4Re::Inhibitor::release (
    l4_umword_t id )
```

Release a specific inhibitor lock.

## Parameters

<i>id</i>	The ID of the inhibitor lock to release.
-----------	--

## Return values

<code>0</code>	Success
<code>-L4_ENODEV</code>	Lock with the given <code>id</code> does not exist.

The documentation for this class was generated from the following file:

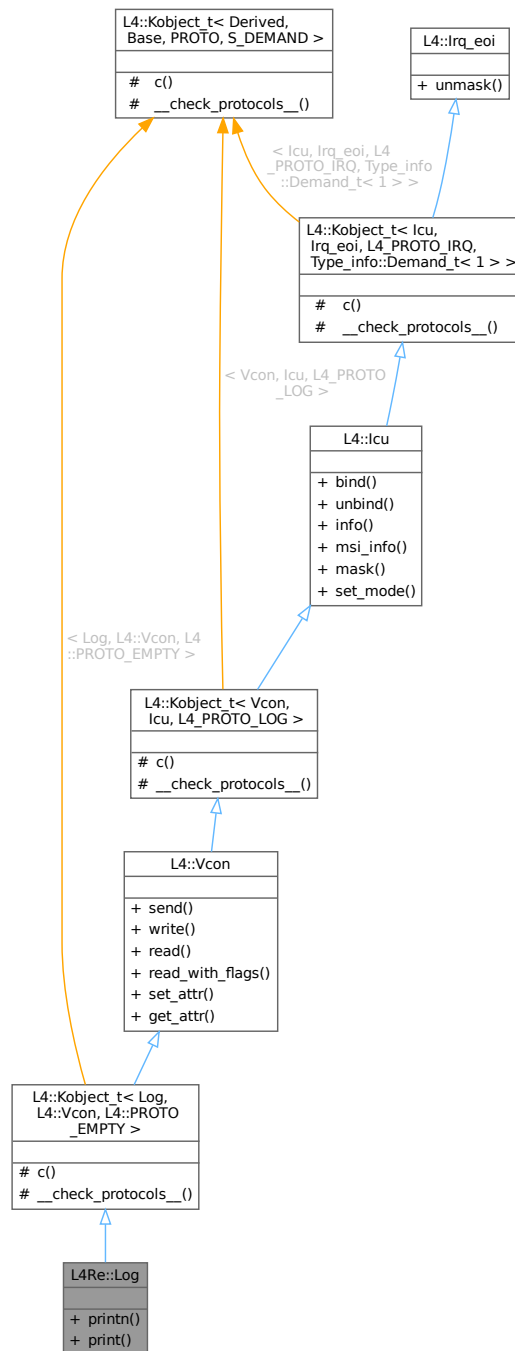
- `l4/re/inhibitor`

## 15.286 L4Re::Log Class Reference

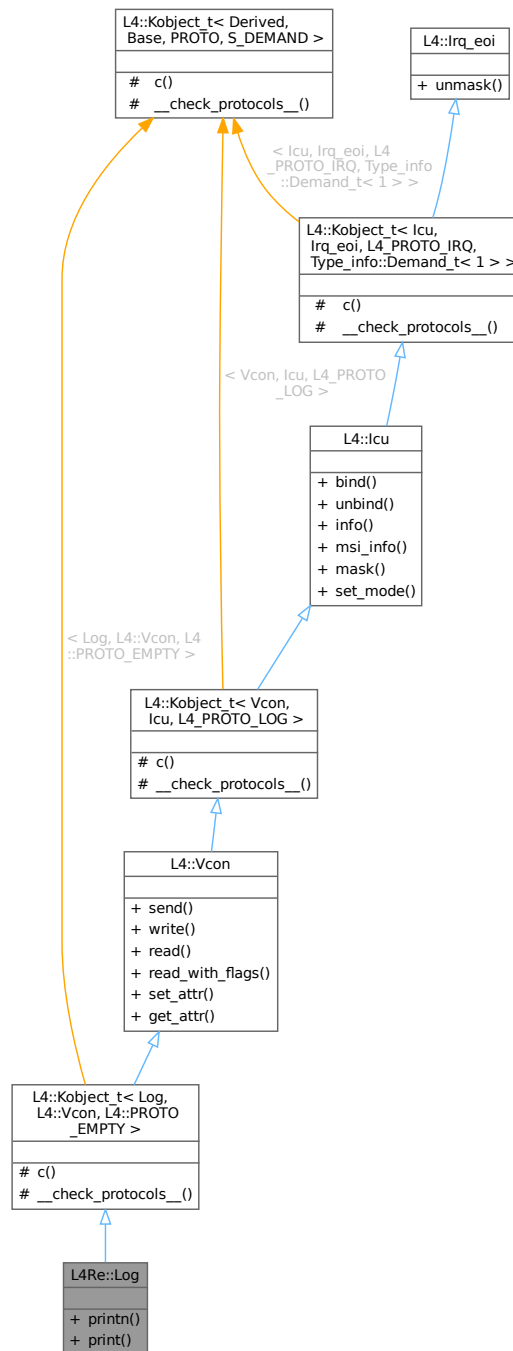
[Log](#) interface class.

```
#include <log>
```

Inheritance diagram for L4Re::Log:



Collaboration diagram for L4Re::Log:



## Public Member Functions

- void `printn` (char const \*string, int len) const noexcept  
Print string with length len, NULL characters don't matter.
- void `print` (char const \*string) const noexcept  
Print NULL-terminated string.

## Public Member Functions inherited from L4::Vcon

- [l4\\_msgtag\\_t send](#) (char const \*buf, unsigned size, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) const noexcept  
*Send data to `this` virtual console.*
- long [write](#) (char const \*buf, unsigned size, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) const noexcept  
*Write data to `this` virtual console.*
- int [read](#) (char \*buf, unsigned size, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) const noexcept  
*Read data from `this` virtual console.*
- int [read\\_with\\_flags](#) (char \*buf, unsigned size, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) const noexcept  
*Read data from `this` virtual console which also returns flags.*
- [l4\\_msgtag\\_t set\\_attr](#) ([l4\\_vcon\\_attr\\_t](#) const \*attr, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) const noexcept  
*Set the attributes of `this` virtual console.*
- [l4\\_msgtag\\_t get\\_attr](#) ([l4\\_vcon\\_attr\\_t](#) \*attr, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) const noexcept  
*Get attributes of `this` virtual console.*

## Public Member Functions inherited from L4::Icu

- [l4\\_msgtag\\_t bind](#) (unsigned irqnum, [L4::Cap< Triggerable >](#) irq, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- [l4\\_msgtag\\_t unbind](#) (unsigned irqnum, [L4::Cap< Triggerable >](#) irq, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Remove binding of an interrupt line from the interrupt controller object.*
- [l4\\_msgtag\\_t info](#) ([l4\\_icu\\_info\\_t](#) \*info, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Get information about the ICU features.*
- [l4\\_msgtag\\_t msi\\_info](#) ([l4\\_umword\\_t](#) irqnum, [l4\\_uint64\\_t](#) source, [l4\\_icu\\_msi\\_info\\_t](#) \*msi\_info)  
*Get MSI info about IRQ.*
- [l4\\_msgtag\\_t mask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Mask an IRQ line.*
- [l4\\_msgtag\\_t set\\_mode](#) (unsigned irqnum, [l4\\_umword\\_t](#) mode, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Set interrupt mode.*

## Public Member Functions inherited from L4::Irq\_eoi

- [l4\\_msgtag\\_t unmask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Unmask the given interrupt line.*

## Additional Inherited Members

## Protected Types inherited from L4::Kobject\_t< Log, L4::Vcon, L4::PROTO\_EMPTY >

- typedef Log **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, Log > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Types inherited from [L4::Kobject\\_t< Vcon, Icu, L4\\_PROTO\\_LOG >](#)

- typedef [Vcon](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, [Vcon](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_Iface](#) >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Types inherited from

#### [L4::Kobject\\_t< Icu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)

- typedef [Icu](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, [Icu](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_Iface](#) >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from

#### [L4::Kobject\\_t< Log, L4::Vcon, L4::PROTO\\_EMPTY >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from

#### [L4::Kobject\\_t< Vcon, Icu, L4\\_PROTO\\_LOG >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from

#### [L4::Kobject\\_t< Icu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

### Static Protected Member Functions inherited from

#### [L4::Kobject\\_t< Log, L4::Vcon, L4::PROTO\\_EMPTY >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

### Static Protected Member Functions inherited from

#### [L4::Kobject\\_t< Vcon, Icu, L4\\_PROTO\\_LOG >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

**Static Protected Member Functions inherited from****[L4::Kobject\\_t< Icu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)**

- static void **\_\_check\_protocols\_\_** () noexcept  
*Helper to check for protocol conflicts.*

**15.286.1 Detailed Description**

[Log](#) interface class.

Definition at line [44](#) of file [log](#).

**15.286.2 Member Function Documentation****15.286.2.1 print()**

```
void L4Re::Log::print (
    char const * string ) const    [noexcept]
```

Print NULL-terminated string.

**Parameters**

<i>string</i>	string to print
---------------	-----------------

**15.286.2.2 printn()**

```
void L4Re::Log::printn (
    char const * string,
    int len ) const    [noexcept]
```

Print string with length len, NULL characters don't matter.

**Parameters**

<i>string</i>	string to print
<i>len</i>	length of string

The documentation for this class was generated from the following file:

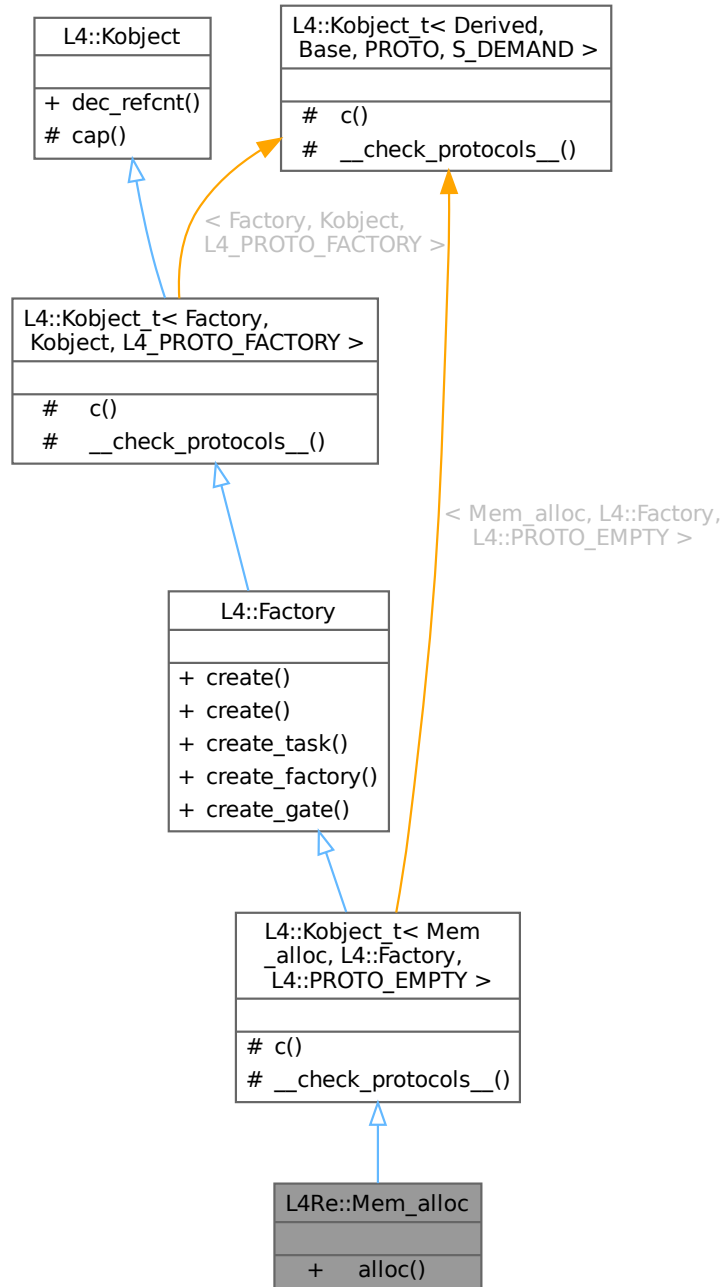
- [l4/re/log](#)

**15.287 L4Re::Mem\_alloc Class Reference**

Memory allocation interface.

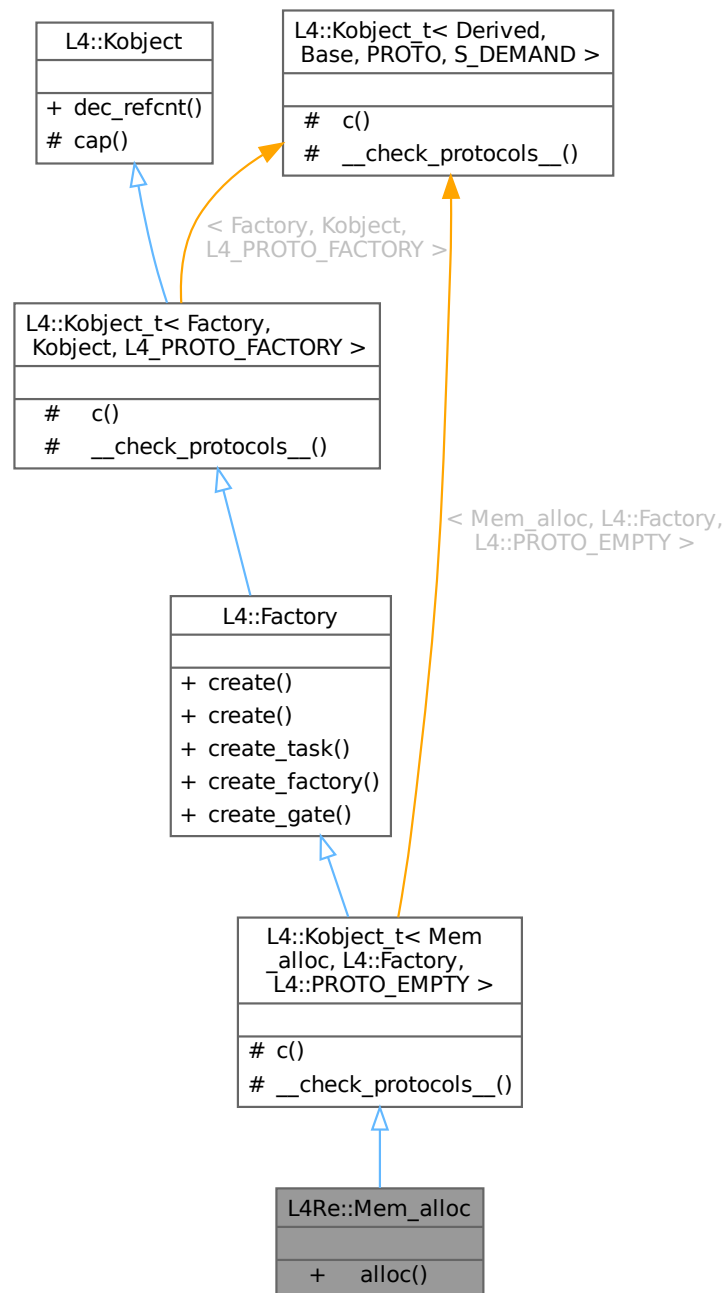
```
#include <mem_alloc>
```

Inheritance diagram for L4Re::Mem\_alloc:





Collaboration diagram for L4Re::Mem\_alloc:



## Public Types

- enum **Mem\_alloc\_flags** { **Continuous** = 0x01 , **Pinned** = 0x02 , **Super\_pages** = 0x04 , **Fixed\_paddr** = 0x08 }  
*Flags for the allocator.*

## Public Member Functions

- `long alloc (long size, L4::Cap< Dataspace > mem, unsigned long flags=0, unsigned long align=0, l4_addr_t paddr=0) const noexcept`

*Allocate anonymous memory.*

## Public Member Functions inherited from L4::Factory

- `S create (Cap< void > target, long obj, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Generic create call to the factory.*
- `template<typename OBJ >  
S create (Cap< OBJ > target, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Create call for typed capabilities.*
- `l4_msgtag_t create_task (Cap< Task > const &target_cap, l4_fpage_t *utcb_area, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Create a new task.*
- `l4_msgtag_t create_factory (Cap< Factory > const &target_cap, unsigned long limit, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Create a new factory.*
- `l4_msgtag_t create_gate (Cap< void > const &target_cap, Cap< Thread > const &thread_cap, l4_umword_t label, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Create a new IPC gate.*

## Public Member Functions inherited from L4::Kobject

- `l4_msgtag_t dec_refcnt (l4_mword_t diff, l4_utcb_t *utcb=l4_utcb())`  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

### Protected Types inherited from

**L4::Kobject\_t< Mem\_alloc, L4::Factory, L4::PROTO\_EMPTY >**

- `typedef Mem_alloc Class`  
*The target interface type (inheriting from Kobject\_t)*
- `typedef Typeid::Iface< PROTO, Mem_alloc > __iface`  
*The interface description for the derived class.*
- `typedef Typeid::Merge_list< Typeid::Iface_list< __iface >, typename Base::__iface_list > __iface_list`  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Types inherited from L4::Kobject\_t< Factory, Kobject, L4\_PROTO\_FACTORY >

- `typedef Factory Class`  
*The target interface type (inheriting from Kobject\_t)*
- `typedef Typeid::Iface< PROTO, Factory > __iface`  
*The interface description for the derived class.*
- `typedef Typeid::Merge_list< Typeid::Iface_list< __iface >, typename Base::__iface_list > __iface_list`  
*The list of all RPC interfaces provided directly or through inheritance.*

**Protected Member Functions inherited from****L4::Kobject\_t< Mem\_alloc, L4::Factory, L4::PROTO\_EMPTY >**

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

**Protected Member Functions inherited from****L4::Kobject\_t< Factory, Kobject, L4\_PROTO\_FACTORY >**

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

**Protected Member Functions inherited from L4::Kobject**

- [l4\\_cap\\_idx\\_t cap \(\)](#) const noexcept  
*Return capability selector.*

**Static Protected Member Functions inherited from****L4::Kobject\_t< Mem\_alloc, L4::Factory, L4::PROTO\_EMPTY >**

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

**Static Protected Member Functions inherited from****L4::Kobject\_t< Factory, Kobject, L4\_PROTO\_FACTORY >**

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

**15.287.1 Detailed Description**

Memory allocation interface.

The memory-allocator API is the basic API to allocate memory from the [L4Re](#) subsystem. The memory is allocated in terms of dataspace (see [L4Re::Dataspace](#)). The provided dataspace have at least the property that data written to such a dataspace is available as long as the dataspace is not freed or the data is not overwritten. In particular, the memory backing a dataspace from an allocator need not be allocated instantly, but may be allocated lazily on demand.

A memory allocator can provide dataspace with additional properties, such as physically contiguous memory, pre-allocated memory, or pinned memory. To request memory with an additional property the [L4Re::Mem\\_alloc::alloc\(\)](#) method provides a flags parameter. If the concrete implementation of a memory allocator does not support or allow allocation of memory with a certain property, the allocation may be refused.

Definition at line 61 of file [mem\\_alloc](#).

**15.287.2 Member Enumeration Documentation****15.287.2.1 Mem\_alloc\_flags**

```
enum L4Re::Mem_alloc::Mem_alloc_flags
```

Flags for the allocator.

They describe requested properties of the allocated memory. Support of these properties by the dataspace provider is optional.

## Enumerator

Continuous	Allocate physically contiguous memory.
Pinned	Deprecated, use <a href="#">L4Re::Dma_space</a> instead.
Super_pages	Allocate super pages.
Fixed_paddr	Allocate at fixed physical address. Only honored on no-MMU systems. Will fail on MMU systems.

Definition at line 71 of file [mem\\_alloc](#).

### 15.287.3 Member Function Documentation

#### 15.287.3.1 alloc()

```
long L4Re::Mem_alloc::alloc (
    long size,
    L4::Cap< Dataspace > mem,
    unsigned long flags = 0,
    unsigned long align = 0,
    l4_addr_t paddr = 0 ) const [noexcept]
```

Allocate anonymous memory.

## Parameters

	<i>size</i>	Size in bytes to be requested. Allocation granularity is (super)pages, however, the allocator will store the byte-granular given size as the size of the dataspace and consecutively will use this byte-granular size for servicing the dataspace. Allocators may optionally also implement a maximum allocation strategy: if <i>size</i> is a negative value and <i>flags</i> set the <a href="#">Mem_alloc_flags::Continuous</a> bit, the allocator tries to allocate as much memory as possible leaving an amount of at least <i>-size</i> bytes within the associated quota.
out	<i>mem</i>	Capability slot where the capability to the dataspace is received.
	<i>flags</i>	Special dataspace properties, see <a href="#">Mem_alloc_flags</a>
	<i>align</i>	Log2 alignment of dataspace if supported by allocator, will be at least L4_PAGESHIFT, with Super_pages flag set at least L4_SUPERPAGESHIFT
	<i>paddr</i>	The physical address where the dataspace should be allocated if <a href="#">Mem_alloc_flags::Fixed</a> flag is set.

## Return values

0	Success
-L4_ERANGE	Given size not supported.
-L4_ENOMEM	Not enough memory available.
<0	IPC error

Definition at line 35 of file [mem\\_alloc\\_impl.h](#).

References [l4\\_error\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

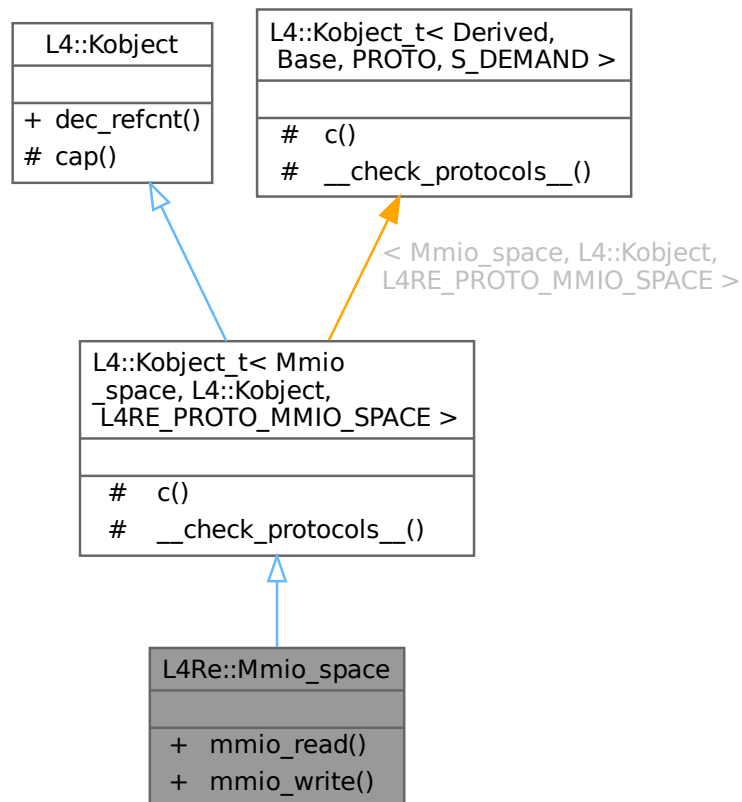
- [l4/re/mem\\_alloc](#)
- [l4/re/impl/mem\\_alloc\\_impl.h](#)

## 15.288 L4Re::Mmio\_space Struct Reference

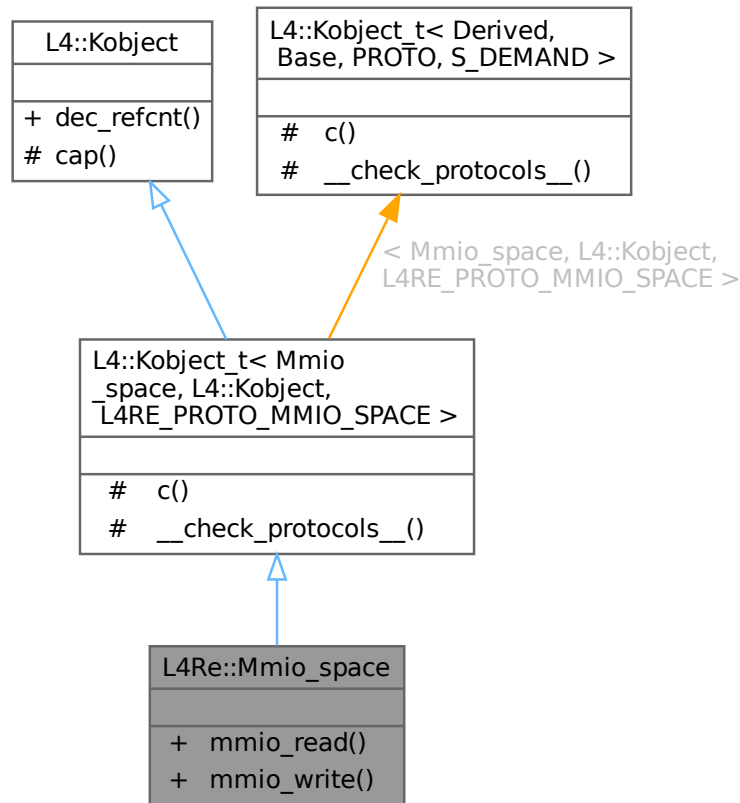
Interface for memory-like address space accessible via IPC.

```
#include <mmio_space>
```

Inheritance diagram for L4Re::Mmio\_space:



Collaboration diagram for L4Re::Mmio\_space:



## Public Types

- enum `Access_width` { `Wd_8bit` = 0 , `Wd_16bit` = 1 , `Wd_32bit` = 2 , `Wd_64bit` = 3 }  
*Actual size of the value to read or write.*
- typedef `l4_uint64_t Addr`  
*Device address.*

## Public Member Functions

- long `mmio_read` (`Addr` addr, char width, `l4_uint64_t` \*value)  
*Read a value from the given address.*
- long `mmio_write` (`Addr` addr, char width, `l4_uint64_t` value)  
*Write a value to the given address.*

## Public Member Functions inherited from L4::Kobject

- `l4_msgtag_t dec_refcnt` (`l4_mword_t` diff, `l4_utcb_t` \*utcb=`l4_utcb()`)  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

### Protected Types inherited from

[L4::Kobject\\_t](#)< [Mmio\\_space](#), [L4::Kobject](#), [L4RE\\_PROTO\\_MMIO\\_SPACE](#) >

- typedef [Mmio\\_space](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef [Typeid::Iface](#)< [PROTO](#), [Mmio\\_space](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef [Typeid::Merge\\_list](#)< [Typeid::Iface\\_list](#)< **\_\_Iface** >, [typename](#) [Base::\\_\\_Iface\\_list](#) > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Mmio\\_space](#), [L4::Kobject](#), [L4RE\\_PROTO\\_MMIO\\_SPACE](#) >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t](#) **cap** () const noexcept  
*Return capability selector.*

### Static Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Mmio\\_space](#), [L4::Kobject](#), [L4RE\\_PROTO\\_MMIO\\_SPACE](#) >

- static void **\_\_check\_protocols\_\_** () noexcept  
*Helper to check for protocol conflicts.*

## 15.288.1 Detailed Description

Interface for memory-like address space accessible via IPC.

This interface defines methods for indirect access to MMIO regions.

Memory mapped IO (MMIO) is used by device drivers to control hardware devices. Access to MMIO regions is assigned to user-level device drivers via mappings of memory pages.

However, there are hardware platforms where MMIO regions for different devices share the same memory page. With respect to security and safety, it is often not allowed to map a memory page to multiple device drivers because the driver of one device could then influence operation of another device, which violates security boundaries.

A solution to that problem is to implement a third (trusted) component that gets exclusive access to the shared memory page, and that drivers can access via IPC with the [Mmio\\_space](#) protocol. This proxy-component can then enforce an access policy.

#### Include File

```
#include <l4/re/mmio_space>
```

Definition at line 46 of file [mmio\\_space](#).

## 15.288.2 Member Enumeration Documentation

### 15.288.2.1 Access\_width

```
enum L4Re::Mmio\_space::Access\_width
```

Actual size of the value to read or write.

## Enumerator

Wd_8bit	Value is a byte.
Wd_16bit	Value is a 2-byte word.
Wd_32bit	Value is a 4-byte word.
Wd_64bit	Value is a 8-byte word.

Definition at line 50 of file [mmio\\_space](#).

## 15.288.3 Member Function Documentation

### 15.288.3.1 mmio\_read()

```
long L4Re::Mmio_space::mmio_read (
    Addr addr,
    char width,
    l4_uint64_t * value )
```

Read a value from the given address.

## Parameters

	<i>addr</i>	Device virtual address to read from. The address must be aligned relative to the access width.
	<i>width</i>	Access width of value to be read, see <a href="#">Access_width</a> .
out	<i>value</i>	Return value. If width is smaller than 64 bit,the upper bits are guaranteed to be 0.

## Return values

<a href="#">L4_EOK</a>	Success.
<a href="#">-L4_EPERM</a>	Insufficient read rights.
<a href="#">-L4_EINVAL</a>	Address does not exist or cannot be accessed with the given width.

### 15.288.3.2 mmio\_write()

```
long L4Re::Mmio_space::mmio_write (
    Addr addr,
    char width,
    l4_uint64_t value )
```

Write a value to the given address.

## Parameters

<i>addr</i>	Device virtual address to write to. The address must be aligned relative to the access width.
<i>width</i>	Access width of value to write, see <a href="#">Access_width</a> .
<i>value</i>	Value to write. If width is smaller than 64 bit, the upper bits are ignored.



## Return values

<a href="#">L4_EOK</a>	Success.
<a href="#">-L4_EPERM</a>	Insufficient write rights.
<a href="#">-L4_EINVAL</a>	Address does not exist or cannot be accessed with the given width.

The documentation for this struct was generated from the following file:

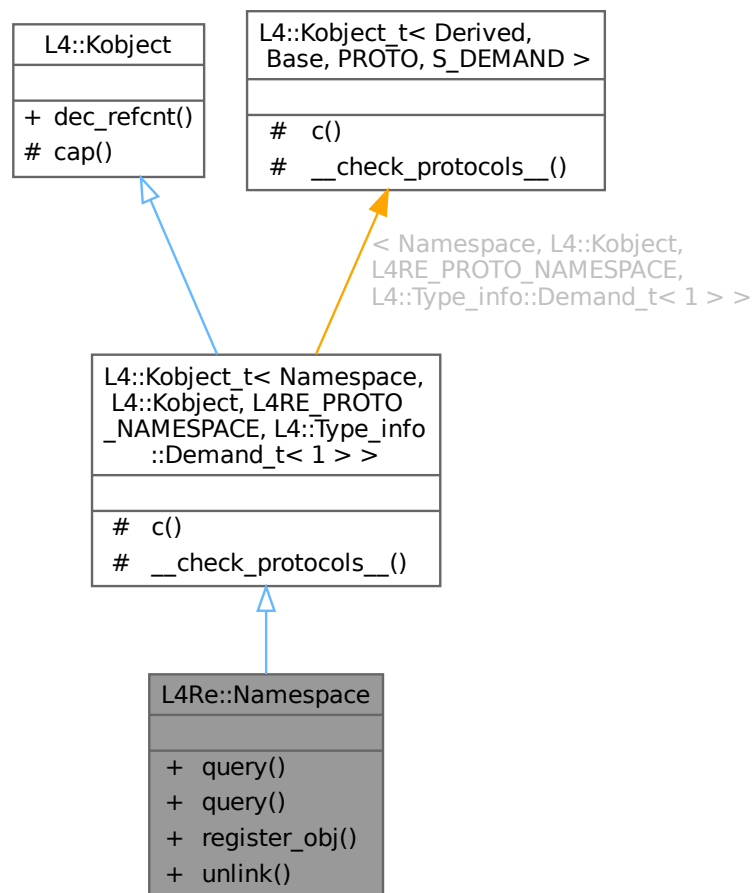
- [l4/re/mmio\\_space](#)

## 15.289 L4Re::Namespace Class Reference

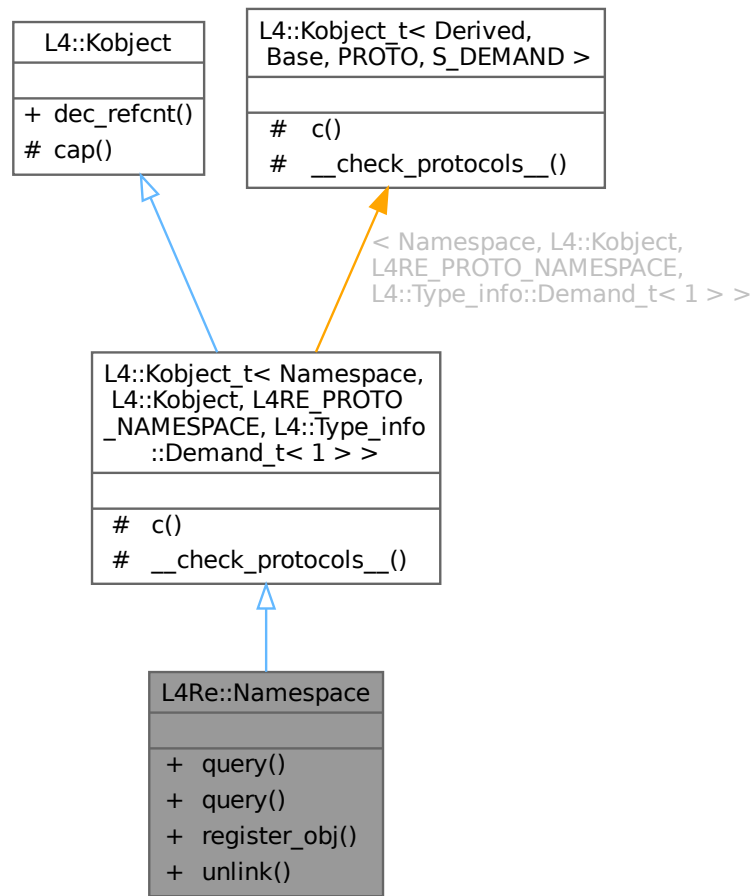
Name-space interface.

```
#include <namespace>
```

Inheritance diagram for L4Re::Namespace:



Collaboration diagram for L4Re::Namespace:



## Public Types

- enum `Register_flags` {  
`Ro` = `L4_CAP_FPAGE_RO` , `Rw` = `L4_CAP_FPAGE_RW` , `Rs` = `L4_CAP_FPAGE_RS` , `Rws` = `L4_CAP_FPAGE_RWS` ,  
`Strong` = `L4_CAP_FPAGE_S` , `Trusted` = `0x008` , `Cap_flags` = `Ro | Rw | Strong | Trusted` , `Link` = `0x100` ,  
`Overwrite` = `0x200` }  
*Flags for registering name spaces.*
- enum `Query_result_flags` { `Partly_resolved` = `0x020` }  
*Flags returned by query IPC, only used internally.*
- enum `Query_timeout` { `To_default` = `3600000` , `To_non_blocking` = `0` }  
*Timeout values for query operation.*

## Public Member Functions

- long `query` (char const \*name, `L4::Cap`< void > const &`cap`, int timeout=`To_default`, `l4_umword_t` \*local\_id=0, bool iterate=true) const noexcept

*Query the name space for a named object.*

- long [query](#) (char const \*name, unsigned len, [L4::Cap](#)< void > const &cap, int timeout=[To\\_default](#), [l4\\_umword\\_t](#) \*local\_id=0, bool iterate=true) const noexcept

*Query the name space for a named object.*

- long [register\\_obj](#) (char const \*name, [L4::lpc::Cap](#)< void > obj, unsigned flags=[Rw](#)) const noexcept

*Register an object with a name.*

- long [unlink](#) (char const \*name)

*Remove an entry from the name space.*

## Public Member Functions inherited from [L4::Kobject](#)

- [l4\\_msgtag\\_t dec\\_refcnt](#) ([l4\\_mword\\_t](#) diff, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)())

*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

## Protected Types inherited from

[L4::Kobject\\_t](#)< [Namespace](#), [L4::Kobject](#), [L4RE\\_PROTO\\_NAMESPACE](#), [L4::Type\\_info::Demand\\_t](#)< 1 >

- typedef [Namespace](#) **Class**

*The target interface type (inheriting from [Kobject\\_t](#))*

- typedef [Typeid::Iface](#)< [PROTO](#), [Namespace](#) > **\_\_Iface**

*The interface description for the derived class.*

- typedef [Typeid::Merge\\_list](#)< [Typeid::Iface\\_list](#)< **\_\_Iface** >, typename [Base::\\_\\_Iface\\_list](#) > **\_\_Iface\_list**

*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Namespace](#), [L4::Kobject](#), [L4RE\\_PROTO\\_NAMESPACE](#), [L4::Type\\_info::Demand\\_t](#)< 1 >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept

*Get the capability to ourselves.*

## Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t cap](#) () const noexcept

*Return capability selector.*

## Static Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Namespace](#), [L4::Kobject](#), [L4RE\\_PROTO\\_NAMESPACE](#), [L4::Type\\_info::Demand\\_t](#)< 1 >

- static void **\_\_check\_protocols** () noexcept

*Helper to check for protocol conflicts.*

## 15.289.1 Detailed Description

Name-space interface.

All name space objects must provide this interface. However, it is not mandatory that a name space object allows to register new capabilities.

The name lookup is done iteratively, this means the hierarchical names are resolved component wise by the client itself.

Definition at line 60 of file [namespace](#).

## 15.289.2 Member Enumeration Documentation

### 15.289.2.1 Query\_result\_flags

```
enum L4Re::Namespace::Query_result_flags
```

Flags returned by query IPC, only used internally.

Enumerator

Partly_resolved	Name was only partly resolved.
-----------------	--------------------------------

Definition at line 88 of file [namespace](#).

### 15.289.2.2 Query\_timeout

```
enum L4Re::Namespace::Query_timeout
```

Timeout values for query operation.

Enumerator

To_default	Default timeout.
To_non_blocking	Expect callee to answer immediately.

Definition at line 94 of file [namespace](#).

### 15.289.2.3 Register\_flags

```
enum L4Re::Namespace::Register_flags
```

Flags for registering name spaces.

Enumerator

Ro	Read-only.
----	------------

## Enumerator

Rw	Read-write.
Rs	Read-only + strong.
Rws	Read-write + strong.
Strong	Strong.
Trusted	Obsolete, do not use.
Link	Obsolete, do not use.
Overwrite	If entry already exists, overwrite it.

Definition at line 68 of file [namespace](#).

### 15.289.3 Member Function Documentation

#### 15.289.3.1 query() [1/2]

```
long L4Re::Namespace::query (
    char const * name,
    L4::Cap< void > const & cap,
    int timeout = To_default,
    l4_umword_t * local_id = 0,
    bool iterate = true ) const [noexcept]
```

Query the name space for a named object.

## Parameters

in	<i>name</i>	String to query (without any leading slashes).
out	<i>cap</i>	Capability slot where the received capability will be put.
in	<i>timeout</i>	Timeout of query in milliseconds. The client will only wait if a name has already been registered with the server but no object has yet been attached.
out	<i>local_id</i>	If given, <a href="#">L4_RCV_ITEM_LOCAL_ID</a> will be set for the IPC from the name space, so that if the capability that was received is a local item, the capability ID will be returned with this parameter.
in	<i>iterate</i>	If true, the client will try to resolve names by iteratively calling the name spaces until the name is fully resolved.

## Return values

0	Name could be fully resolved.
>0	Name could only be partly resolved. The number of remaining characters is returned.
-L4_ENOENT	Entry could not be found.
-L4_EAGAIN	Entry exists but no object is yet attached. Try again later.
<0	IPC errors, see <a href="#">l4_error_code_t</a> .

Definition at line 125 of file [namespace\\_impl.h](#).

**15.289.3.2 query()** [2/2]

```
long L4Re::Namespace::query (
    char const * name,
    unsigned len,
    L4::Cap< void > const & cap,
    int timeout = To_default,
    l4_umword_t * local_id = 0,
    bool iterate = true ) const [noexcept]
```

Query the name space for a named object.

The query string does not necessarily need to be null-terminated.

**Parameters**

in	<i>len</i>	Length of the string to query without any terminating null characters.
in	<i>name</i>	String to query (without any leading slashes).
out	<i>cap</i>	Capability slot where the received capability will be put.
in	<i>timeout</i>	Timeout of query in milliseconds. The client will only wait if a name has already been registered with the server but no object has yet been attached.
out	<i>local_id</i>	If given, <a href="#">L4_RCV_ITEM_LOCAL_ID</a> will be set for the IPC from the name space, so that if the capability that was received is a local item, the capability ID will be returned with this parameter.
in	<i>iterate</i>	If true, the client will try to resolve names by iteratively calling the name spaces until the name is fully resolved.

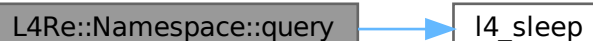
**Return values**

0	Name could be fully resolved.
>0	Name could only be partly resolved. The number of remaining characters is returned.
-L4_ENOENT	Entry could not be found.
-L4_EAGAIN	Entry exists but no object is yet attached. Try again later.
<0	IPC errors, see <a href="#">l4_error_code_t</a> .

Definition at line 77 of file [namespace\\_impl.h](#).

References [L4\\_EAGAIN](#), [L4\\_EINVAL](#), [l4\\_sleep\(\)](#), and [L4\\_UNLIKELY](#).

Here is the call graph for this function:



### 15.289.3.3 register\_obj()

```
long L4Re::Namespace::register_obj (
    char const * name,
    L4::Ipc::Cap< void > obj,
    unsigned flags = Rw ) const [inline], [noexcept]
```

Register an object with a name.

#### Parameters

<i>name</i>	Name under which the object should be registered.
<i>obj</i>	Capability to object to register. An invalid capability may be given to only reserve the name for later use.
<i>flags</i>	Flags to assign to the entry, see <a href="#">L4Re::Namespace::Register_flags</a> . Note that the rights that are assigned to a capability are not only determined by the rights given in these flags but also by the rights with which the <code>obj</code> capability was mapped to the name space.

#### Return values

0	Object was successfully registered with <i>name</i> .
-L4_EEXIST	Name already registered.
-L4_EPERM	Caller does not have <a href="#">L4_CAP_FPAGE_W</a> right on the invoked capability.
-L4_ENOMEM	Server has insufficient resources.
-L4_EINVAL	Invalid parameter.
<0	IPC errors, see <a href="#">l4_error_code_t</a> .

#### Precondition

requires capability rights: {RW}

Definition at line 176 of file [namespace](#).

### 15.289.3.4 unlink()

```
long L4Re::Namespace::unlink (
    char const * name ) [inline]
```

Remove an entry from the name space.

#### Parameters

<i>name</i>	Name of the entry to remove.
-------------	------------------------------

#### Return values

0	Entry successfully removed.
-L4_ENOENT	Given name does not exist.
-L4_EPERM	Caller does not have <a href="#">L4_CAP_FPAGE_W</a> right on the invoked capability.
-L4_EACCESS	Name cannot be removed.

## Return values

<code>&lt; 0</code>	IPC errors, see <a href="#">l4_error_code_t</a> .
---------------------	---

## Precondition

requires capability rights: {RW}

Definition at line 203 of file [namespace](#).

The documentation for this class was generated from the following files:

- [l4/re/namespace](#)
- [l4/re/impl/namespace\\_impl.h](#)

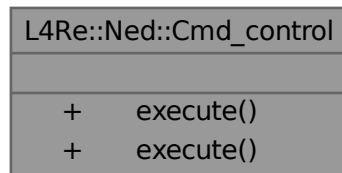
## 15.290 L4Re::Ned::Cmd\_control Class Reference

Direct control interface for Ned.

```
#include <cmd_control>
```

Inherits L4::Kobject\_0t< Derived, PROTO, S\_DEMAND >.

Collaboration diagram for L4Re::Ned::Cmd\_control:



### Public Member Functions

- long [execute](#) (L4::lpc::String<> cmd) noexcept  
*Execute the given Lua code.*
- long [execute](#) (L4::lpc::String<> cmd, L4::lpc::String< char > \*result) noexcept  
*Execute the given Lua code.*

### 15.290.1 Detailed Description

Direct control interface for Ned.

Definition at line 20 of file [cmd\\_control](#).



## 15.290.2 Member Function Documentation

### 15.290.2.1 execute() [1/2]

```
long L4Re::Ned::Cmd_control::execute (  
    L4::Ipc::String<> cmd ) [inline], [noexcept]
```

Execute the given Lua code.

**Parameters**

in	<i>cmd</i>	String with Lua code to execute.
----	------------	----------------------------------

**Return values**

<i>L4_EOK</i>	Code was successfully executed.
<i>-L4_EINVAL</i>	Code could not be parsed.
<i>-L4_EIO</i>	Error during code execution.

The code is executed using the global Lua state of ned which is retained between successive calls to execute. Thus you may define data in one call to execute and use it in a subsequent call.

This function does not return any results from the execution of the Lua code itself.

Definition at line 43 of file [cmd\\_control](#).

**15.290.2.2 execute() [2/2]**

```
long L4Re::Ned::Cmd_control::execute (
    L4::Ipc::String<> cmd,
    L4::Ipc::String< char > * result ) [inline], [noexcept]
```

Execute the given Lua code.

**Parameters**

in	<i>cmd</i>	String with Lua code to execute.
out	<i>result</i>	The first return value of the Lua code block as string.

**Return values**

<i>L4_EOK</i>	Code was successfully executed.
<i>-L4_EINVAL</i>	Code could not be parsed.
<i>-L4_EIO</i>	Error during code execution.

The code is executed using the global Lua state of ned which is retained between successive calls to execute. Thus you may define data in one call to execute and use it in a subsequent call.

Definition at line 65 of file [cmd\\_control](#).

The documentation for this class was generated from the following file:

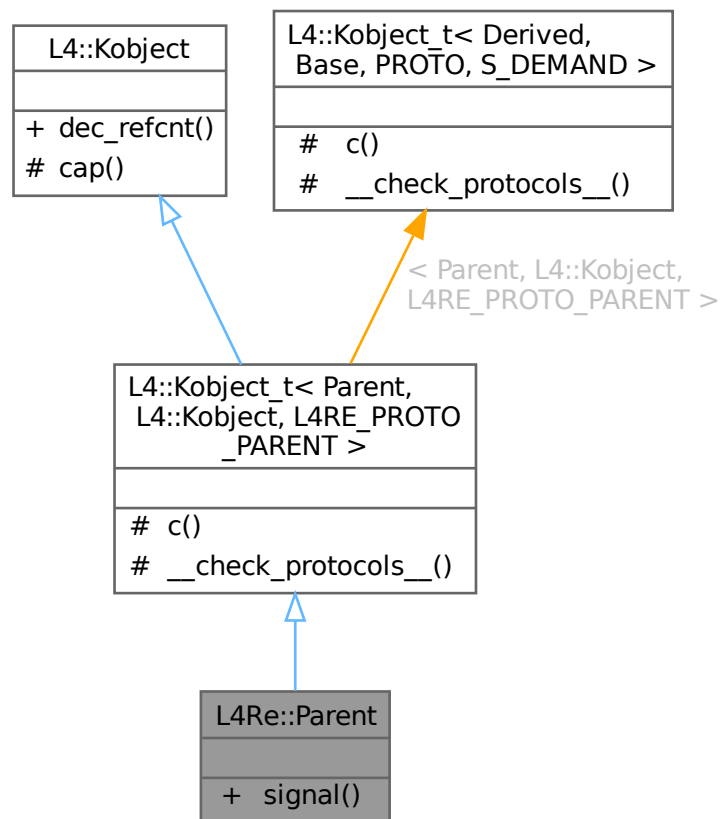
- pkg/l4re-core/ned/lib/include/cmd\_control

**15.291 L4Re::Parent Class Reference**

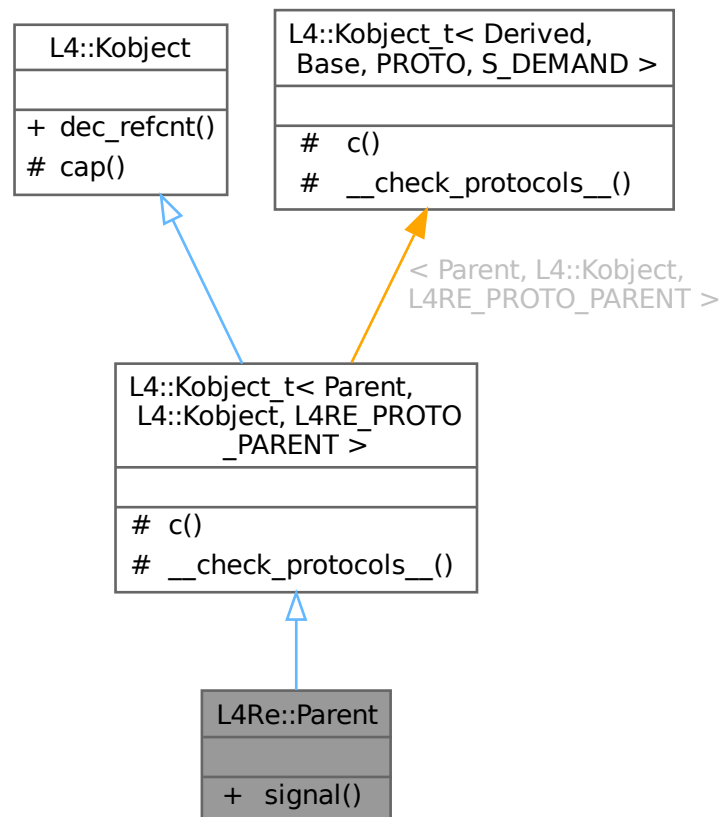
[Parent](#) interface.

```
#include <parent>
```

Inheritance diagram for L4Re::Parent:



Collaboration diagram for L4Re::Parent:



### Public Member Functions

- long [signal](#) (unsigned long sig, unsigned long val)  
*Send a signal to the parent.*

### Public Member Functions inherited from [L4::Kobject](#)

- [l4\\_msgtag\\_t dec\\_refcnt](#) ([l4\\_mword\\_t](#) diff, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#))  
*Decrement the in kernel reference counter for the object.*

### Additional Inherited Members

### Protected Types inherited from

#### [L4::Kobject\\_t< Parent, L4::Kobject, L4RE\\_PROTO\\_PARENT >](#)

- typedef Parent **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, Parent > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_Iface](#) >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from [L4::Kobject\\_t< Parent, L4::Kobject, L4RE\\_PROTO\\_PARENT >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

## Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t cap \(\)](#) const noexcept  
*Return capability selector.*

## Static Protected Member Functions inherited from [L4::Kobject\\_t< Parent, L4::Kobject, L4RE\\_PROTO\\_PARENT >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

### 15.291.1 Detailed Description

[Parent](#) interface.

See also

[Parent API](#) for more details about the purpose.

Definition at line 53 of file [parent](#).

### 15.291.2 Member Function Documentation

#### 15.291.2.1 [signal\(\)](#)

```
long L4Re::Parent::signal (
    unsigned long sig,
    unsigned long val )
```

Send a signal to the parent.

#### Parameters

<i>sig</i>	Signal to send
<i>val</i>	Value of the signal

#### Return values

0	Success
<0	IPC error

**Note**

The implementations of this interface in Moe and Ned only recognize the signal 0, in which case they will terminate the application from which the interface was invoked and not return. In this case, `val` is treated as the application's return code. For any other value of `sig`, the method just returns successfully.

The documentation for this class was generated from the following file:

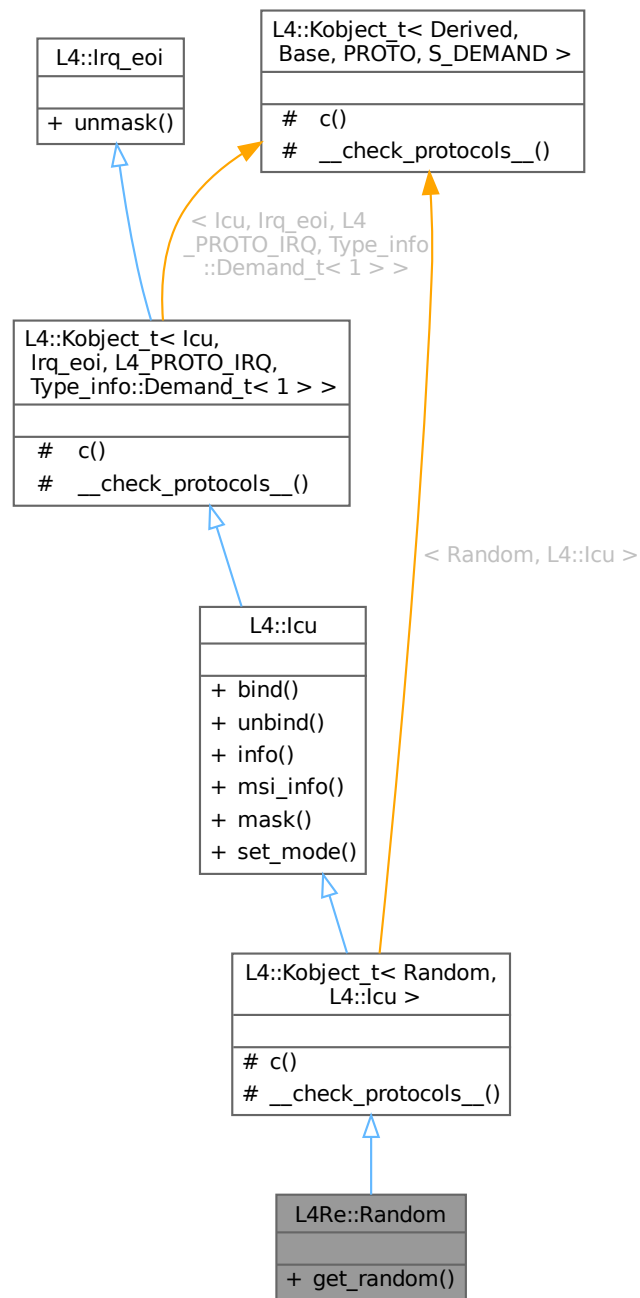
- [l4/re/parent](#)

## 15.292 L4Re::Random Struct Reference

Low-bandwidth interface for random number generators.

```
#include <random>
```

Inheritance diagram for L4Re::Random:







- Bind an interrupt line of an interrupt controller to an interrupt object.*

• [l4\\_msgtag\\_t unbind](#) (unsigned irqnum, [L4::Cap](#)< [Triggerable](#) > irq, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept

*Remove binding of an interrupt line from the interrupt controller object.*
- [l4\\_msgtag\\_t info](#) ([l4\\_icu\\_info\\_t](#) \*info, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept

*Get information about the ICU features.*
- [l4\\_msgtag\\_t msi\\_info](#) ([l4\\_umword\\_t](#) irqnum, [l4\\_uint64\\_t](#) source, [l4\\_icu\\_msi\\_info\\_t](#) \*msi\_info)

*Get MSI info about IRQ.*
- [l4\\_msgtag\\_t mask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept

*Mask an IRQ line.*
- [l4\\_msgtag\\_t set\\_mode](#) (unsigned irqnum, [l4\\_umword\\_t](#) mode, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept

*Set interrupt mode.*

## Public Member Functions inherited from [L4::lrq\\_eoi](#)

- [l4\\_msgtag\\_t unmask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept

*Unmask the given interrupt line.*

## Additional Inherited Members

## Protected Types inherited from [L4::Kobject\\_t](#)< [Random](#), [L4::lcu](#) >

- typedef [Random](#) **Class**

*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< [PROTO\\_ANY](#), [Random](#) > **\_\_Iface**

*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**

*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Types inherited from [L4::Kobject\\_t](#)< [lcu](#), [lrq\\_eoi](#), [L4\\_PROTO\\_IRQ](#), [Type\\_info::Demand\\_t](#)< 1 > >

- typedef [lcu](#) **Class**

*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< [PROTO](#), [lcu](#) > **\_\_Iface**

*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**

*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from [L4::Kobject\\_t](#)< [Random](#), [L4::lcu](#) >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept

*Get the capability to ourselves.*

### Protected Member Functions inherited from

[L4::Kobject\\_t< Icu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept

*Get the capability to ourselves.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< Random, L4::Icu >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept

*Helper to check for protocol conflicts.*

### Static Protected Member Functions inherited from

[L4::Kobject\\_t< Icu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept

*Helper to check for protocol conflicts.*

## 15.292.1 Detailed Description

Low-bandwidth interface for random number generators.

The interface offers an ICU interface where a client can register an interrupt to get notified when entropy is available. Support for notifications is optional. If a service does not implement notification, it must return 0 for the number of interrupts in the [info\(\)](#) call. The notification interrupt must have index 0.

#### Include File

```
#include <l4/re/random>
```

Definition at line 33 of file [random](#).

## 15.292.2 Member Function Documentation

### 15.292.2.1 [get\\_random\(\)](#)

```
long L4Re::Random::get_random (
    l4_size_t size,
    L4::Ipc::Array< char, unsigned long > * buffer )
```

Get a random number.

#### Parameters

	<i>size</i>	Number of bytes of entropy requested.
out	<i>buffer</i>	Buffer containing the random number. Each byte in the buffer contains 8 bits of randomness.

## Return values

$\geq 0$	Actual size of the returned random number in bytes. This may be less than the requested size. The return value may also be 0 if temporarily no entropy is available.
<code>-L4_EIO</code>	Source of randomness permanently unavailable.
$< 0$	IPC error.

This function should never block. It should immediately return as much entropy as is available. If the call returns less than the requested bytes and a notification interrupt was installed, then the service triggers an interrupt as soon as the remaining entropy is available. That means that when an interrupt is triggered, the service must guarantee that the next call to [get\\_random\(\)](#) returns at least the number of missing bytes for the call that initially triggered the notification.

If [get\\_random\(\)](#) is called while a notification is pending, then the behaviour is implementation-defined.

The documentation for this struct was generated from the following file:

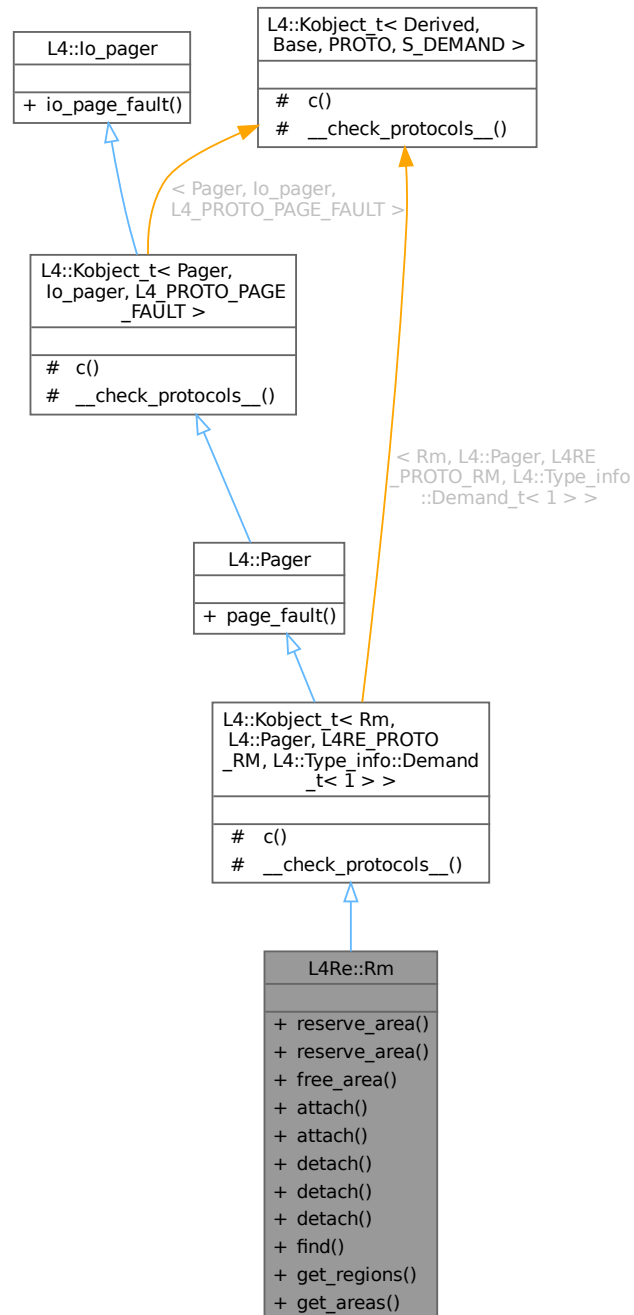
- [l4/re/random](#)

## 15.293 L4Re::Rm Class Reference

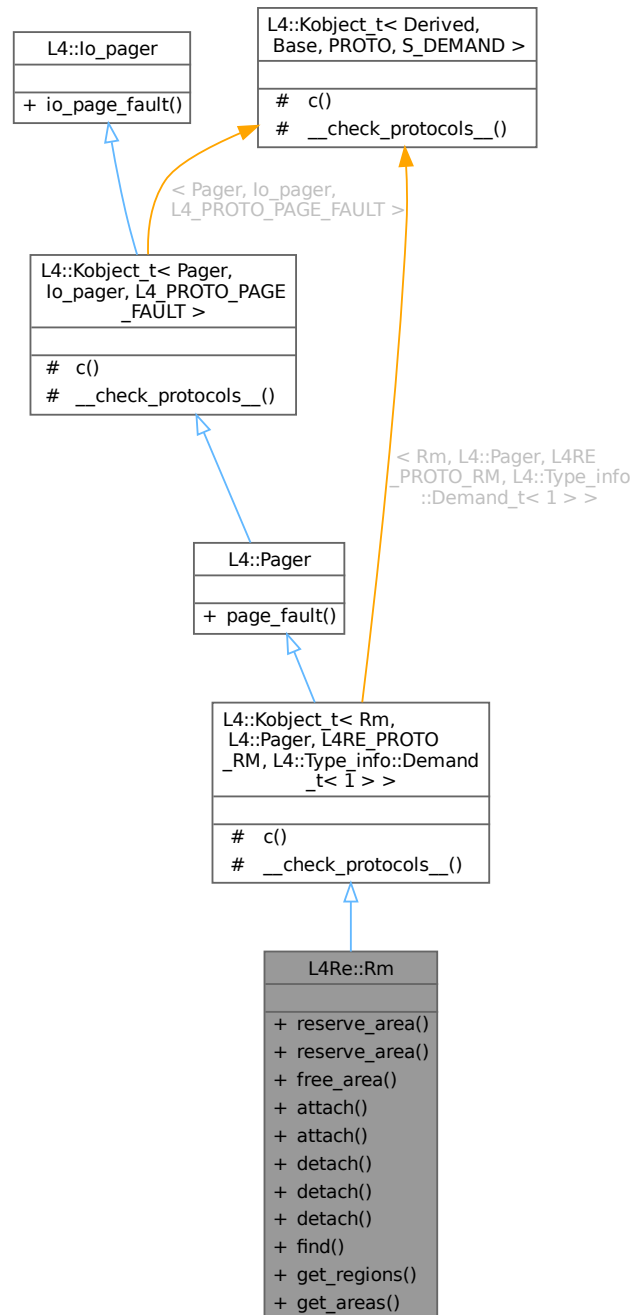
Region map.

```
#include <l4/re/rm>
```

Inheritance diagram for L4Re::Rm:



Collaboration diagram for L4Re::Rm:



## Data Structures

- struct [F](#)  
*Rm flags definitions.*
- struct [Range](#)  
*A range of virtual addresses.*
- class [Unique\\_region](#)  
*Unique region.*

## Public Types

- enum `Detach_result` { `Detached_ds` = 0 , `Kept_ds` = 1 , `Split_ds` = 2 , `Detach_result_mask` = 3 , `Detach_again` = 4 }
- Result values for detach operation.*
- enum `Region_flag_shifts` { `Caching_shift` = `Dataspace::F::Caching_shift` }
- Region flag shifts.*
- enum `Detach_flags` { `Detach_exact` = 1 , `Detach_overlap` = 2 , `Detach_keep` = 4 }
- Flags for detach operation.*
- using `Region` = `Range`
- A region is a range of virtual addresses which is backed by a dataspace.*
- using `Area` = `Range`
- An area is a range of virtual addresses which is reserved, see `L4Re::Rm::reserve_area()`.*

## Public Member Functions

- long `reserve_area` (`l4_addr_t` \*start, unsigned long size, Flags flags=`Flags(0)`, unsigned char align=`L4_PAGESHIFT`) const noexcept
- Reserve the given area in the region map.*
- template<typename T >  
long `reserve_area` (T \*\*start, unsigned long size, Flags flags=`Flags(0)`, unsigned char align=`L4_PAGESHIFT`) const noexcept
- Reserve the given area in the region map.*
- long `free_area` (`l4_addr_t` addr)
- Free an area from the region map.*
- long `attach` (`l4_addr_t` \*start, unsigned long size, Flags flags, `L4::lpc::Cap`< `Dataspace` > mem, Offset offs=0, unsigned char align=`L4_PAGESHIFT`, `L4::Cap`< `L4::Task` > const task=`L4::Cap`< `L4::Task` >::`Invalid`) const noexcept
- Attach a data space to a region.*
- template<typename T >  
long `attach` (T \*\*start, unsigned long size, Flags flags, `L4::lpc::Cap`< `Dataspace` > mem, Offset offs=0, unsigned char align=`L4_PAGESHIFT`, `L4::Cap`< `L4::Task` > const task=`L4::Cap`< `L4::Task` >::`Invalid`) const noexcept
- Attach a data space to a region.*
- int `detach` (`l4_addr_t` addr, `L4::Cap`< `Dataspace` > \*mem, `L4::Cap`< `L4::Task` > const &task=`This_task`) const noexcept
- Detach and unmap a region from the address space.*
- int `detach` (void \*addr, `L4::Cap`< `Dataspace` > \*mem, `L4::Cap`< `L4::Task` > const &task=`This_task`) const noexcept
- Detach and unmap a region from the address space.*
- int `detach` (`l4_addr_t` start, unsigned long size, `L4::Cap`< `Dataspace` > \*mem, `L4::Cap`< `L4::Task` > const &task) const noexcept
- Detach and unmap all parts of the regions within the specified interval.*
- long `find` (`l4_addr_t` \*addr, unsigned long \*size, Offset \*offset, `L4Re::Rm::Flags` \*flags, `L4::Cap`< `Dataspace` > \*m) noexcept
- Find a region given an address and size.*
- long `get_regions` (`l4_addr_t` start, `L4::lpc::Ret_array`< `Range` > regions)
- Return the list of regions whose starting addresses are higher or equal to `start` in the address space managed by this region map.*
- long `get_areas` (`l4_addr_t` start, `L4::lpc::Ret_array`< `Range` > areas)
- Return the list of areas whose starting addresses are higher or equal to `start` in the address space managed by this region map.*

## Public Member Functions inherited from [L4::Pager](#)

- [l4\\_msgtag\\_t page\\_fault](#) ([l4\\_umword\\_t](#) pfa, [l4\\_umword\\_t](#) pc, [L4::lpc::Rcv\\_fpage](#) rwin, [L4::lpc::Opt<L4::lpc::Snd\\_fpage & > fp](#))

*Page-fault protocol message.*

## Public Member Functions inherited from [L4::io\\_pager](#)

- [l4\\_msgtag\\_t io\\_page\\_fault](#) ([l4\\_fpage\\_t](#) io\_pfa, [l4\\_umword\\_t](#) pc, [L4::lpc::Rcv\\_fpage](#) rwin, [L4::lpc::Opt<L4::lpc::Snd\\_fpage & > fp](#))

*IO page fault protocol message.*

## Additional Inherited Members

### Protected Types inherited from

[L4::Kobject\\_t< Rm, L4::Pager, L4RE\\_PROTO\\_RM, L4::Type\\_info::Demand\\_t< 1 > >](#)

- typedef Rm **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, Rm > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Types inherited from

[L4::Kobject\\_t< Pager, io\\_pager, L4\\_PROTO\\_PAGE\\_FAULT >](#)

- typedef [Pager](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, [Pager](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from

[L4::Kobject\\_t< Rm, L4::Pager, L4RE\\_PROTO\\_RM, L4::Type\\_info::Demand\\_t< 1 > >](#)

- [L4::Cap< Class > c](#) () const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from

[L4::Kobject\\_t< Pager, io\\_pager, L4\\_PROTO\\_PAGE\\_FAULT >](#)

- [L4::Cap< Class > c](#) () const noexcept  
*Get the capability to ourselves.*

**Static Protected Member Functions inherited from****[L4::Kobject\\_t< Rm, L4::Pager, L4RE\\_PROTO\\_RM, L4::Type\\_info::Demand\\_t< 1 > >](#)**

- static void **`__check_protocols__`** () noexcept  
*Helper to check for protocol conflicts.*

**Static Protected Member Functions inherited from****[L4::Kobject\\_t< Pager, Io\\_pager, L4\\_PROTO\\_PAGE\\_FAULT >](#)**

- static void **`__check_protocols__`** () noexcept  
*Helper to check for protocol conflicts.*

**15.293.1 Detailed Description**

Region map.

See also

[Region map API](#) .

Definition at line 92 of file [rm](#).

**15.293.2 Member Typedef Documentation****15.293.2.1 Area**

```
using L4Re::Rm::Area = Range
```

An area is a range of virtual addresses which is reserved, see [L4Re::Rm::reserve\\_area\(\)](#).

See also

[Region map API](#)

Definition at line 701 of file [rm](#).

**15.293.2.2 Region**

```
using L4Re::Rm::Region = Range
```

A region is a range of virtual addresses which is backed by a dataspace.

See also

[Region map API](#)

Definition at line 693 of file [rm](#).

**15.293.3 Member Enumeration Documentation****15.293.3.1 Detach\_flags**

```
enum L4Re::Rm::Detach\_flags
```

Flags for detach operation.



## Enumerator

Detach_exact	Do an unmap of the exact region given.
Detach_overlap	Do an unmap of all overlapping regions.
Detach_keep	Do not free the detached data space, ignore the <a href="#">F::Detach_free</a> .

Definition at line 229 of file [rm](#).

## 15.293.3.2 Detach\_result

enum [L4Re::Rm::Detach\\_result](#)

Result values for detach operation.

## Enumerator

Detached_ds	Detached data sapce.
Kept_ds	Kept data space.
Split_ds	Splitted data space, and done.
Detach_again	Detached data space, more to do.

Definition at line 100 of file [rm](#).

## 15.293.3.3 Region\_flag\_shifts

enum [L4Re::Rm::Region\\_flag\\_shifts](#)

Region flag shifts.

## Enumerator

Caching_shift	Start of <a href="#">Rm</a> cache bits.
---------------	---

Definition at line 112 of file [rm](#).

## 15.293.4 Member Function Documentation

## 15.293.4.1 attach() [1/2]

```
long L4Re::Rm::attach (
    l4\_addr\_t * start,
    unsigned long size,
    Rm::Flags flags,
    L4::Ipc::Cap< Dataspace > mem,
    Rm::Offset offs = 0,
    unsigned char align = L4\_PAGESHIFT,
    L4::Cap< L4::Task > const task = L4::Cap<L4::Task>::Invalid ) const [noexcept]
```

Attach a data space to a region.

## Parameters

<i>in, out</i>	<i>start</i>	Virtual start address where the region manager shall attach the data space. Will be rounded down to the nearest start of a page. If <a href="#">L4Re::Rm::F::Search_addr</a> is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If <a href="#">L4Re::Rm::F::In_area</a> is given the value is used as a selector for the area (see <a href="#">L4Re::Rm::reserve_area</a> ) to attach the data space to.
	<i>size</i>	Size of the data space to attach (in bytes). Will be rounded up to the nearest multiple of the page size.
	<i>flags</i>	The flags control how and with which rights the dataspace is attached to the region. See <a href="#">L4Re::Rm::F::Attach_flags</a> and <a href="#">L4Re::Rm::F::Region_flags</a> . The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the <a href="#">F::Eager_map</a> flag is set this function may also return <a href="#">L4Re::Dataspace::map</a> error codes if the mapping fails.
	<i>mem</i>	Data space.
	<i>offs</i>	Offset into the data space to use.
	<i>align</i>	Alignment of the virtual region, log2-size, default: a page ( <a href="#">L4_PAGESHIFT</a> ). This is only meaningful if the <a href="#">L4Re::Rm::F::Search_addr</a> flag is used.
	<i>task</i>	Optional destination task of mapping if <a href="#">F::Eager_map</a> flag was passed. If invalid, the mapping is established in the current task. This parameter is only useful if the region manager is for a foreign task.

## Return values

<i>0</i>	Success
<i>-L4_ENOENT</i>	No area could be found (see <a href="#">L4Re::Rm::F::In_area</a> )
<i>-L4_EPERM</i>	Operation not allowed.
<i>-L4_EINVAL</i>	
<i>-L4_EADDRNOTAVAIL</i>	The given address is not available.
<i>&lt;0</i>	IPC errors

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

## Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see [L4Re::Rm::find](#)).

Definition at line 44 of file [rm\\_impl.h](#).

## 15.293.4.2 attach() [2/2]

```
template<typename T>
long L4Re::Rm::attach (
    T ** start,
    unsigned long size,
    Flags flags,
    L4::Ipc::Cap< Dataspace > mem,
```

```

    Offset offs = 0,
    unsigned char align = L4_PAGESHIFT,
    L4::Cap< L4::Task > const task = L4::Cap<L4::Task>::Invalid ) const [inline],
[noexcept]

```

Attach a data space to a region.

#### Parameters

<i>in, out</i>	<i>start</i>	Virtual start address where the region manager shall attach the data space. Will be rounded down to the nearest start of a page. If <a href="#">L4Re::Rm::F::Search_addr</a> is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If <a href="#">L4Re::Rm::F::In_area</a> is given the value is used as a selector for the area (see <a href="#">L4Re::Rm::reserve_area</a> ) to attach the data space to.
	<i>size</i>	Size of the data space to attach (in bytes). Will be rounded up to the nearest multiple of the page size.
	<i>flags</i>	The flags control how and with which rights the dataspace is attached to the region. See <a href="#">L4Re::Rm::F::Attach_flags</a> and <a href="#">L4Re::Rm::F::Region_flags</a> . The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the <a href="#">F::Eager_map</a> flag is set this function may also return <a href="#">L4Re::Dataspace::map</a> error codes if the mapping fails.
	<i>mem</i>	Data space.
	<i>offs</i>	Offset into the data space to use.
	<i>align</i>	Alignment of the virtual region, log2-size, default: a page ( <a href="#">L4_PAGESHIFT</a> ). This is only meaningful if the <a href="#">L4Re::Rm::F::Search_addr</a> flag is used.
	<i>task</i>	Optional destination task of mapping if <a href="#">F::Eager_map</a> flag was passed. If invalid, the mapping is established in the current task. This parameter is only useful if the region manager is for a foreign task.

#### Return values

<i>0</i>	Success
<i>-L4_ENOENT</i>	No area could be found (see <a href="#">L4Re::Rm::F::In_area</a> )
<i>-L4_EPERM</i>	Operation not allowed.
<i>-L4_EINVAL</i>	
<i>-L4_EADDRNOTAVAIL</i>	The given address is not available.
<i>&lt;0</i>	IPC errors

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

#### Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see [L4Re::Rm::find](#)).

Definition at line 411 of file [rm](#).

**15.293.4.3 detach()** [1/3]

```
int L4Re::Rm::detach (
    l4_addr_t addr,
    L4::Cap< Dataspace > * mem,
    L4::Cap< L4::Task > const & task = This_task ) const [inline], [noexcept]
```

Detach and unmap a region from the address space.

**Parameters**

	<i>addr</i>	Virtual address of region, any address within the region is valid.
out	<i>mem</i>	<a href="#">Dataspace</a> that is affected. Give 0 if not interested.
	<i>task</i>	This argument specifies the task where the pages are unmapped. Provide <a href="#">L4::Cap&lt;L4::Task&gt;::Invalid</a> for none. The default is the current task.

**Return values**

<a href="#">L4Re::Rm::Detach_result</a>	On success.
<a href="#">-L4_ENOENT</a>	No region found.
<a href="#">&lt;0</a>	IPC errors

Frees a region in the virtual address space given by *addr* (address type). The corresponding part of the address space is now available again.

Definition at line 746 of file [rm](#).

**15.293.4.4 detach()** [2/3]

```
int L4Re::Rm::detach (
    l4_addr_t start,
    unsigned long size,
    L4::Cap< Dataspace > * mem,
    L4::Cap< L4::Task > const & task ) const [inline], [noexcept]
```

Detach and unmap all parts of the regions within the specified interval.

**Parameters**

	<i>start</i>	Start of area to detach, must be within region.
	<i>size</i>	Size of of area to detach (in bytes).
out	<i>mem</i>	<a href="#">Dataspace</a> that is affected. Give 0 if not interested.
	<i>task</i>	This argument specifies the task where the pages are unmapped. Provide <a href="#">L4::Cap&lt;L4::Task&gt;::Invalid</a> for none. The default is the current task.

**Return values**

<a href="#">L4Re::Rm::Detach_result</a>	On success.
<a href="#">-L4_ENOENT</a>	No region found.
<a href="#">&lt;0</a>	IPC errors

Frees all regions within the interval given by start and size. If a region overlaps the start or the end of the interval this region is only detached partly. If the interval is within one region the original region is split up into two separate regions.

Definition at line 759 of file [rm](#).

#### 15.293.4.5 detach() [3/3]

```
int L4Re::Rm::detach (
    void * addr,
    L4::Cap< Dataspace > * mem,
    L4::Cap< L4::Task > const & task = This_task ) const [inline], [noexcept]
```

Detach and unmap a region from the address space.

##### Parameters

	<i>addr</i>	Virtual address of region, any address within the region is valid.
out	<i>mem</i>	<a href="#">Dataspace</a> that is affected. Give 0 if not interested.
	<i>task</i>	This argument specifies the task where the pages are unmapped. Provide <a href="#">L4::Cap&lt;L4::Task&gt;::Invalid</a> for none. The default is the current task.

##### Return values

<a href="#">L4Re::Rm::Detach_result</a>	On success.
<a href="#">-L4_ENOENT</a>	No region found.
<0	IPC errors

Frees a region in the virtual address space given by addr (address type). The corresponding part of the address space is now available again.

Definition at line 751 of file [rm](#).

#### 15.293.4.6 find()

```
long L4Re::Rm::find (
    l4_addr_t * addr,
    unsigned long * size,
    Offset * offset,
    L4Re::Rm::Flags * flags,
    L4::Cap< Dataspace > * m ) [inline], [noexcept]
```

Find a region given an address and size.

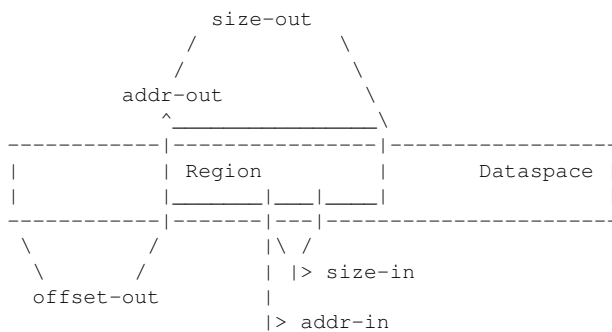
##### Parameters

in, out	<i>addr</i>	Address to look for. Returns the start address of the found region.
in, out	<i>size</i>	Size of the area to look for (in bytes). Returns the size of the found region (in bytes).
out	<i>offset</i>	Offset at the beginning of the region within the associated dataspace.
out	<i>flags</i>	Region flags, see <a href="#">F::Region_flags</a> (and <a href="#">F::In_area</a> ).
out	<i>m</i>	Associated dataspace or paging service.

## Return values

0	Success
-L4_EPERM	Operation not allowed.
-L4_ENOENT	No region found.
<0	IPC errors

This function returns the properties of the region that contains the area described by the `addr` and `size` parameter. If no such region is found but a reserved area, the area is returned and `F::ln_area` is set in `flags`. Note, in the case of an area the `offset` and `m` return values are invalid.



## Note

The value of the size input parameter should be 1 to assure that a region can be determined unambiguously.

Definition at line 668 of file `rm`.

## 15.293.4.7 free\_area()

```
long L4Re::Rm::free_area (
    l4_addr_t addr )
```

Free an area from the region map.

## Parameters

<i>addr</i>	An address within the area to free.
-------------	-------------------------------------

## Return values

0	Success
-L4_ENOENT	No area found.
<0	IPC errors

## Note

The data spaces that are attached to that area are not detached by this operation.

See also

[reserve\\_area\(\)](#) for more information about areas.

#### 15.293.4.8 get\_areas()

```
long L4Re::Rm::get_areas (
    l4_addr_t start,
    L4::Ipc::Ret_array< Range > areas )
```

Return the list of areas whose starting addresses are higher or equal to `start` in the address space managed by this region map.

##### Parameters

	<i>start</i>	Virtual address from where to start searching.
out	<i>areas</i>	List of areas found in this region map.

##### Return values

<code>&gt;=0</code>	Number of returned areas in the <code>areas</code> array.
<code>&lt;0</code>	IPC errors

##### Note

The returned list of areas might not be complete and the caller shall use the function repeatedly with a start address one larger than the end address of the last area from the previous call.

#### 15.293.4.9 get\_regions()

```
long L4Re::Rm::get_regions (
    l4_addr_t start,
    L4::Ipc::Ret_array< Range > regions )
```

Return the list of regions whose starting addresses are higher or equal to `start` in the address space managed by this region map.

##### Parameters

	<i>start</i>	Virtual address from where to start searching.
out	<i>regions</i>	List of regions found in this region map.

##### Return values

<code>&gt;=0</code>	Number of returned regions in the <code>regions</code> array.
<code>&lt;0</code>	IPC errors

**Note**

The returned list of regions might not be complete and the caller shall use the function repeatedly with a start address one larger than the end address of the last region from the previous call.

**15.293.4.10 reserve\_area() [1/2]**

```
long L4Re::Rm::reserve_area (
    l4_addr_t * start,
    unsigned long size,
    Flags flags = Flags(0),
    unsigned char align = L4_PAGESHIFT ) const [inline], [noexcept]
```

Reserve the given area in the region map.

**Parameters**

<i>in, out</i>	<i>start</i>	The virtual start address of the area to reserve. Returns the start address of the area.
	<i>size</i>	The size of the area to reserve (in bytes).
	<i>flags</i>	Flags for the reserved area (see <a href="#">L4Re::Rm::F::Region_flags</a> and <a href="#">L4Re::Rm::F::Attach_flags</a> ).
	<i>align</i>	Alignment of area if searched as bits (log2 value).

**Return values**

0	Success
-L4_EADDRNOTAVAIL	The given area cannot be reserved.
<0	IPC errors

This function reserves an area within the virtual address space managed by the region map. There are two kinds of areas available:

- Reserved areas (*flags* = [L4Re::Rm::F::Reserved](#)), where no data spaces can be attached
- Special purpose areas (*flags* = 0), where data spaces can be attached to the area via the [L4Re::Rm::F::In\\_area](#) flag and a start address within the area itself.

**Note**

When searching for a free place in the virtual address space (with *flags* = [L4Re::Rm::F::Search\\_addr](#)), the space between *start* and the end of the virtual address space is searched.

Definition at line [288](#) of file [rm](#).

**15.293.4.11 reserve\_area() [2/2]**

```
template<typename T >
long L4Re::Rm::reserve_area (
    T ** start,
    unsigned long size,
    Flags flags = Flags(0),
    unsigned char align = L4_PAGESHIFT ) const [inline], [noexcept]
```

Reserve the given area in the region map.



## Parameters

<i>in, out</i>	<i>start</i>	The virtual start address of the area to reserve. Returns the start address of the area.
	<i>size</i>	The size of the area to reserve (in bytes).
	<i>flags</i>	Flags for the reserved area (see <a href="#">F::Region_flags</a> and <a href="#">F::Attach_flags</a> ).
	<i>align</i>	Alignment of area if searched as bits (log2 value).

## Return values

<i>0</i>	Success
<i>-L4_EADDRNOTAVAIL</i>	The given area cannot be reserved.
<i>&lt;0</i>	IPC errors

For more information, please refer to the analogous function

## See also

[L4Re::Rm::reserve\\_area](#).

Definition at line [314](#) of file [rm](#).

The documentation for this class was generated from the following files:

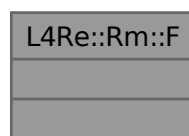
- [l4/re/rm](#)
- [l4/re/impl/rm\\_impl.h](#)

## 15.294 L4Re::Rm::F Struct Reference

[Rm](#) flags definitions.

```
#include <rm>
```

Collaboration diagram for L4Re::Rm::F:



## Public Types

- enum [Attach\\_flags](#) : `l4_uint32_t` {  
[Search\\_addr](#) = 0x20000 , [In\\_area](#) = 0x40000 , [Eager\\_map](#) = 0x80000 , [No\\_eager\\_map](#) = 0x100000 ,  
[Attach\\_mask](#) = 0x1f0000 }  
*Flags for attach operation.*
- enum [Region\\_flags](#) : `l4_uint16_t` {  
[Rights\\_mask](#) = 0x0f , [R](#) = `Dataspace::F::R` , [W](#) = `Dataspace::F::W` , [X](#) = `Dataspace::F::X` ,  
[RW](#) = `Dataspace::F::RW` , [RX](#) = `Dataspace::F::RX` , [RWX](#) = `Dataspace::F::RWX` , [Detach\\_free](#) = 0x200 ,  
[Pager](#) = 0x400 , [Reserved](#) = 0x800 , [Caching\\_mask](#) = `Dataspace::F::Caching_mask` , [Cache\\_normal](#) =  
`Dataspace::F::Normal` ,  
[Cache\\_buffered](#) = `Dataspace::F::Bufferable` , [Cache\\_uncached](#) = `Dataspace::F::Uncacheable` , [Ds\\_map\\_mask](#)  
= 0xff , [Region\\_flags\\_mask](#) = 0xffff }  
*Region flags (permissions, cacheability, special).*

### 15.294.1 Detailed Description

[Rm](#) flags definitions.

Definition at line 119 of file [rm](#).

### 15.294.2 Member Enumeration Documentation

#### 15.294.2.1 [Attach\\_flags](#)

```
enum L4Re::Rm::F::Attach\_flags : l4\_uint32\_t
```

Flags for attach operation.

##### Enumerator

<a href="#">Search_addr</a>	Search for a suitable address range.
<a href="#">In_area</a>	Search only in area, or map into area.
<a href="#">Eager_map</a>	Eagerly map the attached data space in.
<a href="#">No_eager_map</a>	Prevent eager mapping of the attached data space.
<a href="#">Attach_mask</a>	Mask of all attach flags.

Definition at line 122 of file [rm](#).

#### 15.294.2.2 [Region\\_flags](#)

```
enum L4Re::Rm::F::Region\_flags : l4\_uint16\_t
```

Region flags (permissions, cacheability, special).

##### Enumerator

<a href="#">Rights_mask</a>	Region rights.
<a href="#">R</a>	Readable region.

## Enumerator

W	Writable region.
X	Executable region.
RW	Readable and writable region.
RX	Readable and executable region.
RWX	Readable, writable and executable region.
Detach_free	Free the portion of the data space after detach.
Pager	Region has a pager.
Reserved	Region is reserved (blocked)
Caching_mask	Mask of all <a href="#">Rm</a> cache bits.
Cache_normal	Cache bits for normal cacheable memory.
Cache_buffered	Cache bits for buffered (write combining) memory.
Cache_uncached	Cache bits for uncached memory.
Ds_map_mask	Mask for all bits for cache options and rights.
Region_flags_mask	Mask of all region flags.

Definition at line [139](#) of file [rm](#).

The documentation for this struct was generated from the following file:

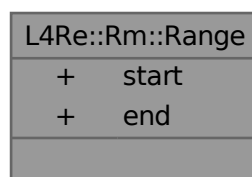
- [l4/re/rm](#)

## 15.295 L4Re::Rm::Range Struct Reference

A range of virtual addresses.

```
#include <rm>
```

Collaboration diagram for L4Re::Rm::Range:



### Data Fields

- [l4\\_addr\\_t](#) **start**  
*First address of the range.*
- [l4\\_addr\\_t](#) **end**  
*Last address of the range.*

## 15.295.1 Detailed Description

A range of virtual addresses.

Definition at line 680 of file [rm](#).

The documentation for this struct was generated from the following file:

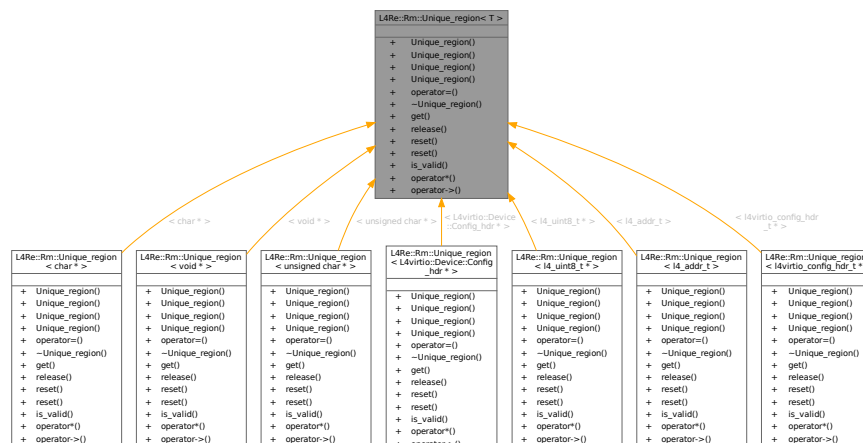
- [l4/re/rm](#)

## 15.296 L4Re::Rm::Unique\_region< T > Class Template Reference

Unique region.

```
#include <rm>
```

Inheritance diagram for L4Re::Rm::Unique\_region< T >:



Collaboration diagram for L4Re::Rm::Unique\_region< T >:

L4Re::Rm::Unique_region< T >	
+	Unique_region()
+	Unique_region()
+	Unique_region()
+	Unique_region()
+	operator=()
+	~Unique_region()
+	get()
+	release()
+	reset()
+	reset()
+	is_valid()
+	operator*()
+	operator->()

## Public Member Functions

- **Unique\_region** () noexcept  
*Construct an invalid [Unique\\_region](#)*
- **Unique\_region** (T addr) noexcept  
*Construct a [Unique\\_region](#) from an address.*
- **Unique\_region** (T addr, L4::Cap< Rm > const &rm) noexcept  
*Construct a valid [Unique\\_region](#) from an address and a region manager.*
- **Unique\_region** (Unique\_region &&o) noexcept  
*Move-Construct a [Unique\\_region](#)*
- **Unique\_region** & operator= (Unique\_region &&o) noexcept  
*Move-assign a [Unique\\_region](#)*
- **~Unique\_region** () noexcept  
*Destructor.*
- T **get** () const noexcept  
*Return the address.*
- T **release** () noexcept  
*Return the address and invalidate the [Unique\\_region](#)*
- void **reset** (T addr, L4::Cap< Rm > const &rm) noexcept  
*Set new address and region manager.*
- void **reset** () noexcept  
*Make the [Unique\\_region](#) invalid.*
- bool **is\_valid** () const noexcept  
*Check if the [Unique\\_region](#) is valid.*

- **T operator\*** () const noexcept  
*Dereference the address.*
- **T operator->** () const noexcept  
*Member access for the address.*

### 15.296.1 Detailed Description

```
template<typename T>
class L4Re::Rm::Unique_region< T >
```

Unique region.

Capture a single region with automatic detach on destruction and unique ownership. Stores the start address and the region-mapper capability internally. A unique region is valid precisely if the internal region-mapper capability is valid. The features for unique ownership and automatic detach are only active for valid unique regions.

Definition at line [434](#) of file [rm](#).

### 15.296.2 Constructor & Destructor Documentation

#### 15.296.2.1 Unique\_region() [1/3]

```
template<typename T >
L4Re::Rm::Unique_region< T >::Unique_region (
    T addr ) [inline], [explicit], [noexcept]
```

Construct a [Unique\\_region](#) from an address.

No region manager is set.

Parameters

<i>addr</i>	The new address
-------------	-----------------

Definition at line [455](#) of file [rm](#).

#### 15.296.2.2 Unique\_region() [2/3]

```
template<typename T >
L4Re::Rm::Unique_region< T >::Unique_region (
    T addr,
    L4::Cap< Rm > const & rm ) [inline], [noexcept]
```

Construct a valid [Unique\\_region](#) from an address and a region manager.

Parameters

<i>addr</i>	The address
<i>rm</i>	The region manager

Definition at line 464 of file [rm](#).

### 15.296.2.3 Unique\_region() [3/3]

```
template<typename T >
L4Re::Rm::Unique_region< T >::Unique_region (
    Unique_region< T > && o ) [inline], [noexcept]
```

Move-Construct a [Unique\\_region](#)

#### Parameters

<i>o</i>	L-value reference to other region.
----------	------------------------------------

Definition at line 472 of file [rm](#).

### 15.296.2.4 ~Unique\_region()

```
template<typename T >
L4Re::Rm::Unique_region< T >::~~Unique_region ( ) [inline], [noexcept]
```

Destructor.

If the region is valid, call `detach`.

Definition at line 497 of file [rm](#).

References [L4::Cap\\_base::is\\_valid\(\)](#).

Here is the call graph for this function:



## 15.296.3 Member Function Documentation

### 15.296.3.1 get()

```
template<typename T >
T L4Re::Rm::Unique_region< T >::get ( ) const [inline], [noexcept]
```

Return the address.

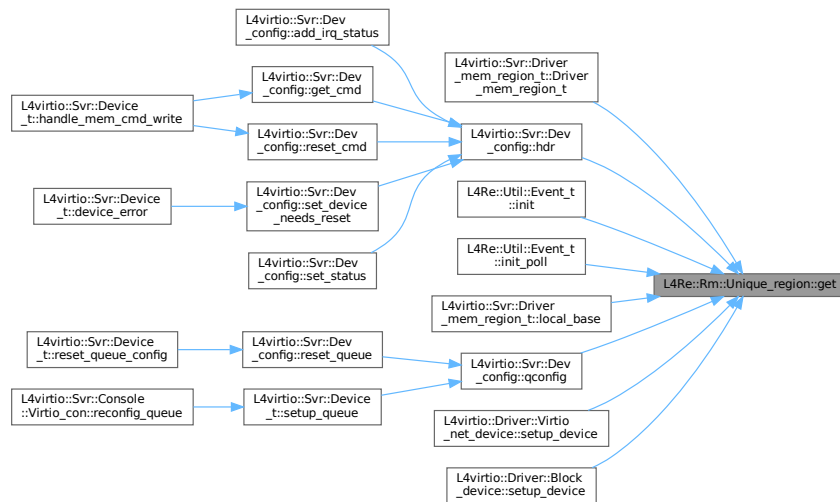
**Returns**

the address

Definition at line 508 of file [rm](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_region\\_t< DATA >::Driver\\_mem\\_region\\_t\(\)](#), [L4virtio::Svr::Dev\\_config::hdr\(\)](#), [L4Re::Util::Event\\_t< PAYLOAD >::init\(\)](#), [L4Re::Util::Event\\_t< PAYLOAD >::init\\_poll\(\)](#), [L4virtio::Svr::Driver\\_mem\\_region\\_t< DATA >::L4virtio::Svr::Dev\\_config::qconfig\(\)](#), [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#), and [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#).

Here is the caller graph for this function:

**15.296.3.2 is\_valid()**

```

template<typename T >
bool L4Re::Rm::Unique_region< T >::is_valid ( ) const [inline], [noexcept]

```

Check if the [Unique\\_region](#) is valid.

**Returns**

true iff the [Unique\\_region](#) is valid

Definition at line 548 of file [rm](#).

References [L4::Cap\\_base::is\\_valid\(\)](#).

Here is the call graph for this function:





### 15.296.3.3 operator=()

```
template<typename T >
Unique_region & L4Re::Rm::Unique_region< T >::operator= (
    Unique_region< T > && o ) [inline], [noexcept]
```

Move-assign a [Unique\\_region](#)

#### Parameters

<i>o</i>	L-value reference to region to assign from
----------	--

Definition at line [480](#) of file [rm](#).

References [L4::Cap\\_base::is\\_valid\(\)](#).

Here is the call graph for this function:



### 15.296.3.4 release()

```
template<typename T >
T L4Re::Rm::Unique_region< T >::release ( ) [inline], [noexcept]
```

Return the address and invalidate the [Unique\\_region](#)

#### Returns

the address

Definition at line [516](#) of file [rm](#).

### 15.296.3.5 reset()

```
template<typename T >
void L4Re::Rm::Unique_region< T >::reset (
    T addr,
    L4::Cap< Rm > const & rm ) [inline], [noexcept]
```

Set new address and region manager.

## Parameters

<i>addr</i>	The new address
<i>rm</i>	The new region manager

Definition at line 528 of file [rm](#).

References [L4::Cap\\_base::is\\_valid\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

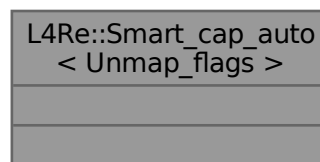
- [l4/re/rm](#)

## 15.297 L4Re::Smart\_cap\_auto< Unmap\_flags > Class Template Reference

Helper for Unique\_cap and Unique\_del\_cap.

```
#include <cap_alloc>
```

Collaboration diagram for L4Re::Smart\_cap\_auto< Unmap\_flags >:



### 15.297.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Smart_cap_auto< Unmap_flags >
```

Helper for Unique\_cap and Unique\_del\_cap.

Definition at line 147 of file [cap\\_alloc](#).

The documentation for this class was generated from the following file:

- [l4/re/cap\\_alloc](#)

## 15.298 L4Re::Smart\_count\_cap< Unmap\_flags > Class Template Reference

Helper for Ref\_cap and Ref\_del\_cap.

```
#include <cap_alloc>
```

Collaboration diagram for L4Re::Smart\_count\_cap< Unmap\_flags >:

L4Re::Smart_count_cap < Unmap_flags >	
+	free()
+	copy()
+	invalidate()

### Public Member Functions

- void **free** ([L4::Cap\\_base](#) &c) noexcept  
*Free operation for [L4::Smart\\_cap](#) (decrement ref count and delete if 0).*
- [L4::Cap\\_base](#) **copy** ([L4::Cap\\_base](#) const &src)  
*Copy operation for [L4::Smart\\_cap](#) (increment ref count).*

### Static Public Member Functions

- static void **invalidate** ([L4::Cap\\_base](#) &c) noexcept  
*Invalidate operation for [L4::Smart\\_cap](#).*

### 15.298.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Smart_count_cap< Unmap_flags >
```

Helper for Ref\_cap and Ref\_del\_cap.

Definition at line 176 of file [cap\\_alloc](#).

The documentation for this class was generated from the following file:

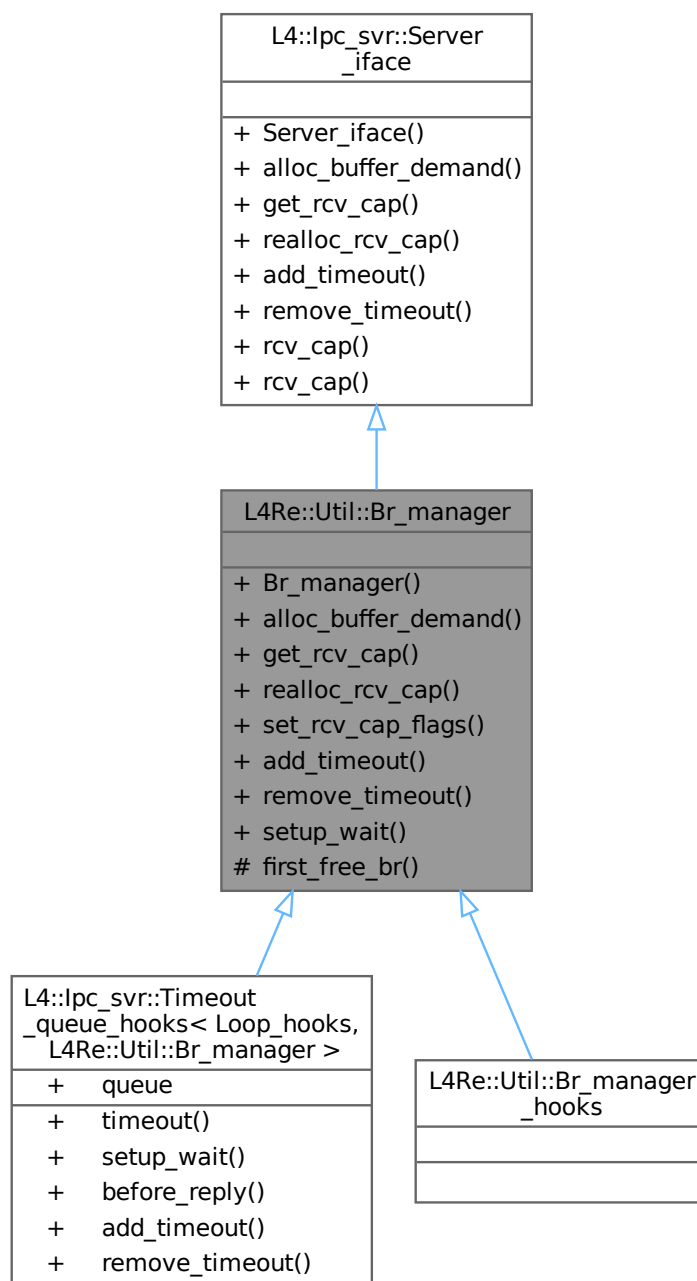
- [l4/re/cap\\_alloc](#)

## 15.299 L4Re::Util::Br\_manager Class Reference

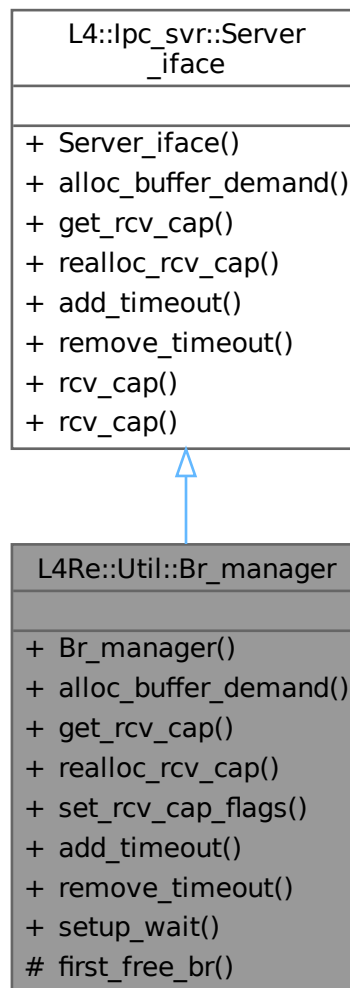
Buffer-register (BR) manager for [L4::Server](#).

```
#include <br_manager>
```

Inheritance diagram for L4Re::Util::Br\_manager:



Collaboration diagram for L4Re::Util::Br\_manager:



## Public Member Functions

- **Br\_manager** ()  
*Make a buffer-register (BR) manager.*
- int **alloc\_buffer\_demand** (**Demand** const &d) override  
*Tells the server to allocate buffers for the given demand.*
- **L4::Cap**< void > **get\_rcv\_cap** (int i) const override  
*Get capability slot allocated to the given receive buffer.*
- int **realloc\_rcv\_cap** (int i) override  
*Allocate a new capability for the given receive buffer.*
- void **set\_rcv\_cap\_flags** (unsigned long flags)  
*Set the receive flags for the buffers.*
- int **add\_timeout** (**L4::lpc\_svr::Timeout** \*, **l4\_kernel\_clock\_t**) override  
*No timeouts handled by us.*

- int **remove\_timeout** ([L4::lpc\\_svr::Timeout](#) \*) override  
*No timeouts handled by us.*
- void **setup\_wait** ([l4\\_utcb\\_t](#) \*utcb, [L4::lpc\\_svr::Reply\\_mode](#))  
*setup\_wait() used the server loop ([L4::Server](#))*

## Public Member Functions inherited from [L4::lpc\\_svr::Server\\_iface](#)

- **Server\_iface** ()  
*Make a server interface.*
- template<typename T >  
[L4::Cap](#)< T > **rcv\_cap** (int index) const  
*Get given receive buffer as typed capability.*
- [L4::Cap](#)< void > **rcv\_cap** (int index) const  
*Get receive cap with the given index as generic (void) type.*

## Protected Member Functions

- unsigned **first\_free\_br** () const  
*Used for assigning BRs for a timeout.*

## Additional Inherited Members

## Public Types inherited from [L4::lpc\\_svr::Server\\_iface](#)

- typedef [L4::Type\\_info::Demand](#) **Demand**  
*Data type expressing server-side demand for receive buffers.*

### 15.299.1 Detailed Description

Buffer-register (BR) manager for [L4::Server](#).

Implementation of the [L4::lpc\\_svr::Server\\_iface](#) API for managing the server-side receive buffers needed for a set of server objects running within a server.

Definition at line 36 of file [br\\_manager](#).

### 15.299.2 Member Function Documentation

#### 15.299.2.1 [alloc\\_buffer\\_demand\(\)](#)

```
int L4Re::Util::Br_manager::alloc_buffer_demand (
    Demand const & demand ) [inline], [override], [virtual]
```

Tells the server to allocate buffers for the given demand.

## Parameters

<i>demand</i>	The total server-side demand of receive buffers needed for a given interface, see Demand.
---------------	---

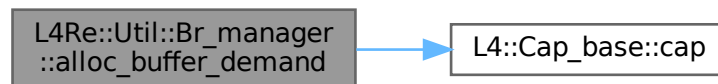
This function is not called by user applications directly. Usually the server implementation or the registry implementation calls this function whenever a new object is registered at the server.

Implements [L4::lpc\\_svr::Server\\_iface](#).

Definition at line 65 of file [br\\_manager](#).

References [L4::Cap\\_base::cap\(\)](#), [L4Re::Util::cap\\_alloc](#), [L4::Type\\_info::Demand::caps](#), [L4\\_EINVAL](#), [L4\\_ENOMEM](#), [L4\\_EOK](#), [L4\\_ERANGE](#), [L4::Type\\_info::Demand::mem](#), and [L4::Type\\_info::Demand::ports](#).

Here is the call graph for this function:



### 15.299.2.2 get\_rcv\_cap()

```
L4::Cap< void > L4Re::Util::Br_manager::get_rcv_cap (
    int index ) const [inline], [override], [virtual]
```

Get capability slot allocated to the given receive buffer.

## Parameters

<i>index</i>	The receive buffer index of the expected capability argument ( $0 \leq \text{index} < \text{caps}$ registered with <a href="#">alloc_buffer_demand()</a> ).
--------------	---

## Precondition

$0 \leq \text{index} < \text{caps}$  registered with [alloc\\_buffer\\_demand\(\)](#)

## Returns

Capability slot currently allocated to the given receive buffer.

Implements [L4::lpc\\_svr::Server\\_iface](#).

Definition at line 97 of file [br\\_manager](#).

References [L4\\_CAP\\_MASK](#).



**15.299.2.3 realloc\_rcv\_cap()**

```
int L4Re::Util::Br_manager::realloc_rcv_cap (
    int index ) [inline], [override], [virtual]
```

Allocate a new capability for the given receive buffer.

**Parameters**

<i>index</i>	The receive buffer index of the expected capability argument ( $0 \leq \text{index} < \text{caps}$ registered with <a href="#">alloc_buffer_demand()</a> ).
--------------	---

**Precondition**

$0 \leq \text{index} < \text{caps}$  registered with [alloc\\_buffer\\_demand\(\)](#)

**Returns**

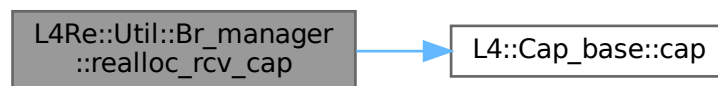
0 on success, < 0 on error.

Implements [L4::lpc\\_svr::Server\\_iface](#).

Definition at line 105 of file [br\\_manager](#).

References [L4::Cap\\_base::cap\(\)](#), [L4Re::Util::cap\\_alloc](#), [L4\\_EINVAL](#), [L4\\_ENOMEM](#), and [L4\\_EOK](#).

Here is the call graph for this function:

**15.299.2.4 set\_rcv\_cap\_flags()**

```
void L4Re::Util::Br_manager::set_rcv_cap_flags (
    unsigned long flags ) [inline]
```

Set the receive flags for the buffers.

**Precondition**

Must be called before any handlers are registered.

## Parameters

<i>flags</i>	New receive capability flags, see <a href="#">l4_msg_item_consts_t</a> .
--------------	--

Definition at line 129 of file [br\\_manager](#).

References [l4\\_assert](#).

The documentation for this class was generated from the following file:

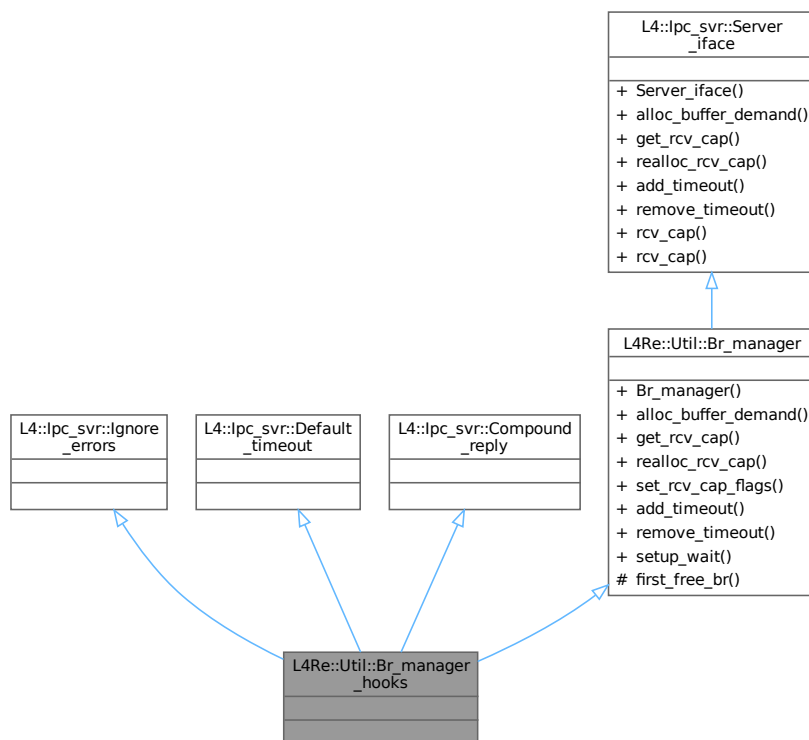
- [l4/re/util/br\\_manager](#)

## 15.300 L4Re::Util::Br\_manager\_hooks Struct Reference

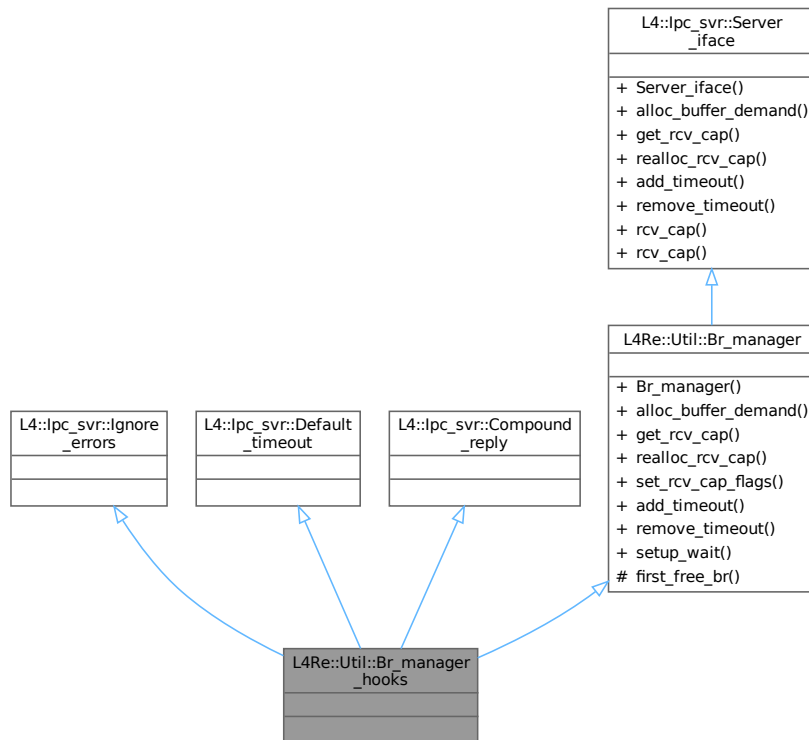
Predefined server-loop hooks for a server loop using the [Br\\_manager](#).

```
#include <br_manager>
```

Inheritance diagram for L4Re::Util::Br\_manager\_hooks:



Collaboration diagram for L4Re::Util::Br\_manager\_hooks:



### Additional Inherited Members

### Public Types inherited from L4::lpc\_svr::Server\_iface

- typedef L4::Type\_info::Demand Demand  
*Data type expressing server-side demand for receive buffers.*

### Public Member Functions inherited from L4Re::Util::Br\_manager

- **Br\_manager** ()  
*Make a buffer-register (BR) manager.*
- int **alloc\_buffer\_demand** (Demand const &d) override  
*Tells the server to allocate buffers for the given demand.*
- L4::Cap< void > **get\_rcv\_cap** (int i) const override  
*Get capability slot allocated to the given receive buffer.*
- int **realloc\_rcv\_cap** (int i) override  
*Allocate a new capability for the given receive buffer.*
- void **set\_rcv\_cap\_flags** (unsigned long flags)  
*Set the receive flags for the buffers.*
- int **add\_timeout** (L4::lpc\_svr::Timeout \*, l4\_kernel\_clock\_t) override  
*No timeouts handled by us.*
- int **remove\_timeout** (L4::lpc\_svr::Timeout \*) override  
*No timeouts handled by us.*
- void **setup\_wait** (l4\_utcb\_t \*utcb, L4::lpc\_svr::Reply\_mode)  
*setup\_wait() used the server loop (L4::Server)*

## Public Member Functions inherited from [L4::lpc\\_svr::Server\\_iface](#)

- **Server\_iface** ()  
*Make a server interface.*
- `template<typename T >`  
[L4::Cap](#)< T > [rcv\\_cap](#) (int index) const  
*Get given receive buffer as typed capability.*
- [L4::Cap](#)< void > [rcv\\_cap](#) (int index) const  
*Get receive cap with the given index as generic (void) type.*

## Protected Member Functions inherited from [L4Re::Util::Br\\_manager](#)

- unsigned **first\_free\_br** () const  
*Used for assigning BRs for a timeout.*

### 15.300.1 Detailed Description

Predefined server-loop hooks for a server loop using the [Br\\_manager](#).

This class can be used whenever a server loop including full management of receive buffer resources is needed.

Definition at line 176 of file [br\\_manager](#).

The documentation for this struct was generated from the following file:

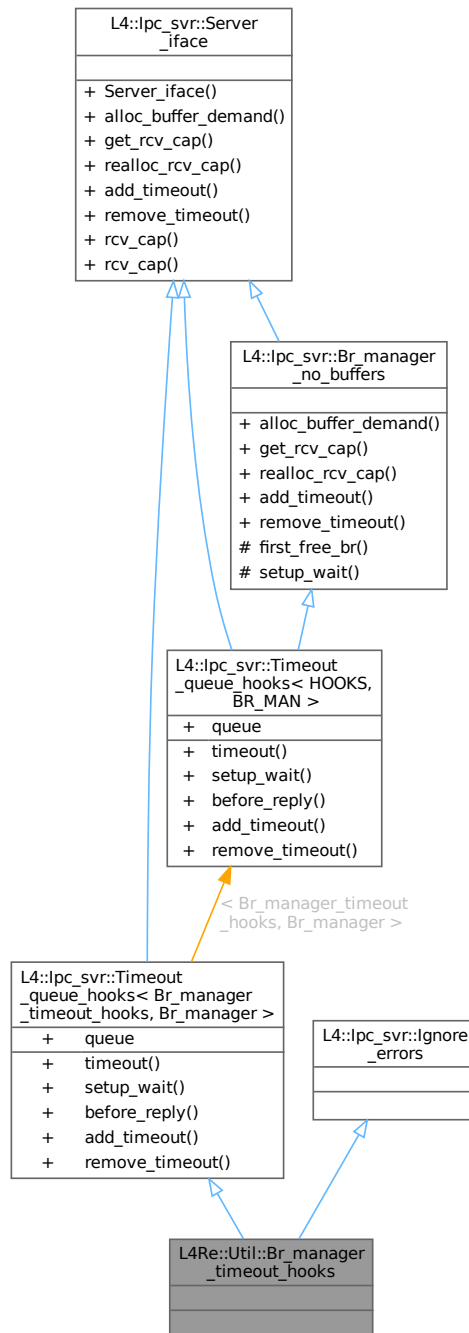
- `l4/re/util/br_manager`

## 15.301 [L4Re::Util::Br\\_manager\\_timeout\\_hooks](#) Struct Reference

Predefined server-loop hooks for a server with using the [Br\\_manager](#) and a timeout queue.

```
#include <br_manager>
```

Inheritance diagram for L4Re::Util::Br\_manager\_timeout\_hooks:





**Public Member Functions inherited from****L4::lpc\_svr::Timeout\_queue\_hooks< Br\_manager\_timeout\_hooks, Br\_manager >**

- **l4\_timeout\_t** **timeout** ()  
*get the time for the next timeout*
- void **setup\_wait** (l4\_utcb\_t \*utcb, L4::lpc\_svr::Reply\_mode mode)  
*setup\_wait() for the server loop*
- L4::lpc\_svr::Reply\_mode **before\_reply** (l4\_msgtag\_t, l4\_utcb\_t \*)  
*server loop hook*
- int **add\_timeout** (Timeout \*timeout, l4\_kernel\_clock\_t time) override  
*Add a timeout to the queue for time time.*
- int **remove\_timeout** (Timeout \*timeout) override  
*Remove timeout from the queue.*

**Public Member Functions inherited from L4::lpc\_svr::Server\_iface**

- **Server\_iface** ()  
*Make a server interface.*
- virtual int **alloc\_buffer\_demand** (Demand const &demand)=0  
*Tells the server to allocate buffers for the given demand.*
- virtual L4::Cap< void > **get\_rcv\_cap** (int index) const =0  
*Get capability slot allocated to the given receive buffer.*
- virtual int **realloc\_rcv\_cap** (int index)=0  
*Allocate a new capability for the given receive buffer.*
- template<typename T >  
L4::Cap< T > **rcv\_cap** (int index) const  
*Get given receive buffer as typed capability.*
- L4::Cap< void > **rcv\_cap** (int index) const  
*Get receive cap with the given index as generic (void) type.*

**Data Fields inherited from****L4::lpc\_svr::Timeout\_queue\_hooks< Br\_manager\_timeout\_hooks, Br\_manager >**

- **Timeout\_queue** **queue**  
*Use this timeout queue.*

**15.301.1 Detailed Description**

Predefined server-loop hooks for a server with using the [Br\\_manager](#) and a timeout queue.

This class can be used for server loops that need the full package of buffer-register management and a timeout queue.

Definition at line 190 of file [br\\_manager](#).

The documentation for this struct was generated from the following file:

- [l4/re/util/br\\_manager](#)

## 15.302 L4Re::Util::Cap\_alloc\_base Class Reference

Capability allocator.

```
#include <bitmap_cap_alloc>
```

Inherited by L4Re::Util::Cap\_alloc< Size >.

Collaboration diagram for L4Re::Util::Cap\_alloc\_base:

L4Re::Util::Cap_alloc_base	
+	alloc()
+	free()

### Public Member Functions

- template<typename T >  
[L4::Cap](#)< T > **alloc** () noexcept  
*Allocate a capability slot.*
- template<typename T >  
void **free** ([L4::Cap](#)< T > const &cap, [l4\\_cap\\_idx\\_t](#) task=[L4\\_INVALID\\_CAP](#), [l4\\_umword\\_t](#) unmap\_↔  
flags=[L4\\_FP\\_ALL\\_SPACES](#)) noexcept  
*Free a capability slot.*

### 15.302.1 Detailed Description

Capability allocator.

Definition at line 39 of file [bitmap\\_cap\\_alloc](#).

The documentation for this class was generated from the following file:

- [l4/re/util/bitmap\\_cap\\_alloc](#)

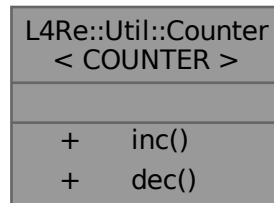


## 15.303 L4Re::Util::Counter< COUNTER > Struct Template Reference

Counter for [Counting\\_cap\\_alloc](#) with variable data width.

```
#include <counting_cap_alloc>
```

Collaboration diagram for L4Re::Util::Counter< COUNTER >:



### Public Member Functions

- bool [inc](#) ()  
*Increment counter if not yet saturated.*
- Type [dec](#) ()  
*Decrement counter if not saturated.*

### 15.303.1 Detailed Description

```
template<typename COUNTER = unsigned char>
struct L4Re::Util::Counter< COUNTER >
```

Counter for [Counting\\_cap\\_alloc](#) with variable data width.

This version is not thread safe.

Definition at line 38 of file [counting\\_cap\\_alloc](#).

### 15.303.2 Member Function Documentation

#### 15.303.2.1 dec()

```
template<typename COUNTER = unsigned char>
Type L4Re::Util::Counter< COUNTER >::dec ( ) [inline]
```

Decrement counter if not saturated.

Once the counter reached the saturated state, the counter value isn't changed.

Definition at line 78 of file [counting\\_cap\\_alloc](#).

**15.303.2.2 inc()**

```
template<typename COUNTER = unsigned char>  
bool L4Re::Util::Counter< COUNTER >::inc ( ) [inline]
```

Increment counter if not yet saturated.

Once the counter reached the saturated state, the counter value isn't changed.

## Return values

<i>false</i>	The counter just went saturated after it was increased.
<i>true</i>	Either the counter was already saturated – in that case the counter value was not changed, or the counter was not saturated – in that case the counter was increased.

Definition at line 61 of file [counting\\_cap\\_alloc](#).

The documentation for this struct was generated from the following file:

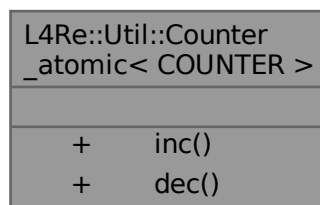
- [l4/re/util/counting\\_cap\\_alloc](#)

## 15.304 L4Re::Util::Counter\_atomic< COUNTER > Struct Template Reference

Thread safe version of counter for [Counting\\_cap\\_alloc](#).

```
#include <counting_cap_alloc>
```

Collaboration diagram for L4Re::Util::Counter\_atomic< COUNTER >:



### Public Member Functions

- bool [inc](#) ()  
*Increment counter if not yet saturated.*
- Type [dec](#) ()  
*Decrement counter if not saturated.*

### 15.304.1 Detailed Description

```
template<typename COUNTER = unsigned char>
struct L4Re::Util::Counter_atomic< COUNTER >
```

Thread safe version of counter for [Counting\\_cap\\_alloc](#).

Despite using atomic instructions, this version has to make sure that capability slots are not reused too early. If the last reference is gone, the capability slot has to be unmapped. The slot must only be allocated again when the unmap has completed. This is accomplished by starting with an initial count of 2. The last reference will decrease the counter to 1. Only then, after the slot was unmapped, will the counter be set to 0. This will allow other threads to reallocate the slot.

Definition at line 109 of file [counting\\_cap\\_alloc](#).

## 15.304.2 Member Function Documentation

### 15.304.2.1 dec()

```
template<typename COUNTER = unsigned char>
Type L4Re::Util::Counter_atomic< COUNTER >::dec ( ) [inline]
```

Decrement counter if not saturated.

Once the counter reached the saturated state, the counter value isn't changed.

Definition at line 153 of file [counting\\_cap\\_alloc](#).

### 15.304.2.2 inc()

```
template<typename COUNTER = unsigned char>
bool L4Re::Util::Counter_atomic< COUNTER >::inc ( ) [inline]
```

Increment counter if not yet saturated.

Once the counter reached the saturated state, the counter value isn't changed.

#### Return values

<i>false</i>	The counter just went saturated after it was increased.
<i>true</i>	Either the counter was already saturated – in that case the counter value was not changed, or the counter was not saturated – in that case the counter was increased.

Definition at line 132 of file [counting\\_cap\\_alloc](#).

The documentation for this struct was generated from the following file:

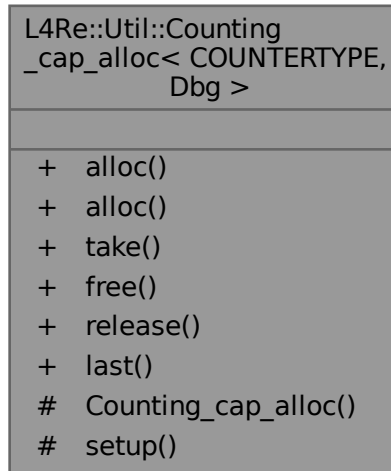
- [l4/re/util/counting\\_cap\\_alloc](#)

## 15.305 L4Re::Util::Counting\_cap\_alloc< COUNTERTYPE, Dbg > Class Template Reference

Internal reference-counting cap allocator.

```
#include <counting_cap_alloc>
```

Collaboration diagram for L4Re::Util::Counting\_cap\_alloc< COUNTERTYPE, Dbg >:



## Public Member Functions

- `L4::Cap< void > alloc () noexcept`  
*Allocate a new capability slot.*
- `template<typename T >`  
`L4::Cap< T > alloc () noexcept`  
*Allocate a new capability slot.*
- `void take (L4::Cap< void > cap) noexcept`  
*Increase the reference counter for the capability.*
- `bool free (L4::Cap< void > cap, l4_cap_idx_t task=L4_INVALID_CAP, unsigned unmap_flags=L4_FP_ALL_SPACES) noexcept`  
*Free the capability.*
- `bool release (L4::Cap< void > cap, l4_cap_idx_t task=L4_INVALID_CAP, unsigned unmap_flags=L4_FP_ALL_SPACES) noexcept`  
*Decrease the reference counter for a capability.*
- `long last () noexcept`  
*Return highest capability id managed by this allocator.*

## Protected Member Functions

- `Counting_cap_alloc () noexcept`  
*Create a new, empty allocator.*
- `void setup (void *m, long capacity, long bias, Dbg *dbg) noexcept`  
*Set up the backing memory for the allocator and the area of managed capability slots.*

### 15.305.1 Detailed Description

```
template<typename COUNTERTYPE, typename Dbg>
class L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >
```

Internal reference-counting cap allocator.

This is intended for internal use only. [L4Re](#) applications should use [L4Re::Util::cap\\_alloc\(\)](#).

Allocator for capability slots that automatically frees the slot and optionally unmaps the capability when the reference count goes down to zero. Reference counting must be done manually via [take\(\)](#) and [release\(\)](#). The backing store for the reference counters must be provided in the [setup\(\)](#) method. The allocator can recognize capability slots that are not managed by itself and does nothing on such slots.

#### Note

The user must ensure that the backing store is zero-initialized.

The user must ensure that the capability slots managed by this allocator are not used by a different allocator, see [setup\(\)](#).

The operations in this class are not thread-safe.

Definition at line 202 of file [counting\\_cap\\_alloc](#).

### 15.305.2 Constructor & Destructor Documentation

#### 15.305.2.1 Counting\_cap\_alloc()

```
template<typename COUNTERTYPE , typename Dbg >
L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::Counting_cap_alloc ( ) [inline], [protected],
[noexcept]
```

Create a new, empty allocator.

Needs to be initialized with [setup\(\)](#) before it can be used.

Definition at line 231 of file [counting\\_cap\\_alloc](#).

### 15.305.3 Member Function Documentation

#### 15.305.3.1 alloc() [1/2]

```
template<typename COUNTERTYPE , typename Dbg >
L4::Cap< void > L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::alloc ( ) [inline],
[noexcept]
```

Allocate a new capability slot.

**Returns**

The newly allocated capability slot, invalid if the allocator was exhausted.

Definition at line 269 of file [counting\\_cap\\_alloc](#).

References [L4\\_CAP\\_SHIFT](#).

Referenced by [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE, Dbg >::alloc\(\)](#).

Here is the caller graph for this function:

**15.305.3.2 alloc() [2/2]**

```

template<typename COUNTERTYPE , typename Dbg >
template<typename T >
L4::Cap< T > L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::alloc ( ) [inline], [noexcept]
  
```

Allocate a new capability slot.

**Returns**

The newly allocated capability slot, invalid if the allocator was exhausted.

Definition at line 294 of file [counting\\_cap\\_alloc](#).

References [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE, Dbg >::alloc\(\)](#).

Here is the call graph for this function:

**15.305.3.3 free()**

```

template<typename COUNTERTYPE , typename Dbg >
bool L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::free (
    L4::Cap< void > cap,
    l4_cap_idx_t task = L4_INVALID_CAP,
    unsigned unmap_flags = L4_FP_ALL_SPACES ) [inline], [noexcept]
  
```

Free the capability.

**Parameters**

<i>cap</i>	Capability to free.
<i>task</i>	If set, task to unmap the capability from.
<i>unmap_flags</i>	Flags for unmap, see <code>l4_unmap_flags_t</code> .

**Precondition**

The capability has been allocated. Calling free twice on a capability managed by this allocator results in undefined behaviour.

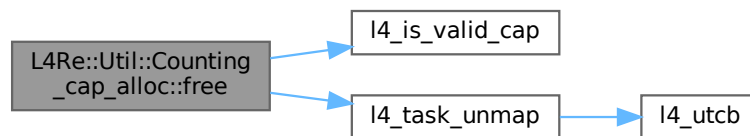
**Returns**

True, if the capability was managed by this allocator.

Definition at line 333 of file `counting_cap_alloc`.

References `l4_assert`, `l4_is_valid_cap()`, and `l4_task_unmap()`.

Here is the call graph for this function:

**15.305.3.4 release()**

```

template<typename COUNTERTYPE , typename Dbg >
bool L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::release (
    L4::Cap< void > cap,
    l4_cap_idx_t task = L4_INVALID_CAP,
    unsigned unmap_flags = L4_FP_ALL_SPACES ) [inline], [noexcept]
  
```

Decrease the reference counter for a capability.

**Parameters**

<i>cap</i>	Capability to release.
<i>task</i>	If set, task to unmap the capability from.
<i>unmap_flags</i>	Flags for unmap, see <code>l4_unmap_flags_t</code> .



**Precondition**

The capability has been allocated. Calling release on a free capability results in undefined behaviour.

**Returns**

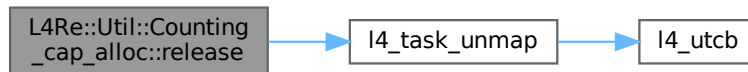
True, if the capability was freed as a result of this operation. If false is returned the capability is either still in use or is not managed by this allocator.

Does nothing apart from returning false if the capability is not managed by this allocator.

Definition at line 371 of file [counting\\_cap\\_alloc](#).

References [l4\\_assert](#), [L4\\_INVALID\\_CAP](#), and [l4\\_task\\_unmap\(\)](#).

Here is the call graph for this function:

**15.305.3.5 setup()**

```

template<typename COUNTERTYPE , typename Dbg >
void L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::setup (
    void * m,
    long capacity,
    long bias,
    Dbg * dbg ) [inline], [protected], [noexcept]
  
```

Set up the backing memory for the allocator and the area of managed capability slots.

**Parameters**

<i>m</i>	Pointer to backing memory.
<i>capacity</i>	Number of capabilities that can be stored.
<i>bias</i>	First capability id to use by this allocator.
<i>dbg</i>	Logger for warnings if counter got saturated.

The allocator will manage the capability slots between `bias` and `bias + capacity - 1` (inclusive). It is the responsibility of the user to ensure that these slots are not used otherwise.

Definition at line 254 of file [counting\\_cap\\_alloc](#).

**15.305.3.6 take()**

```

template<typename COUNTERTYPE , typename Dbg >
  
```

```
void L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::take (
    L4::Cap< void > cap ) [inline], [noexcept]
```

Increase the reference counter for the capability.

#### Parameters

<i>cap</i>	Capability, whose reference counter should be increased.
------------	--

If the capability was still free, it will be automatically allocated. Silently does nothing if the capability is not managed by this allocator.

Definition at line 308 of file [counting\\_cap\\_alloc](#).

References [L4\\_CAP\\_SHIFT](#), and [L4\\_UNLIKELY](#).

The documentation for this class was generated from the following file:

- [l4/re/util/counting\\_cap\\_alloc](#)

## 15.306 L4Re::Util::Dataspace\_svr Class Reference

[Dataspace](#) server class.

```
#include <dataspace_svr>
```

Collaboration diagram for L4Re::Util::Dataspace\_svr:

L4Re::Util::Dataspace_svr	
+	map()
+	map_hook()
+	take()
+	release()
+	copy()
+	clear()
+	allocate()
+	page_shift()
+	is_static()
+	map_info()

## Public Member Functions

- `int map` (`Dataspace::Offset` offset, `Dataspace::Map_addr` local\_addr, `Dataspace::Flags` flags, `Dataspace::Map_addr` min\_addr, `Dataspace::Map_addr` max\_addr, `L4::lpc::Snd_fpage` &memory)  
*Map a region of the dataspace.*
- `virtual int map_hook` (`Dataspace::Offset` offs, unsigned order, `Dataspace::Flags` flags, `Dataspace::Map_addr` \*base, unsigned \*send\_order)  
*A hook that is called for acquiring the data to be mapped.*
- `virtual void take` () noexcept  
*Take a reference to this dataspace.*
- `virtual unsigned long release` () noexcept  
*Release a reference to this dataspace.*
- `virtual long copy` (`l4_addr_t` dst\_offs, `l4_umword_t` src\_id, `l4_addr_t` src\_offs, unsigned long size) noexcept  
*Copy from src dataspace to this destination dataspace.*
- `virtual long clear` (unsigned long offs, unsigned long size) const noexcept  
*Clear a region in the dataspace.*
- `virtual long allocate` (`l4_addr_t` offset, `l4_size_t` size, unsigned access) noexcept  
*Allocate a region within a dataspace.*
- `virtual unsigned long page_shift` () const noexcept  
*Define the size of the flexpage to map.*
- `virtual bool is_static` () const noexcept  
*Return whether the dataspace is static.*
- `virtual long map_info` (`l4_addr_t` &start\_addr, `l4_addr_t` &end\_addr) noexcept  
*Return mapping information for no-MMU systems.*

### 15.306.1 Detailed Description

`Dataspace` server class.

The default implementation of the interface provides a continuous dataspace with contiguous pages.

Definition at line 40 of file `dataspace_svr`.

### 15.306.2 Member Function Documentation

#### 15.306.2.1 `allocate()`

```
virtual long L4Re::Util::Dataspace_svr::allocate (
    l4_addr_t offset,
    l4_size_t size,
    unsigned access ) [inline], [virtual], [noexcept]
```

Allocate a region within a dataspace.

#### Parameters

<i>offset</i>	Offset in the dataspace, in bytes.
<i>size</i>	Size of the range, in bytes.
<i>access</i>	Access mode with which the memory backing the dataspace region should be allocated.

**Return values**

0	Success
<0	Error

Definition at line 235 of file [dataspace\\_svr](#).

References [L4\\_ENODEV](#).

**15.306.2.2 clear()**

```
virtual long L4Re::Util::Dataspace_svr::clear (
    unsigned long offs,
    unsigned long size ) const [inline], [virtual], [noexcept]
```

Clear a region in the dataspace.

**Parameters**

<i>offs</i>	Start of the region
<i>size</i>	Size of the region

**Return values**

0	Success
<0	Error

Definition at line 203 of file [dataspace\\_svr](#).

References [L4\\_ERANGE](#).

**15.306.2.3 copy()**

```
virtual long L4Re::Util::Dataspace_svr::copy (
    l4_addr_t dst_offs,
    l4_umword_t src_id,
    l4_addr_t src_offs,
    unsigned long size ) [inline], [virtual], [noexcept]
```

Copy from src dataspace to this destination dataspace.

**Parameters**

<i>dst_offs</i>	Offset into the destination dataspace
<i>src_id</i>	Local id of the source dataspace
<i>src_offs</i>	Offset into the source dataspace
<i>size</i>	Number of bytes to copy

## Return values

$\geq 0$	Number of bytes copied
$< 0$	An error occurred. The error code may depend on the implementation.

Definition at line 186 of file [dataspace\\_svr](#).

References [L4\\_ENODEV](#).

## 15.306.2.4 is\_static()

```
virtual bool L4Re::Util::Dataspace_svr::is_static ( ) const [inline], [virtual], [noexcept]
```

Return whether the dataspace is static.

## Returns

True if dataspace is static

Definition at line 255 of file [dataspace\\_svr](#).

## 15.306.2.5 map()

```
int L4Re::Util::Dataspace_svr::map (
    Dataspace::Offset offset,
    Dataspace::Map_addr local_addr,
    Dataspace::Flags flags,
    Dataspace::Map_addr min_addr,
    Dataspace::Map_addr max_addr,
    L4::Ipc::Snd_fpage & memory ) [inline]
```

Map a region of the dataspace.

## Parameters

	<i>offset</i>	Offset to start within data space
	<i>local_addr</i>	Local address to map to.
	<i>flags</i>	<a href="#">Dataspace</a> flags, see <a href="#">L4Re::Dataspace::F::Flags</a> .
	<i>min_addr</i>	Defines start of receive window.
	<i>max_addr</i>	Defines end of receive window.
out	<i>memory</i>	Send fpage to map

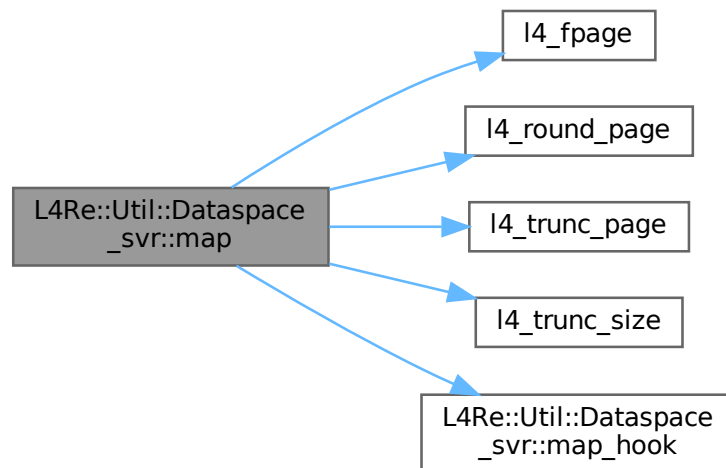
## Return values

0	Success
$< 0$	Error

Definition at line 68 of file [dataspace\\_svr](#).

References [L4\\_EOK](#), [L4\\_ERANGE](#), [l4\\_fpage\(\)](#), [L4\\_PAGESHIFT](#), [l4\\_round\\_page\(\)](#), [l4\\_trunc\\_page\(\)](#), [l4\\_trunc\\_size\(\)](#), and [map\\_hook\(\)](#).

Here is the call graph for this function:



### 15.306.2.6 map\_hook()

```

virtual int L4Re::Util::Dataspace_svr::map_hook (
    Dataspace::Offset offs,
    unsigned order,
    Dataspace::Flags flags,
    Dataspace::Map_addr * base,
    unsigned * send_order ) [inline], [virtual]
  
```

A hook that is called for acquiring the data to be mapped.

#### Parameters

<i>offs</i>	Offset in dataspace to supply
<i>order</i>	Log2-size of data to supply
<i>flags</i>	Flags for the mapping
<i>base</i>	Start address of the flexpage to be mapped
<i>send_order</i>	Order (log2 of size) of the flexpage to be mapped

#### Return values

$< 0$	Error and the map request will be aborted with that error.
$\geq 0$	Success

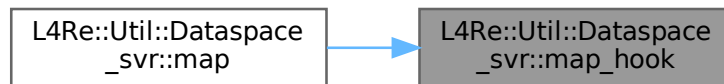
See also

[map](#)

Definition at line 147 of file [dataspace\\_svr](#).

Referenced by [map\(\)](#).

Here is the caller graph for this function:



### 15.306.2.7 map\_info()

```
virtual long L4Re::Util::Dataspace_svr::map_info (
    l4_addr_t & start_addr,
    l4_addr_t & end_addr ) [inline], [virtual], [noexcept]
```

Return mapping information for no-MMU systems.

The method is only called on no-MMU systems. It should return the address of the underlying backing buffer so that the caller might map the dataspace.

The default implementation always returns an error because the derived class must provide the required information.

See also

[L4Re::Dataspace::map\\_info\(\)](#)

Definition at line 270 of file [dataspace\\_svr](#).

References [L4\\_EPERM](#).

### 15.306.2.8 page\_shift()

```
virtual unsigned long L4Re::Util::Dataspace_svr::page_shift ( ) const [inline], [virtual],
[noexcept]
```

Define the size of the flexpage to map.

Returns

flexpage size

Definition at line 247 of file [dataspace\\_svr](#).

References [L4\\_LOG2\\_PAGESIZE](#).

#### 15.306.2.9 `release()`

```
virtual unsigned long L4Re::Util::Dataspace_svr::release ( ) [inline], [virtual], [noexcept]
```

Release a reference to this dataspace.

##### Returns

Number of references to the dataspace

Default does nothing and returns always zero.

Definition at line 171 of file [dataspace\\_svr](#).

#### 15.306.2.10 `take()`

```
virtual void L4Re::Util::Dataspace_svr::take ( ) [inline], [virtual], [noexcept]
```

Take a reference to this dataspace.

Default does nothing.

Definition at line 161 of file [dataspace\\_svr](#).

The documentation for this class was generated from the following file:

- `l4/re/util/dataspace_svr`

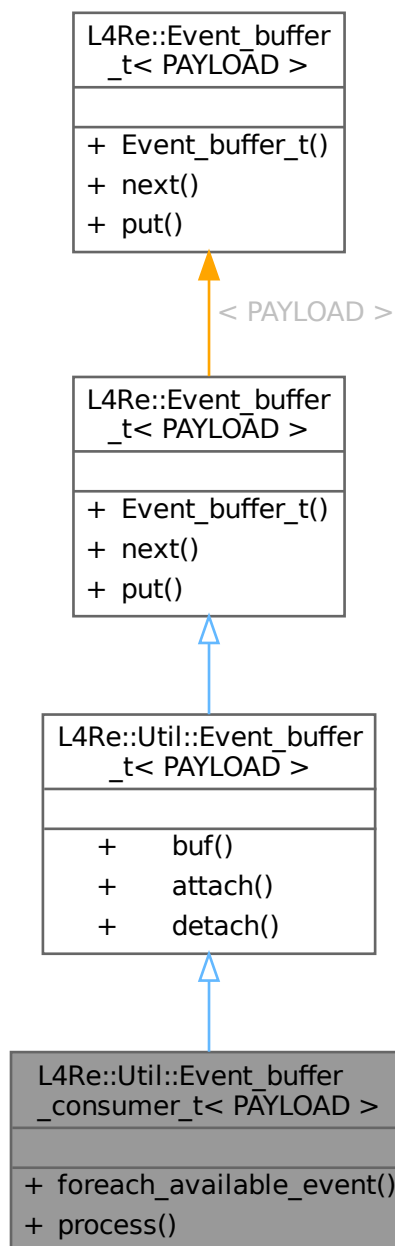
### 15.307 `L4Re::Util::Event_buffer_consumer_t< PAYLOAD >` Class Template Reference

An event buffer consumer.

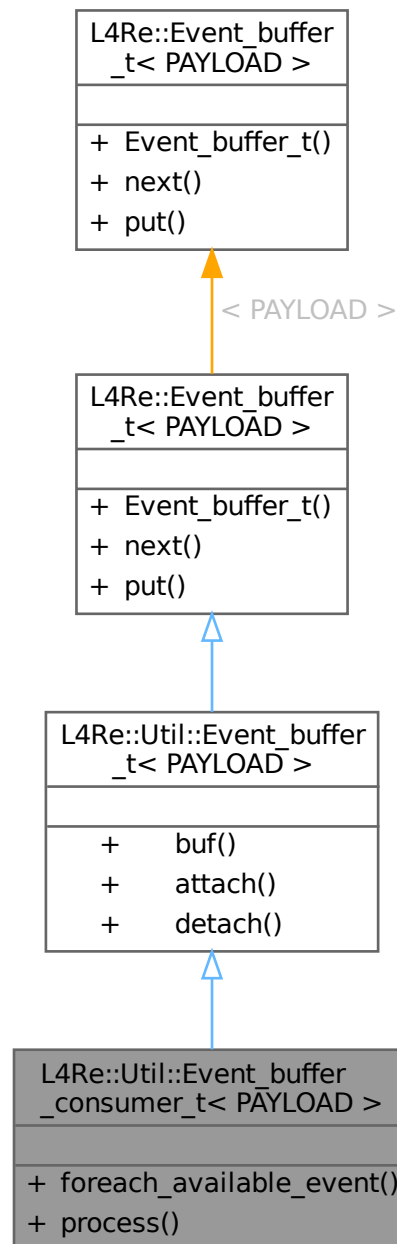
```
#include <event_buffer>
```



Inheritance diagram for L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >:



Collaboration diagram for L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >:



## Public Member Functions

- `template<typename CB , typename D >`  
`void foreach_available_event (CB const &cb, D data=D())`  
*Call function on every available event.*
- `template<typename CB , typename D >`  
`void process (L4Re::Cap< L4Re::Irq > irq, L4Re::Cap< L4Re::Thread > thread, CB const &cb, D data=D())`  
*Continuously wait for events and process them.*

**Public Member Functions inherited from [L4Re::Util::Event\\_buffer\\_t< PAYLOAD >](#)**

- void \* [buf](#) () const noexcept  
*Return the buffer.*
- long [attach](#) (L4::Cap< [L4Re::Dataspace](#) > ds, L4::Cap< [L4Re::Rm](#) > rm) noexcept  
*Attach event buffer from address space.*
- long [detach](#) (L4::Cap< [L4Re::Rm](#) > rm) noexcept  
*Detach event buffer from address space.*

**Public Member Functions inherited from [L4Re::Event\\_buffer\\_t< PAYLOAD >](#)**

- [Event\\_buffer\\_t](#) (void \*buffer, [l4\\_addr\\_t](#) size)  
*Initialize event buffer.*
- [Event](#) \* [next](#) () noexcept  
*Next event in buffer.*
- bool [put](#) ([Event](#) const &ev) noexcept  
*Put event into buffer at current position.*

**15.307.1 Detailed Description**

```
template<typename PAYLOAD>
class L4Re::Util::Event_buffer_consumer_t< PAYLOAD >
```

An event buffer consumer.

Definition at line 94 of file [event\\_buffer](#).

**15.307.2 Member Function Documentation****15.307.2.1 [foreach\\_available\\_event\(\)](#)**

```
template<typename PAYLOAD >
template<typename CB , typename D >
void L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::foreach_available_event (
    CB const & cb,
    D data = D() ) [inline]
```

Call function on every available event.

**Parameters**

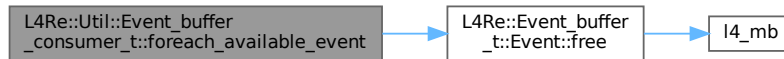
<i>cb</i>	Function callback.
<i>data</i>	Data to pass as an argument to the callback.

Definition at line 105 of file [event\\_buffer](#).

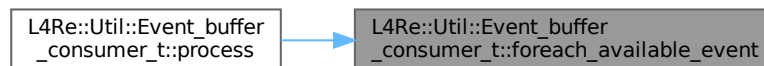
References [L4Re::Event\\_buffer\\_t< PAYLOAD >::Event::free\(\)](#).

Referenced by [L4Re::Util::Event\\_buffer\\_consumer\\_t< PAYLOAD >::process\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.307.2.2 process()

```

template<typename PAYLOAD >
template<typename CB , typename D >
void L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process (
    L4::Cap< L4::Irq > irq,
    L4::Cap< L4::Thread > thread,
    CB const & cb,
    D data = D() ) [inline]
  
```

Continuously wait for events and process them.

#### Parameters

<i>irq</i>	<a href="#">Event</a> signal to wait for.
<i>thread</i>	Thread capability of the thread calling this function.
<i>cb</i>	Callback function that is called for each received event.
<i>data</i>	Data to pass as an argument to the processing callback.

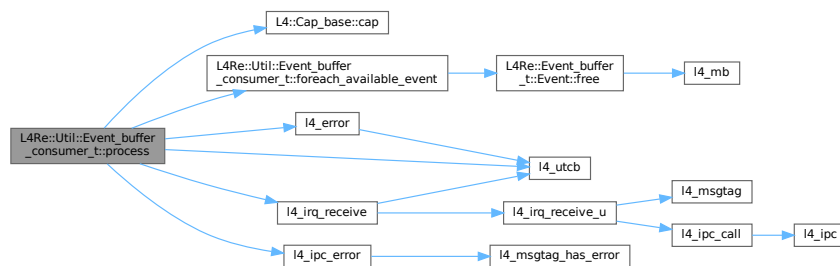
#### Note

This function never returns.

Definition at line 126 of file [event\\_buffer](#).

References [L4::Cap\\_base::cap\(\)](#), [L4Re::Util::Event\\_buffer\\_consumer\\_t< PAYLOAD >::foreach\\_available\\_event\(\)](#), [l4\\_error\(\)](#), [l4\\_ipc\\_error\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_irq\\_receive\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

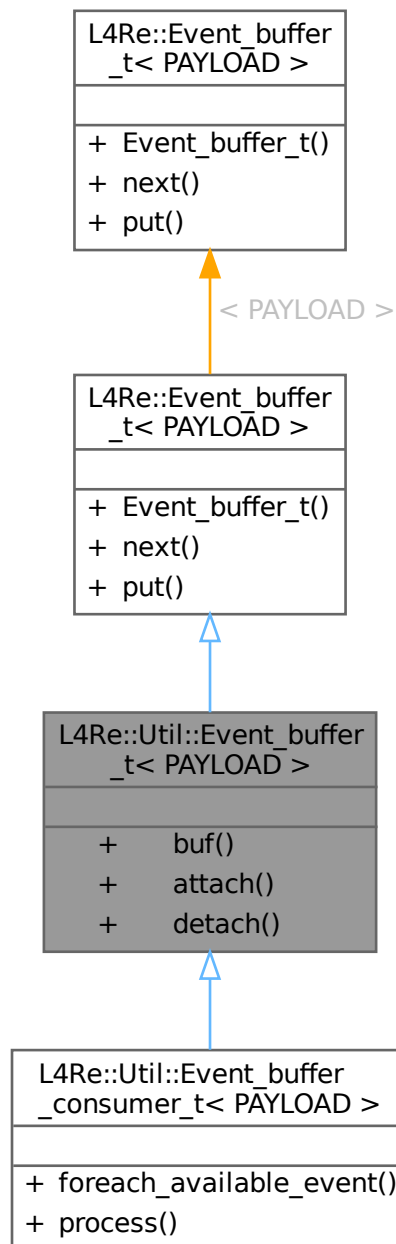
- `I4/re/util/event_buffer`

## 15.308 L4Re::Util::Event\_buffer\_t< PAYLOAD > Class Template Reference

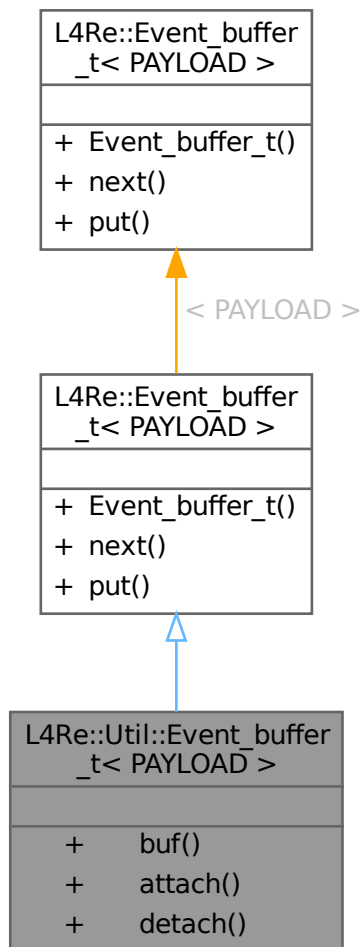
Event\_buffer utility class.

```
#include <event_buffer>
```

Inheritance diagram for L4Re::Util::Event\_buffer\_t< PAYLOAD >:



Collaboration diagram for L4Re::Util::Event\_buffer\_t< PAYLOAD >:



### Public Member Functions

- void \* `buf()` const noexcept  
*Return the buffer.*
- long `attach` (L4::Cap< L4Re::Dataspace > ds, L4::Cap< L4Re::Rm > rm) noexcept  
*Attach event buffer from address space.*
- long `detach` (L4::Cap< L4Re::Rm > rm) noexcept  
*Detach event buffer from address space.*

### Public Member Functions inherited from `L4Re::Event_buffer_t< PAYLOAD >`

- `Event_buffer_t` (void \*buffer, l4\_addr\_t size)  
*Initialize event buffer.*
- `Event * next()` noexcept  
*Next event in buffer.*
- bool `put` (`Event` const &ev) noexcept  
*Put event into buffer at current position.*

### 15.308.1 Detailed Description

```
template<typename PAYLOAD>
class L4Re::Util::Event_buffer_t< PAYLOAD >
```

Event\_buffer utility class.

Definition at line 36 of file [event\\_buffer](#).

### 15.308.2 Member Function Documentation

#### 15.308.2.1 attach()

```
template<typename PAYLOAD >
long L4Re::Util::Event_buffer_t< PAYLOAD >::attach (
    L4::Cap< L4Re::Dataspace > ds,
    L4::Cap< L4Re::Rm > rm ) [inline], [noexcept]
```

Attach event buffer from address space.

##### Parameters

<i>ds</i>	<a href="#">Dataspace</a> of the event buffer.
<i>rm</i>	Region manager to attach buffer to.

##### Returns

0 on success, negative error code otherwise.

Definition at line 56 of file [event\\_buffer](#).

References [L4::lpc::make\\_cap\\_rw\(\)](#), [L4Re::Rm::F::RW](#), and [L4Re::Rm::F::Search\\_addr](#).

Here is the call graph for this function:



#### 15.308.2.2 buf()

```
template<typename PAYLOAD >
void * L4Re::Util::Event_buffer_t< PAYLOAD >::buf ( ) const [inline], [noexcept]
```

Return the buffer.



**Returns**

Pointer to the event buffer.

Definition at line 46 of file [event\\_buffer](#).

**15.308.2.3 detach()**

```
template<typename PAYLOAD >
long L4Re::Util::Event_buffer_t< PAYLOAD >::detach (
    L4::Cap< L4Re::Rm > rm ) [inline], [noexcept]
```

Detach event buffer from address space.

**Parameters**

<i>rm</i>	Region manager to detach buffer from.
-----------	---------------------------------------

**Returns**

0 on success, negative error code otherwise.

Definition at line 79 of file [event\\_buffer](#).

The documentation for this class was generated from the following file:

- I4/re/util/event\_buffer

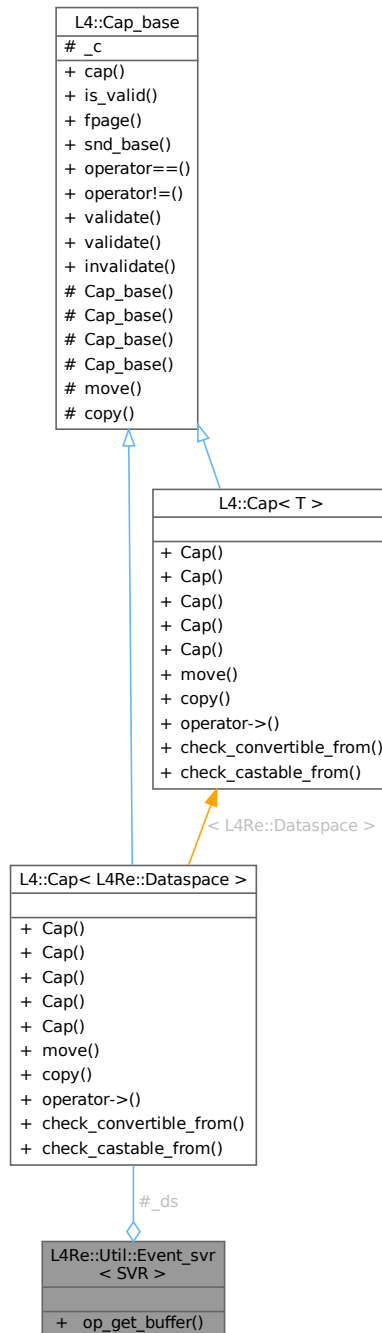
**15.309 L4Re::Util::Event\_svr< SVR > Class Template Reference**

Convenience wrapper for implementing an event server.

```
#include <event_svr>
```

Inherits L4Re::Util::Icu\_cap\_array\_svr< ICU >.

Collaboration diagram for L4Re::Util::Event\_svr< SVR >:



## Public Member Functions

- long **op\_get\_buffer** (L4Re::Event::Rights, [L4::Ipc::Cap](#)< [L4Re::Dataspace](#) > &ds)  
Handle [L4Re::Event](#) protocol.

### 15.309.1 Detailed Description

```
template<typename SVR>
class L4Re::Util::Event_svr< SVR >
```

Convenience wrapper for implementing an event server.

See also

[L4Re::Event](#), [L4Re::Util::Event\\_t](#)

Definition at line 39 of file [event\\_svr](#).

The documentation for this class was generated from the following file:

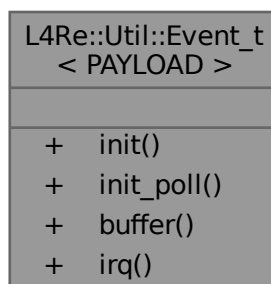
- [l4/re/util/event\\_svr](#)

## 15.310 L4Re::Util::Event\_t< PAYLOAD > Class Template Reference

Convenience wrapper for getting access to an event object.

```
#include <event>
```

Collaboration diagram for L4Re::Util::Event\_t< PAYLOAD >:



### Public Types

- enum [Mode](#) { [Mode\\_irq](#) , [Mode\\_polling](#) }  
*Modes of operation.*

## Public Member Functions

- `template<typename IRQ_TYPE >`  
`int init (L4::Cap< L4Re::Event > event, L4Re::Env const *env=L4Re::Env::env(), L4Re::Cap_alloc *ca=L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc))`  
*Initialise an event object.*
- `int init_poll (L4::Cap< L4Re::Event > event, L4Re::Env const *env=L4Re::Env::env(), L4Re::Cap_alloc *ca=L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc))`  
*Initialise an event object in polling mode.*
- `L4Re::Event_buffer_t< PAYLOAD > & buffer ()`  
*Get event buffer.*
- `L4::Cap< L4::Triggerable > irq () const`  
*Get event IRQ.*

### 15.310.1 Detailed Description

```
template<typename PAYLOAD>
class L4Re::Util::Event_t< PAYLOAD >
```

Convenience wrapper for getting access to an event object.

After calling `init()` the class supplies the event-buffer and the associated IRQ object.

Definition at line 43 of file `event`.

### 15.310.2 Member Enumeration Documentation

#### 15.310.2.1 Mode

```
template<typename PAYLOAD >
enum L4Re::Util::Event_t::Mode
```

Modes of operation.

Enumerator

Mode_irq	Create an IRQ and attach, to get notifications.
Mode_polling	Do not use an IRQ.

Definition at line 49 of file `event`.

### 15.310.3 Member Function Documentation

#### 15.310.3.1 buffer()

```
template<typename PAYLOAD >
L4Re::Event_buffer_t< PAYLOAD > & L4Re::Util::Event_t< PAYLOAD >::buffer ( ) [inline]
```

Get event buffer.

## Returns

[Event](#) buffer object.

Definition at line 159 of file [event](#).

## 15.310.3.2 init()

```
template<typename PAYLOAD >
template<typename IRQ_TYPE >
int L4Re::Util::Event_t< PAYLOAD >::init (
    L4::Cap< L4Re::Event > event,
    L4Re::Env const * env = L4Re::Env::env(),
    L4Re::Cap_alloc * ca = L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc) )
[inline]
```

Initialise an event object.

## Template Parameters

<i>IRQ_TYPE</i>	Type used for handling notifications from the event provider. This must be derived from <a href="#">L4::Triggerable</a> .
-----------------	---

## Parameters

<i>event</i>	Capability to event.
<i>env</i>	Pointer to L4Re-Environment
<i>ca</i>	Pointer to capability allocator.

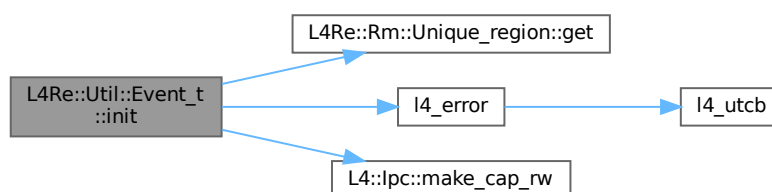
## Return values

0	Success
-L4_ENOMEM	No memory to allocate required capabilities.
<0	Other IPC errors.

Definition at line 70 of file [event](#).

References [L4Re::Rm::Unique\\_region< T >::get\(\)](#), [L4\\_ENOMEM](#), [l4\\_error\(\)](#), [L4::lpc::make\\_cap\\_rw\(\)](#), [L4Re::Rm::F::RW](#), and [L4Re::Rm::F::Search\\_addr](#).

Here is the call graph for this function:



### 15.310.3.3 `init_poll()`

```
template<typename PAYLOAD >
int L4Re::Util::Event_t< PAYLOAD >::init_poll (
    L4::Cap< L4Re::Event > event,
    L4Re::Env const * env = L4Re::Env::env(),
    L4Re::Cap_alloc * ca = L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc) )
[inline]
```

Initialise an event object in polling mode.

#### Parameters

<i>event</i>	Capability to event.
<i>env</i>	Pointer to L4Re-Environment
<i>ca</i>	Pointer to capability allocator.

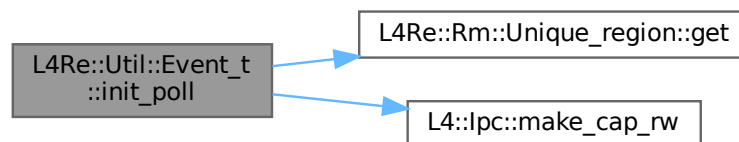
#### Return values

0	Success
-L4_ENOMEM	No memory to allocate required capabilities.
<0	Other IPC errors.

Definition at line 123 of file [event](#).

References [L4Re::Rm::Unique\\_region< T >::get\(\)](#), [L4\\_ENOMEM](#), [L4::lpc::make\\_cap\\_rw\(\)](#), [L4Re::Rm::F::RW](#), and [L4Re::Rm::F::Search\\_addr](#).

Here is the call graph for this function:



### 15.310.3.4 `irq()`

```
template<typename PAYLOAD >
L4::Cap< L4::Triggerable > L4Re::Util::Event_t< PAYLOAD >::irq ( ) const [inline]
```

Get event IRQ.

#### Returns

[Event](#) IRQ.

Definition at line 166 of file [event](#).

The documentation for this class was generated from the following file:

- [l4/re/util/event](#)

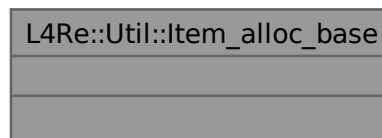
## 15.311 L4Re::Util::Item\_alloc\_base Class Reference

Item allocator.

```
#include <item_alloc>
```

Inherited by L4Re::Util::Item\_alloc< Bits >.

Collaboration diagram for L4Re::Util::Item\_alloc\_base:



### 15.311.1 Detailed Description

Item allocator.

Definition at line 38 of file [item\\_alloc](#).

The documentation for this class was generated from the following file:

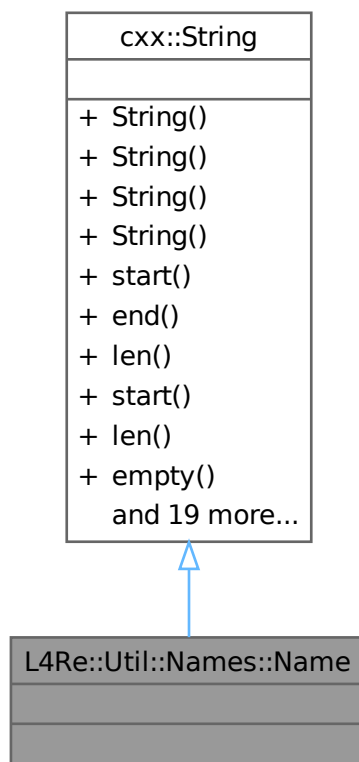
- [l4/re/util/item\\_alloc](#)

## 15.312 L4Re::Util::Names::Name Class Reference

[Name](#) class.

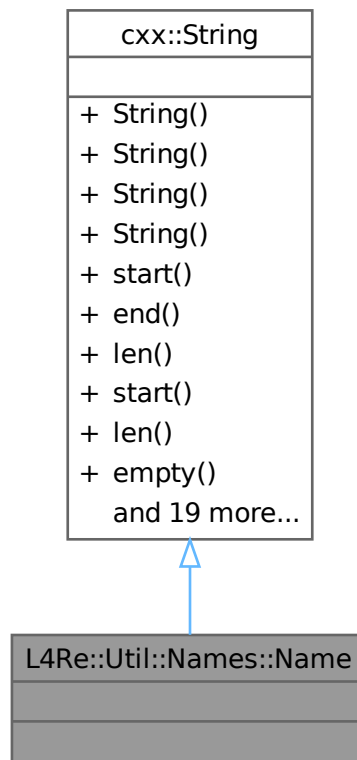
```
#include <name_space_svr>
```

Inheritance diagram for L4Re::Util::Names::Name:





Collaboration diagram for L4Re::Util::Names::Name:



### Additional Inherited Members

### Public Types inherited from `cxx::String`

- `typedef char const * Index`  
*Character index type.*

### Public Member Functions inherited from `cxx::String`

- **`String`** (`char const *s`) `noexcept`  
*Initialize from a zero-terminated string.*
- **`String`** (`char const *s`, unsigned long `len`) `noexcept`  
*Initialize from a pointer to first character and a length.*
- **`String`** (`char const *s`, `char const *e`) `noexcept`  
*Initialize with start and end pointer.*
- **`String`** ()  
*Zero-initialize. Create an invalid string.*
- **`Index start`** () `const`  
*Pointer to first character.*

- **Index end** () const  
*Pointer to first byte behind the string.*
- int **len** () const  
*Length.*
- void **start** (char const \*s)  
*Set start.*
- void **len** (unsigned long len)  
*Set length.*
- bool **empty** () const  
*Check if the string has length zero.*
- **String head** (**Index end**) const  
*Return prefix up to index.*
- **String head** (unsigned long **end**) const  
*Prefix of length **end**.*
- **String substr** (unsigned long idx, unsigned long **len**=~0UL) const  
*Substring of length **len** starting at **idx**.*
- **String substr** (char const \***start**, unsigned long **len**=0) const  
*Substring of length **len** starting at **start**.*
- template<typename F >  
char const \* **find\_match** (F &&match) const  
*Find matching character. **match** should be a function such as **isspace**.*
- char const \* **find** (char const \*c) const  
*Find character. Return **end()** if not found.*
- char const \* **find** (int c) const  
*Find character. Return **end()** if not found.*
- char const \* **rfind** (char const \*c) const  
*Find right-most character. Return **end()** if not found.*
- **Index starts\_with** (cxx::String const &c) const  
*Check if **c** is a prefix of string.*
- char const \* **find** (int c, char const \*s) const  
*Find character **c** starting at position **s**. Return **end()** if not found.*
- char const \* **find** (char const \*c, char const \*s) const  
*Find character set at position.*
- char const & **operator[]** (unsigned long idx) const  
*Get character at **idx**.*
- char const & **operator[]** (int idx) const  
*Get character at **idx**.*
- char const & **operator[]** (**Index** idx) const  
*Get character at **idx**.*
- bool **eof** (char const \*s) const  
*Check if pointer **s** points behind string.*
- template<typename INT >  
int **from\_dec** (INT \*v) const  
*Convert decimal string to integer.*
- template<typename INT >  
int **from\_hex** (INT \*v) const  
*Convert hex string to integer.*
- bool **operator==** (**String** const &o) const  
*Equality.*
- bool **operator!=** (**String** const &o) const  
*Inequality.*

### 15.312.1 Detailed Description

[Name](#) class.

Definition at line 39 of file [name\\_space\\_svr](#).

The documentation for this class was generated from the following file:

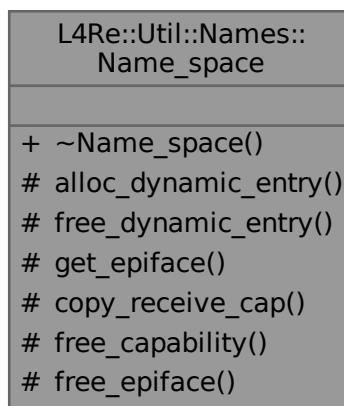
- [l4/re/util/name\\_space\\_svr](#)

## 15.313 L4Re::Util::Names::Name\_space Class Reference

Abstract server-side implementation of the L4::Namespace interface.

```
#include <name_space_svr>
```

Collaboration diagram for L4Re::Util::Names::Name\_space:



### Public Member Functions

- virtual `~Name_space()`  
*The destructor of the derived class is responsible for freeing resources.*

### Protected Member Functions

- virtual `Entry * alloc_dynamic_entry (Name const &n, unsigned flags)=0`  
*Allocate a new entry for the given name.*
- virtual `void free_dynamic_entry (Entry *e)=0`  
*Free an entry previously allocated with [alloc\\_dynamic\\_entry\(\)](#).*
- virtual `int get_epiface (l4_umword_t data, bool is_local, L4::Epiface **lo)=0`  
*Return a pointer to the epiface assigned to a given label.*
- virtual `int copy_receive_cap (L4::Cap< void > *cap)=0`  
*Return the receive capability for permanent use.*
- virtual `void free_capability (L4::Cap< void > cap)=0`  
*Free a capability previously acquired with [copy\\_receive\\_cap\(\)](#).*
- virtual `void free_epiface (L4::Epiface *epiface)=0`  
*Free epiface previously acquired with [get\\_epiface\(\)](#).*

### 15.313.1 Detailed Description

Abstract server-side implementation of the L4::Namespace interface.

#### Note

The derived class is responsible for resource management through the provided interfaces. This includes freeing all resources on destruction!

Definition at line 191 of file [name\\_space\\_svr](#).

### 15.313.2 Member Function Documentation

#### 15.313.2.1 alloc\_dynamic\_entry()

```
virtual Entry * L4Re::Util::Names::Name_space::alloc_dynamic_entry (
    Name const & n,
    unsigned flags ) [protected], [pure virtual]
```

Allocate a new entry for the given name.

#### Parameters

<i>n</i>	<a href="#">Name</a> of the entry, must be copied.
<i>flags</i>	Entry flags, see <a href="#">Obj::Flags</a> .

#### Returns

A pointer to the newly allocated entry or NULL on error.

This method is called when a new entry was received. It must allocate memory, copy *n* out of the receive buffer and wrap everything in an Entry.

Referenced by [free\\_epiface\(\)](#).

Here is the caller graph for this function:



#### 15.313.2.2 copy\_receive\_cap()

```
virtual int L4Re::Util::Names::Name_space::copy_receive_cap (
    L4::Cap< void > * cap ) [protected], [pure virtual]
```

Return the receive capability for permanent use.

## Parameters

out	cap	Capability slot with the received capability. Must be permanently available until <a href="#">free_capability()</a> is called.
-----	-----	--

This method is called when a new entry is registered together with a capability mapping. It must decide whether and where to store the capability and return the final capability slot. Typical implementations return the capability slot in the receive window and allocate a new receive window.

**15.313.2.3 free\_capability()**

```
virtual void L4Re::Util::Names::Name_space::free_capability (
    L4::Cap< void > cap ) [protected], [pure virtual]
```

Free a capability previously acquired with [copy\\_receive\\_cap\(\)](#).

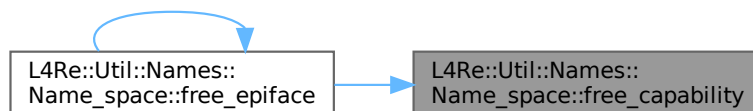
## Parameters

in	cap	Capability to free.
----	-----	---------------------

Counterpart of [copy\\_receive\\_cap](#). Free the capability slot when the entry is deleted or changed.

Referenced by [free\\_epiface\(\)](#).

Here is the caller graph for this function:

**15.313.2.4 free\_dynamic\_entry()**

```
virtual void L4Re::Util::Names::Name_space::free_dynamic_entry (
    Entry * e ) [protected], [pure virtual]
```

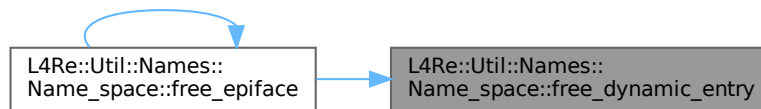
Free an entry previously allocated with [alloc\\_dynamic\\_entry\(\)](#).

## Parameters

e	Entry to free.
---	----------------

Referenced by [free\\_epiface\(\)](#).

Here is the caller graph for this function:



### 15.313.2.5 free\_epiface()

```
virtual void L4Re::Util::Names::Name_space::free_epiface (
    L4::Epiface * epiface ) [protected], [pure virtual]
```

Free epiface previously acquired with [get\\_epiface\(\)](#).

#### Parameters

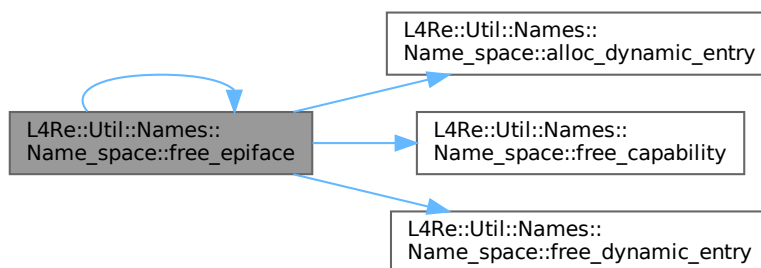
in	<i>epiface</i>	Epiface to free.
----	----------------	------------------

Called when an entry that points to an epiface is deleted allowing implementations that hold resources to free them.

References [alloc\\_dynamic\\_entry\(\)](#), [free\\_capability\(\)](#), [free\\_dynamic\\_entry\(\)](#), [free\\_epiface\(\)](#), [L4\\_BASE\\_TASK\\_CAP](#), [L4\\_EEXIST](#), [L4\\_ENOMEM](#), and [L4Re::Namespace::Overwrite](#).

Referenced by [free\\_epiface\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.313.2.6 get\_epiface()

```
virtual int L4Re::Util::Names::Name_space::get_epiface (
    l4_umword_t data,
    bool is_local,
    L4::Epiface ** lo ) [protected], [pure virtual]
```

Return a pointer to the epiface assigned to a given label.

#### Parameters

in	<i>data</i>	Label or in the local case the capability slot of the receiving capability.
in	<i>is_local</i>	If true, a local capability slot was supplied, if false the label of a locally bound IPC gate.
out	<i>lo</i>	Pointer to epiface responsible for the capability.

#### Returns

[L4\\_EOK](#) if a valid interface could be found or an error message otherwise.

This method is called when a new entry is registered and some local ID was received for the capability. In this case, the generic implementation needs to get the epiface in order to get the capability.

The callee must make sure that the epiface remains valid until `free_epiface` is called. In particular, the capability slot must not be reallocated as long as the namespace server holds a reference to the epiface.

The documentation for this class was generated from the following file:

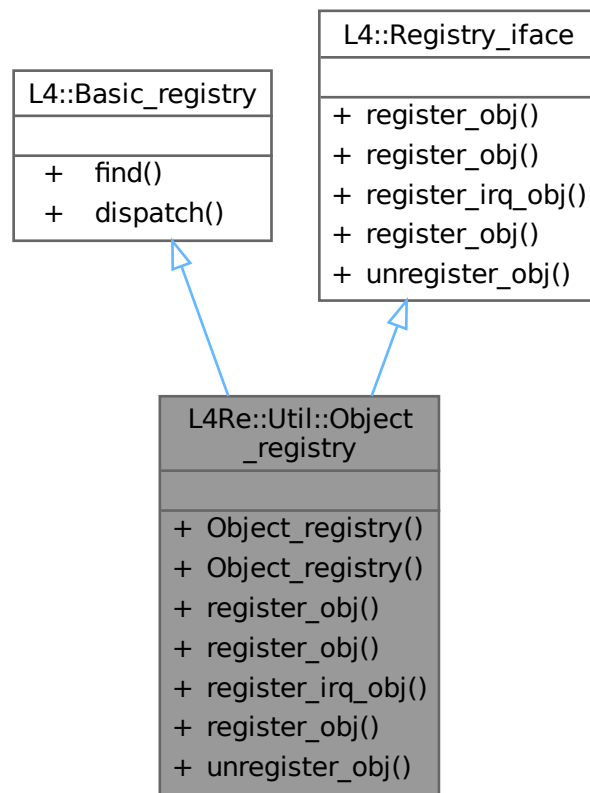
- `l4/re/util/name_space_svr`

## 15.314 L4Re::Util::Object\_registry Class Reference

A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread.

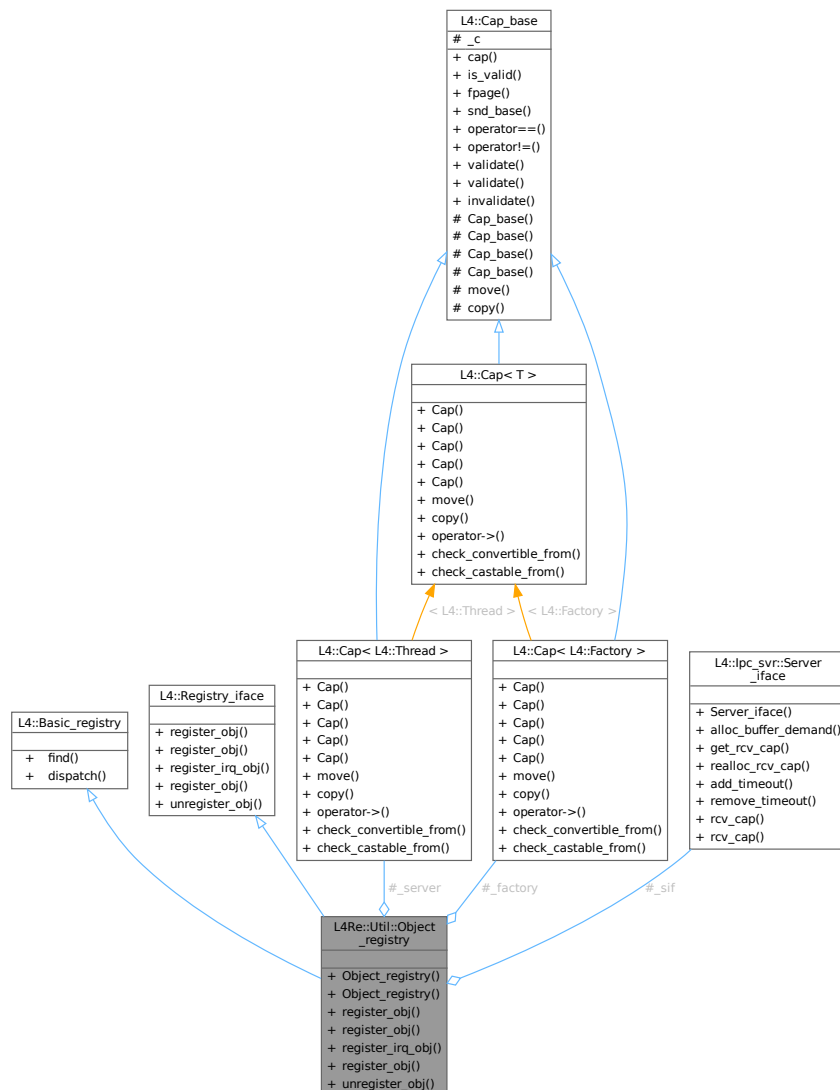
```
#include <object_registry>
```

Inheritance diagram for L4Re::Util::Object\_registry:





Collaboration diagram for L4Re::Util::Object\_registry:



## Public Member Functions

- [Object\\_registry](#) ([L4::lpc\\_svr::Server\\_iface](#) \*sif)  
Create a registry for the main thread of the task using the default factory.
- [Object\\_registry](#) ([L4::lpc\\_svr::Server\\_iface](#) \*sif, [L4::Cap](#)< [L4::Thread](#) > server, [L4::Cap](#)< [L4::Factory](#) > factory)  
Create a registry for arbitrary threads.
- [L4::Cap](#)< void > [register\\_obj](#) ([L4::Epiface](#) \*o, char const \*service) override  
Register a new server object to a pre-allocated receive endpoint.
- [L4::Cap](#)< void > [register\\_obj](#) ([L4::Epiface](#) \*o) override  
Register a new server object on a newly allocated capability.
- [L4::Cap](#)< [L4::Irq](#) > [register\\_irq\\_obj](#) ([L4::Epiface](#) \*o) override  
Register a handler for an interrupt.
- [L4::Cap](#)< [L4::Rcv\\_endpoint](#) > [register\\_obj](#) ([L4::Epiface](#) \*o, [L4::Cap](#)< [L4::Rcv\\_endpoint](#) > ep) override

*Register a handler for an already existing interrupt.*

- void `unregister_obj` (`L4::Epiface *o`, bool `unmap=true`) override

*Remove a server object from the handler list.*

## Additional Inherited Members

## Static Public Member Functions inherited from `L4::Basic_registry`

- static `Value * find` (`l4_umword_t label`)

*Get the server object for an `lpc_gate` label.*

- static `l4_msgtag_t dispatch` (`l4_msgtag_t tag`, `l4_umword_t label`, `l4_utcb_t *utcb`)

*The dispatch function called by the server loop.*

### 15.314.1 Detailed Description

A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread.

This class manages most of the setup of a server object. If necessary, an IPC gate is created, the specified thread is bound to the IPC gate. Incoming IPC is dispatched to the server object based on the label of the IPC gate.

The object registry is also able to manage IRQ endpoints. They require a different method for the object creation. Otherwise they are handled in the same way as IPC gates: a server object is responsible to process the incoming interrupts.

Definition at line 52 of file `object_registry`.

### 15.314.2 Constructor & Destructor Documentation

#### 15.314.2.1 `Object_registry()` [1/2]

```
L4Re::Util::Object_registry::Object_registry (
    L4::Ipc_svr::Server_iface * sif ) [inline], [explicit]
```

Create a registry for the main thread of the task using the default factory.

#### Parameters

<i>sif</i>	Server loop interface.
------------	------------------------

Definition at line 78 of file `object_registry`.

#### 15.314.2.2 `Object_registry()` [2/2]

```
L4Re::Util::Object_registry::Object_registry (
    L4::Ipc_svr::Server_iface * sif,
    L4::Cap< L4::Thread > server,
    L4::Cap< L4::Factory > factory ) [inline]
```

Create a registry for arbitrary threads.

## Parameters

<i>sif</i>	Server loop interface.
<i>server</i>	Capability to the thread that executes the server objects.
<i>factory</i>	Capability to a factory object capable of creating new IPC gates.

Definition at line 92 of file [object\\_registry](#).

### 15.314.3 Member Function Documentation

#### 15.314.3.1 register\_irq\_obj()

```
L4::Cap< L4::Irq > L4Re::Util::Object_registry::register_irq_obj (
    L4::Epiface * o ) [inline], [override], [virtual]
```

Register a handler for an interrupt.

## Parameters

<i>o</i>	Server object that handles IRQs.
----------	----------------------------------

## Return values

<i>L4::Cap&lt;L4::Irq&gt;</i>	Capability to a new IRQ object on success.
<i>L4::Cap&lt;L4::Irq&gt;::Invalid</i>	The allocation of the IRQ has failed.

The IRQ will be newly allocated using the registry's factory object. The caller must call [unregister\\_obj\(\)](#) to free all resources.

Implements [L4::Registry\\_iface](#).

Definition at line 238 of file [object\\_registry](#).

#### 15.314.3.2 register\_obj() [1/3]

```
L4::Cap< void > L4Re::Util::Object_registry::register_obj (
    L4::Epiface * o ) [inline], [override], [virtual]
```

Register a new server object on a newly allocated capability.

## Parameters

<i>o</i>	Server object that handles IPC requests.
----------	--

## Return values

<i>L4::Cap&lt;void&gt;</i>	A valid capability to a new IPC gate.
<i>L4::Cap&lt;void&gt;::Invalid</i>	The allocation of the IPC gate has failed.

The IPC gate will be allocated using the registry's factory. The caller must call [unregister\\_obj\(\)](#) to free all resources.

Implements [L4::Registry\\_iface](#).

Definition at line 222 of file [object\\_registry](#).

### 15.314.3.3 register\_obj() [2/3]

```
L4::Cap< void > L4Re::Util::Object_registry::register_obj (
    L4::Epiface * o,
    char const * service ) [inline], [override], [virtual]
```

Register a new server object to a pre-allocated receive endpoint.

#### Parameters

<i>o</i>	Server object that handles IPC requests.
<i>service</i>	Name of a pre-allocated receive endpoint.

#### Return values

<i>L4::Cap&lt;void&gt;</i>	The capability known as <i>service</i> on success.
<i>L4::Cap&lt;void&gt;::Invalid</i>	No capability with the given name found.

The interface must be freed with [unregister\\_obj\(\)](#) by the caller to unbind the thread from the capability.

Implements [L4::Registry\\_iface](#).

#### Examples

[examples/clntsrv/server.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 205 of file [object\\_registry](#).

### 15.314.3.4 register\_obj() [3/3]

```
L4::Cap< L4::Rcv_endpoint > L4Re::Util::Object_registry::register_obj (
    L4::Epiface * o,
    L4::Cap< L4::Rcv_endpoint > ep ) [inline], [override], [virtual]
```

Register a handler for an already existing interrupt.

#### Parameters

<i>o</i>	Server object that handles the IPC.
<i>ep</i>	Capability to a receive endpoint, may be a hardware or software interrupt or an IPC gate.

## Return values

<i>L4::Cap&lt;L4::Rcv_endpoint&gt;</i>	Capability ep on success.
<i>L4::Cap&lt;L4::Rcv_endpoint&gt;::Invalid</i>	The IRQ attach operation has failed.

The interface must be freed with [unregister\\_obj\(\)](#) by the caller to unbind the thread from the capability.

Implements [L4::Registry\\_iface](#).

Definition at line 260 of file [object\\_registry](#).

## 15.314.3.5 unregister\_obj()

```
void L4Re::Util::Object_registry::unregister_obj (
    L4::Epiface * o,
    bool unmap = true ) [inline], [override], [virtual]
```

Remove a server object from the handler list.

## Parameters

<i>o</i>	Server object to unbind.
<i>unmap</i>	Specifies if the object capability shall be unmapped (true) or not. The default (true) is to unmap the capability.

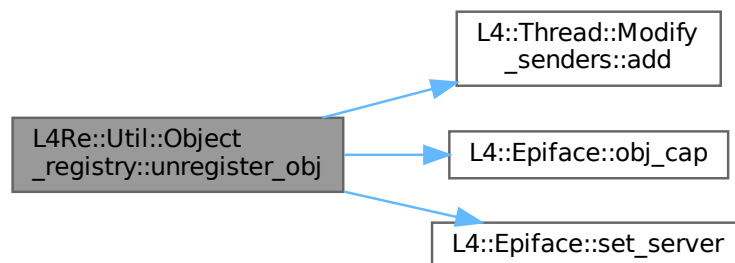
The capability used by the server object will be unmapped if `unmap` is true.

Implements [L4::Registry\\_iface](#).

Definition at line 276 of file [object\\_registry](#).

References [L4::Thread::Modify\\_senders::add\(\)](#), [L4Re::Util::cap\\_alloc](#), [L4\\_FP\\_ALL\\_SPACES](#), [L4::Epiface::obj\\_cap\(\)](#), and [L4::Epiface::set\\_server\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

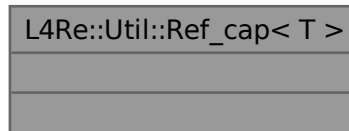
- `l4/re/util/object_registry`

## 15.315 L4Re::Util::Ref\_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap of the capability selector.

```
#include <cap_alloc>
```

Collaboration diagram for L4Re::Util::Ref\_cap< T >:



### 15.315.1 Detailed Description

```
template<typename T>
struct L4Re::Util::Ref_cap< T >
```

Automatic capability that implements automatic free and unmap of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object that is referred by the capability.
----------	--

This kind of automatic capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

#### Usage:

```
L4Re::Util::Ref_cap<L4Re::Dataspace>::Cap global_ds_cap;

{
    L4Re::Util::Ref_cap<L4Re::Dataspace>::Cap
    ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
```

Definition at line 153 of file [cap\\_alloc](#).

The documentation for this struct was generated from the following file:

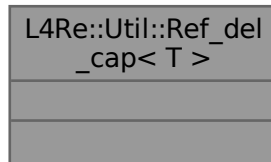
- [l4/re/util/cap\\_alloc](#)

## 15.316 L4Re::Util::Ref\_del\_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap+delete of the capability selector.

```
#include <cap_alloc>
```

Collaboration diagram for L4Re::Util::Ref\_del\_cap< T >:



### 15.316.1 Detailed Description

```
template<typename T>
struct L4Re::Util::Ref_del_cap< T >
```

Automatic capability that implements automatic free and unmap+delete of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object that is referred by the capability.
----------	--

This kind of automatic capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to [Ref\\_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

#### Usage:

```

L4Re::Util::Ref_del_cap<L4Re::Dataspace>::Cap global_ds_cap;

{
    L4Re::Util::Ref_del_cap<L4Re::Dataspace>::Cap
    ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
...
global_ds_cap = L4_INVALID_CAP;
// reference count dropped to 0 (data space shall be deleted).
```

Definition at line 194 of file [cap\\_alloc](#).

The documentation for this struct was generated from the following file:

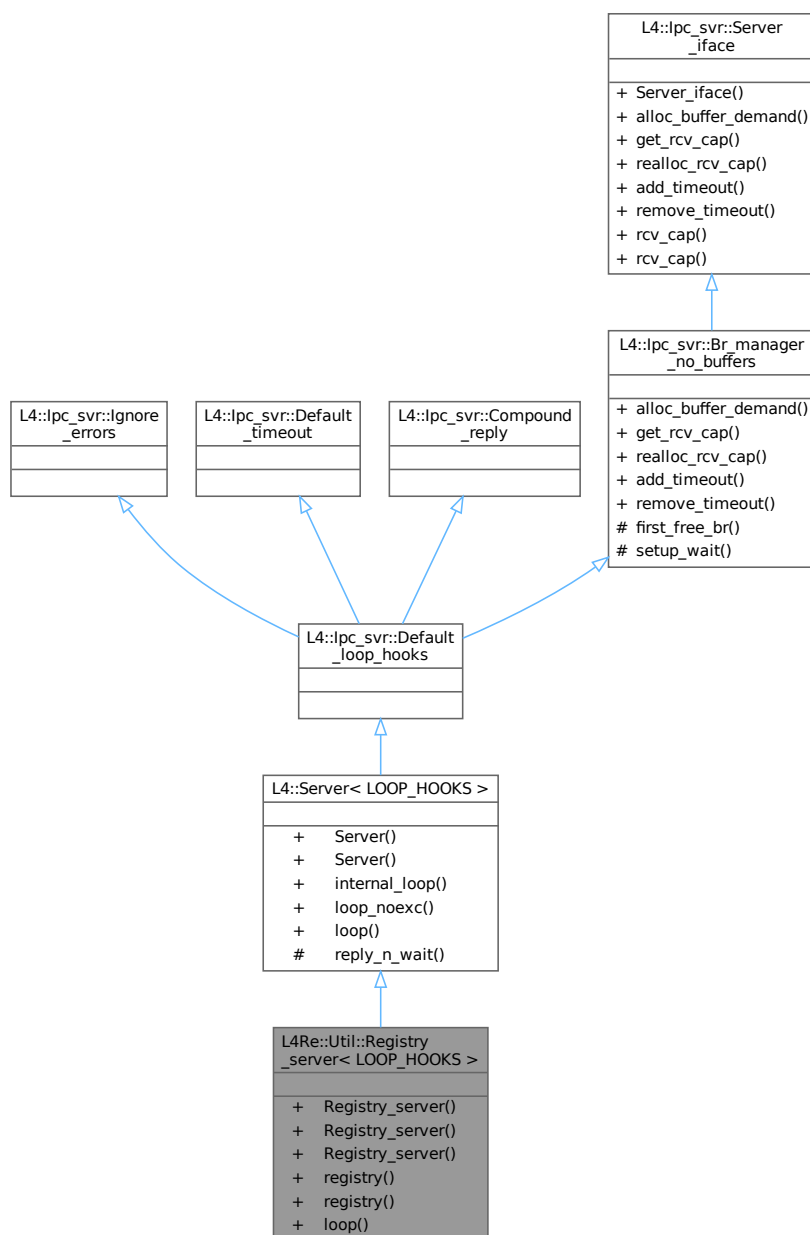
- [l4/re/util/cap\\_alloc](#)

## 15.317 L4Re::Util::Registry\_server< LOOP\_HOOKS > Class Template Reference

A server loop object which has a [Object\\_registry](#) included.

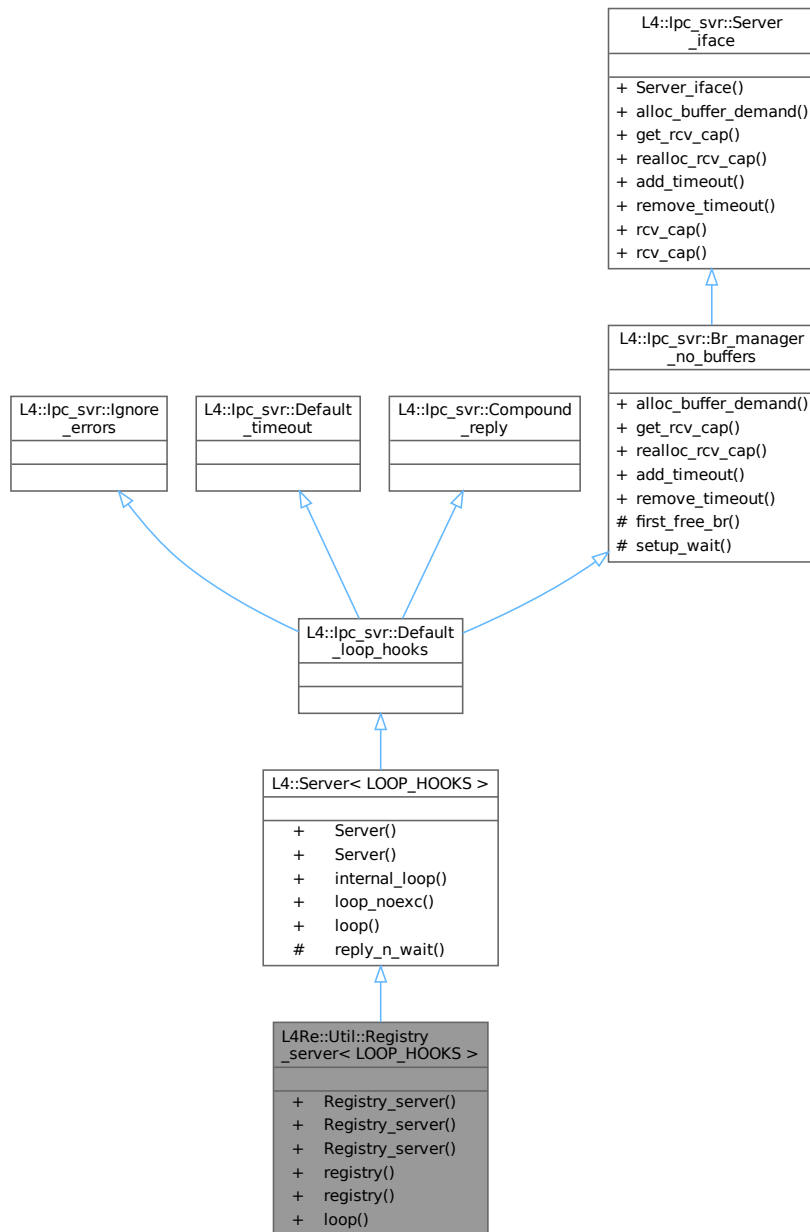
```
#include <object_registry>
```

Inheritance diagram for L4Re::Util::Registry\_server< LOOP\_HOOKS >:





Collaboration diagram for L4Re::Util::Registry\_server< LOOP\_HOOKS >:



## Public Member Functions

- [Registry\\_server \(\)](#)  
Create a new server loop object for the main thread of the task.
- [Registry\\_server \(l4\\_utcb\\_t \\*, L4::Cap< L4::Thread > server, L4::Cap< L4::Factory > factory\)](#)  
Create a new server loop object for an arbitrary thread and factory.
- [Registry\\_server \(L4::Cap< L4::Thread > server, L4::Cap< L4::Factory > factory\)](#)  
Create a new server loop object for an arbitrary thread and factory.
- [Object\\_registry](#) const \* **registry** () const  
Return registry of this server loop.

- [Object\\_registry](#) \* **registry** ()  
*Return registry of this server loop.*
- void [L4\\_NORETURN loop](#) ([l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)())  
*Start the server loop.*

## Public Member Functions inherited from [L4::Server< LOOP\\_HOOKS >](#)

- [Server](#) ([l4\\_utcb\\_t](#) \*)  
*Initializes the server loop.*
- **Server** ()  
*Initializes the server loop.*
- template<typename DISPATCH >  
[L4\\_NORETURN](#) void [internal\\_loop](#) (DISPATCH dispatch, [l4\\_utcb\\_t](#) \*)  
*The server loop.*
- template<typename R >  
[L4\\_NORETURN](#) void **loop\_noexc** (R r, [l4\\_utcb\\_t](#) \*u=[l4\\_utcb](#)())  
*Server loop without exception handling.*
- template<typename EXC , typename R >  
[L4\\_NORETURN](#) void **loop** (R r, [l4\\_utcb\\_t](#) \*u=[l4\\_utcb](#)())  
*Server loop with internal exception handling.*

## Public Member Functions inherited from [L4::lpc\\_svr::Br\\_manager\\_no\\_buffers](#)

- int [alloc\\_buffer\\_demand](#) ([Demand](#) const &demand) override  
*Tells the server to allocate buffers for the given demand.*
- [L4::Cap< void >](#) **get\_rcv\_cap** (int) const override  
*Returns [L4::Cap<void>::Invalid](#), we have no buffer management.*
- int **realloc\_rcv\_cap** (int) override  
*Returns -L4\_ENOMEM, we have no buffer management.*
- int **add\_timeout** ([Timeout](#) \*, [l4\\_kernel\\_clock\\_t](#)) override  
*Returns -L4\_ENOSYS, we have no timeout queue.*
- int **remove\_timeout** ([Timeout](#) \*) override  
*Returns -L4\_ENOSYS, we have no timeout queue.*

## Public Member Functions inherited from [L4::lpc\\_svr::Server\\_iface](#)

- **Server\_iface** ()  
*Make a server interface.*
- template<typename T >  
[L4::Cap< T >](#) **rcv\_cap** (int index) const  
*Get given receive buffer as typed capability.*
- [L4::Cap< void >](#) **rcv\_cap** (int index) const  
*Get receive cap with the given index as generic (void) type.*

## Additional Inherited Members

## Public Types inherited from [L4::lpc\\_svr::Server\\_iface](#)

- typedef [L4::Type\\_info::Demand](#) **Demand**  
*Data type expressing server-side demand for receive buffers.*

## Protected Member Functions inherited from [L4::Server< LOOP\\_HOOKS >](#)

- [l4\\_msgtag\\_t](#) **reply\_n\_wait** ([l4\\_msgtag\\_t](#) reply, [l4\\_umword\\_t](#) \*p, [l4\\_utcb\\_t](#) \*)

*Internal implementation for reply and wait.*

## Protected Member Functions inherited from [L4::lpc\\_svr::Br\\_manager\\_no\\_buffers](#)

- unsigned **first\_free\_br** () const  
*Returns 1 as first free buffer.*
- void **setup\_wait** ([l4\\_utcb\\_t](#) \*utcb, [L4::lpc\\_svr::Reply\\_mode](#))  
*Setup wait function for the server loop (Server<>).*

### 15.317.1 Detailed Description

```
template<typename LOOP_HOOKS = L4::lpc_svr::Default_loop_hooks>
class L4Re::Util::Registry_server< LOOP_HOOKS >
```

A server loop object which has a [Object\\_registry](#) included.

#### Examples

[examples/clntsrv/server.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 307 of file [object\\_registry](#).

### 15.317.2 Constructor & Destructor Documentation

#### 15.317.2.1 Registry\_server() [1/3]

```
template<typename LOOP_HOOKS = L4::lpc_svr::Default_loop_hooks>
L4Re::Util::Registry\_server< LOOP_HOOKS >::Registry_server ( ) [inline]
```

Create a new server loop object for the main thread of the task.

#### Precondition

Must be called from the main thread or behaviour is undefined.

Definition at line 319 of file [object\\_registry](#).

#### 15.317.2.2 Registry\_server() [2/3]

```
template<typename LOOP_HOOKS = L4::lpc_svr::Default_loop_hooks>
L4Re::Util::Registry\_server< LOOP_HOOKS >::Registry_server (
    l4\_utcb\_t * ,
    L4::Cap< L4::Thread > server,
    L4::Cap< L4::Factory > factory ) [inline]
```

Create a new server loop object for an arbitrary thread and factory.

## Parameters

<i>server</i>	Capability to thread running the server loop.
<i>factory</i>	Capability to factory object used to create new IPC gates.

**Deprecated** Note that this variant of the constructor is deprecated, please do not supply the UTCB pointer, it's not used.

Definition at line 331 of file [object\\_registry](#).

### 15.317.2.3 Registry\_server() [3/3]

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
L4Re::Util::Registry_server< LOOP_HOOKS >::Registry_server (
    L4::Cap< L4::Thread > server,
    L4::Cap< L4::Factory > factory ) [inline]
```

Create a new server loop object for an arbitrary thread and factory.

## Parameters

<i>server</i>	Capability to thread running the server loop.
<i>factory</i>	Capability to factory object used to create new IPC gates.

Definition at line 342 of file [object\\_registry](#).

## 15.317.3 Member Function Documentation

### 15.317.3.1 loop()

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
void L4_NORETURN L4Re::Util::Registry_server< LOOP_HOOKS >::loop (
    l4_utcb_t * utcb = l4_utcb() ) [inline]
```

Start the server loop.

## Parameters

<i>utcb</i>	The UTCB of the thread running the server loop, defaults to <a href="#">l4_utcb()</a> .
-------------	---

## Examples

[examples/clntsrv/server.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 358 of file [object\\_registry](#).

The documentation for this class was generated from the following file:

- [l4/re/util/object\\_registry](#)

## 15.318 L4Re::Util::Smart\_cap\_auto< Unmap\_flags > Class Template Reference

Helper for Unique\_cap and Unique\_del\_cap.

```
#include <cap_alloc>
```

Collaboration diagram for L4Re::Util::Smart\_cap\_auto< Unmap\_flags >:

L4Re::Util::Smart_cap_auto< Unmap_flags >	
+	free()
+	invalidate()

### Static Public Member Functions

- static void **free** ([L4::Cap\\_base](#) &c)  
*Free the provided capability.*
- static void **invalidate** ([L4::Cap\\_base](#) &c)  
*Invalidate the provided capability.*

### 15.318.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>  
class L4Re::Util::Smart_cap_auto< Unmap_flags >
```

Helper for Unique\_cap and Unique\_del\_cap.

Definition at line 56 of file [cap\\_alloc](#).

The documentation for this class was generated from the following file:

- [l4/re/util/cap\\_alloc](#)

## 15.319 L4Re::Util::Smart\_count\_cap< Unmap\_flags > Class Template Reference

Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).

```
#include <cap_alloc>
```

Collaboration diagram for L4Re::Util::Smart\_count\_cap< Unmap\_flags >:

L4Re::Util::Smart_count_cap< Unmap_flags >	
+	free()
+	invalidate()
+	copy()

### Static Public Member Functions

- static void **free** ([L4::Cap\\_base](#) &c) noexcept  
*Free operation for [L4::Smart\\_cap](#) (decrement ref count and delete if 0).*
- static void **invalidate** ([L4::Cap\\_base](#) &c) noexcept  
*Invalidate operation for [L4::Smart\\_cap](#).*
- static [L4::Cap\\_base](#) **copy** ([L4::Cap\\_base](#) const &src)  
*Copy operation for [L4::Smart\\_cap](#) (increment ref count).*

### 15.319.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Util::Smart_count_cap< Unmap_flags >
```

Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).

Definition at line 87 of file [cap\\_alloc](#).

The documentation for this class was generated from the following file:

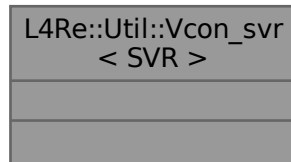
- [l4/re/util/cap\\_alloc](#)

## 15.320 L4Re::Util::Vcon\_svr< SVR > Class Template Reference

[Console](#) server template class.

```
#include <vcon_svr>
```

Collaboration diagram for L4Re::Util::Vcon\_svr< SVR >:



### 15.320.1 Detailed Description

```
template<typename SVR>
class L4Re::Util::Vcon_svr< SVR >
```

[Console](#) server template class.

This template uses `vcon_write()` and `vcon_read()` to get and deliver data from the implementor.

`vcon_read()` needs to update the status argument with the `L4_vcon_read_stat` flags, especially the `L4_VCON_READ_STAT_DONE` flag to indicate that there's nothing more to read for the other end.

`vcon_write()` gets the live data from the UTCB. Make sure to copy out the data before using the UTCB again.

The size parameter of both functions is given in bytes.

Definition at line 47 of file [vcon\\_svr](#).

The documentation for this class was generated from the following file:

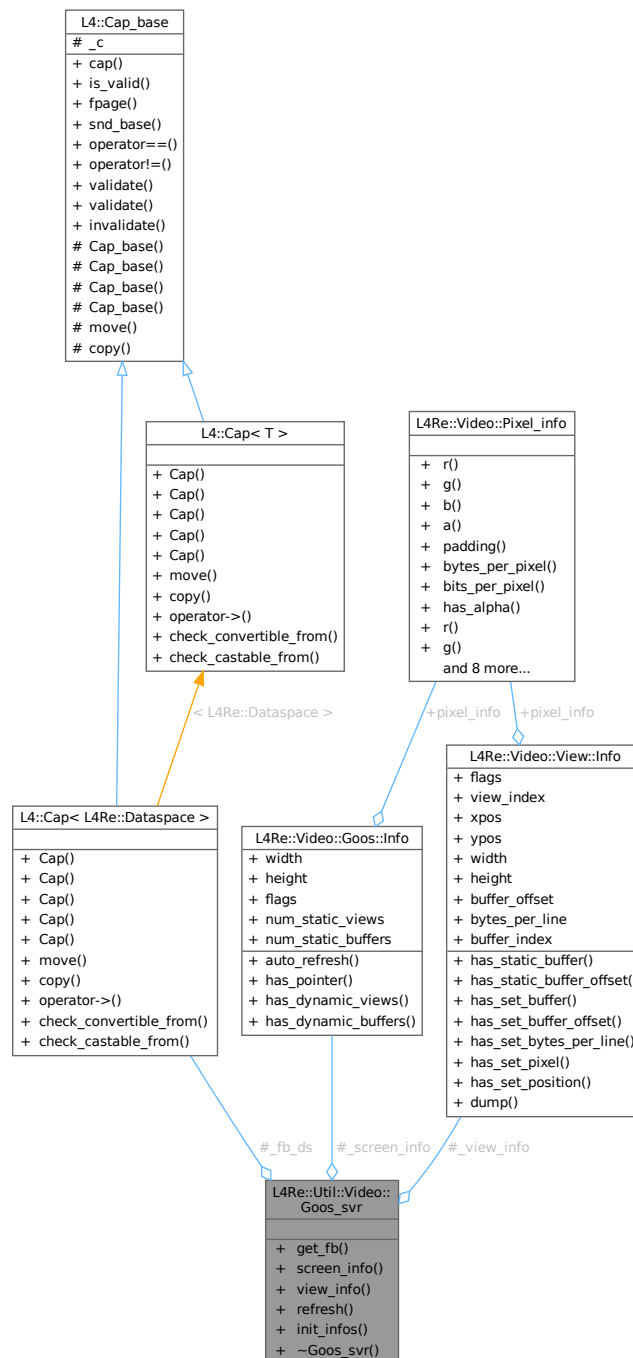
- `l4/re/util/vcon_svr`

## 15.321 L4Re::Util::Video::Goos\_svr Class Reference

Goos server class.

```
#include <goos_svr>
```

Collaboration diagram for L4Re::Util::Video::Goos\_svr:





## Public Member Functions

- [L4::Cap](#)< [L4Re::Dataspace](#) > [get\\_fb](#) () const  
*Return framebuffer memory dataspace.*
- [L4Re::Video::Goos::Info](#) const \* [screen\\_info](#) () const  
*Goos information structure.*
- [L4Re::Video::View::Info](#) const \* [view\\_info](#) () const  
*View information structure.*
- virtual int [refresh](#) (int x, int y, int w, int h)  
*Refresh area of the framebuffer.*
- void [init\\_infos](#) ()  
*Initialize the view information structure of this object.*
- virtual ~[Goos\\_svr](#) ()  
*Destructor.*

## Protected Attributes

- [L4::Cap](#)< [L4Re::Dataspace](#) > [\\_fb\\_ds](#)  
*Goos memory dataspace.*
- [L4Re::Video::Goos::Info](#) [\\_screen\\_info](#)  
*Goos information.*
- [L4Re::Video::View::Info](#) [\\_view\\_info](#)  
*View information.*

## 15.321.1 Detailed Description

Goos server class.

Definition at line 36 of file [goos\\_svr](#).

## 15.321.2 Member Function Documentation

### 15.321.2.1 [get\\_fb\(\)](#)

```
L4::Cap< L4Re::Dataspace > L4Re::Util::Video::Goos\_svr::get\_fb ( ) const [inline]
```

Return framebuffer memory dataspace.

#### Returns

Goos memory dataspace

Definition at line 53 of file [goos\\_svr](#).

References [\\_fb\\_ds](#).

### 15.321.2.2 init\_infos()

```
void L4Re::Util::Video::Goos_svr::init_infos ( ) [inline]
```

Initialize the view information structure of this object.

This function initializes the view info structure of this goos object based on the information in the goos information, i.e. the width, height and pixel\_info of the goos information has to contain valid values before calling init\_info().

Definition at line 89 of file [goos\\_svr](#).

References [\\_screen\\_info](#), [\\_view\\_info](#), [L4Re::Video::View::Info::buffer\\_index](#), [L4Re::Video::View::Info::flags](#), [L4Re::Video::View::Info::height](#), [L4Re::Video::Goos::Info::height](#), [L4Re::Video::View::Info::pixel\\_info](#), [L4Re::Video::Goos::Info::pixel\\_info](#), [L4Re::Video::View::Info::view\\_index](#), [L4Re::Video::View::Info::width](#), [L4Re::Video::Goos::Info::width](#), [L4Re::Video::View::Info::xpos](#), and [L4Re::Video::View::Info::ypos](#).

### 15.321.2.3 refresh()

```
virtual int L4Re::Util::Video::Goos_svr::refresh (
    int x,
    int y,
    int w,
    int h ) [inline], [virtual]
```

Refresh area of the framebuffer.

#### Parameters

<i>x</i>	X coordinate (pixels)
<i>y</i>	Y coordinate (pixels)
<i>w</i>	Width of area in pixels
<i>h</i>	Height of area in pixels

#### Returns

0 on success, negative error code otherwise

Definition at line 77 of file [goos\\_svr](#).

References [L4\\_ENOSYS](#).

### 15.321.2.4 screen\_info()

```
L4Re::Video::Goos::Info const * L4Re::Util::Video::Goos_svr::screen_info ( ) const [inline]
```

Goos information structure.

#### Returns

Return goos information structure.

Definition at line 59 of file [goos\\_svr](#).

References [\\_screen\\_info](#).

## 15.321.2.5 view\_info()

```
L4Re::Video::View::Info const * L4Re::Util::Video::Goos_svr::view_info ( ) const [inline]
```

View information structure.

## Returns

Return view information structure.

Definition at line 65 of file `goos_svr`.

References `_view_info`.

The documentation for this class was generated from the following file:

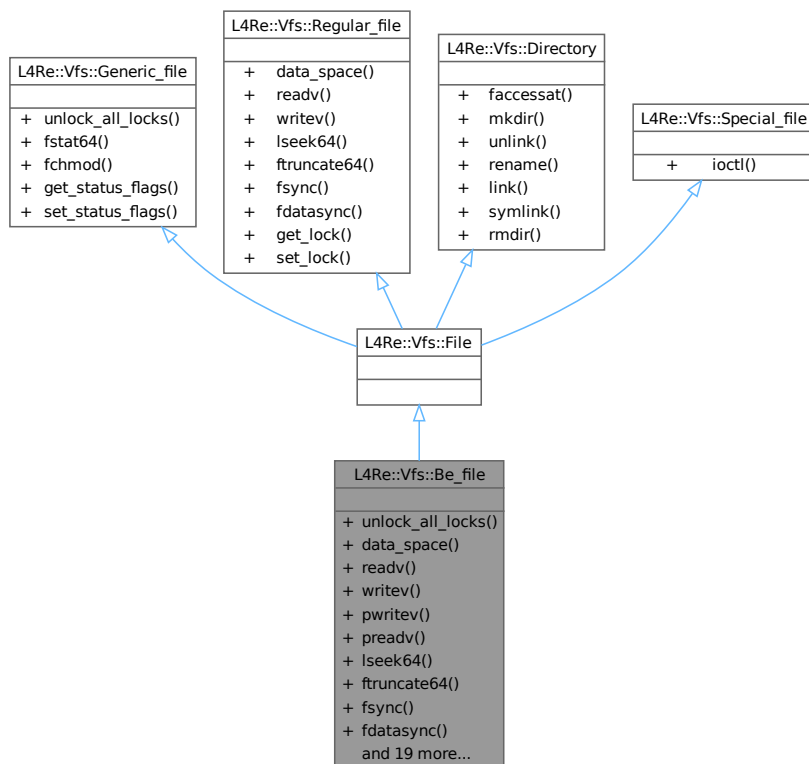
- `l4/re/util/video/goos_svr`

## 15.322 L4Re::Vfs::Be\_file Class Reference

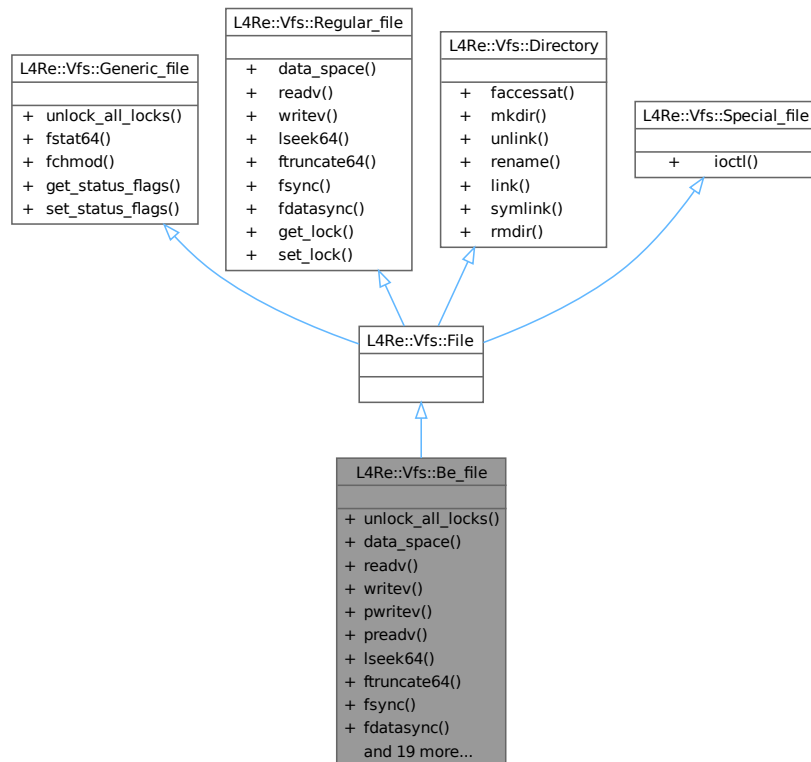
Boiler plate class for implementing an open file for `L4Re::Vfs`.

```
#include <backend>
```

Inheritance diagram for `L4Re::Vfs::Be_file`:



Collaboration diagram for L4Re::Vfs::Be\_file:



## Public Member Functions

- `int unlock_all_locks ()` noexcept override  
*Unlock all locks on the file.*
- `L4::Cap< L4Re::Dataspace > data_space ()` noexcept override  
*Get an [L4Re::Dataspace](#) object for the file.*
- `ssize_t readv (const struct iovec *, int)` noexcept override  
*Default backend for POSIX read and readv functions.*
- `ssize_t writev (const struct iovec *, int)` noexcept override  
*Default backend for POSIX write and writev functions.*
- `ssize_t pwritev (const struct iovec *, int, off64_t)` noexcept override  
*Default backend for POSIX pwrite and pwritev functions.*
- `ssize_t preadv (const struct iovec *, int, off64_t)` noexcept override  
*Default backend for POSIX pread and preadv functions.*
- `off64_t lseek64 (off64_t, int)` noexcept override  
*Default backend for POSIX seek and lseek functions.*
- `int ftruncate64 (off64_t)` noexcept override  
*Default backend for the POSIX truncate, ftruncate and similar functions.*
- `int fsync ()` const noexcept override  
*Default backend for POSIX fsync.*
- `int fdatasync ()` const noexcept override  
*Default backend for POSIX fdatasync.*

- int **ioctl** (unsigned long, va\_list) noexcept override  
*Default backend for POSIX ioctl.*
- int **fstat64** (struct stat64 \*) const noexcept override  
*Get status information for the file.*
- int **fchmod** (mode\_t) noexcept override  
*Default backend for POSIX chmod and fchmod.*
- int **get\_status\_flags** () const noexcept override  
*Default backend for POSIX fcntl subfunctions.*
- int **set\_status\_flags** (long) noexcept override  
*Default backend for POSIX fcntl subfunctions.*
- int **get\_lock** (struct flock64 \*) noexcept override  
*Default backend for POSIX fcntl subfunctions.*
- int **set\_lock** (struct flock64 \*, bool) noexcept override  
*Default backend for POSIX fcntl subfunctions.*
- int **faccessat** (const char \*, int, int) noexcept override  
*Default backend for POSIX access and faccessat functions.*
- int **fchmodat** (const char \*, mode\_t, int) noexcept override  
*Default backend for POSIX fchmodat function.*
- int **utime** (const struct utimbuf \*) noexcept override  
*Default backend for POSIX utime.*
- int **utimes** (const struct timeval[2]) noexcept override  
*Default backend for POSIX utimes.*
- int **utimensat** (const char \*, const struct timespec[2], int) noexcept override  
*Default backend for POSIX utimensat.*
- int **mkdir** (const char \*, mode\_t) noexcept override  
*Default backend for POSIX mkdir and mkdirat.*
- int **unlink** (const char \*) noexcept override  
*Default backend for POSIX unlink, unlinkat.*
- int **rename** (const char \*, const char \*) noexcept override  
*Default backend for POSIX rename, renameat.*
- int **link** (const char \*, const char \*) noexcept override  
*Default backend for POSIX link, linkat.*
- int **symlink** (const char \*, const char \*) noexcept override  
*Default backend for POSIX symlink, symlinkat.*
- int **rmdir** (const char \*) noexcept override  
*Default backend for POSIX rmdir, rmdirat.*
- ssize\_t **readlink** (char \*, size\_t) override  
*Default backend for POSIX readlink, readlinkat.*

### 15.322.1 Detailed Description

Boiler plate class for implementing an open file for [L4Re::Vfs](#).

This class may be used as a base class for everything that a POSIX file descriptor may point to. This are things such as regular files, directories, special device files, streams, pipes, and so on.

#### Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 39 of file [backend](#).

## 15.322.2 Member Function Documentation

### 15.322.2.1 data\_space()

```
L4Re::Cap< L4Re::Dataspace > L4Re::Vfs::Be_file::data_space ( ) [inline], [override], [virtual], [noexcept]
```

Get an [L4Re::Dataspace](#) object for the file.

This is used as a backend for POSIX mmap and mmap2 functions.

#### Note

mmap is not possible if the function returns an invalid capability.

#### Returns

A capability to an [L4Re::Dataspace](#) that represents the file contents in an [L4Re](#) way.

Implements [L4Re::Vfs::Regular\\_file](#).

Definition at line 56 of file [backend](#).

### 15.322.2.2 fstat64()

```
int L4Re::Vfs::Be_file::fstat64 (
    struct stat64 * buf ) const [inline], [override], [virtual], [noexcept]
```

Get status information for the file.

This is the backend for POSIX fstat, stat, fstat64 and friends.

#### Parameters

out	buf	This buffer is filled with the status information.
-----	-----	--

#### Returns

0 on success, or <0 on error.

Implements [L4Re::Vfs::Generic\\_file](#).

Definition at line 95 of file [backend](#).

### 15.322.2.3 unlock\_all\_locks()

```
int L4Re::Vfs::Be_file::unlock_all_locks ( ) [inline], [override], [virtual], [noexcept]
```

Unlock all locks on the file.

**Note**

All locks means all locks independent of which file the locks were taken by.

This method is called by the POSIX close implementation to get the POSIX semantics of releasing all locks taken by this application on a close for any fd referencing the real file.

**Returns**

0 on success, or <0 on error.

Implements [L4Re::Vfs::Generic\\_file](#).

Definition at line 52 of file [backend](#).

The documentation for this class was generated from the following file:

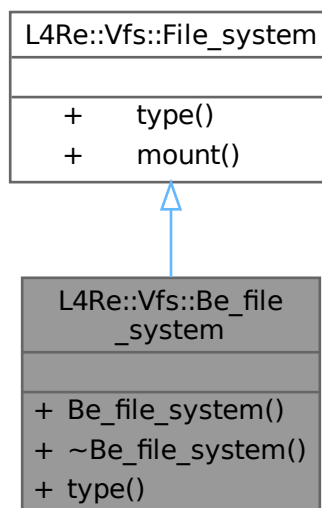
- l4/l4re\_vfs/backend

## 15.323 L4Re::Vfs::Be\_file\_system Class Reference

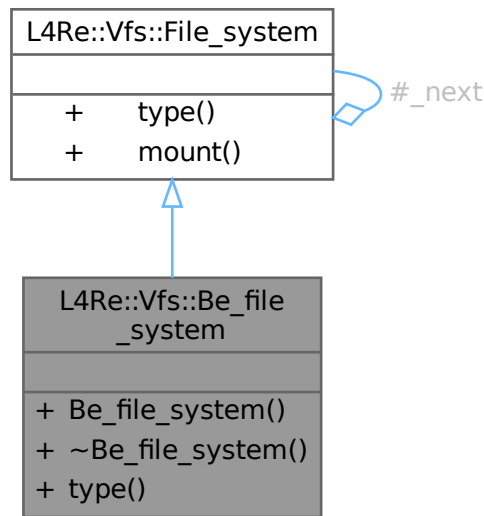
Boilerplate class for implementing a [L4Re::Vfs::File\\_system](#).

```
#include <backend>
```

Inheritance diagram for L4Re::Vfs::Be\_file\_system:



Collaboration diagram for L4Re::Vfs::Be\_file\_system:



### Public Member Functions

- [Be\\_file\\_system](#) (char const \*fstype) noexcept  
*Create a file-system object for the given fstype.*
- [~Be\\_file\\_system](#) () noexcept  
*Destroy a file-system object.*
- char const \* [type](#) () const noexcept override  
*Return the file-system type.*

### Public Member Functions inherited from [L4Re::Vfs::File\\_system](#)

- virtual int [mount](#) (char const \*source, unsigned long mountflags, void const \*data, [cxx::Ref\\_ptr](#)< [File](#) > \*dir) noexcept=0  
*Create a directory object dir representing source mounted with this file system.*

#### 15.323.1 Detailed Description

Boilerplate class for implementing a [L4Re::Vfs::File\\_system](#).

This class already takes care of registering and unregistering the file system in the global registry and implements the [type\(\)](#) method.

#### Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 311 of file [backend](#).



## 15.323.2 Constructor & Destructor Documentation

### 15.323.2.1 Be\_file\_system()

```
L4Re::Vfs::Be_file_system::Be_file_system (
    char const * fstype ) [inline], [explicit], [noexcept]
```

Create a file-system object for the given *fstype*.

#### Parameters

<i>fstype</i>	The type that <a href="#">type()</a> shall return.
---------------	--

This constructor takes care of registering the file system in the registry of L4Re::Vfs::vfs\_ops.

Definition at line 325 of file [backend](#).

### 15.323.2.2 ~Be\_file\_system()

```
L4Re::Vfs::Be_file_system::~~Be_file_system ( ) [inline], [noexcept]
```

Destroy a file-system object.

This destructor takes care of removing this file system from the registry of L4Re::Vfs::vfs\_ops.

Definition at line 337 of file [backend](#).

## 15.323.3 Member Function Documentation

### 15.323.3.1 type()

```
char const * L4Re::Vfs::Be_file_system::type ( ) const [inline], [override], [virtual], [noexcept]
```

Return the file-system type.

Returns the file-system type given as *fstype* in the constructor.

Implements [L4Re::Vfs::File\\_system](#).

Definition at line 347 of file [backend](#).

The documentation for this class was generated from the following file:

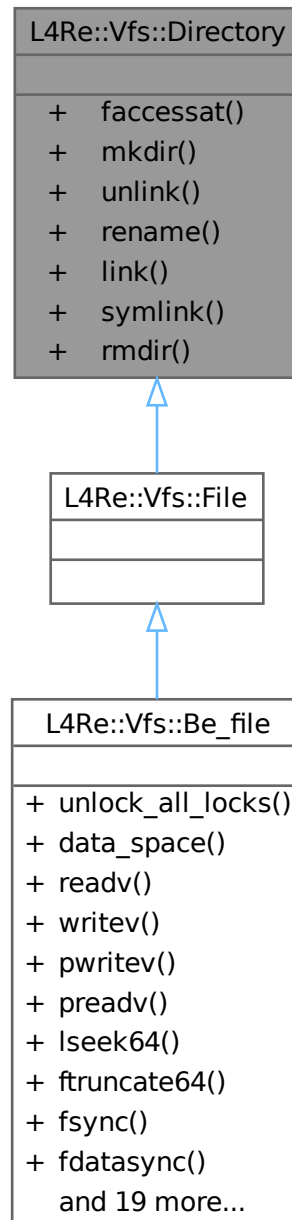
- l4/l4re\_vfs/backend

## 15.324 L4Re::Vfs::Directory Class Reference

Interface for a POSIX file that is a directory.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Directory:



Collaboration diagram for L4Re::Vfs::Directory:

L4Re::Vfs::Directory	
+	faccessat()
+	mkdir()
+	unlink()
+	rename()
+	link()
+	symlink()
+	rmdir()

### Public Member Functions

- virtual int [faccessat](#) (const char \*path, int mode, int flags) noexcept=0  
*Check access permissions on the given file.*
- virtual int [mkdir](#) (const char \*path, mode\_t mode) noexcept=0  
*Create a new subdirectory.*
- virtual int [unlink](#) (const char \*path) noexcept=0  
*Unlink the given file from that directory.*
- virtual int [rename](#) (const char \*src\_path, const char \*dst\_path) noexcept=0  
*Rename the given file.*
- virtual int [link](#) (const char \*src\_path, const char \*dst\_path) noexcept=0  
*Create a hard link (second name) for the given file.*
- virtual int [symlink](#) (const char \*src\_path, const char \*dst\_path) noexcept=0  
*Create a symbolic link for the given file.*
- virtual int [rmdir](#) (const char \*path) noexcept=0  
*Delete an empty directory.*

### 15.324.1 Detailed Description

Interface for a POSIX file that is a directory.

This interface provides functionality for directory files in the [L4Re::Vfs](#). However, real objects always use the combined [L4Re::Vfs::File](#) interface.

Definition at line 140 of file [vfs.h](#).

## 15.324.2 Member Function Documentation

### 15.324.2.1 faccessat()

```
virtual int L4Re::Vfs::Directory::faccessat (
    const char * path,
    int mode,
    int flags ) [pure virtual], [noexcept]
```

Check access permissions on the given file.

Backend function for POSIX access and faccessat functions.

#### Parameters

<i>path</i>	The path relative to this directory. Note: <i>path</i> is relative to this directory and may contain subdirectories.
<i>mode</i>	The access mode to check.
<i>flags</i>	The flags as in POSIX faccessat (AT_EACCESS, AT_SYMLINK_NOFOLLOW).

#### Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 15.324.2.2 link()

```
virtual int L4Re::Vfs::Directory::link (
    const char * src_path,
    const char * dst_path ) [pure virtual], [noexcept]
```

Create a hard link (second name) for the given file.

Backend for the POSIX link and linkat functions.

#### Parameters

<i>src_path</i>	The old name of the file. Note: <i>src_path</i> is relative to this directory and may contain subdirectories.
<i>dst_path</i>	The new (second) name for the file. Note: <i>dst_path</i> is relative to this directory and may contain subdirectories.

#### Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 15.324.2.3 mkdir()

```
virtual int L4Re::Vfs::Directory::mkdir (
    const char * path,
    mode_t mode ) [pure virtual], [noexcept]
```

Create a new subdirectory.

Backend for POSIX mkdir and mkdirat function calls.

#### Parameters

<i>path</i>	The name of the subdirectory to create. Note: <i>path</i> is relative to this directory and may contain subdirectories.
<i>mode</i>	The file mode to use for the new directory.

#### Returns

0 on success, or <0 on error. -ENOTDIR if this or some component in path is not a directory.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 15.324.2.4 rename()

```
virtual int L4Re::Vfs::Directory::rename (
    const char * src_path,
    const char * dst_path ) [pure virtual], [noexcept]
```

Rename the given file.

Backend for the POSIX rename, renameat functions.

#### Parameters

<i>src_path</i>	The old name of the file to rename. Note: <i>src_path</i> is relative to this directory and may contain subdirectories.
<i>dst_path</i>	The new name for the file. Note: <i>dst_path</i> is relative to this directory and may contain subdirectories.

#### Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 15.324.2.5 rmdir()

```
virtual int L4Re::Vfs::Directory::rmdir (
    const char * path ) [pure virtual], [noexcept]
```

Delete an empty directory.

Backend for POSIX rmdir, rmdirat functions.

**Parameters**

<i>path</i>	The name of the directory to remove. Note: <i>path</i> is relative to this directory and may contain subdirectories.
-------------	--

**Returns**

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

**15.324.2.6 symlink()**

```
virtual int L4Re::Vfs::Directory::symlink (
    const char * src_path,
    const char * dst_path ) [pure virtual], [noexcept]
```

Create a symbolic link for the given file.

Backend for the POSIX symlink and symlinkat functions.

**Parameters**

<i>src_path</i>	The old name of the file. Note: <i>src_path</i> shall be an absolute path.
<i>dst_path</i>	The name for symlink. Note: <i>dst_path</i> is relative to this directory and may contain subdirectories.

**Returns**

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

**15.324.2.7 unlink()**

```
virtual int L4Re::Vfs::Directory::unlink (
    const char * path ) [pure virtual], [noexcept]
```

Unlink the given file from that directory.

Backend for the POSIX unlink and unlinkat functions.

**Parameters**

<i>path</i>	The name of the file to unlink. Note: <i>path</i> is relative to this directory and may contain subdirectories.
-------------	---

**Returns**

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

The documentation for this class was generated from the following file:

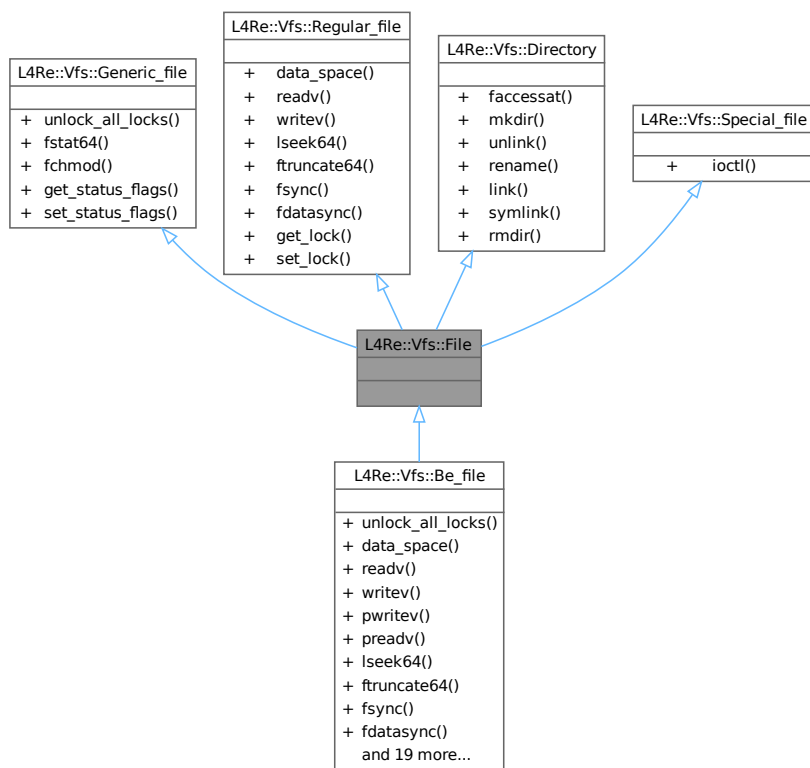
- l4/l4re\_vfs/vfs.h

## 15.325 L4Re::Vfs::File Class Reference

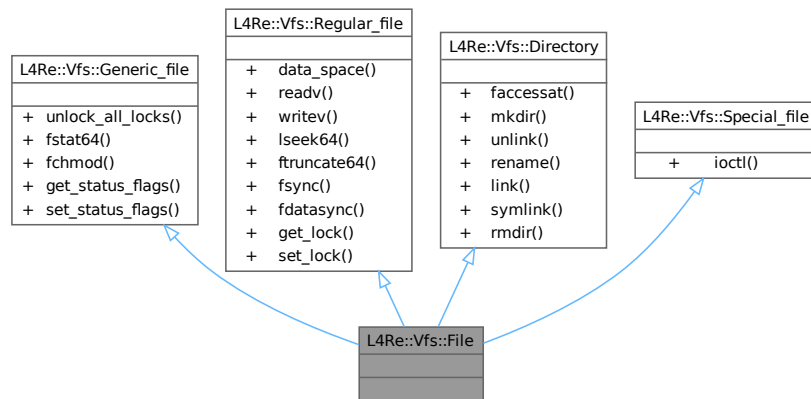
The basic interface for an open POSIX file.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::File:



Collaboration diagram for L4Re::Vfs::File:



### Additional Inherited Members

#### Public Member Functions inherited from L4Re::Vfs::Generic\_file

- virtual int [unlock\\_all\\_locks](#) () noexcept=0  
*Unlock all locks on the file.*
- virtual int [fstat64](#) (struct stat64 \*buf) const noexcept=0  
*Get status information for the file.*
- virtual int [fchmod](#) (mode\_t) noexcept=0  
*Change POSIX access rights on the file.*
- virtual int [get\\_status\\_flags](#) () const noexcept=0  
*Get file status flags (fcntl F\_GETFL).*
- virtual int [set\\_status\\_flags](#) (long flags) noexcept=0  
*Set file status flags (fcntl F\_SETFL).*

#### Public Member Functions inherited from L4Re::Vfs::Regular\_file

- virtual [L4::Cap< L4Re::Dataspace > data\\_space](#) () noexcept=0  
*Get an L4Re::Dataspace object for the file.*
- virtual ssize\_t [readv](#) (const struct iovec \*, int iovcnt) noexcept=0  
*Read one or more blocks of data from the file.*
- virtual ssize\_t [writev](#) (const struct iovec \*, int iovcnt) noexcept=0  
*Write one or more blocks of data to the file.*
- virtual off64\_t [lseek64](#) (off64\_t, int) noexcept=0  
*Change the file pointer.*
- virtual int [ftruncate64](#) (off64\_t pos) noexcept=0  
*Truncate the file at the given position.*
- virtual int [fsync](#) () const noexcept=0  
*Sync the data and meta data to persistent storage.*
- virtual int [fdatasync](#) () const noexcept=0  
*Sync the data to persistent storage.*
- virtual int [get\\_lock](#) (struct flock64 \*lock) noexcept=0  
*Test if the given lock can be placed in the file.*
- virtual int [set\\_lock](#) (struct flock64 \*lock, bool wait) noexcept=0  
*Acquire or release the given lock on the file.*



## Public Member Functions inherited from [L4Re::Vfs::Directory](#)

- virtual int [faccessat](#) (const char \*path, int mode, int flags) noexcept=0  
*Check access permissions on the given file.*
- virtual int [mkdir](#) (const char \*path, mode\_t mode) noexcept=0  
*Create a new subdirectory.*
- virtual int [unlink](#) (const char \*path) noexcept=0  
*Unlink the given file from that directory.*
- virtual int [rename](#) (const char \*src\_path, const char \*dst\_path) noexcept=0  
*Rename the given file.*
- virtual int [link](#) (const char \*src\_path, const char \*dst\_path) noexcept=0  
*Create a hard link (second name) for the given file.*
- virtual int [symlink](#) (const char \*src\_path, const char \*dst\_path) noexcept=0  
*Create a symbolic link for the given file.*
- virtual int [rmdir](#) (const char \*path) noexcept=0  
*Delete an empty directory.*

## Public Member Functions inherited from [L4Re::Vfs::Special\\_file](#)

- virtual int [ioctl](#) (unsigned long cmd, va\_list args) noexcept=0  
*The famous IO control.*

### 15.325.1 Detailed Description

The basic interface for an open POSIX file.

An open POSIX file can be anything that hides behind a POSIX file descriptor. This means that even directories are files. An open file can be anything from a directory to a special device file so see [Generic\\_file](#), [Regular\\_file](#), [Directory](#), and [Special\\_file](#) for more information.

#### Note

For implementing a backend for the [L4Re::Vfs](#) [L4Re::Vfs::Be\\_file](#) may be used as a base class.

Definition at line 435 of file [vfs.h](#).

The documentation for this class was generated from the following file:

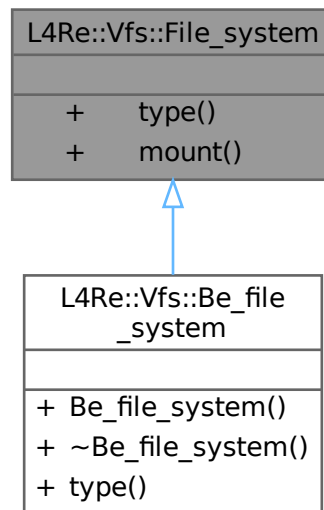
- [l4/l4re\\_vfs/vfs.h](#)

## 15.326 L4Re::Vfs::File\_system Class Reference

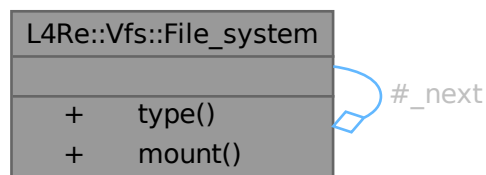
Basic interface for an [L4Re::Vfs](#) file system.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::File\_system:



Collaboration diagram for L4Re::Vfs::File\_system:



### Public Member Functions

- virtual char const \* [type](#) () const noexcept=0  
*Returns the type of the file system used in mount as fstype argument.*
- virtual int [mount](#) (char const \*source, unsigned long mountflags, void const \*data, [cxx::Ref\\_ptr](#)< [File](#) > \*dir) noexcept=0  
*Create a directory object dir representing source mounted with this file system.*

## 15.326.1 Detailed Description

Basic interface for an [L4Re::Vfs](#) file system.

### Note

For implementing a special file system [L4Re::Vfs::Be\\_file\\_system](#) may be used as a base class.

The main purpose of this interface is to have a single object for each supported file-system type (e.g., ext2, vfat) that exists in the application and is registered at the [L4Re::Vfs::Fs](#) singleton available via [L4Re::Vfs::vfs\\_ops](#). Ultimately, the POSIX mount function calls the [File\\_system::mount](#) method matching the file-system type given in mount.

Definition at line 831 of file [vfs.h](#).

## 15.326.2 Member Function Documentation

### 15.326.2.1 mount()

```
virtual int L4Re::Vfs::File_system::mount (
    char const * source,
    unsigned long mountflags,
    void const * data,
    cxx::Ref\_ptr< File > * dir ) [pure virtual], [noexcept]
```

Create a directory object *dir* representing *source* mounted with this file system.

#### Parameters

	<i>source</i>	The path to the source device to mount. This may also be some URL or anything file-system specific.
	<i>mountflags</i>	The mount flags as specified in the POSIX mount call.
	<i>data</i>	The data as specified in the POSIX mount call. The contents are file-system specific.
out	<i>dir</i>	A new directory object representing the file-system root directory.

#### Returns

0 on success, and <0 on error (e.g. -EINVAL).

### 15.326.2.2 type()

```
virtual char const * L4Re::Vfs::File_system::type ( ) const [pure virtual], [noexcept]
```

Returns the type of the file system used in mount as fstype argument.

### Note

This method is already provided by [Be\\_file\\_system](#).

Implemented in [L4Re::Vfs::Be\\_file\\_system](#).

The documentation for this class was generated from the following file:

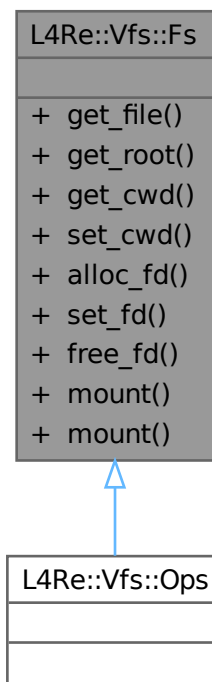
- [l4/l4re\\_vfs/vfs.h](#)

## 15.327 L4Re::Vfs::Fs Class Reference

POSIX File-system related functionality.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Fs:



Collaboration diagram for L4Re::Vfs::Fs:

L4Re::Vfs::Fs
<ul style="list-style-type: none"> <li>+ <a href="#">get_file()</a></li> <li>+ <a href="#">get_root()</a></li> <li>+ <a href="#">get_cwd()</a></li> <li>+ <a href="#">set_cwd()</a></li> <li>+ <a href="#">alloc_fd()</a></li> <li>+ <a href="#">set_fd()</a></li> <li>+ <a href="#">free_fd()</a></li> <li>+ <a href="#">mount()</a></li> <li>+ <a href="#">mount()</a></li> </ul>

## Public Member Functions

- virtual [cxx::Ref\\_ptr< File >](#) [get\\_file](#) (int fd) noexcept=0  
*Get the [L4Re::Vfs::File](#) for the file descriptor fd.*
- virtual [cxx::Ref\\_ptr< File >](#) [get\\_root](#) () noexcept=0  
*Get the directory object for the application's root directory.*
- virtual [cxx::Ref\\_ptr< File >](#) [get\\_cwd](#) () noexcept  
*Get the directory object for the application's current working directory.*
- virtual void [set\\_cwd](#) ([cxx::Ref\\_ptr< File >](#) const &) noexcept  
*Set the current working directory for the application.*
- virtual int [alloc\\_fd](#) ([cxx::Ref\\_ptr< File >](#) const &f=[cxx::Ref\\_ptr<>::Nil](#)) noexcept=0  
*Allocate the next free file descriptor.*
- virtual [cxx::Pair< cxx::Ref\\_ptr< File >, int >](#) [set\\_fd](#) (int fd, [cxx::Ref\\_ptr< File >](#) const &f=[cxx::Ref\\_ptr<>::Nil](#)) noexcept=0  
*Set the file object referenced by the file descriptor fd.*
- virtual [cxx::Ref\\_ptr< File >](#) [free\\_fd](#) (int fd) noexcept=0  
*Free the file descriptor fd.*
- virtual int [mount](#) (char const \*path, [cxx::Ref\\_ptr< File >](#) const &dir) noexcept=0  
*Mount a given file object at the given global path in the VFS.*
- int [mount](#) (char const \*source, char const \*target, char const \*fstype, unsigned long mountflags, void const \*data) noexcept  
*Backend for the POSIX mount call.*

### 15.327.1 Detailed Description

POSIX File-system related functionality.

**Note**

This class usually exists as a singleton and as a superclass of [L4Re::Vfs::Ops](#) (

**See also**

[L4Re::Vfs::vfs\\_ops](#)).

Definition at line 920 of file [vfs.h](#).

## 15.327.2 Member Function Documentation

### 15.327.2.1 `alloc_fd()`

```
virtual int L4Re::Vfs::Fs::alloc_fd (
    cxx::Ref_ptr< File > const & f = cxx::Ref_ptr<>::Nil ) [pure virtual], [noexcept]
```

Allocate the next free file descriptor.

**Parameters**

<i>f</i>	The file to assign to that file descriptor.
----------	---

**Returns**

The allocated file descriptor, or -EMFILE on error.

### 15.327.2.2 `free_fd()`

```
virtual cxx::Ref_ptr< File > L4Re::Vfs::Fs::free_fd (
    int fd ) [pure virtual], [noexcept]
```

Free the file descriptor *fd*.

**Parameters**

<i>fd</i>	The file descriptor to free.
-----------	------------------------------

**Returns**

A pointer to the file object that was assigned to the *fd*.

### 15.327.2.3 `get_file()`

```
virtual cxx::Ref_ptr< File > L4Re::Vfs::Fs::get_file (
    int fd ) [pure virtual], [noexcept]
```

Get the [L4Re::Vfs::File](#) for the file descriptor *fd*.

## Parameters

<i>fd</i>	The POSIX file descriptor number.
-----------	-----------------------------------

## Returns

A pointer to the [File](#) object, or 0 if *fd* is not open.

## 15.327.2.4 mount()

```
virtual int L4Re::Vfs::Fs::mount (
    char const * path,
    cxx::Ref_ptr< File > const & dir ) [pure virtual], [noexcept]
```

Mount a given file object at the given global path in the VFS.

## Parameters

<i>path</i>	The global path to mount <i>dir</i> at.
<i>dir</i>	A pointer to the file/directory object that shall be mounted at <i>path</i> .

## Returns

0 on success, or <0 on error.

## 15.327.2.5 set\_fd()

```
virtual cxx::Pair< cxx::Ref_ptr< File >, int > L4Re::Vfs::Fs::set_fd (
    int fd,
    cxx::Ref_ptr< File > const & f = cxx::Ref_ptr<>::Nil ) [pure virtual], [noexcept]
```

Set the file object referenced by the file descriptor *fd*.

## Parameters

<i>fd</i>	The file descriptor to set to <i>f</i> .
<i>f</i>	The file object to assign.

## Returns

A pair of a pointer to the file object that was previously assigned to *fd* (*first*) and a return value (*second*). *second* contains `-#EBADF` if the passed file descriptor is outside the valid range. *first* contains a Nil pointer in that case. On success, *second* contains 0.

The documentation for this class was generated from the following files:

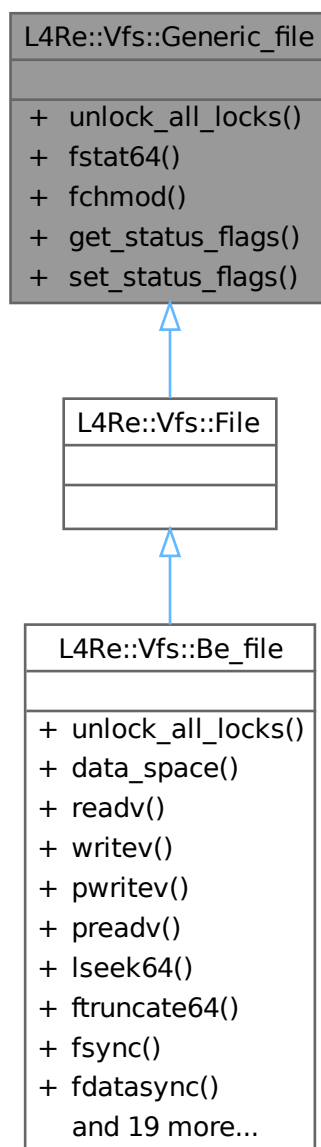
- l4/l4re\_vfs/vfs.h
- l4/l4re\_vfs/impl/vfs\_impl.h

## 15.328 L4Re::Vfs::Generic\_file Class Reference

The common interface for an open POSIX file.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Generic\_file:





Collaboration diagram for L4Re::Vfs::Generic\_file:

L4Re::Vfs::Generic_file
<ul style="list-style-type: none"> <li>+ unlock_all_locks()</li> <li>+ fstat64()</li> <li>+ fchmod()</li> <li>+ get_status_flags()</li> <li>+ set_status_flags()</li> </ul>

### Public Member Functions

- virtual int [unlock\\_all\\_locks](#) () noexcept=0  
*Unlock all locks on the file.*
- virtual int [fstat64](#) (struct stat64 \*buf) const noexcept=0  
*Get status information for the file.*
- virtual int [fchmod](#) (mode\_t) noexcept=0  
*Change POSIX access rights on the file.*
- virtual int [get\\_status\\_flags](#) () const noexcept=0  
*Get file status flags (fcntl F\_GETFL).*
- virtual int [set\\_status\\_flags](#) (long flags) noexcept=0  
*Set file status flags (fcntl F\_SETFL).*

## 15.328.1 Detailed Description

The common interface for an open POSIX file.

This interface is common to all kinds of open files, independent of the file type (e.g., directory, regular file etc.). However, in the [L4Re::Vfs](#) the interface [File](#) is used for every real object.

See also

[L4Re::Vfs::File](#) for more information.

Definition at line 62 of file [vfs.h](#).

## 15.328.2 Member Function Documentation

### 15.328.2.1 fchmod()

```
virtual int L4Re::Vfs::Generic_file::fchmod (
    mode_t ) [pure virtual], [noexcept]
```

Change POSIX access rights on the file.

Backend for POSIX chmod and fchmod.

Implemented in [L4Re::Vfs::Be\\_file](#).

**15.328.2.2 fstat64()**

```
virtual int L4Re::Vfs::Generic_file::fstat64 (
    struct stat64 * buf ) const [pure virtual], [noexcept]
```

Get status information for the file.

This is the backend for POSIX fstat, stat, fstat64 and friends.

**Parameters**

out	buf	This buffer is filled with the status information.
-----	-----	--

**Returns**

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

**15.328.2.3 get\_status\_flags()**

```
virtual int L4Re::Vfs::Generic_file::get_status_flags ( ) const [pure virtual], [noexcept]
```

Get file status flags (fcntl F\_GETFL).

This function is used by the fcntl implementation for the F\_GETFL command.

**Returns**

flags such as O\_RDONLY, O\_WRONLY, O\_RDWR, O\_DIRECT, O\_ASYNC, O\_NOATIME, O\_NONBLOCK, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

**15.328.2.4 set\_status\_flags()**

```
virtual int L4Re::Vfs::Generic_file::set_status_flags (
    long flags ) [pure virtual], [noexcept]
```

Set file status flags (fcntl F\_SETFL).

This function is used by the fcntl implementation for the F\_SETFL command.

**Parameters**

flags	The file status flags to set. This must be a combination of O_RDONLY, O_WRONLY, O_RDWR, O_APPEND, O_ASYNC, O_DIRECT, O_NOATIME, O_NONBLOCK.
-------	---

**Note**

Creation flags such as `O_CREAT`, `O_EXCL`, `O_NOCTTY`, `O_TRUNC` are ignored.

**Returns**

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

**15.328.2.5 unlock\_all\_locks()**

```
virtual int L4Re::Vfs::Generic_file::unlock_all_locks ( ) [pure virtual], [noexcept]
```

Unlock all locks on the file.

**Note**

All locks means all locks independent of which file the locks were taken by.

This method is called by the POSIX close implementation to get the POSIX semantics of releasing all locks taken by this application on a close for any fd referencing the real file.

**Returns**

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

The documentation for this class was generated from the following file:

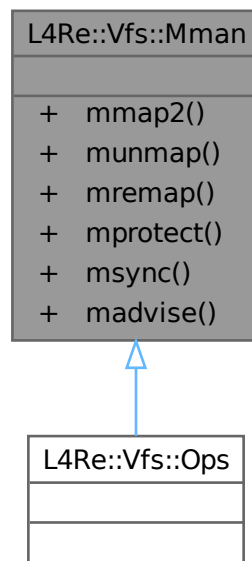
- `l4/l4re_vfs/vfs.h`

## 15.329 L4Re::Vfs::Mman Class Reference

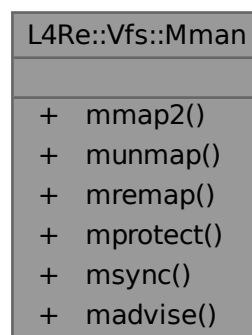
Interface for POSIX memory management.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Mman:



Collaboration diagram for L4Re::Vfs::Mman:



## Public Member Functions

- virtual int **mmap2** (void \*start, size\_t len, int prot, int flags, int fd, off\_t offset, void \*\*ptr) noexcept=0  
*Backend for the mmap2 system call.*
- virtual int **munmap** (void \*start, size\_t len) noexcept=0  
*Backend for the munmap system call.*

- virtual int **mremap** (void \*old, size\_t old\_sz, size\_t new\_sz, int flags, void \*\*new\_addr) noexcept=0  
*Backend for the mremap system call.*
- virtual int **mprotect** (const void \*a, size\_t sz, int prot) noexcept=0  
*Backend for the mprotect system call.*
- virtual int **msync** (void \*addr, size\_t len, int flags) noexcept=0  
*Backend for the msync system call.*
- virtual int **madvice** (void \*addr, size\_t len, int advice) noexcept=0  
*Backend for the madvice system call.*

### 15.329.1 Detailed Description

Interface for POSIX memory management.

#### Note

This interface usually exists as a singleton and as a superclass of [L4Re::Vfs::Ops](#).

An implementation for this interface is in [l4/l4re\\_vfs/impl/vfs\\_impl.h](#) and used by the l4re\_vfs library or by the VFS implementation in ldso.

Definition at line 747 of file [vfs.h](#).

The documentation for this class was generated from the following file:

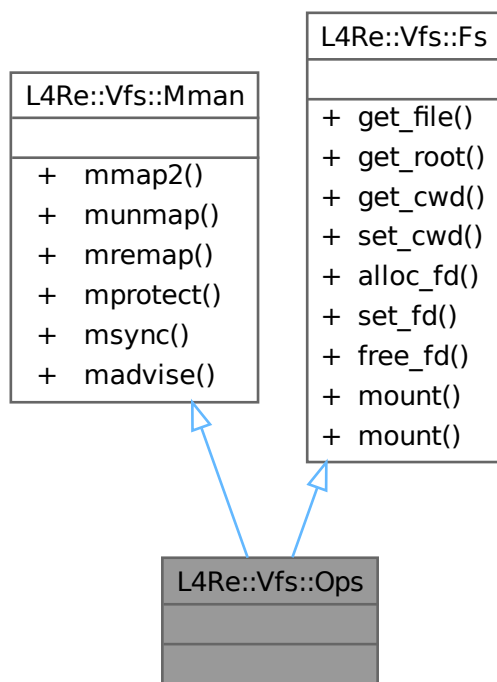
- l4/l4re\_vfs/vfs.h

## 15.330 L4Re::Vfs::Ops Class Reference

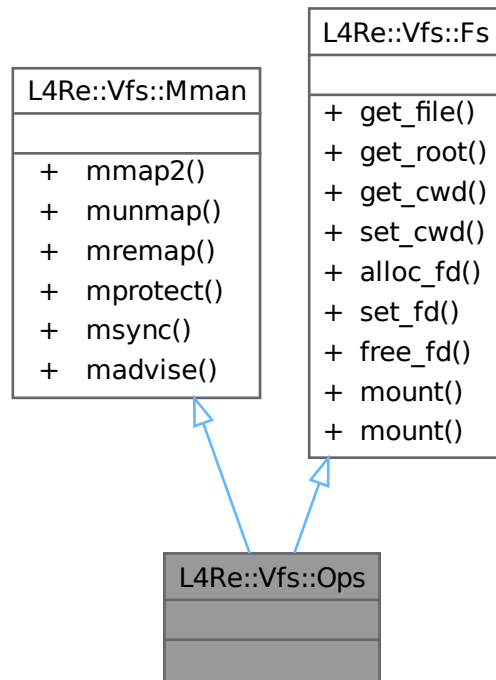
Interface for the POSIX backends of an application.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Ops:



Collaboration diagram for L4Re::Vfs::Ops:



### Additional Inherited Members

#### Public Member Functions inherited from L4Re::Vfs::Mman

- virtual int **mmap2** (void \*start, size\_t len, int prot, int flags, int fd, off\_t offset, void \*\*ptr) noexcept=0  
*Backend for the mmap2 system call.*
- virtual int **munmap** (void \*start, size\_t len) noexcept=0  
*Backend for the munmap system call.*
- virtual int **mremap** (void \*old, size\_t old\_sz, size\_t new\_sz, int flags, void \*\*new\_addr) noexcept=0  
*Backend for the mremap system call.*
- virtual int **mprotect** (const void \*a, size\_t sz, int prot) noexcept=0  
*Backend for the mprotect system call.*
- virtual int **msync** (void \*addr, size\_t len, int flags) noexcept=0  
*Backend for the msync system call.*
- virtual int **madvise** (void \*addr, size\_t len, int advice) noexcept=0  
*Backend for the madvice system call.*

#### Public Member Functions inherited from L4Re::Vfs::Fs

- virtual [cxx::Ref\\_ptr< File > get\\_file](#) (int fd) noexcept=0  
*Get the [L4Re::Vfs::File](#) for the file descriptor fd.*

- virtual `cxx::Ref_ptr< File > get_root ()` noexcept=0  
*Get the directory object for the application's root directory.*
- virtual `cxx::Ref_ptr< File > get_cwd ()` noexcept  
*Get the directory object for the application's current working directory.*
- virtual void `set_cwd (cxx::Ref_ptr< File > const &)` noexcept  
*Set the current working directory for the application.*
- virtual int `alloc_fd (cxx::Ref_ptr< File > const &f=cxx::Ref_ptr<>::Nil)` noexcept=0  
*Allocate the next free file descriptor.*
- virtual `cxx::Pair< cxx::Ref_ptr< File >, int > set_fd (int fd, cxx::Ref_ptr< File > const &f=cxx::Ref_ptr<>::Nil)` noexcept=0  
*Set the file object referenced by the file descriptor fd.*
- virtual `cxx::Ref_ptr< File > free_fd (int fd)` noexcept=0  
*Free the file descriptor fd.*
- virtual int `mount (char const *path, cxx::Ref_ptr< File > const &dir)` noexcept=0  
*Mount a given file object at the given global path in the VFS.*
- int `mount (char const *source, char const *target, char const *fstype, unsigned long mountflags, void const *data)` noexcept  
*Backend for the POSIX mount call.*

### 15.330.1 Detailed Description

Interface for the POSIX backends of an application.

#### Note

There usually exists a single instance of this interface available via `L4Re::Vfs::vfs_ops` that is used for all kinds of C-Library functions.

Definition at line 1083 of file `vfs.h`.

The documentation for this class was generated from the following file:

- `l4/l4re_vfs/vfs.h`

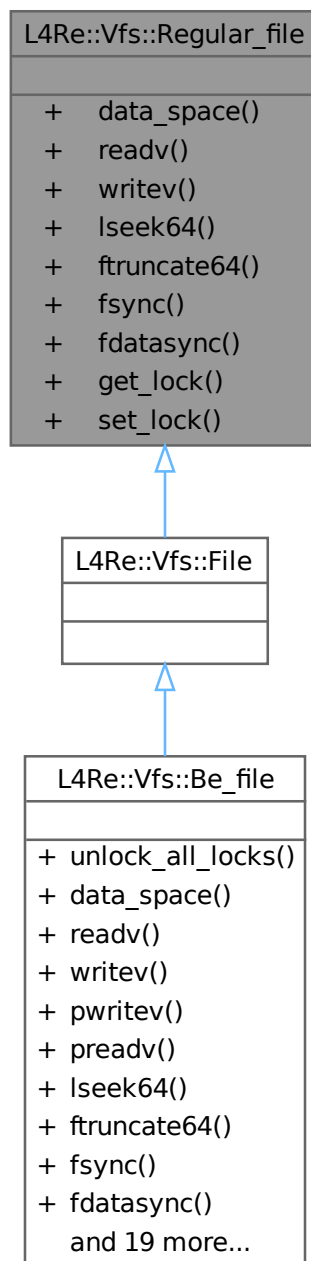
## 15.331 L4Re::Vfs::Regular\_file Class Reference

Interface for a POSIX file that provides regular file semantics.

```
#include <vfs.h>
```



Inheritance diagram for L4Re::Vfs::Regular\_file:



Collaboration diagram for L4Re::Vfs::Regular\_file:

L4Re::Vfs::Regular_file
<ul style="list-style-type: none"> <li>+ data_space()</li> <li>+ readv()</li> <li>+ writev()</li> <li>+ lseek64()</li> <li>+ ftruncate64()</li> <li>+ fsync()</li> <li>+ fdatasync()</li> <li>+ get_lock()</li> <li>+ set_lock()</li> </ul>

## Public Member Functions

- virtual [L4::Cap](#)< [L4Re::Dataspace](#) > [data\\_space](#) () noexcept=0  
*Get an [L4Re::Dataspace](#) object for the file.*
- virtual ssize\_t [readv](#) (const struct iovec \*, int iovcnt) noexcept=0  
*Read one or more blocks of data from the file.*
- virtual ssize\_t [writev](#) (const struct iovec \*, int iovcnt) noexcept=0  
*Write one or more blocks of data to the file.*
- virtual off64\_t [lseek64](#) (off64\_t, int) noexcept=0  
*Change the file pointer.*
- virtual int [ftruncate64](#) (off64\_t pos) noexcept=0  
*Truncate the file at the given position.*
- virtual int [fsync](#) () const noexcept=0  
*Sync the data and meta data to persistent storage.*
- virtual int [fdatasync](#) () const noexcept=0  
*Sync the data to persistent storage.*
- virtual int [get\\_lock](#) (struct flock64 \*lock) noexcept=0  
*Test if the given lock can be placed in the file.*
- virtual int [set\\_lock](#) (struct flock64 \*lock, bool wait) noexcept=0  
*Acquire or release the given lock on the file.*

### 15.331.1 Detailed Description

Interface for a POSIX file that provides regular file semantics.

Real objects always use the combined [L4Re::Vfs::File](#) interface.

Definition at line 267 of file [vfs.h](#).

## 15.331.2 Member Function Documentation

### 15.331.2.1 data\_space()

```
virtual L4::Cap< L4Re::Dataspace > L4Re::Vfs::Regular_file::data_space ( ) [pure virtual],  
[noexcept]
```

Get an [L4Re::Dataspace](#) object for the file.

This is used as a backend for POSIX mmap and mmap2 functions.

#### Note

mmap is not possible if the function returns an invalid capability.

#### Returns

A capability to an [L4Re::Dataspace](#) that represents the file contents in an [L4Re](#) way.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 15.331.2.2 fdatsync()

```
virtual int L4Re::Vfs::Regular_file::fdatsync ( ) const [pure virtual], [noexcept]
```

Sync the data to persistent storage.

This is the backend for POSIX fdatsync.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 15.331.2.3 fsync()

```
virtual int L4Re::Vfs::Regular_file::fsync ( ) const [pure virtual], [noexcept]
```

Sync the data and meta data to persistent storage.

This is the backend for POSIX fsync.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 15.331.2.4 ftruncate64()

```
virtual int L4Re::Vfs::Regular_file::ftruncate64 (  
    off64_t pos ) [pure virtual], [noexcept]
```

Truncate the file at the given position.

This function is the backend for truncate and friends.

**Parameters**

<i>pos</i>	The offset at which the file shall be truncated.
------------	--

**Returns**

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

**15.331.2.5 get\_lock()**

```
virtual int L4Re::Vfs::Regular_file::get_lock (
    struct flock64 * lock ) [pure virtual], [noexcept]
```

Test if the given lock can be placed in the file.

This function is used as backend for fcntl F\_GETLK commands.

**Parameters**

<i>lock</i>	The lock that shall be placed on the file. The <i>l_type</i> member will contain F_UNLCK if the lock could be placed.
-------------	---

**Returns**

0 on success, <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

**15.331.2.6 lseek64()**

```
virtual off64_t L4Re::Vfs::Regular_file::lseek64 (
    off64_t ,
    int ) [pure virtual], [noexcept]
```

Change the file pointer.

This is the backend for POSIX seek, lseek and friends.

**Returns**

The new file position, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 15.331.2.7 readv()

```
virtual ssize_t L4Re::Vfs::Regular_file::readv (
    const struct iovec * ,
    int iovcnt ) [pure virtual], [noexcept]
```

Read one or more blocks of data from the file.

This function acts as backend for POSIX read and readv calls and reads data starting from the `f_pos` pointer of that open file. The file pointer is advanced according to the number of bytes read.

#### Returns

The number of bytes read from the file, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 15.331.2.8 set\_lock()

```
virtual int L4Re::Vfs::Regular_file::set_lock (
    struct flock64 * lock,
    bool wait ) [pure virtual], [noexcept]
```

Acquire or release the given lock on the file.

This function is used as backend for `fcntl F_SETLK` and `F_SETLKW` commands.

#### Parameters

<i>lock</i>	The lock that shall be placed on the file.
<i>wait</i>	If true, then block if there is a conflicting lock on the file.

#### Returns

0 on success, <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 15.331.2.9 writev()

```
virtual ssize_t L4Re::Vfs::Regular_file::writev (
    const struct iovec * ,
    int iovcnt ) [pure virtual], [noexcept]
```

Write one or more blocks of data to the file.

This function acts as backend for POSIX write and writev calls. The data is written starting at the current file pointer and the file pointer must be advanced according to the number of written bytes.

**Returns**

The number of bytes written to the file, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

The documentation for this class was generated from the following file:

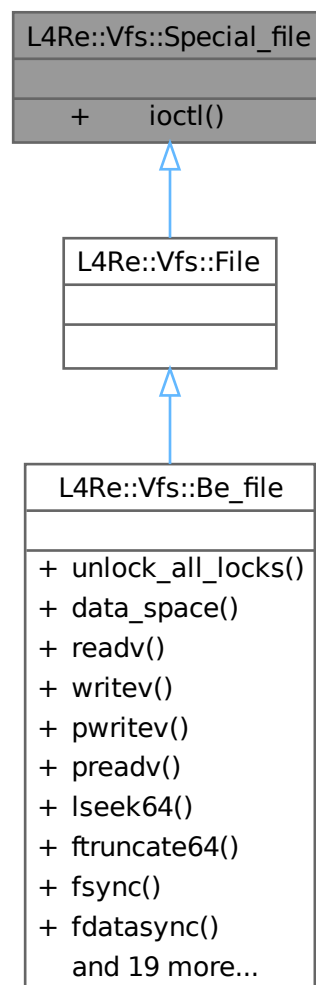
- l4/l4re\_vfs/vfs.h

## 15.332 L4Re::Vfs::Special\_file Class Reference

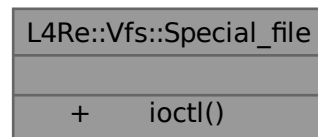
Interface for a POSIX file that provides special file semantics.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Special\_file:



Collaboration diagram for L4Re::Vfs::Special\_file:



### Public Member Functions

- virtual int [ioctl](#) (unsigned long cmd, va\_list args) noexcept=0  
*The famous IO control.*

## 15.332.1 Detailed Description

Interface for a POSIX file that provides special file semantics.

Real objects always use the combined [L4Re::Vfs::File](#) interface.

Definition at line [400](#) of file [vfs.h](#).

## 15.332.2 Member Function Documentation

### 15.332.2.1 ioctl()

```
virtual int L4Re::Vfs::Special_file::ioctl (
    unsigned long cmd,
    va_list args ) [pure virtual], [noexcept]
```

The famous IO control.

Backend for POSIX generic object invocation ioctl.

#### Parameters

<i>cmd</i>	The ioctl command.
<i>args</i>	The arguments for the ioctl, usually some kind of pointer.

#### Returns

>=0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

The documentation for this class was generated from the following file:

- l4/l4re\_vfs/vfs.h

### 15.333 L4Re::Video::Color\_component Class Reference

A color component.

```
#include <colors>
```

Collaboration diagram for L4Re::Video::Color\_component:

L4Re::Video::Color_component
<ul style="list-style-type: none"> <li>+ Color_component()</li> <li>+ Color_component()</li> <li>+ size()</li> <li>+ shift()</li> <li>+ operator==( )</li> <li>+ get()</li> <li>+ set()</li> <li>+ dump()</li> </ul>

#### Public Member Functions

- **Color\_component ( )**  
*Constructor.*
- **Color\_component** (unsigned char bits, unsigned char **shift**)  
*Constructor.*
- unsigned char **size** ( ) const  
*Return the number of bits used by the component.*
- unsigned char **shift** ( ) const  
*Return the position of the component in the pixel.*
- bool **operator==** (**Color\_component** const &o) const  
*Compare for equality.*
- int **get** (unsigned long v) const  
*Get component from value (normalized to 16bits).*
- long unsigned **set** (int v) const  
*Transform 16bit normalized value to the component in the color space.*
- template<typename OUT >  
void **dump** (OUT &s) const  
*Dump information on the view information to a stream.*



### 15.333.1 Detailed Description

A color component.

Definition at line 32 of file [colors](#).

### 15.333.2 Constructor & Destructor Documentation

#### 15.333.2.1 Color\_component()

```
L4Re::Video::Color_component::Color_component (
    unsigned char bits,
    unsigned char shift ) [inline]
```

Constructor.

##### Parameters

<i>bits</i>	Number of bits used by the component
<i>shift</i>	Position in bits of the component in the pixel

Definition at line 47 of file [colors](#).

### 15.333.3 Member Function Documentation

#### 15.333.3.1 dump()

```
template<typename OUT >
void L4Re::Video::Color_component::dump (
    OUT & s ) const [inline]
```

Dump information on the view information to a stream.

##### Parameters

<i>s</i>	Stream
----------	--------

Definition at line 92 of file [colors](#).

#### 15.333.3.2 get()

```
int L4Re::Video::Color_component::get (
    unsigned long v ) const [inline]
```

Get component from value (normalized to 16bits).

**Parameters**

<i>v</i>	Value
----------	-------

**Returns**

Converted value

Definition at line 74 of file [colors](#).

**15.333.3.3 operator==()**

```
bool L4Re::Video::Color_component::operator== (
    Color_component const & o ) const [inline]
```

Compare for equality.

**Returns**

True if the same components are described, false if not.

Definition at line 66 of file [colors](#).

**15.333.3.4 set()**

```
long unsigned L4Re::Video::Color_component::set (
    int v ) const [inline]
```

Transform 16bit normalized value to the component in the color space.

**Parameters**

<i>v</i>	Value return Converted value.
----------	-------------------------------

Definition at line 84 of file [colors](#).

**15.333.3.5 shift()**

```
unsigned char L4Re::Video::Color_component::shift ( ) const [inline]
```

Return the position of the component in the pixel.

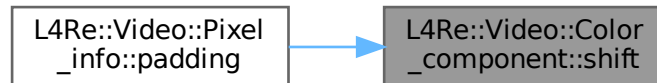
**Returns**

Position in bits of the component in the pixel

Definition at line 60 of file [colors](#).

Referenced by [L4Re::Video::Pixel\\_info::padding\(\)](#).

Here is the caller graph for this function:

**15.333.3.6 size()**

```
unsigned char L4Re::Video::Color_component::size ( ) const [inline]
```

Return the number of bits used by the component.

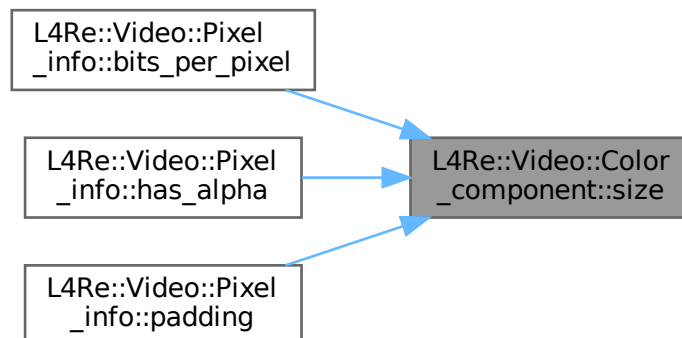
**Returns**

Number of bits used by the component

Definition at line 54 of file [colors](#).

Referenced by [L4Re::Video::Pixel\\_info::bits\\_per\\_pixel\(\)](#), [L4Re::Video::Pixel\\_info::has\\_alpha\(\)](#), and [L4Re::Video::Pixel\\_info::padding\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

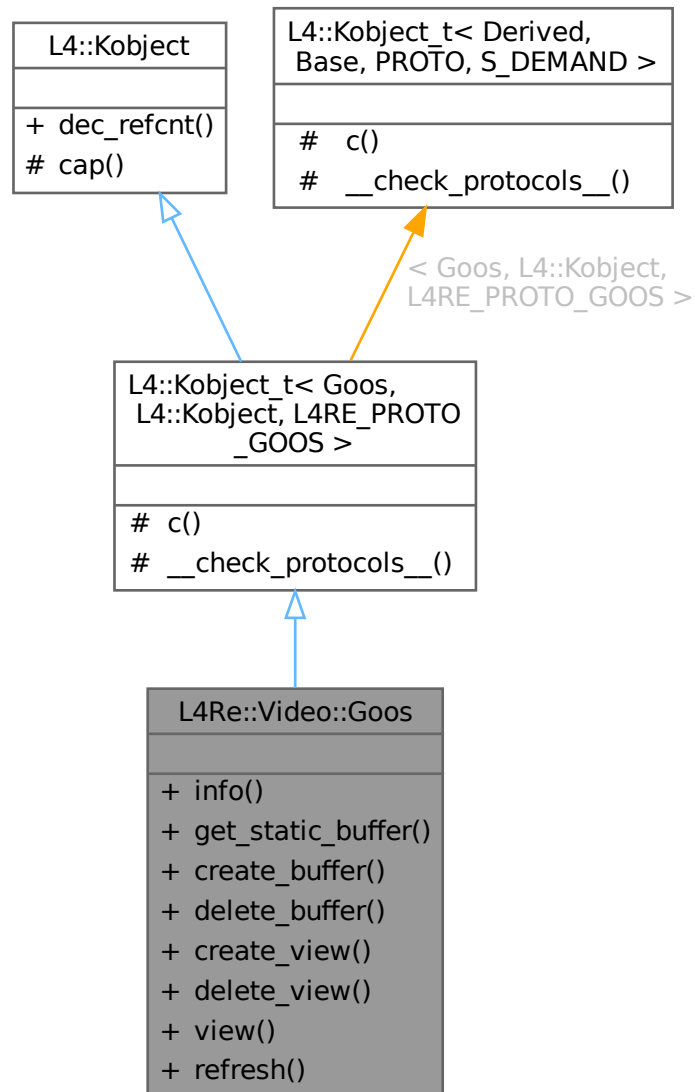
- `I4/re/video/colors`

## 15.334 L4Re::Video::Goos Class Reference

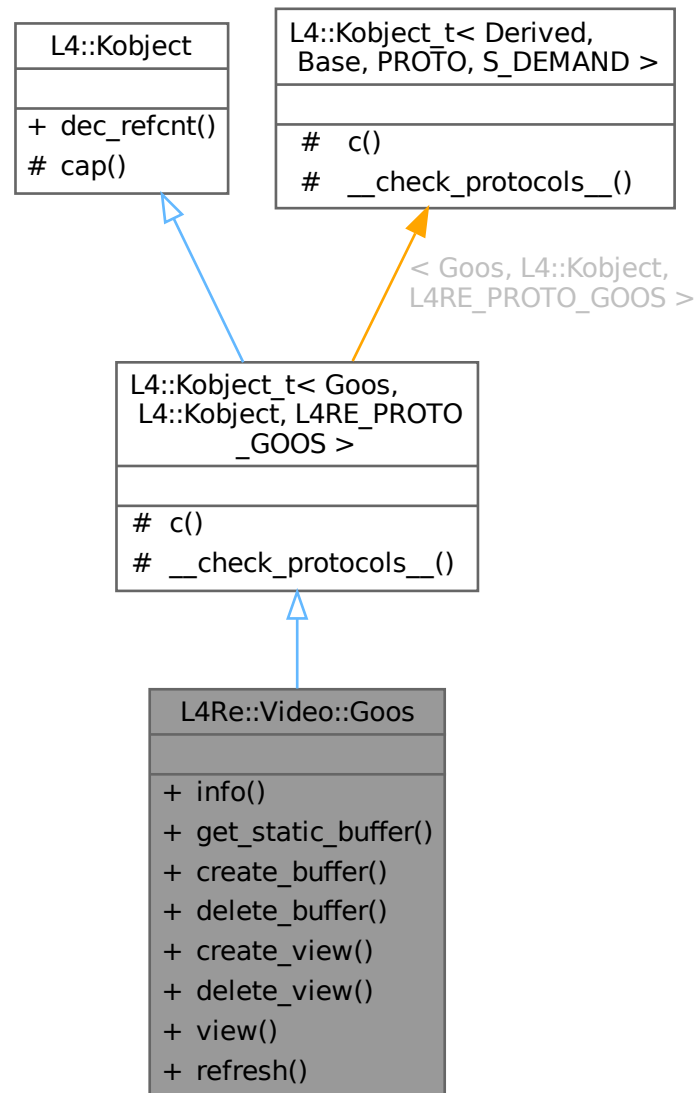
Class that abstracts framebuffers.

```
#include <goos>
```

Inheritance diagram for L4Re::Video::Goos:



Collaboration diagram for L4Re::Video::Goos:



## Data Structures

- struct [Info](#)  
*Information structure of a [Goos](#).*

## Public Types

- enum [Flags](#) { [F\\_auto\\_refresh](#) = 0x01 , [F\\_pointer](#) = 0x02 , [F\\_dynamic\\_views](#) = 0x04 , [F\\_dynamic\\_buffers](#) = 0x08 }
- Flags for a [Goos](#).*

## Public Member Functions

- long **info** (Info \*info)  
*Return the Goos information of the Goos.*
- long **get\_static\_buffer** (unsigned idx, L4::lpc::Out< L4::Cap< L4Re::Dataspace > > rbuf)  
*Return a static buffer of a Goos.*
- long **create\_buffer** (unsigned long size, L4::lpc::Out< L4::Cap< L4Re::Dataspace > > rbuf)  
*Create a buffer.*
- long **delete\_buffer** (unsigned idx)  
*Delete a buffer.*
- int **create\_view** (View \*view, l4\_utcb\_t \*utcb=l4\_utcb()) const noexcept  
*Create a view.*
- int **delete\_view** (View const &v, l4\_utcb\_t \*utcb=l4\_utcb()) const noexcept  
*Delete a view.*
- View **view** (unsigned index) const noexcept  
*Return a view.*
- long **refresh** (int x, int y, int w, int h)  
*Trigger refreshing of the given area on the virtual screen.*

## Public Member Functions inherited from L4::Kobject

- l4\_msgtag\_t **dec\_refcnt** (l4\_mword\_t diff, l4\_utcb\_t \*utcb=l4\_utcb())  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

## Protected Types inherited from

**L4::Kobject\_t< Goos, L4::Kobject, L4RE\_PROTO\_GOOS >**

- typedef Goos **Class**  
*The target interface type (inheriting from Kobject\_t)*
- typedef Typeid::Iface< PROTO, Goos > **\_\_iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< \_\_iface >, typename Base::\_\_iface\_list > **\_\_iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from

**L4::Kobject\_t< Goos, L4::Kobject, L4RE\_PROTO\_GOOS >**

- L4::Cap< Class > **c** () const noexcept  
*Get the capability to ourselves.*

## Protected Member Functions inherited from L4::Kobject

- l4\_cap\_idx\_t **cap** () const noexcept  
*Return capability selector.*

## Static Protected Member Functions inherited from [L4::Kobject\\_t< Goos, L4::Kobject, L4RE\\_PROTO\\_GOOS >](#)

- static void `__check_protocols__()` noexcept  
*Helper to check for protocol conflicts.*

### 15.334.1 Detailed Description

Class that abstracts framebuffers.

A framebuffer is the pixel data that is displayed on a screen and a [Goos](#) object lets the user manipulate that data. A [Goos](#) makes use of two kinds of objects:

- Buffers in the form of [L4Re::Dataspace](#) objects. These hold the bytes for the pixel data.
- [L4Re::Video::View](#) objects.

Both can either be static, that is their number and configuration is fixed and determined by the framebuffer, or they can be dynamic, with the user allocating them.

Definition at line 226 of file [goos](#).

### 15.334.2 Member Enumeration Documentation

#### 15.334.2.1 Flags

```
enum L4Re::Video::Goos::Flags
```

Flags for a [Goos](#).

Enumerator

<code>F_auto_refresh</code>	The graphics display is automatically refreshed.
<code>F_pointer</code>	We have a mouse pointer.
<code>F_dynamic_views</code>	Supports dynamically allocated views.
<code>F_dynamic_buffers</code>	Supports dynamically allocated buffers.

Definition at line 231 of file [goos](#).

### 15.334.3 Member Function Documentation

#### 15.334.3.1 `create_buffer()`

```
long L4Re::Video::Goos::create\_buffer (
    unsigned long size,
    L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > rbuf )
```

Create a buffer.

**Parameters**

<i>size</i>	Size of buffer in bytes.
<i>rbuf</i>	Capability slot to point the buffer dataspace to.

**Return values**

$\geq 0$	Success, the value returned is the buffer index.
$< 0$	Error

**15.334.3.2 create\_view()**

```
int L4Re::Video::Goos::create_view (
    View * view,
    l4_utcb_t * utcb = l4_utcb() ) const [inline], [noexcept]
```

Create a view.

**Parameters**

out	<i>view</i>	A view object.
	<i>utcb</i>	UTCB of the caller. This is a default parameter.

**Return values**

$\geq 0$	Success, the value returned is the view index.
$< 0$	Error

Definition at line 315 of file [goos](#).

**15.334.3.3 delete\_buffer()**

```
long L4Re::Video::Goos::delete_buffer (
    unsigned idx )
```

Delete a buffer.

**Parameters**

<i>idx</i>	Buffer to delete.
------------	-------------------

**Return values**

0	Success
$< 0$	Error



#### 15.334.3.4 delete\_view()

```
int L4Re::Video::Goos::delete_view (
    View const & v,
    l4_utcb_t * utcb = l4_utcb() ) const [inline], [noexcept]
```

Delete a view.

##### Parameters

<i>v</i>	The view object to delete.
<i>utcb</i>	UTCB of the caller. This is a default parameter.

##### Return values

0	Success
<0	Error

Definition at line 335 of file [goos](#).

#### 15.334.3.5 get\_static\_buffer()

```
long L4Re::Video::Goos::get_static_buffer (
    unsigned idx,
    L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > rbuf )
```

Return a static buffer of a [Goos](#).

##### Parameters

<i>idx</i>	Index of the static buffer.
<i>rbuf</i>	Capability slot to point the buffer dataspace to.

##### Return values

0	Success
<0	Error

#### 15.334.3.6 info()

```
long L4Re::Video::Goos::info (
    Info * info )
```

Return the [Goos](#) information of the [Goos](#).

##### Parameters

out	<i>info</i>	<a href="#">Goos</a> information structure pointer.
-----	-------------	---

## Return values

0	Success
<0	Error

**15.334.3.7 view()**

```
View L4Re::Video::Goos::view (  
    unsigned index ) const [inline], [noexcept]
```

Return a view.

## Parameters

<i>index</i>	Index of the view to return.
--------------	------------------------------

## Returns

The view.

Definition at line 366 of file [goos](#).

The documentation for this class was generated from the following file:

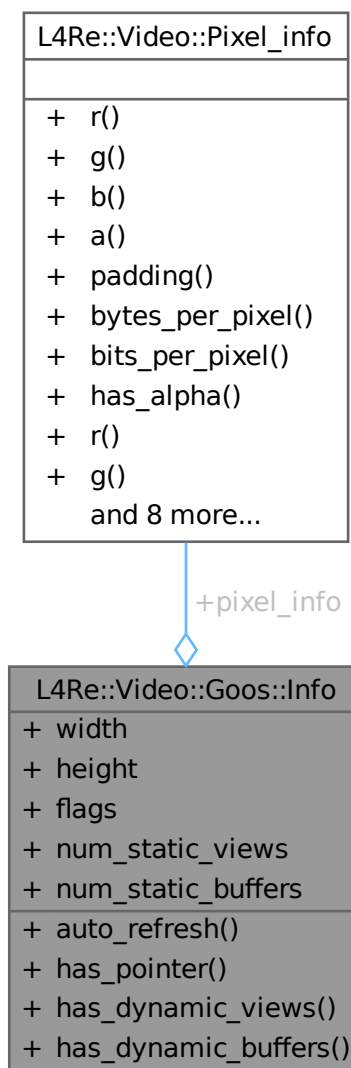
- [l4/re/video/goos](#)

**15.335 L4Re::Video::Goos::Info Struct Reference**

Information structure of a [Goos](#).

```
#include <goos>
```

Collaboration diagram for L4Re::Video::Goos::Info:



## Public Member Functions

- bool **auto\_refresh** () const  
*Return whether this [Goos](#) does auto refreshing or the view refresh functions must be used to make changes visible.*
- bool **has\_pointer** () const  
*Return whether a pointer is used by the provider of the [Goos](#).*
- bool **has\_dynamic\_views** () const  
*Return whether dynamic view are supported.*
- bool **has\_dynamic\_buffers** () const  
*Return whether dynamic buffers are supported.*

## Data Fields

- unsigned long **width**  
*Width.*
- unsigned long **height**  
*Height.*
- unsigned **flags**  
*Flags, see [Flags](#).*
- unsigned **num\_static\_views**  
*Number of static view.*
- unsigned **num\_static\_buffers**  
*Number of static buffers.*
- [Pixel\\_info](#) **pixel\_info**  
*Pixel information.*

### 15.335.1 Detailed Description

Information structure of a [Goos](#).

Definition at line [240](#) of file [goos](#).

The documentation for this struct was generated from the following file:

- [l4/re/video/goos](#)

### 15.336 L4Re::Video::Pixel\_info Class Reference

Pixel information.

```
#include <colors>
```

Collaboration diagram for L4Re::Video::Pixel\_info:

L4Re::Video::Pixel_info
<ul style="list-style-type: none"> <li>+ r()</li> <li>+ g()</li> <li>+ b()</li> <li>+ a()</li> <li>+ padding()</li> <li>+ bytes_per_pixel()</li> <li>+ bits_per_pixel()</li> <li>+ has_alpha()</li> <li>+ r()</li> <li>+ g()</li> <li>and 8 more...</li> </ul>

## Public Member Functions

- [Color\\_component](#) const & [r](#) () const  
*Return the red color compoment of the pixel.*
- [Color\\_component](#) const & [g](#) () const  
*Return the green color compoment of the pixel.*
- [Color\\_component](#) const & [b](#) () const  
*Return the blue color compoment of the pixel.*
- [Color\\_component](#) const & [a](#) () const  
*Return the alpha color compoment of the pixel.*
- [Color\\_component](#) const [padding](#) () const  
*Compute the padding pseudo component.*
- unsigned char [bytes\\_per\\_pixel](#) () const  
*Query size of pixel in bytes.*
- unsigned char [bits\\_per\\_pixel](#) () const  
*Number of bits of the pixel.*
- bool [has\\_alpha](#) () const  
*Return whether the pixel has an alpha channel.*
- void [r](#) ([Color\\_component](#) const &c)  
*Set the red color component of the pixel.*
- void [g](#) ([Color\\_component](#) const &c)  
*Set the green color component of the pixel.*
- void [b](#) ([Color\\_component](#) const &c)  
*Set the blue color component of the pixel.*
- void [a](#) ([Color\\_component](#) const &c)  
*Set the alpha color component of the pixel.*
- void [bytes\\_per\\_pixel](#) (unsigned char bpp)  
*Set the size of the pixel in bytes.*
- **Pixel\_info** ()  
*Constructor.*
- [Pixel\\_info](#) (unsigned char bpp, char [r](#), char [rs](#), char [g](#), char [gs](#), char [b](#), char [bs](#), char [a](#)=0, char [as](#)=0)  
*Constructor.*
- template<typename VBI >  
[Pixel\\_info](#) (VBI const \*vbi)  
*Convenience constructor.*
- bool [operator==](#) ([Pixel\\_info](#) const &o) const  
*Compare for complete equality of the color space.*
- template<typename OUT >  
void [dump](#) (OUT &s) const  
*Dump information on the pixel to a stream.*

### 15.336.1 Detailed Description

Pixel information.

This class wraps the information on a pixel, such as the size and position of each color component in the pixel.

Definition at line 105 of file [colors](#).

## 15.336.2 Constructor & Destructor Documentation

### 15.336.2.1 Pixel\_info() [1/2]

```
L4Re::Video::Pixel_info::Pixel_info (
    unsigned char bpp,
    char r,
    char rs,
    char g,
    char gs,
    char b,
    char bs,
    char a = 0,
    char as = 0 ) [inline]
```

Constructor.

#### Parameters

<i>bpp</i>	Size of pixel in bytes.
<i>r</i>	Red component size.
<i>rs</i>	Red component shift.
<i>g</i>	Green component size.
<i>gs</i>	Green component shift.
<i>b</i>	Blue component size.
<i>bs</i>	Blue component shift.
<i>a</i>	Alpha component size, defaults to 0.
<i>as</i>	Alpha component shift, defaults to 0.

Definition at line [223](#) of file [colors](#).

### 15.336.2.2 Pixel\_info() [2/2]

```
template<typename VBI >
L4Re::Video::Pixel_info::Pixel_info (
    VBI const * vbi ) [inline], [explicit]
```

Convenience constructor.

#### Parameters

<i>vbi</i>	Suitable information structure. Convenience constructor to create the pixel info from a VESA Framebuffer Info.
------------	--

Definition at line [235](#) of file [colors](#).

## 15.336.3 Member Function Documentation

### 15.336.3.1 a() [1/2]

```
Color_component const & L4Re::Video::Pixel_info::a ( ) const [inline]
```

Return the alpha color component of the pixel.

#### Returns

Alpha color component.

Definition at line 134 of file [colors](#).

### 15.336.3.2 a() [2/2]

```
void L4Re::Video::Pixel_info::a (  
    Color_component const & c ) [inline]
```

Set the alpha color component of the pixel.

#### Parameters

c	Alpha color component.
---	------------------------

Definition at line 198 of file [colors](#).

### 15.336.3.3 b() [1/2]

```
Color_component const & L4Re::Video::Pixel_info::b ( ) const [inline]
```

Return the blue color component of the pixel.

#### Returns

Blue color component.

Definition at line 128 of file [colors](#).

### 15.336.3.4 b() [2/2]

```
void L4Re::Video::Pixel_info::b (  
    Color_component const & c ) [inline]
```

Set the blue color component of the pixel.

#### Parameters

c	Blue color component.
---	-----------------------

Definition at line 192 of file [colors](#).

**15.336.3.5 bits\_per\_pixel()**

```
unsigned char L4Re::Video::Pixel_info::bits_per_pixel ( ) const [inline]
```

Number of bits of the pixel.

**Returns**

Number of bits used by the pixel.

Definition at line 167 of file [colors](#).

References [L4Re::Video::Color\\_component::size\(\)](#).

Here is the call graph for this function:

**15.336.3.6 bytes\_per\_pixel() [1/2]**

```
unsigned char L4Re::Video::Pixel_info::bytes_per_pixel ( ) const [inline]
```

Query size of pixel in bytes.

**Returns**

Size of pixel in bytes.

Definition at line 161 of file [colors](#).

**15.336.3.7 bytes\_per\_pixel() [2/2]**

```
void L4Re::Video::Pixel_info::bytes_per_pixel (
    unsigned char bpp ) [inline]
```

Set the size of the pixel in bytes.

**Parameters**

<i>bpp</i>	Size of pixel in bytes.
------------	-------------------------



Definition at line 204 of file [colors](#).

### 15.336.3.8 dump()

```
template<typename OUT >
void L4Re::Video::Pixel_info::dump (
    OUT & s ) const [inline]
```

Dump information on the pixel to a stream.

#### Parameters

<b>s</b>	Stream
----------	--------

Definition at line 257 of file [colors](#).

Referenced by [L4Re::Video::View::Info::dump\(\)](#).

Here is the caller graph for this function:



### 15.336.3.9 g() [1/2]

```
Color_component const & L4Re::Video::Pixel_info::g ( ) const [inline]
```

Return the green color component of the pixel.

#### Returns

Green color component.

Definition at line 122 of file [colors](#).

### 15.336.3.10 g() [2/2]

```
void L4Re::Video::Pixel_info::g (
    Color_component const & c ) [inline]
```

Set the green color component of the pixel.

## Parameters

<code>c</code>	Green color component.
----------------	------------------------

Definition at line 186 of file [colors](#).

**15.336.3.11 has\_alpha()**

```
bool L4Re::Video::Pixel_info::has_alpha ( ) const [inline]
```

Return whether the pixel has an alpha channel.

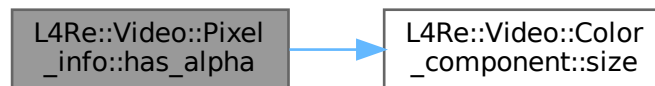
## Returns

True if the pixel has an alpha channel, false if not.

Definition at line 174 of file [colors](#).

References [L4Re::Video::Color\\_component::size\(\)](#).

Here is the call graph for this function:

**15.336.3.12 operator==()**

```
bool L4Re::Video::Pixel_info::operator== (
    Pixel_info const & o ) const [inline]
```

Compare for complete equality of the color space.

## Parameters

<code>o</code>	A <a href="#">Pixel_info</a> to compare to.
----------------	---

## Returns

true if the both [Pixel\\_info](#)'s are equal, false if not.

Definition at line 247 of file [colors](#).

**15.336.3.13 padding()**

```
Color_component const L4Re::Video::Pixel_info::padding ( ) const [inline]
```

Compute the padding pseudo component.

The padding pseudo component represents the tailing bits that are reserved in RGB32 and similar pixel formats.

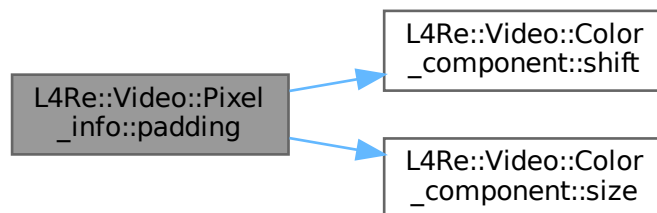
**Returns**

Padding pseudo component.

Definition at line 142 of file [colors](#).

References [L4Re::Video::Color\\_component::shift\(\)](#), and [L4Re::Video::Color\\_component::size\(\)](#).

Here is the call graph for this function:

**15.336.3.14 r() [1/2]**

```
Color_component const & L4Re::Video::Pixel_info::r ( ) const [inline]
```

Return the red color component of the pixel.

**Returns**

Red color component.

Definition at line 116 of file [colors](#).

**15.336.3.15 r() [2/2]**

```
void L4Re::Video::Pixel_info::r (
    Color_component const & c ) [inline]
```

Set the red color component of the pixel.

## Parameters

c	Red color component.
---	----------------------

Definition at line 180 of file [colors](#).

The documentation for this class was generated from the following file:

- [l4/re/video/colors](#)

## 15.337 L4Re::Video::View Class Reference

[View](#) of a framebuffer.

```
#include <goos>
```

Collaboration diagram for L4Re::Video::View:

L4Re::Video::View
<ul style="list-style-type: none"> <li>+ info()</li> <li>+ set_info()</li> <li>+ set_viewport()</li> <li>+ stack()</li> <li>+ push_top()</li> <li>+ push_bottom()</li> <li>+ refresh()</li> <li>+ valid()</li> </ul>

### Data Structures

- struct [Info](#)  
*Information structure of a view.*

### Public Types

- enum [Flags](#) {  
[F\\_none](#) = 0x00 , [F\\_set\\_buffer](#) = 0x01 , [F\\_set\\_buffer\\_offset](#) = 0x02 , [F\\_set\\_bytes\\_per\\_line](#) = 0x04 ,  
[F\\_set\\_pixel](#) = 0x08 , [F\\_set\\_position](#) = 0x10 , [F\\_dyn\\_allocated](#) = 0x20 , [F\\_set\\_background](#) = 0x40 ,  
[F\\_set\\_flags](#) = 0x80 , [F\\_fully\\_dynamic](#) }  
*Flags on a view.*
- enum [V\\_flags](#) { [F\\_above](#) = 0x1000 , [F\\_flags\\_mask](#) = 0xff000 }  
*Property flags of a view.*

## Public Member Functions

- int [info](#) ([Info](#) \*info) const noexcept  
*Return the view information of the view.*
- int [set\\_info](#) ([Info](#) const &info) const noexcept  
*Set the information structure for this view.*
- int [set\\_viewport](#) (int scr\_x, int scr\_y, int w, int h, unsigned long buf\_offset) const noexcept  
*Set the position of the view in the [Goos](#).*
- int [stack](#) ([View](#) const &pivot, bool behind=true) const noexcept  
*Move this view in the view stack.*
- int [push\\_top](#) () const noexcept  
*Make this view the top-most view.*
- int [push\\_bottom](#) () const noexcept  
*Push this view the back.*
- int [refresh](#) (int x, int y, int w, int h) const noexcept  
*Refresh/Redraw the view.*
- bool [valid](#) () const  
*Return whether this view is valid.*

## 15.337.1 Detailed Description

[View](#) of a framebuffer.

A view is a rectangular subset of a framebuffer managed by a [Goos](#) object. The [Goos](#) orders multiple views in a stack which determines which view is on top in case they overlap. The view's pixel data is provided by a backing buffer, which must belong to the [Goos](#). It can be static or dynamically allocated, depending on the framebuffer.

See also

[L4Re::Video::Goos](#)

Definition at line 42 of file [goos](#).

## 15.337.2 Member Enumeration Documentation

### 15.337.2.1 Flags

```
enum L4Re::Video::View::Flags
```

Flags on a view.

Enumerator

F_none	everything for this view is static (the VESA-FB case)
F_set_buffer	buffer object for this view can be changed
F_set_buffer_offset	buffer offset can be set
F_set_bytes_per_line	bytes per line can be set
F_set_pixel	pixel type can be set
F_set_position	position on screen can be set
F_dyn_allocated	<a href="#">View</a> is dynamically allocated.
F_set_background	Set view as background for session.
F_set_flags	Set view flags (.  See also

Definition at line 62 of file [goos](#).

### 15.337.2.2 V\_flags

```
enum L4Re::Video::View::V_flags
```

Property flags of a view.

Such flags can be set or deleted with the [F\\_set\\_flags](#) operation using the [set\\_info\(\)](#) method.

#### Enumerator

F_above	Flag the view as stay on top.
F_flags_mask	Mask containing all possible property flags.

Definition at line 85 of file [goos](#).

## 15.337.3 Member Function Documentation

### 15.337.3.1 info()

```
int L4Re::Video::View::info (  
    Info * info ) const [inline], [noexcept]
```

Return the view information of the view.

#### Parameters

out	<i>info</i>	Information structure pointer.
-----	-------------	--------------------------------

#### Return values

0	Success
<0	Error

Definition at line 370 of file [goos](#).

### 15.337.3.2 refresh()

```
int L4Re::Video::View::refresh (  
    int x,  
    int y,  
    int w,  
    int h ) const [inline], [noexcept]
```

Refresh/Redraw the view.

## Parameters

<i>x</i>	X position.
<i>y</i>	Y position.
<i>w</i>	Width.
<i>h</i>	Height.

## Return values

0	Success
<0	Error

Definition at line 382 of file [goos](#).

**15.337.3.3 set\_info()**

```
int L4Re::Video::View::set_info (
    Info const & info ) const [inline], [noexcept]
```

Set the information structure for this view.

## Parameters

<i>info</i>	Information structure.
-------------	------------------------

## Return values

0	Success
<0	Error

The function will also set the view port according to the values given in the information structure.

Definition at line 374 of file [goos](#).

**15.337.3.4 set\_viewport()**

```
int L4Re::Video::View::set_viewport (
    int scr_x,
    int scr_y,
    int w,
    int h,
    unsigned long buf_offset ) const [inline], [noexcept]
```

Set the position of the view in the [Goos](#).

## Parameters

<i>scr_x</i>	X position
<i>scr_y</i>	Y position
<i>w</i>	Width
<i>h</i>	Height
<i>buf_offset</i>	Offset in the buffer in bytes

## Return values

0	Success
<0	Error

Definition at line 386 of file [goos](#).

References [L4Re::Video::View::Info::buffer\\_index](#), [L4Re::Video::View::Info::buffer\\_offset](#), [L4Re::Video::View::Info::bytes\\_per\\_line](#), [L4Re::Video::View::Info::flags](#), [L4Re::Video::View::Info::height](#), [L4Re::Video::View::Info::pixel\\_info](#), [L4Re::Video::View::Info::view\\_index](#), [L4Re::Video::View::Info::width](#), [L4Re::Video::View::Info::xpos](#), and [L4Re::Video::View::Info::ypos](#).

**15.337.3.5 stack()**

```
int L4Re::Video::View::stack (
    View const & pivot,
    bool behind = true ) const [inline], [noexcept]
```

Move this view in the view stack.

## Parameters

<i>pivot</i>	<a href="#">View</a> to move relative to
<i>behind</i>	When true move the view behind the pivot view, if false move the view before the pivot view.

## Return values

0	Success
<0	Error

Definition at line 378 of file [goos](#).

The documentation for this class was generated from the following file:

- [l4/re/video/goos](#)

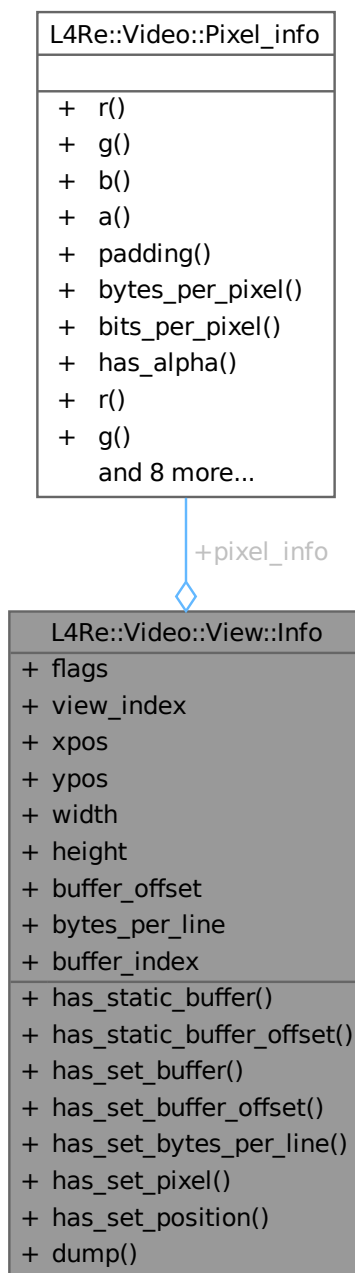
**15.338 L4Re::Video::View::Info Struct Reference**

Information structure of a view.

```
#include <goos>
```



Collaboration diagram for L4Re::Video::View::Info:



## Public Member Functions

- bool **has\_static\_buffer** () const  
*Return whether the view has a static buffer.*
- bool **has\_static\_buffer\_offset** () const  
*Return whether the static buffer offset is available.*
- bool **has\_set\_buffer** () const

- Return whether a buffer is set.*
  - bool **has\_set\_buffer\_offset** () const
  - Return whether the given buffer offset is valid.*
    - bool **has\_set\_bytes\_per\_line** () const
    - Return whether the given bytes-per-line value is valid.*
      - bool **has\_set\_pixel** () const
      - Return whether the given pixel information is valid.*
        - bool **has\_set\_position** () const
        - Return whether the position information given is valid.*
          - template<typename OUT >
          - void **dump** (OUT &s) const
          - Dump information on the view information to a stream.*

## Data Fields

- unsigned **flags** = 0
- Flags, see [Flags](#) and [V\\_flags](#).*
- unsigned **view\_index** = 0
- Index of the view.*
- unsigned long **xpos** = 0
- X position in pixels of the view in the [Goos](#).*
- unsigned long **ypos** = 0
- Y position in pixels of the view in the [Goos](#).*
- unsigned long **width** = 0
- Width of the view in pixels.*
- unsigned long **height** = 0
- Height of the view in pixels.*
- unsigned long **buffer\_offset** = 0
- Offset in the memory buffer in bytes.*
- unsigned long **bytes\_per\_line** = 0
- Bytes per line.*
- [Pixel\\_info](#) **pixel\_info**
- Pixel information.*
- unsigned **buffer\_index** = 0
- Number of the buffer used for this view.*

### 15.338.1 Detailed Description

Information structure of a view.

Definition at line 94 of file [goos](#).

The documentation for this struct was generated from the following file:

- l4/re/video/goos

## 15.339 l4re\_aux\_t Struct Reference

Auxiliary descriptor.

```
#include <l4aux.h>
```

Collaboration diagram for l4re\_aux\_t:

l4re_aux_t
+ binary
+ kip_ds
+ dbg_lvl
+ ldr_flags
+ ldr_base

### Data Fields

- char const \* **binary**  
*Binary name.*
- [l4\\_cap\\_idx\\_t](#) **kip\_ds**  
*Data space of the KIP.*
- [l4\\_umword\\_t](#) **dbg\_lvl**  
*Debug levels for l4re.*
- [l4\\_umword\\_t](#) **ldr\_flags**  
*Flags for l4re, see l4re\_aux\_ldr\_flags\_t.*
- [l4\\_addr\\_t](#) **ldr\_base**  
*Load offset of executable.*

### 15.339.1 Detailed Description

Auxiliary descriptor.

Definition at line 51 of file [l4aux.h](#).

The documentation for this struct was generated from the following file:

- [l4/re/l4aux.h](#)

## 15.340 l4re\_ds\_stats\_t Struct Reference

Information about the data space.

```
#include <dataspace.h>
```

Collaboration diagram for l4re\_ds\_stats\_t:

l4re_ds_stats_t	
+	size
+	flags

### Data Fields

- l4re\_ds\_size\_t **size**  
*size*
- l4re\_ds\_flags\_t **flags**  
*flags*

### 15.340.1 Detailed Description

Information about the data space.

Definition at line 49 of file [dataspace.h](#).

The documentation for this struct was generated from the following file:

- [l4re/c/dataspace.h](#)

## 15.341 l4re\_elf\_aux\_mword\_t Struct Reference

Auxiliary vector element for a single unsigned data word.

```
#include <elf_aux.h>
```

Collaboration diagram for l4re\_elf\_aux\_mword\_t:

l4re_elf_aux_mword_t	

### 15.341.1 Detailed Description

Auxiliary vector element for a single unsigned data word.

Definition at line 129 of file [elf\\_aux.h](#).

The documentation for this struct was generated from the following file:

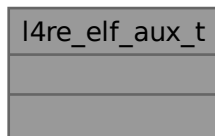
- [l4re/elf\\_aux.h](#)

## 15.342 l4re\_elf\_aux\_t Struct Reference

Generic header for each auxiliary vector element.

```
#include <elf_aux.h>
```

Collaboration diagram for l4re\_elf\_aux\_t:



### 15.342.1 Detailed Description

Generic header for each auxiliary vector element.

Definition at line 109 of file [elf\\_aux.h](#).

The documentation for this struct was generated from the following file:

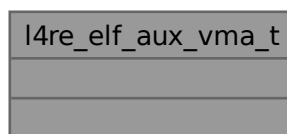
- [l4re/elf\\_aux.h](#)

## 15.343 l4re\_elf\_aux\_vma\_t Struct Reference

Auxiliary vector element for a reserved virtual memory area.

```
#include <elf_aux.h>
```

Collaboration diagram for l4re\_elf\_aux\_vma\_t:



### 15.343.1 Detailed Description

Auxiliary vector element for a reserved virtual memory area.

Definition at line 118 of file [elf\\_aux.h](#).

The documentation for this struct was generated from the following file:

- [l4/re/elf\\_aux.h](#)

## 15.344 l4re\_env\_cap\_entry\_t Struct Reference

Entry in the [L4Re](#) environment array for the named initial objects.

```
#include <env.h>
```

Collaboration diagram for `l4re_env_cap_entry_t`:

<code>l4re_env_cap_entry_t</code>
+ <code>cap</code>
+ <code>flags</code>
+ <code>name</code>
+ <code>l4re_env_cap_entry_t()</code>
+ <code>l4re_env_cap_entry_t()</code>

### Public Member Functions

- `l4re_env_cap_entry_t()` [L4\\_NOTHROW](#)  
*Create an invalid entry.*
- `l4re_env_cap_entry_t(char const *n, l4_cap_idx_t c, l4_umword_t f=0)` [L4\\_NOTHROW](#)  
*Create an entry with the name *n*, capability *c*, and flags *f*.*

### Data Fields

- `l4_cap_idx_t cap`  
*The capability selector for the object.*
- `l4_umword_t flags`  
*Some flags for the object.*
- `char name[16]`  
*The name of the object.*

### 15.344.1 Detailed Description

Entry in the [L4Re](#) environment array for the named initial objects.

Definition at line [50](#) of file [env.h](#).

### 15.344.2 Constructor & Destructor Documentation

#### 15.344.2.1 l4re\_env\_cap\_entry\_t()

```
l4re_env_cap_entry_t::l4re_env_cap_entry_t (
    char const * n,
    l4_cap_idx_t c,
    l4_umword_t f = 0 ) [inline]
```

Create an entry with the name *n*, capability *c*, and flags *f*.

##### Parameters

<i>n</i>	is the name of the initial object.
<i>c</i>	is the capability selector that refers the initial object.
<i>f</i>	are the additional flags for the object.

Definition at line [81](#) of file [env.h](#).

References [name](#).

### 15.344.3 Field Documentation

#### 15.344.3.1 flags

```
l4_umword_t l4re_env_cap_entry_t::flags
```

Some flags for the object.

##### Note

Currently unused.

Definition at line [61](#) of file [env.h](#).

Referenced by [l4re\\_env\\_get\\_cap\\_l\(\)](#).

The documentation for this struct was generated from the following file:

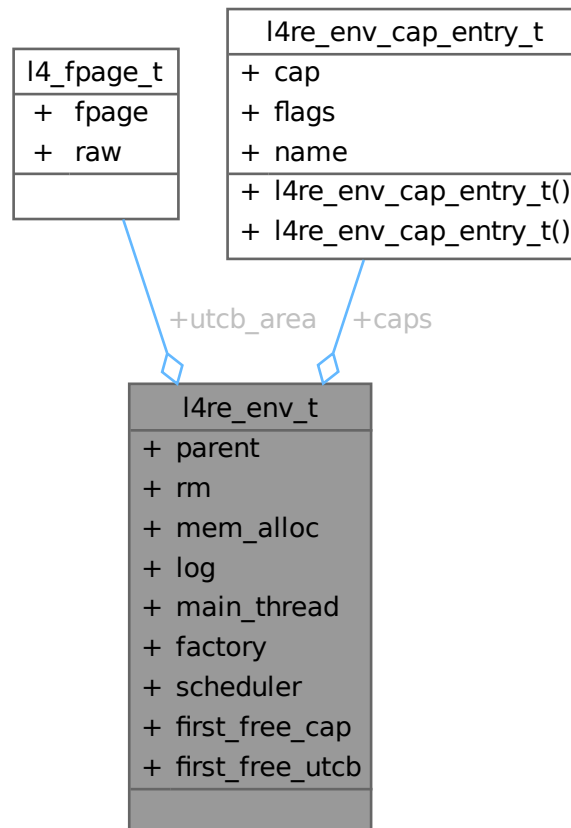
- [l4/re/env.h](#)

## 15.345 l4re\_env\_t Struct Reference

Initial environment data structure.

```
#include <env.h>
```

Collaboration diagram for l4re\_env\_t:



### Data Fields

- [l4\\_cap\\_idx\\_t](#) **parent**  
*Parent object-capability.*
- [l4\\_cap\\_idx\\_t](#) **rm**  
*Region map object-capability.*
- [l4\\_cap\\_idx\\_t](#) **mem\_alloc**  
*Memory allocator object-capability.*
- [l4\\_cap\\_idx\\_t](#) **log**  
*Logging object-capability.*
- [l4\\_cap\\_idx\\_t](#) **main\_thread**  
*Object-capability of the first user thread.*



- [l4\\_cap\\_idx\\_t factory](#)  
*Object-capability of the factory available to the task.*
- [l4\\_cap\\_idx\\_t scheduler](#)  
*Object capability for the scheduler set to use.*
- [l4\\_cap\\_idx\\_t first\\_free\\_cap](#)  
*First capability index available to the application.*
- [l4\\_fpage\\_t utcb\\_area](#)  
*UTCB area of the task.*
- [l4\\_addr\\_t first\\_free\\_utcb](#)  
*First UTCB within the UTCB area available to the application.*
- [l4re\\_env\\_cap\\_entry\\_t \\* caps](#)  
*Pointer to the first entry in the initial objects array which contains [l4re\\_env\\_cap\\_entry\\_t](#) elements.*

## 15.345.1 Detailed Description

Initial environment data structure.

See also

[Initial environment](#)

Definition at line 109 of file [env.h](#).

## 15.345.2 Field Documentation

### 15.345.2.1 caps

```
l4re\_env\_cap\_entry\_t\* l4re\_env\_t::caps
```

Pointer to the first entry in the initial objects array which contains [l4re\\_env\\_cap\\_entry\\_t](#) elements.

The array is terminated by an invalid entry with a `flags` value of `~0`.

Definition at line 126 of file [env.h](#).

Referenced by [L4Re::Env::initial\\_caps\(\)](#), and [L4Re::Env::initial\\_caps\(\)](#).

The documentation for this struct was generated from the following file:

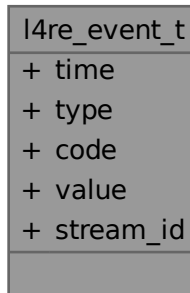
- [l4/re/env.h](#)

## 15.346 l4re\_event\_t Struct Reference

Event structure used in buffer.

```
#include <event.h>
```

Collaboration diagram for l4re\_event\_t:



### Data Fields

- long long **time**  
*Time stamp of the event.*
- unsigned short **type**  
*Type of the event.*
- unsigned short **code**  
*Code of the event.*
- int **value**  
*Value of the event.*
- [l4\\_umword\\_t](#) **stream\_id**  
*Stream ID.*

### 15.346.1 Detailed Description

Event structure used in buffer.

Definition at line 40 of file [event.h](#).

The documentation for this struct was generated from the following file:

- [l4re/c/event.h](#)

## 15.347 l4re\_video\_color\_component\_t Struct Reference

Color component structure.

```
#include <colors.h>
```

Collaboration diagram for l4re\_video\_color\_component\_t:

l4re_video_color_component_t	
+	size
+	shift

### Data Fields

- unsigned char **size**  
*Size in bits.*
- unsigned char **shift**  
*offset in pixel*

### 15.347.1 Detailed Description

Color component structure.

Definition at line 31 of file [colors.h](#).

The documentation for this struct was generated from the following file:

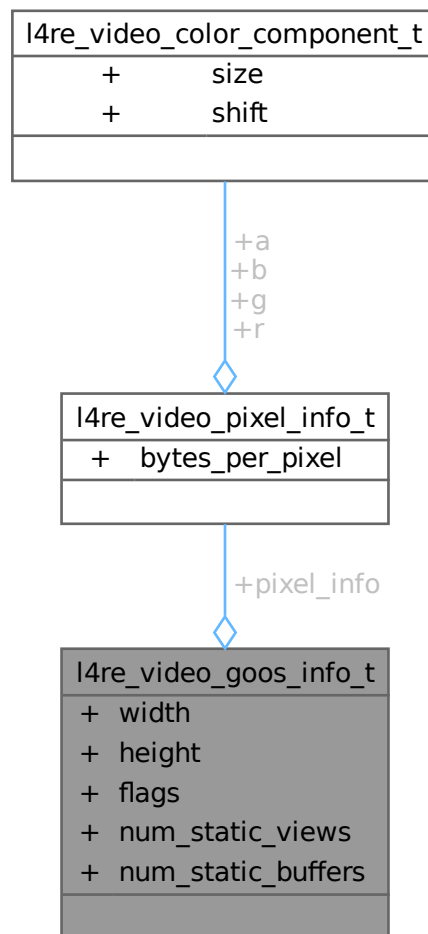
- [l4/re/c/video/colors.h](#)

## 15.348 l4re\_video\_goos\_info\_t Struct Reference

Goos information structure.

```
#include <goos.h>
```

Collaboration diagram for `l4re_video_goos_info_t`:



## Data Fields

- unsigned long **width**  
*Width of the goos.*
- unsigned long **height**  
*Height of the goos.*
- unsigned **flags**  
*Flags of the framebuffer, see [l4re\\_video\\_goos\\_info\\_flags\\_t](#).*
- unsigned **num\_static\_views**  
*Number of static views.*
- unsigned **num\_static\_buffers**  
*Number of static buffers.*
- [l4re\\_video\\_pixel\\_info\\_t](#) **pixel\_info**  
*Pixel layout of the goos.*

### 15.348.1 Detailed Description

Goos information structure.

Definition at line 51 of file [goos.h](#).

The documentation for this struct was generated from the following file:

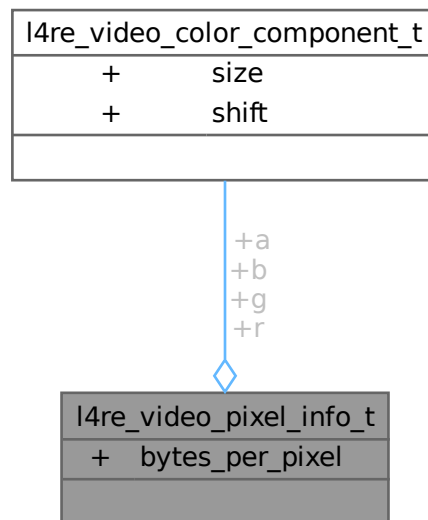
- [l4re/c/video/goos.h](#)

## 15.349 l4re\_video\_pixel\_info\_t Struct Reference

Pixel\_info structure.

```
#include <colors.h>
```

Collaboration diagram for l4re\_video\_pixel\_info\_t:



### Data Fields

- [l4re\\_video\\_color\\_component\\_t](#) **a**  
*Colors.*
- unsigned char **bytes\_per\_pixel**  
*Bytes per pixel.*

### 15.349.1 Detailed Description

Pixel\_info structure.

Definition at line 41 of file [colors.h](#).

The documentation for this struct was generated from the following file:

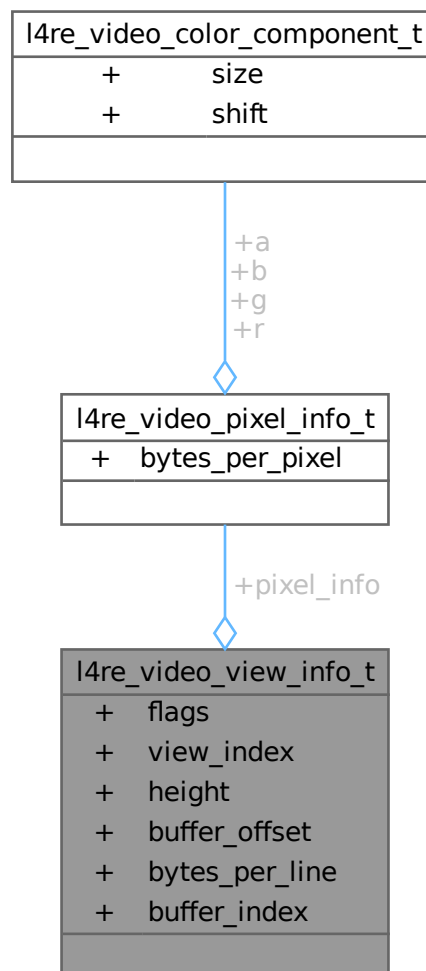
- [l4re/c/video/colors.h](#)

## 15.350 l4re\_video\_view\_info\_t Struct Reference

View information structure.

```
#include <view.h>
```

Collaboration diagram for l4re\_video\_view\_info\_t:



## Data Fields

- unsigned **flags**  
*Flags.*
- unsigned **view\_index**  
*Number of view in the goos.*
- unsigned long **height**  
*Position in goos and size of view.*
- unsigned long **buffer\_offset**  
*Memory offset in goos buffer.*
- unsigned long **bytes\_per\_line**  
*Size of line in view.*
- [l4re\\_video\\_pixel\\_info\\_t](#) **pixel\_info**  
*Pixel info.*
- unsigned **buffer\_index**  
*Number of buffer of goos.*

### 15.350.1 Detailed Description

View information structure.

Definition at line 59 of file [view.h](#).

The documentation for this struct was generated from the following file:

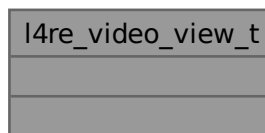
- [l4re/c/video/view.h](#)

## 15.351 l4re\_video\_view\_t Struct Reference

C representation of a goos view.

```
#include <view.h>
```

Collaboration diagram for l4re\_video\_view\_t:



### 15.351.1 Detailed Description

C representation of a goos view.

A view is a visible rectangle that provides a view to the contents of a buffer (frame buffer) memory object and is placed on a real screen.

Definition at line 78 of file [view.h](#).

The documentation for this struct was generated from the following file:

- [l4/re/c/video/view.h](#)

## 15.352 l4shmc\_ringbuf\_head\_t Struct Reference

Head field of a ring buffer.

```
#include <ringbuf.h>
```

Collaboration diagram for l4shmc\_ringbuf\_head\_t:

l4shmc_ringbuf_head_t	
+	next_read
+	next_write
+	bytes_filled
+	sender_waits
+	data

### Data Fields

- unsigned **next\_read**  
*offset to next read packet*
- unsigned **next\_write**  
*offset to next write packet*
- unsigned **bytes\_filled**  
*bytes filled in buffer*
- unsigned **sender\_waits**  
*sender waiting?*
- char **data** []  
*tail pointer -> data*



### 15.352.1 Detailed Description

Head field of a ring buffer.

Definition at line 58 of file [ringbuf.h](#).

The documentation for this struct was generated from the following file:

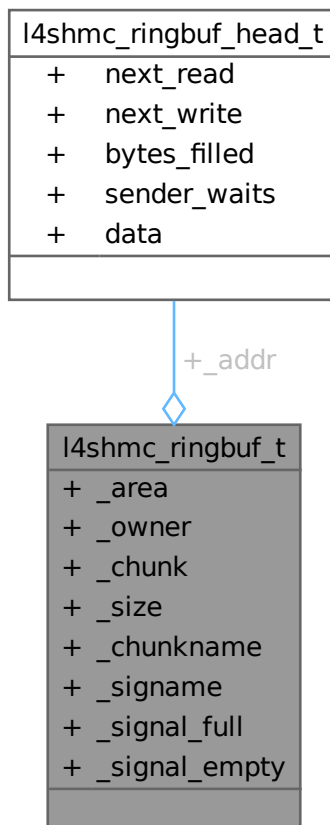
- [l4shmc/ringbuf.h](#)

## 15.353 l4shmc\_ringbuf\_t Struct Reference

Ring buffer.

```
#include <ringbuf.h>
```

Collaboration diagram for l4shmc\_ringbuf\_t:



## Data Fields

- `l4shmc_area_t * _area`  
*L4SHM area this buffer is located in.*
- `l4_cap_idx_t _owner`  
*owner (attached to send/recv signal)*
- `l4shmc_chunk_t _chunk`  
*chunk descriptor*
- `unsigned _size`  
*chunk size // XXX do we need this?*
- `char * _chunkname`  
*name of the ring buffer chunk*
- `char * _signame`  
*base name of the ring buffer signals*
- `l4shmc_ringbuf_head_t * _addr`  
*pointer to ring buffer head*
- `l4shmc_signal_t _signal_full`  
*"full" signal - triggered when data is produced*
- `l4shmc_signal_t _signal_empty`  
*"empty" signal - triggered when data is consumed*

### 15.353.1 Detailed Description

Ring buffer.

Definition at line 84 of file [ringbuf.h](#).

The documentation for this struct was generated from the following file:

- [l4/shmc/ringbuf.h](#)

## 15.354 l4util\_idt\_desc\_t Struct Reference

IDT entry.

```
#include <idt.h>
```

Collaboration diagram for `l4util_idt_desc_t`:

l4util_idt_desc_t	
+	b
+	b

## Data Fields

- [l4\\_uint64\\_t](#) **b**  
*see Intel doc*
- [l4\\_uint32\\_t](#) **b**  
*see Intel doc*

### 15.354.1 Detailed Description

IDT entry.

Definition at line 33 of file [idt.h](#).

The documentation for this struct was generated from the following files:

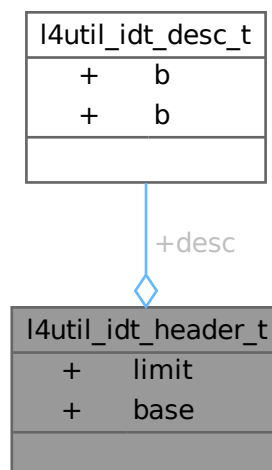
- [amd64/l4/util/idt.h](#)
- [x86/l4/util/idt.h](#)

## 15.355 l4util\_idt\_header\_t Struct Reference

Header of an IDT table.

```
#include <idt.h>
```

Collaboration diagram for l4util\_idt\_header\_t:



## Data Fields

- [l4\\_uint16\\_t](#) **limit**  
*limit field (see Intel doc)*
- void \* **base**  
*idt base (see Intel doc)*

### 15.355.1 Detailed Description

Header of an IDT table.

Definition at line 40 of file [idt.h](#).

The documentation for this struct was generated from the following files:

- amd64/l4/util/[idt.h](#)
- x86/l4/util/[idt.h](#)

## 15.356 l4util\_l4mod\_info Struct Reference

Base module structure.

```
#include <l4mod.h>
```

Collaboration diagram for l4util\_l4mod\_info:

l4util_l4mod_info
+ flags
+ cmdline
+ mods_addr
+ mods_count
+ vbe_ctrl_info
+ vbe_mode_info

**Data Fields**

- [l4\\_uint64\\_t](#) **flags**  
*Flags.*
- [l4\\_uint64\\_t](#) **cmdline**  
*Pointer to kernel command line.*
- [l4\\_uint64\\_t](#) **mods\_addr**  
*Module list.*
- [l4\\_uint32\\_t](#) **mods\_count**  
*Number of modules.*
- [l4\\_uint64\\_t](#) **vbe\_ctrl\_info**  
*VESA video info, valid if one of vbe\_ctrl\_info or vbe\_mode\_info is not zero.*
- [l4\\_uint64\\_t](#) **vbe\_mode\_info**  
*VESA video mode info.*

**15.356.1 Detailed Description**

Base module structure.

Definition at line 35 of file [l4mod.h](#).

**15.356.2 Field Documentation****15.356.2.1 vbe\_ctrl\_info**

```
l4\_uint64\_t l4util_l4mod_info::vbe_ctrl_info
```

VESA video info, valid if one of vbe\_ctrl\_info or vbe\_mode\_info is not zero.

VESA video controller info

Definition at line 47 of file [l4mod.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/l4mod.h](#)

**15.357 l4util\_l4mod\_mod Struct Reference**

A single module.

```
#include <l4mod.h>
```

Collaboration diagram for l4util\_l4mod\_mod:

l4util_l4mod_mod
+ flags
+ mod_start
+ mod_end
+ cmdline

## Data Fields

- [l4\\_uint64\\_t](#) **flags**  
*Module flags ([l4util\\_l4mod\\_mod\\_info\\_flag](#))*
- [l4\\_uint64\\_t](#) **mod\_start**  
*Starting address of module in memory.*
- [l4\\_uint64\\_t](#) **mod\_end**  
*End address of module in memory.*
- [l4\\_uint64\\_t](#) **cmdline**  
*Module command line.*

### 15.357.1 Detailed Description

A single module.

Definition at line 26 of file [l4mod.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/l4mod.h](#)

### 15.358 l4util\_mb\_addr\_range\_t Struct Reference

INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached.

```
#include <mb_info.h>
```

Collaboration diagram for `l4util_mb_addr_range_t`:

l4util_mb_addr_range_t	
+	struct_size
+	addr
+	size
+	type

**Data Fields**

- [l4\\_uint32\\_t struct\\_size](#)  
*Size of structure.*
- [l4\\_uint64\\_t addr](#)  
*Start address.*
- [l4\\_uint64\\_t size](#)  
*Size of memory range.*
- [l4\\_uint32\\_t type](#)  
*type of memory range*

**15.358.1 Detailed Description**

INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached.

Definition at line 50 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

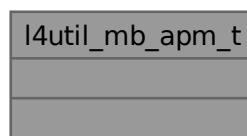
- [l4/util/mb\\_info.h](#)

**15.359 l4util\_mb\_apm\_t Struct Reference**

APM BIOS info.

```
#include <mb_info.h>
```

Collaboration diagram for l4util\_mb\_apm\_t:

**15.359.1 Detailed Description**

APM BIOS info.

Definition at line 92 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/mb\\_info.h](#)

## 15.360 l4util\_mb\_drive\_t Struct Reference

Drive Info structure.

```
#include <mb_info.h>
```

Collaboration diagram for l4util\_mb\_drive\_t:

l4util_mb_drive_t
+ drive_number
+ drive_mode
+ drive_cylinders
+ drive_heads
+ drive_sectors
+ drive_ports

### Data Fields

- [l4\\_uint8\\_t drive\\_number](#)  
<The size of this structure.
- [l4\\_uint8\\_t drive\\_mode](#)  
<The BIOS drive number.
- [l4\\_uint16\\_t drive\\_cylinders](#)  
<The access mode (see below).
- [l4\\_uint8\\_t drive\\_heads](#)  
<number of cylinders
- [l4\\_uint8\\_t drive\\_sectors](#)  
<number of heads
- [l4\\_uint16\\_t drive\\_ports](#) [0]  
<number of sectors per track

### 15.360.1 Detailed Description

Drive Info structure.

Definition at line 75 of file [mb\\_info.h](#).



## 15.360.2 Field Documentation

### 15.360.2.1 drive\_cylinders

`l4_uint16_t l4util_mb_drive_t::drive_cylinders`

<The access mode (see below).

Definition at line 80 of file [mb\\_info.h](#).

### 15.360.2.2 drive\_mode

`l4_uint8_t l4util_mb_drive_t::drive_mode`

<The BIOS drive number.

Definition at line 79 of file [mb\\_info.h](#).

### 15.360.2.3 drive\_number

`l4_uint8_t l4util_mb_drive_t::drive_number`

<The size of this structure.

Definition at line 78 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/mb\\_info.h](#)

## 15.361 l4util\_mb\_info\_t Struct Reference

MultiBoot Info description.

```
#include <mb_info.h>
```

Collaboration diagram for l4util\_mb\_info\_t:

l4util_mb_info_t
+ flags
+ mem_lower
+ mem_upper
+ boot_device
+ cmdline
+ mods_count
+ mods_addr
+ tabsize
+ num
+ mmap_length
and 12 more...

### Data Fields

- [l4\\_uint32\\_t](#) **flags**  
*MultiBoot info version number.*
- [l4\\_uint32\\_t](#) **mem\_lower**  
*available memory below 1MB*
- [l4\\_uint32\\_t](#) **mem\_upper**  
*available memory starting from 1MB [kB]*
- [l4\\_uint32\\_t](#) **boot\_device**  
*"root" partition*
- [l4\\_uint32\\_t](#) **cmdline**  
*Kernel command line.*
- [l4\\_uint32\\_t](#) **mods\_count**  
*number of modules*
- [l4\\_uint32\\_t](#) **mods\_addr**  
*module list*
- [l4\\_uint32\\_t](#) **mmap\_length**  
*size of memory mapping buffer*
- [l4\\_uint32\\_t](#) **mmap\_addr**  
*address of memory mapping buffer*

- [l4\\_uint32\\_t drives\\_length](#)  
*size of drive info buffer*
- [l4\\_uint32\\_t drives\\_addr](#)  
*address of driver info buffer*
- [l4\\_uint32\\_t config\\_table](#)  
*ROM configuration table.*
- [l4\\_uint32\\_t boot\\_loader\\_name](#)  
*Boot Loader Name.*
- [l4\\_uint32\\_t apm\\_table](#)  
*APM table.*
- [l4\\_uint32\\_t vbe\\_ctrl\\_info](#)  
*VESA video controller info.*
- [l4\\_uint32\\_t vbe\\_mode\\_info](#)  
*VESA video mode info.*
- [l4\\_uint16\\_t vbe\\_mode](#)  
*VESA video mode number.*
- [l4\\_uint16\\_t vbe\\_interface\\_seg](#)  
*VESA segment of prot BIOS interface.*
- [l4\\_uint16\\_t vbe\\_interface\\_off](#)  
*VESA offset of prot BIOS interface.*
- [l4\\_uint16\\_t vbe\\_interface\\_len](#)  
*VESA lenght of prot BIOS interface.*

### 15.361.1 Detailed Description

MultiBoot Info description.

This is the struct passed to the boot image. This is done by placing its address in the EAX register.

Definition at line 249 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

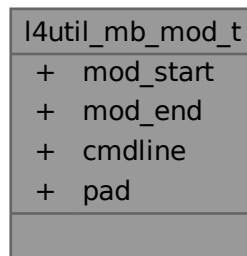
- [l4/util/mb\\_info.h](#)

## 15.362 l4util\_mb\_mod\_t Struct Reference

The structure type "mod\_list" is used by the [multiboot\\_info](#) structure.

```
#include <mb_info.h>
```

Collaboration diagram for `l4util_mb_mod_t`:



## Data Fields

- [l4\\_uint32\\_t](#) **mod\_start**  
*Starting address of module in memory.*
- [l4\\_uint32\\_t](#) **mod\_end**  
*End address of module in memory.*
- [l4\\_uint32\\_t](#) **cmdline**  
*Module command line.*
- [l4\\_uint32\\_t](#) **pad**  
*padding to take it to 16 bytes*

## 15.362.1 Detailed Description

The structure type "mod\_list" is used by the [multiboot\\_info](#) structure.

Definition at line 35 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

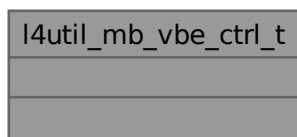
- [l4/util/mb\\_info.h](#)

## 15.363 l4util\_mb\_vbe\_ctrl\_t Struct Reference

VBE controller information.

```
#include <mb_info.h>
```

Collaboration diagram for l4util\_mb\_vbe\_ctrl\_t:



### 15.363.1 Detailed Description

VBE controller information.

Definition at line 108 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/mb\\_info.h](#)

## 15.364 l4util\_mb\_vbe\_mode\_t Struct Reference

VBE mode information.

```
#include <mb_info.h>
```

Collaboration diagram for `l4util_mb_vbe_mode_t`:

<code>l4util_mb_vbe_mode_t</code>
+ mode_attributes
+ win_a_attributes
+ win_b_attributes
+ win_granularity
+ win_size
+ win_a_segment
+ win_b_segment
+ win_func
+ bytes_per_scanline
+ x_resolution
+ y_resolution
+ x_char_size
+ y_char_size
+ number_of_planes
+ bits_per_pixel
+ number_of_banks
+ memory_model
+ bank_size
+ number_of_image_pages
+ reserved0
+ red_mask_size
+ red_field_position
+ green_mask_size
+ green_field_position
+ blue_mask_size
+ blue_field_position
+ reserved_mask_size
+ reserved_field_position
+ direct_color_mode_info
+ phys_base
+ reserved1
+ reserved2
+ linear_bytes_per_scanline
+ banked_number_of_image_pages
+ linear_number_of_image_pages
+ linear_red_mask_size
+ linear_red_field_position
+ linear_green_mask_size
+ linear_green_field_position
+ linear_blue_mask_size
+ linear_blue_field_position
+ linear_reserved_mask_size
+ linear_reserved_field_position
+ max_pixel_clock
+ reserved3
* mode_attributes
* win_a_attributes
* win_b_attributes
* win_granularity
* win_size
* win_a_segment
* win_b_segment
* win_func
* bytes_per_scanline
* x_resolution
* y_resolution
* x_char_size
* y_char_size
* number_of_planes
* bits_per_pixel
* number_of_banks
* memory_model
* bank_size
* number_of_image_pages
* reserved0
* red_mask_size
* red_field_position
* green_mask_size
* green_field_position
* blue_mask_size
* blue_field_position
* reserved_mask_size
* reserved_field_position
* direct_color_mode_info
* phys_base
* reserved1
* reserved2
* linear_bytes_per_scanline
* banked_number_of_image_pages
* linear_number_of_image_pages
* linear_red_mask_size
* linear_red_field_position
* linear_green_mask_size
* linear_green_field_position
* linear_blue_mask_size
* linear_blue_field_position
* linear_reserved_mask_size
* linear_reserved_field_position
* max_pixel_clock
* reserved3

## Data Fields

### all VESA versions

- [l4\\_uint16\\_t mode\\_attributes](#)  
*Mode attributes.*
- [l4\\_uint8\\_t win\\_a\\_attributes](#)  
*Window A attributes.*

- [l4\\_uint8\\_t win\\_b\\_attributes](#)  
*Window B attributes.*
- [l4\\_uint16\\_t win\\_granularity](#)  
*Window granularity.*
- [l4\\_uint16\\_t win\\_size](#)  
*Window size.*
- [l4\\_uint16\\_t win\\_a\\_segment](#)  
*Window A start segment.*
- [l4\\_uint16\\_t win\\_b\\_segment](#)  
*Window B start segment.*
- [l4\\_uint32\\_t win\\_func](#)  
*Real mode pointer to window function.*
- [l4\\_uint16\\_t bytes\\_per\\_scanline](#)  
*Bytes per scan line.*

#### >= VESA version 1.2

- [l4\\_uint16\\_t x\\_resolution](#)  
*Horizontal resolution in pixels or characters.*
- [l4\\_uint16\\_t y\\_resolution](#)  
*Vertical resolution in pixels or characters.*
- [l4\\_uint8\\_t x\\_char\\_size](#)  
*Character cell width in pixels.*
- [l4\\_uint8\\_t y\\_char\\_size](#)  
*Character cell height in pixels.*
- [l4\\_uint8\\_t number\\_of\\_planes](#)  
*Number of memory planes.*
- [l4\\_uint8\\_t bits\\_per\\_pixel](#)  
*Bits per pixel.*
- [l4\\_uint8\\_t number\\_of\\_banks](#)  
*Number of banks.*
- [l4\\_uint8\\_t memory\\_model](#)  
*Memory model type.*
- [l4\\_uint8\\_t bank\\_size](#)  
*Bank size in KiB.*
- [l4\\_uint8\\_t number\\_of\\_image\\_pages](#)  
*Number of images.*
- [l4\\_uint8\\_t reserved0](#)  
*Reserved for page function.*

#### direct color

- [l4\\_uint8\\_t red\\_mask\\_size](#)  
*Size of direct color red mask in bits.*
- [l4\\_uint8\\_t red\\_field\\_position](#)  
*Bit position of LSB of red mask.*
- [l4\\_uint8\\_t green\\_mask\\_size](#)  
*Size of direct color green mask in bits.*
- [l4\\_uint8\\_t green\\_field\\_position](#)  
*Bit position of LSB of green mask.*
- [l4\\_uint8\\_t blue\\_mask\\_size](#)  
*Size of direct color blue mask in bits.*
- [l4\\_uint8\\_t blue\\_field\\_position](#)  
*Bit position of LSB of blue mask.*
- [l4\\_uint8\\_t reserved\\_mask\\_size](#)  
*Size of direct color reserved mask in bits.*
- [l4\\_uint8\\_t reserved\\_field\\_position](#)  
*Bit position of LSB of reserved mask.*

- [l4\\_uint8\\_t direct\\_color\\_mode\\_info](#)

*Direct color mode attributes.*

#### >= VESA version 2.0

- [l4\\_uint32\\_t phys\\_base](#)  
*Physical address for flat memory memory frame buffer.*
- [l4\\_uint32\\_t reserved1](#)  
*Reserved – always set to 0.*
- [l4\\_uint16\\_t reversed2](#)  
*Reserved – always set to 0.*

#### >= VESA version 3.0

- [l4\\_uint16\\_t linear\\_bytes\\_per\\_scanline](#)  
*Bytes per scan line for linear modes.*
- [l4\\_uint8\\_t banked\\_number\\_of\\_image\\_pages](#)  
*Number of images for banked modes.*
- [l4\\_uint8\\_t linear\\_number\\_of\\_image\\_pages](#)  
*Number of images for linear modes.*
- [l4\\_uint8\\_t linear\\_red\\_mask\\_size](#)  
*Size of direct color red mask (linear modes).*
- [l4\\_uint8\\_t linear\\_red\\_field\\_position](#)  
*Bit position of LSB of red mask (linear modes).*
- [l4\\_uint8\\_t linear\\_green\\_mask\\_size](#)  
*Size of direct color green mask (linear modes).*
- [l4\\_uint8\\_t linear\\_green\\_field\\_position](#)  
*Bit position of LSB of green mask (linear modes).*
- [l4\\_uint8\\_t linear\\_blue\\_mask\\_size](#)  
*Size of direct color blue mask (linear modes).*
- [l4\\_uint8\\_t linear\\_blue\\_field\\_position](#)  
*Bit position of LSB of blue mask (linear modes).*
- [l4\\_uint8\\_t linear\\_reserved\\_mask\\_size](#)  
*Size of direct color reserved mask (linear modes).*
- [l4\\_uint8\\_t linear\\_reserved\\_field\\_position](#)  
*Bit position of LSB of reserved mask (linear modes).*
- [l4\\_uint32\\_t max\\_pixel\\_clock](#)  
*Maximum pixel clock (in Hz) for graphics mode.*
- [l4\\_uint8\\_t reserved3](#) [190]  
*Reserved (padding)*

### 15.364.1 Detailed Description

VBE mode information.

Definition at line 127 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/mb\\_info.h](#)

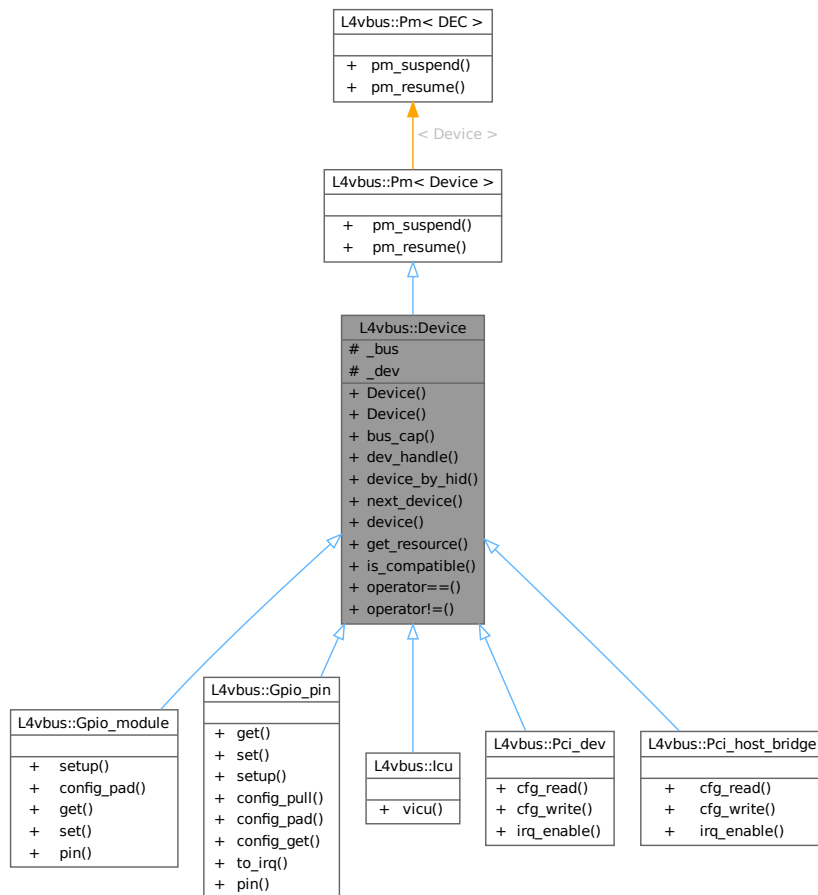


## 15.365 L4vbus::Device Class Reference

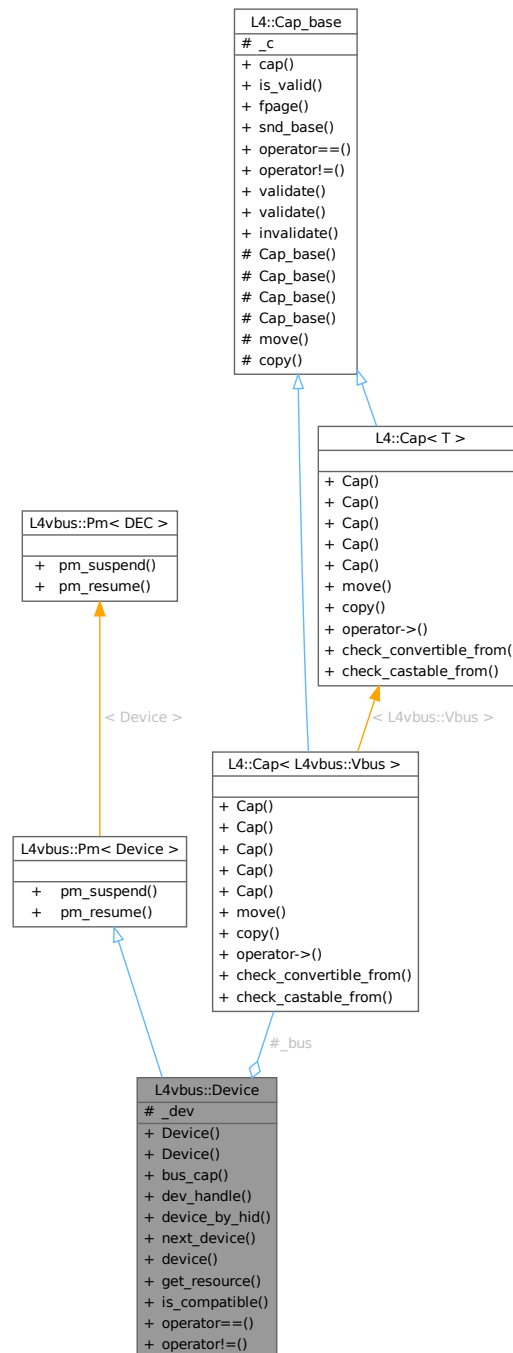
Device on a [L4vbus::Vbus](#).

```
#include <vbus>
```

Inheritance diagram for L4vbus::Device:



Collaboration diagram for L4vbus::Device:



## Public Member Functions

- **Device ( )**  
Construct a new vbus device using the NULL device `L4VBUS_NULL`.
- **Device (L4::Cap< Vbus > bus, l4vbus\_device\_handle\_t dev)**  
Construct a new vbus device using a device handle.
- **L4::Cap< Vbus > bus\_cap ( ) const**

- Access the *Vbus* capability of the underlying virtual bus.

  - `l4vbus_device_handle_t dev_handle` () const

Access the device handle of this device.
- int `device_by_hid` (*Device* \*child, char const \*hid, int depth=L4VBUS\_MAX\_DEPTH, *l4vbus\_device\_t* \*devinfo=0) const

Find a device by the hardware interface identifier (HID).
- int `next_device` (*Device* \*child, int depth=L4VBUS\_MAX\_DEPTH, *l4vbus\_device\_t* \*devinfo=0) const

Find next child following *child*.
- int `device` (*l4vbus\_device\_t* \*devinfo) const

Obtain detailed information about a *Vbus* device.
- int `get_resource` (unsigned res\_idx, *l4vbus\_resource\_t* \*res) const

Obtain the resource description of an individual device resource.
- int `is_compatible` (char const \*cid) const

Check if the given device has a compatibility ID (CID) or HID that matches *cid*.
- bool `operator==` (*Device* const &o) const

Test if two devices are the same *Vbus* device.
- bool `operator!=` (*Device* const &o) const

Test if two *Vbus* devices are not the same.

## Public Member Functions inherited from `L4vbus::Pm< Device >`

- int `pm_suspend` () const

Suspend the device.
- int `pm_resume` () const

Resume the device.

## Protected Attributes

- `L4::Cap< Vbus > _bus`

The *Vbus* capability where this device is located on.
- `l4vbus_device_handle_t _dev`

The device handle for this device.

## 15.365.1 Detailed Description

`Device` on a `L4vbus::Vbus`.

Definition at line 85 of file `vbus`.

## 15.365.2 Constructor & Destructor Documentation

### 15.365.2.1 Device()

```
L4vbus::Device::Device (
    L4::Cap< Vbus > bus,
    l4vbus_device_handle_t dev ) [inline]
```

Construct a new vbus device using a device handle.

Specifying the special root bus device handle `L4VBUS_ROOT_BUS` forms the root device of the corresponding vbus, see `Vbus::root`.

## Parameters

<i>bus</i>	The vbus capability where this device is assigned.
<i>dev</i>	The device handle of the device.

Definition at line 102 of file [vbus](#).

### 15.365.3 Member Function Documentation

#### 15.365.3.1 bus\_cap()

```
L4::Cap< Vbus > L4vbus::Device::bus_cap ( ) const [inline]
```

Access the [Vbus](#) capability of the underlying virtual bus.

## Returns

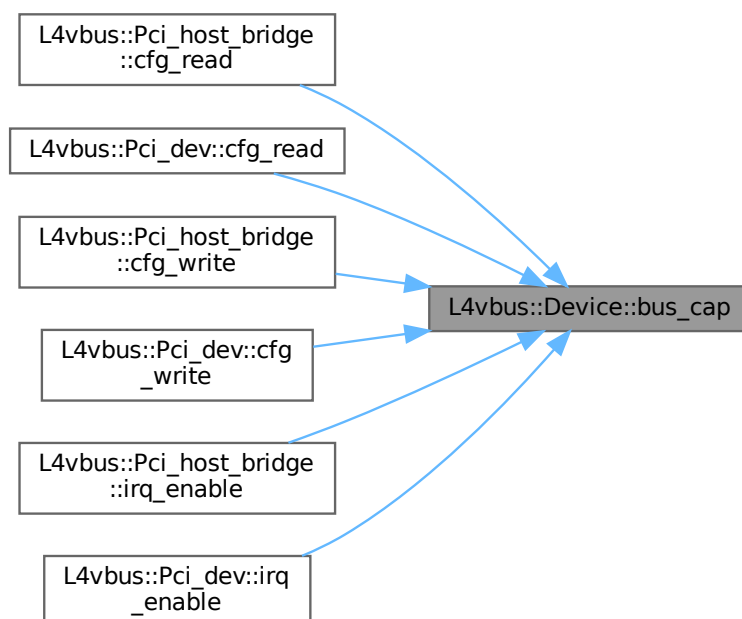
the capability to the underlying [Vbus](#).

Definition at line 109 of file [vbus](#).

References [\\_bus](#).

Referenced by [L4vbus::Pci\\_host\\_bridge::cfg\\_read\(\)](#), [L4vbus::Pci\\_dev::cfg\\_read\(\)](#), [L4vbus::Pci\\_host\\_bridge::cfg\\_write\(\)](#), [L4vbus::Pci\\_dev::cfg\\_write\(\)](#), [L4vbus::Pci\\_host\\_bridge::irq\\_enable\(\)](#), and [L4vbus::Pci\\_dev::irq\\_enable\(\)](#).

Here is the caller graph for this function:



### 15.365.3.2 dev\_handle()

```
l4vbus_device_handle_t L4vbus::Device::dev_handle ( ) const [inline]
```

Access the device handle of this device.

#### Returns

the device handle for this device.

The device handle is used to directly address the device on its virtual bus.

Definition at line 118 of file [vbus](#).

References [\\_dev](#).

### 15.365.3.3 device()

```
int L4vbus::Device::device (
    l4vbus_device_t * devinfo ) const [inline]
```

Obtain detailed information about a [Vbus](#) device.

#### Parameters

out	<i>devinfo</i>	Information structure which contains details about the device. The pointer might be NULL.
-----	----------------	---

#### Return values

0	Success.
-L4_ENODEV	No device with the given device handle <i>dev</i> could be found.

Definition at line 191 of file [vbus](#).

References [\\_bus](#), [\\_dev](#), and [l4vbus\\_get\\_device\(\)](#).

Here is the call graph for this function:



### 15.365.3.4 device\_by\_hid()

```
int L4vbus::Device::device_by_hid (
    Device * child,
    char const * hid,
    int depth = L4VBUS_MAX_DEPTH,
    l4vbus_device_t * devinfo = 0 ) const [inline]
```

Find a device by the hardware interface identifier (HID).

This function searches the vbus for a device with the given HID and returns a handle to the first matching device. The HID usually conforms to an ACPI HID or a Linux device tree compatible identifier.

It is possible to have multiple devices with the same HID on a vbus. In order to find all matching devices this function has to be called repeatedly with `child` pointing to the device found in the previous iteration. The iteration starts at `child` that might be any device node in the tree.

#### Parameters

in, out	<i>child</i>	Handle of the device from where in the device tree the search should start. To start searching from the beginning <i>child</i> must be initialized using the default ( <a href="#">L4VBUS_NULL</a> ). If a matching device is found, its handle is returned through this parameter.
	<i>hid</i>	HID of the device
	<i>depth</i>	Maximum depth for the recursive lookup
out	<i>devinfo</i>	<a href="#">Device</a> information structure (might be NULL)

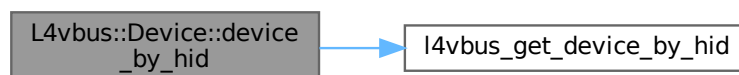
#### Return values

$\geq 0$	A device with the given HID was found.
<code>-L4_ENOENT</code>	No device with the given HID could be found on the vbus.
<code>-L4_EINVAL</code>	Invalid or no HID provided.
<code>-L4_ENODEV</code>	Function called on a non-existing device.

Definition at line [150](#) of file [vbus](#).

References [\\_bus](#), [\\_dev](#), and [l4vbus\\_get\\_device\\_by\\_hid\(\)](#).

Here is the call graph for this function:



### 15.365.3.5 get\_resource()

```
int L4vbus::Device::get_resource (
    unsigned res_idx,
    l4vbus_resource_t * res ) const [inline]
```

Obtain the resource description of an individual device resource.

#### Parameters

	<i>res_idx</i>	Index of the resource for which the resource description should be returned. The total number of resources for a device is available in the <a href="#">l4vbus_device_t</a> structure that is returned by <a href="#">L4vbus::Device::device_by_hid()</a> and <a href="#">L4vbus::Device::next_device()</a> .
out	<i>res</i>	Descriptor of the resource.

This function returns the resource descriptor of an individual device resource selected by the *res\_idx* parameter.

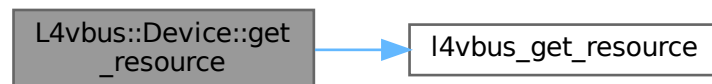
#### Return values

<i>0</i>	Success.
<i>-L4_ENOENT</i>	Invalid resource index <i>res_idx</i> .

Definition at line 211 of file [vbus](#).

References [\\_bus](#), [\\_dev](#), and [l4vbus\\_get\\_resource\(\)](#).

Here is the call graph for this function:



### 15.365.3.6 is\_compatible()

```
int L4vbus::Device::is_compatible (
    char const * cid ) const [inline]
```

Check if the given device has a compatibility ID (CID) or HID that matches *cid*.

#### Parameters

<i>cid</i>	the compatibility ID to test
------------	------------------------------

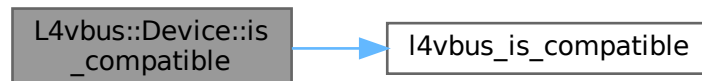
**Returns**

1 when the given ID (*cid*) matches this device, 0 when the given ID does not match, <0 on error.

Definition at line 225 of file [vbus](#).

References [\\_bus](#), [\\_dev](#), and [l4vbus\\_is\\_compatible\(\)](#).

Here is the call graph for this function:

**15.365.3.7 next\_device()**

```

int L4vbus::Device::next_device (
    Device * child,
    int depth = L4VBUS_MAX_DEPTH,
    l4vbus_device_t * devinfo = 0 ) const [inline]
  
```

Find next child following *child*.

**Parameters**

in, out	<i>child</i>	Handle of the device that precedes the device that shall be returned. To start from the beginning, <i>child</i> must be initialized with <a href="#">L4VBUS_NULL</a> ( <a href="#">Device::Device</a> ). If a device is found, its handle is returned through this parameter.
	<i>depth</i>	Depth to look for
out	<i>devinfo</i>	<a href="#">Device</a> information (might be NULL)

**Returns**

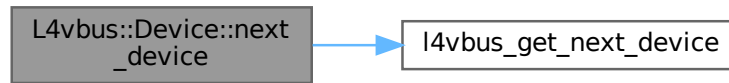
0 on success, else failure

Definition at line 173 of file [vbus](#).

References [\\_bus](#), [\\_dev](#), and [l4vbus\\_get\\_next\\_device\(\)](#).



Here is the call graph for this function:



### 15.365.3.8 `operator!=()`

```
bool L4vbus::Device::operator!= (
    Device const & o ) const [inline]
```

Test if two [Vbus](#) devices are not the same.

#### Returns

true if the two devices are different, false else.

Definition at line [241](#) of file [vbus](#).

References [\\_bus](#), and [\\_dev](#).

### 15.365.3.9 `operator==()`

```
bool L4vbus::Device::operator== (
    Device const & o ) const [inline]
```

Test if two devices are the same [Vbus](#) device.

#### Returns

true if the two devices are the same, false else.

Definition at line [232](#) of file [vbus](#).

References [\\_bus](#), and [\\_dev](#).

The documentation for this class was generated from the following file:

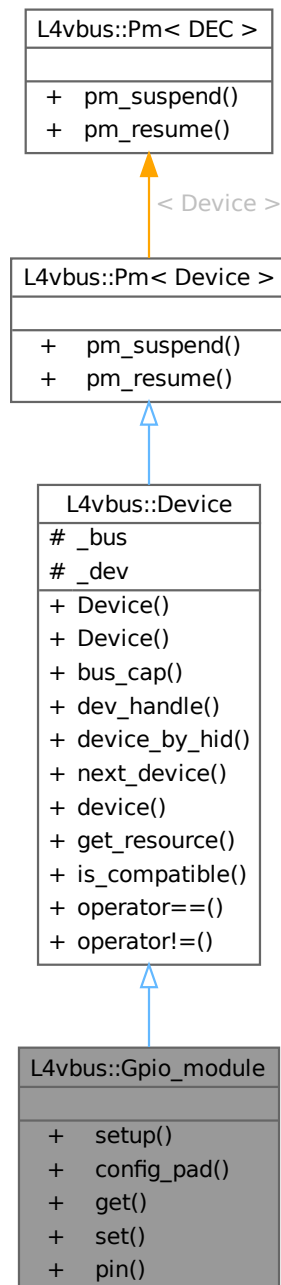
- [l4/vbus/vbus](#)

## 15.366 L4vbus::Gpio\_module Class Reference

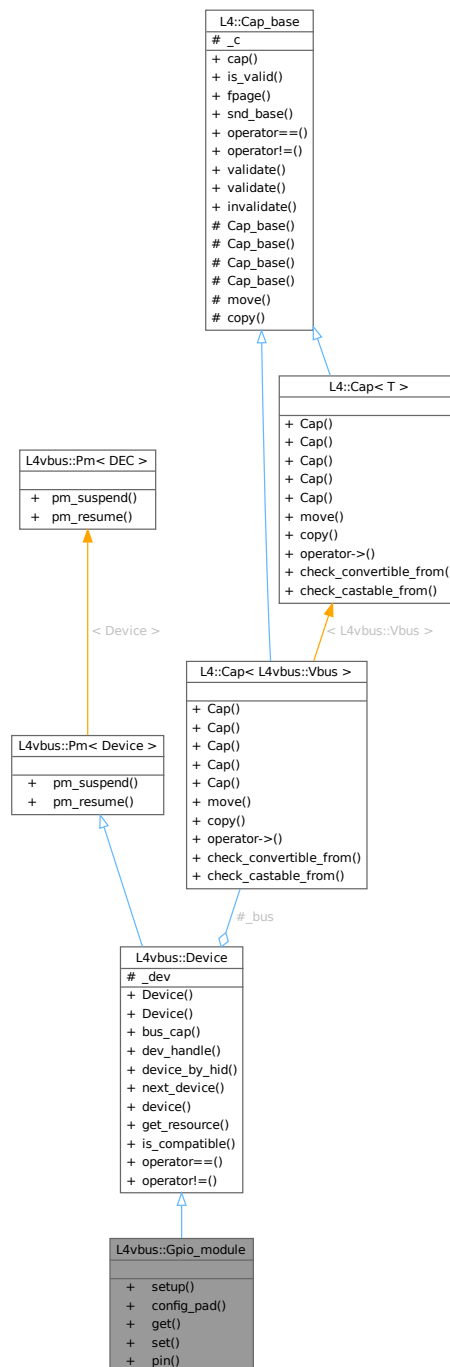
A [Gpio\\_module](#) groups multiple GPIO pins together.

```
#include <vbus_gpio>
```

Inheritance diagram for L4vbus::Gpio\_module:



Collaboration diagram for L4vbus::Gpio\_module:



## Data Structures

- struct `Pin_slice`  
*A slice of the pins provided by this module.*

## Public Member Functions

- int **setup** (Pin\_slice const &mask, unsigned mode, unsigned value) const

- *Configure function of multiple GPIO pins at once.*  
int [config\\_pad](#) ([Pin\\_slice](#) const &mask, unsigned func, unsigned value) const  
*Hardware specific configuration function for multiple GPIO pins.*
- int [get](#) (unsigned offset, unsigned \*data) const  
*Read values of multiple GPIO pins at once.*
- int [set](#) ([Pin\\_slice](#) const &mask, unsigned data)  
*Set multiple GPIO output pins at once.*
- [Gpio\\_pin](#) pin (unsigned pin) const  
*Get [Gpio\\_pin](#) for a specific pin of this [Gpio\\_module](#).*

## Public Member Functions inherited from [L4vbus::Device](#)

- [Device](#) ()  
*Construct a new vbus device using the NULL device [L4VBUS\\_NULL](#).*
- [Device](#) ([L4::Cap](#)< [Vbus](#) > bus, [l4vbus\\_device\\_handle\\_t](#) dev)  
*Construct a new vbus device using a device handle.*
- [L4::Cap](#)< [Vbus](#) > [bus\\_cap](#) () const  
*Access the [Vbus](#) capability of the underlying virtual bus.*
- [l4vbus\\_device\\_handle\\_t](#) [dev\\_handle](#) () const  
*Access the device handle of this device.*
- int [device\\_by\\_hid](#) ([Device](#) \*child, char const \*hid, int depth=[L4VBUS\\_MAX\\_DEPTH](#), [l4vbus\\_device\\_t](#) \*devinfo=0) const  
*Find a device by the hardware interface identifier (HID).*
- int [next\\_device](#) ([Device](#) \*child, int depth=[L4VBUS\\_MAX\\_DEPTH](#), [l4vbus\\_device\\_t](#) \*devinfo=0) const  
*Find next child following [child](#).*
- int [device](#) ([l4vbus\\_device\\_t](#) \*devinfo) const  
*Obtain detailed information about a [Vbus](#) device.*
- int [get\\_resource](#) (unsigned res\_idx, [l4vbus\\_resource\\_t](#) \*res) const  
*Obtain the resource description of an individual device resource.*
- int [is\\_compatible](#) (char const \*cid) const  
*Check if the given device has a compatibility ID (CID) or HID that matches [cid](#).*
- bool [operator==](#) ([Device](#) const &o) const  
*Test if two devices are the same [Vbus](#) device.*
- bool [operator!=](#) ([Device](#) const &o) const  
*Test if two [Vbus](#) devices are not the same.*

## Public Member Functions inherited from [L4vbus::Pm](#)< [Device](#) >

- int [pm\\_suspend](#) () const  
*Suspend the device.*
- int [pm\\_resume](#) () const  
*Resume the device.*

## Additional Inherited Members

## Protected Attributes inherited from [L4vbus::Device](#)

- [L4::Cap](#)< [Vbus](#) > [\\_bus](#)  
*The [Vbus](#) capability where this device is located on.*
- [l4vbus\\_device\\_handle\\_t](#) [\\_dev](#)  
*The device handle for this device.*

### 15.366.1 Detailed Description

A [Gpio\\_module](#) groups multiple GPIO pins together.

Definition at line 135 of file [vbus\\_gpio](#).

### 15.366.2 Member Function Documentation

#### 15.366.2.1 config\_pad()

```
int L4vbus::Gpio_module::config_pad (
    Pin_slice const & mask,
    unsigned func,
    unsigned value ) const [inline]
```

Hardware specific configuration function for multiple GPIO pins.

##### Parameters

<i>mask</i>	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation.
<i>func</i>	Hardware specific configuration register, usually offset to the GPIO chip's base address.
<i>value</i>	Value which is written into the hardware specific configuration register for the specified pins

##### Returns

0 if OK, error code otherwise

Definition at line 187 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_multi\\_config\\_pad\(\)](#).

Here is the call graph for this function:



#### 15.366.2.2 get()

```
int L4vbus::Gpio_module::get (
    unsigned offset,
    unsigned * data ) const [inline]
```

Read values of multiple GPIO pins at once.

## Parameters

	<i>offset</i>	Pin corresponding to the LSB in <i>data</i> . Note: allowed may be hardware specific.
<i>out</i>	<i>data</i>	Each bit returns the value (0 or 1) for the corresponding GPIO pin. The value of pins that are not accessible is undefined.

## Returns

0 if OK, error code otherwise

Definition at line 203 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_multi\\_get\(\)](#).

Here is the call graph for this function:



### 15.366.2.3 pin()

```
Gpio_pin L4vbus::Gpio_module::pin (  
    unsigned pin ) const [inline]
```

Get [Gpio\\_pin](#) for a specific pin of this [Gpio\\_module](#).

## Parameters

<i>pin</i>	GPIO pin number to get <a href="#">Gpio_pin</a> for.
------------	--

## Returns

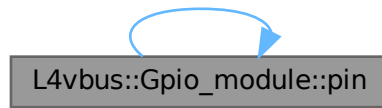
[Gpio\\_pin](#)

Definition at line 231 of file [vbus\\_gpio](#).

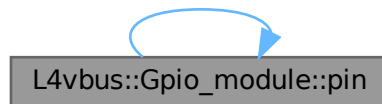
References [pin\(\)](#).

Referenced by [pin\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 15.366.2.4 set()

```
int L4vbus::Gpio_module::set (
    Pin_slice const & mask,
    unsigned data ) [inline]
```

Set multiple GPIO output pins at once.

##### Parameters

<i>mask</i>	Mask of GPIO pins to set. A bit set to 1 selects this pin. A maximum of 32 pins can be set at once. The real number depends on the hardware and the driver implementation.
<i>data</i>	Each bit corresponds to the GPIO pin in <i>mask</i> . The value of each bit is written to the GPIO pin if its bit in <i>mask</i> is set.

##### Returns

0 if OK, error code otherwise

Definition at line 219 of file `vbus_gpio`.

References `L4vbus::Device::_bus`, `L4vbus::Device::_dev`, and `l4vbus_gpio_multi_set()`.

Here is the call graph for this function:



### 15.366.2.5 setup()

```

int L4vbus::Gpio_module::setup (
    Pin_slice const & mask,
    unsigned mode,
    unsigned value ) const [inline]
  
```

Configure function of multiple GPIO pins at once.

#### Parameters

<i>mask</i>	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation.
<i>mode</i>	GPIO function, see <a href="#">L4vbus_gpio_generic_func</a> for generic functions. Hardware specific functions must be provided in the lower 8 bits.
<i>value</i>	Optional value to set the GPIO pins to if they are configured as output pins

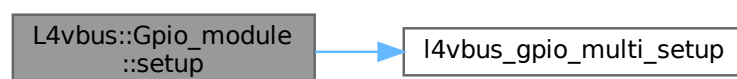
#### Returns

0 if OK, error code otherwise

Definition at line 168 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_multi\\_setup\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- [l4/vbus/vbus\\_gpio](#)

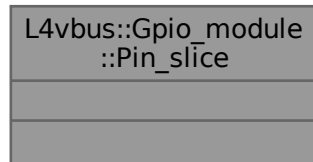


## 15.367 L4vbus::Gpio\_module::Pin\_slice Struct Reference

A slice of the pins provided by this module.

```
#include <vbus_gpio>
```

Collaboration diagram for L4vbus::Gpio\_module::Pin\_slice:



### 15.367.1 Detailed Description

A slice of the pins provided by this module.

Data type to specify a selection of pins for the 'multi' methods.

Definition at line 148 of file [vbus\\_gpio](#).

The documentation for this struct was generated from the following file:

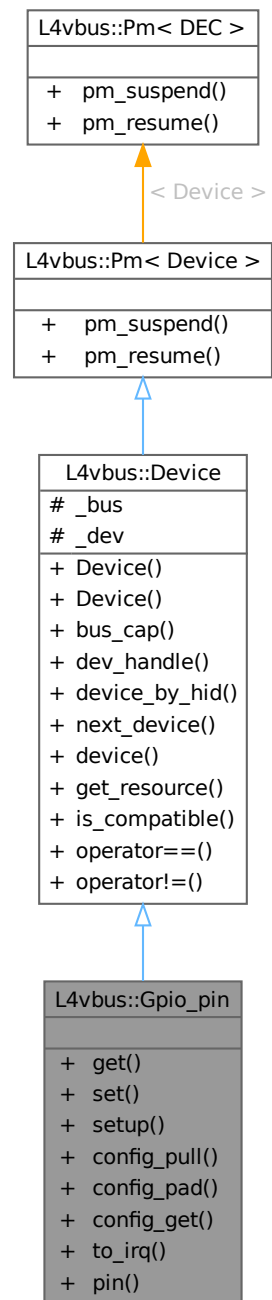
- I4/vbus/vbus\_gpio

## 15.368 L4vbus::Gpio\_pin Class Reference

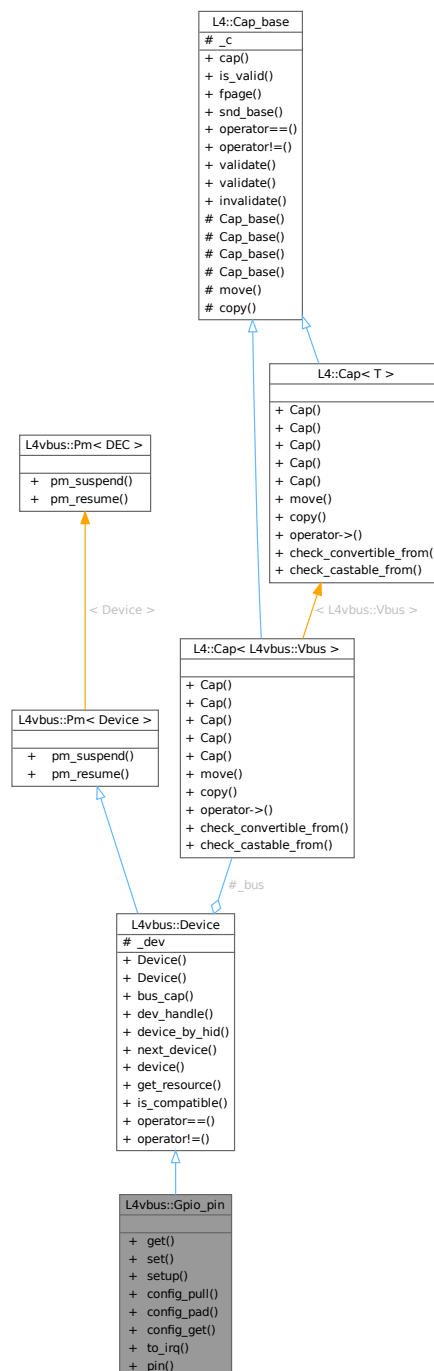
A GPIO pin.

```
#include <vbus_gpio>
```

Inheritance diagram for L4vbus::Gpio\_pin:



Collaboration diagram for L4vbus::Gpio\_pin:



## Public Member Functions

- `int get () const`  
Read value of GPIO input pin.
- `int set (int value) const`  
Set GPIO output pin.
- `int setup (unsigned mode, unsigned value) const`

- *Configure the function of a GPIO pin.*
- int `config_pull` (unsigned mode) const  
*Generic function to set pull up/down mode.*
- int `config_pad` (unsigned func, unsigned value) const  
*Hardware specific configuration function.*
- int `config_get` (unsigned func, unsigned \*value) const  
*Read hardware specific configuration.*
- int `to_irq` () const  
*Create IRQ for GPIO pin.*
- unsigned `pin` () const  
*Get pin number.*

## Public Member Functions inherited from `L4vbus::Device`

- `Device` ()  
*Construct a new vbus device using the NULL device `L4VBUS_NULL`.*
- `Device` (`L4::Cap`< `Vbus` > bus, `l4vbus_device_handle_t` dev)  
*Construct a new vbus device using a device handle.*
- `L4::Cap`< `Vbus` > `bus_cap` () const  
*Access the `Vbus` capability of the underlying virtual bus.*
- `l4vbus_device_handle_t` `dev_handle` () const  
*Access the device handle of this device.*
- int `device_by_hid` (`Device` \*child, char const \*hid, int depth=L4VBUS\_MAX\_DEPTH, `l4vbus_device_t` \*devinfo=0) const  
*Find a device by the hardware interface identifier (HID).*
- int `next_device` (`Device` \*child, int depth=L4VBUS\_MAX\_DEPTH, `l4vbus_device_t` \*devinfo=0) const  
*Find next child following `child`.*
- int `device` (`l4vbus_device_t` \*devinfo) const  
*Obtain detailed information about a `Vbus` device.*
- int `get_resource` (unsigned res\_idx, `l4vbus_resource_t` \*res) const  
*Obtain the resource description of an individual device resource.*
- int `is_compatible` (char const \*cid) const  
*Check if the given device has a compatibility ID (CID) or HID that matches `cid`.*
- bool `operator==` (`Device` const &o) const  
*Test if two devices are the same `Vbus` device.*
- bool `operator!=` (`Device` const &o) const  
*Test if two `Vbus` devices are not the same.*

## Public Member Functions inherited from `L4vbus::Pm`< `Device` >

- int `pm_suspend` () const  
*Suspend the device.*
- int `pm_resume` () const  
*Resume the device.*

## Additional Inherited Members

## Protected Attributes inherited from [L4vbus::Device](#)

- [L4::Cap< Vbus > \\_bus](#)  
*The [Vbus](#) capability where this device is located on.*
- [l4vbus\\_device\\_handle\\_t \\_dev](#)  
*The device handle for this device.*

### 15.368.1 Detailed Description

A GPIO pin.

Definition at line 28 of file [vbus\\_gpio](#).

### 15.368.2 Member Function Documentation

#### 15.368.2.1 `config_get()`

```
int L4vbus::Gpio_pin::config_get (
    unsigned func,
    unsigned * value ) const [inline]
```

Read hardware specific configuration.

#### Parameters

	<i>func</i>	Hardware specific configuration register to read from. Usually this is an offset to the GPIO chip's base address.
<i>out</i>	<i>value</i>	The configuration value.

#### Returns

0 if OK, error code otherwise

Definition at line 104 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_config\\_get\(\)](#).

Here is the call graph for this function:



### 15.368.2.2 config\_pad()

```
int L4vbus::Gpio_pin::config_pad (
    unsigned func,
    unsigned value ) const [inline]
```

Hardware specific configuration function.

#### Parameters

<i>func</i>	Hardware specific configuration register, usually offset to the GPIO chip's base address
<i>value</i>	Value which is written into the hardware specific configuration register for the specified pin

#### Returns

0 if OK, error code otherwise

Definition at line 91 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_config\\_pad\(\)](#).

Here is the call graph for this function:



### 15.368.2.3 config\_pull()

```
int L4vbus::Gpio_pin::config_pull (
    unsigned mode ) const [inline]
```

Generic function to set pull up/down mode.

#### Parameters

<i>mode</i>	mode for pull up/down resistors, see <a href="#">L4vbus_gpio_pull_modes</a>
-------------	---

#### Returns

0 if OK, error code otherwise

Definition at line 77 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_config\\_pull\(\)](#).

Here is the call graph for this function:



#### 15.368.2.4 `get()`

```
int L4vbus::Gpio_pin::get ( ) const [inline]
```

Read value of GPIO input pin.

##### Returns

Value of GPIO pin (usually 0 or 1), negative error code otherwise.

Definition at line 40 of file `vbus_gpio`.

References `L4vbus::Device::_bus`, `L4vbus::Device::_dev`, and `l4vbus_gpio_get()`.

Here is the call graph for this function:



#### 15.368.2.5 `pin()`

```
unsigned L4vbus::Gpio_pin::pin ( ) const [inline]
```

Get pin number.

##### Returns

GPIO pin number

Definition at line 124 of file `vbus_gpio`.

#### 15.368.2.6 `set()`

```
int L4vbus::Gpio_pin::set (
    int value ) const [inline]
```

Set GPIO output pin.

**Parameters**

<i>value</i>	Value to write to the GPIO pin (usually 0 or 1)
--------------	---

**Returns**

0 if OK, error code otherwise

Definition at line 51 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_set\(\)](#).

Here is the call graph for this function:

**15.368.2.7 setup()**

```
int L4vbus::Gpio_pin::setup (
    unsigned mode,
    unsigned value ) const [inline]
```

Configure the function of a GPIO pin.

**Parameters**

<i>mode</i>	GPIO function, see <a href="#">L4vbus_gpio_generic_func</a> for generic functions. Hardware specific functions must be provided in the lower 8 bits.
<i>value</i>	Optional value to set the GPIO pin to if it is configured as an output pin

**Returns**

0 if OK, error code otherwise

Definition at line 66 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_setup\(\)](#).



Here is the call graph for this function:



### 15.368.2.8 to\_irq()

```
int L4vbus::Gpio_pin::to_irq ( ) const [inline]
```

Create IRQ for GPIO pin.

#### Returns

IRQ number if OK, negative error code otherwise

Definition at line 114 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_to\\_irq\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

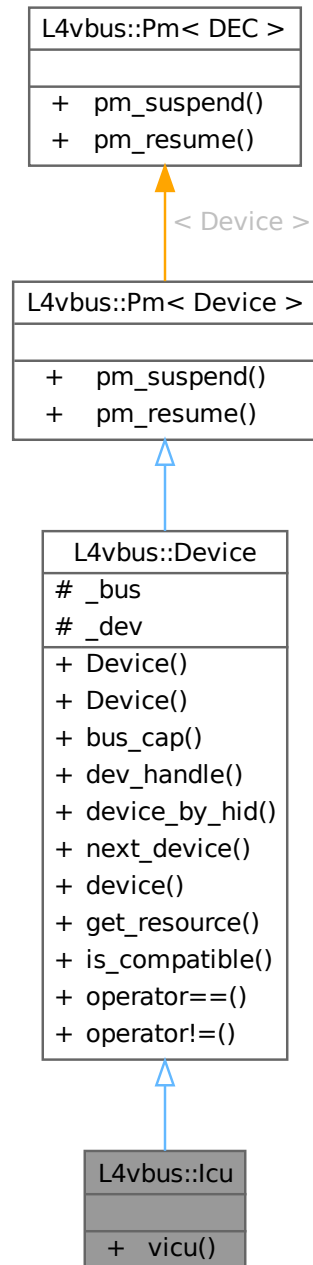
- `I4/vbus/vbus_gpio`

## 15.369 L4vbus::lcu Class Reference

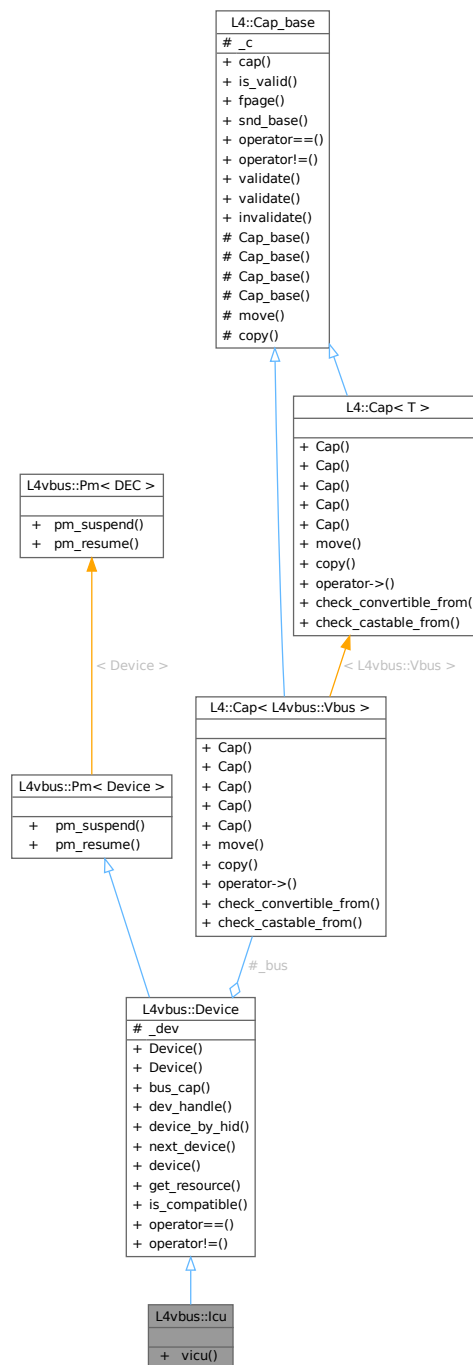
Vbus Interrupt controller API.

```
#include <vbus>
```

Inheritance diagram for L4vbus::lcu:



Collaboration diagram for L4vbus::lcu:



## Public Types

- enum `Src_types` { `Src_dev_handle` = `L4VBUS_ICU_SRC_DEV_HANDLE` }
- Flags that can be used with the ICU on a vbus device.

## Public Member Functions

- int `vicu` (L4::Cap< L4::lcu > icu) const

Request an [L4::Icu](#) capability for this [Vbus](#)'s virtual ICU.

## Public Member Functions inherited from [L4vbus::Device](#)

- [Device](#) ()  
*Construct a new vbus device using the NULL device [L4VBUS\\_NULL](#).*
- [Device](#) ([L4::Cap](#)< [Vbus](#) > bus, [l4vbus\\_device\\_handle\\_t](#) dev)  
*Construct a new vbus device using a device handle.*
- [L4::Cap](#)< [Vbus](#) > [bus\\_cap](#) () const  
*Access the [Vbus](#) capability of the underlying virtual bus.*
- [l4vbus\\_device\\_handle\\_t](#) [dev\\_handle](#) () const  
*Access the device handle of this device.*
- int [device\\_by\\_hid](#) ([Device](#) \*child, char const \*hid, int depth=L4VBUS\_MAX\_DEPTH, [l4vbus\\_device\\_t](#) \*devinfo=0) const  
*Find a device by the hardware interface identifier (HID).*
- int [next\\_device](#) ([Device](#) \*child, int depth=L4VBUS\_MAX\_DEPTH, [l4vbus\\_device\\_t](#) \*devinfo=0) const  
*Find next child following [child](#).*
- int [device](#) ([l4vbus\\_device\\_t](#) \*devinfo) const  
*Obtain detailed information about a [Vbus](#) device.*
- int [get\\_resource](#) (unsigned res\_idx, [l4vbus\\_resource\\_t](#) \*res) const  
*Obtain the resource description of an individual device resource.*
- int [is\\_compatible](#) (char const \*cid) const  
*Check if the given device has a compatibility ID (CID) or HID that matches [cid](#).*
- bool [operator==](#) ([Device](#) const &o) const  
*Test if two devices are the same [Vbus](#) device.*
- bool [operator!=](#) ([Device](#) const &o) const  
*Test if two [Vbus](#) devices are not the same.*

## Public Member Functions inherited from [L4vbus::Pm](#)< [Device](#) >

- int [pm\\_suspend](#) () const  
*Suspend the device.*
- int [pm\\_resume](#) () const  
*Resume the device.*

## Additional Inherited Members

## Protected Attributes inherited from [L4vbus::Device](#)

- [L4::Cap](#)< [Vbus](#) > [\\_bus](#)  
*The [Vbus](#) capability where this device is located on.*
- [l4vbus\\_device\\_handle\\_t](#) [\\_dev](#)  
*The device handle for this device.*

## 15.369.1 Detailed Description

[Vbus](#) Interrupt controller API.

Every [Vbus](#) contains a virtual interrupt control unit that manages the IRQs of the devices on the bus. This class provides access to a capability to an [L4::Icu](#) object which can then be used to interface with the IRQs.

See also

[L4::Icu](#)

Definition at line 262 of file [vbus](#).

## 15.369.2 Member Enumeration Documentation

### 15.369.2.1 Src\_types

```
enum L4vbus::Icu::Src_types
```

Flags that can be used with the ICU on a vbus device.

Enumerator

Src_dev_handle	Flag to denote that the value should be interpreted as a device handle. This flag may be used in the <code>source</code> parameter in <a href="#">L4::Icu::msi_info()</a> to denote that the ICU should interpret the source ID as a device handle.
----------------	---

Definition at line 266 of file [vbus](#).

## 15.369.3 Member Function Documentation

### 15.369.3.1 vicu()

```
int L4vbus::Icu::vicu (
    L4::Cap< L4::Icu > icu ) const [inline]
```

Request an [L4::Icu](#) capability for this [Vbus](#)'s virtual ICU.

Parameters

out	icu	Capability slot where the <a href="#">L4::Icu</a> capability shall be stored.
-----	-----	---

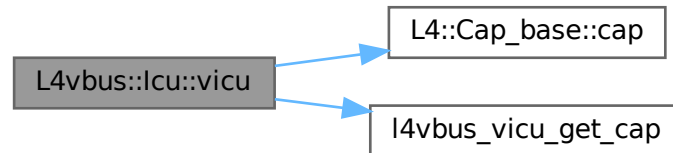
Return values

0	Success.
otherwise	IPC error.

Definition at line 287 of file [vbus](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), [L4::Cap\\_base::cap\(\)](#), and [l4vbus\\_vicu\\_get\\_cap\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

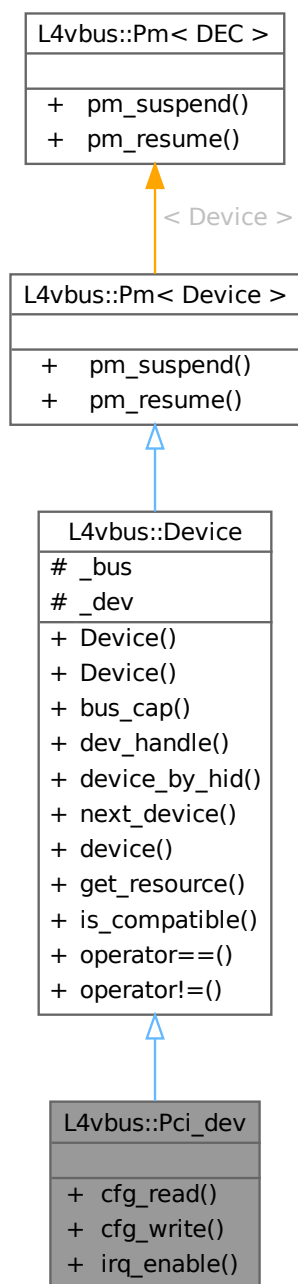
- `l4/vbus/vbus`

## 15.370 L4vbus::Pci\_dev Class Reference

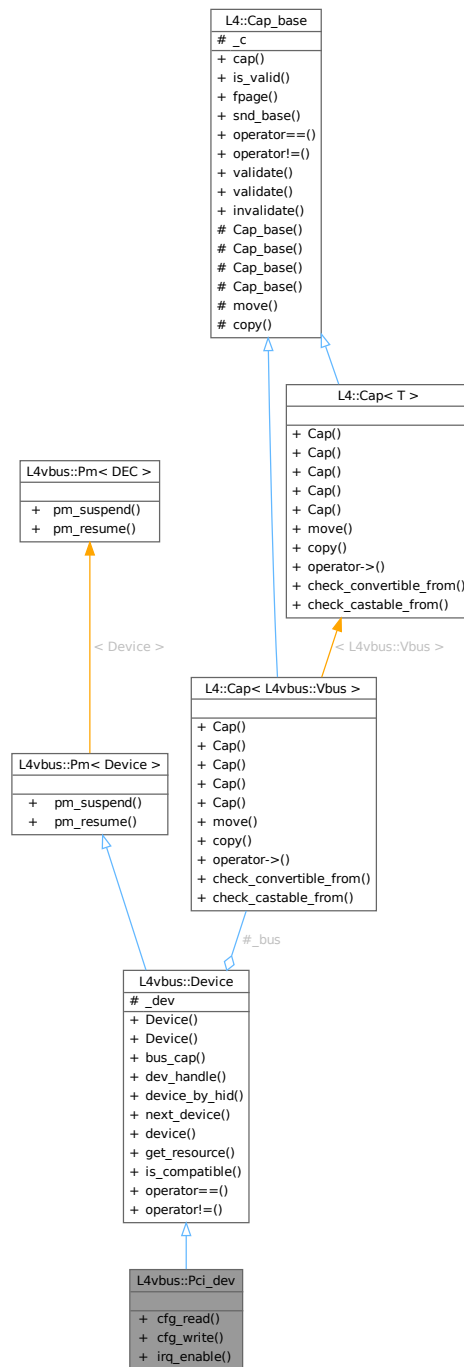
A PCI device.

```
#include <vbus_pci>
```

Inheritance diagram for L4vbus::Pci\_dev:



Collaboration diagram for L4vbus::Pci\_dev:



## Public Member Functions

- `int cfg_read (l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width) const`  
Read from the device's vPCI configuration space.
- `int cfg_write (l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width) const`  
Write to the device's vPCI configuration space.
- `int irq_enable (unsigned char *trigger, unsigned char *polarity) const`  
Enable the device's PCI interrupt.



## Public Member Functions inherited from L4vbus::Device

- **Device** ()  
*Construct a new vbus device using the NULL device `L4VBUS_NULL`.*
- **Device** (L4::Cap< Vbus > bus, l4vbus\_device\_handle\_t dev)  
*Construct a new vbus device using a device handle.*
- **L4::Cap< Vbus > bus\_cap** () const  
*Access the `Vbus` capability of the underlying virtual bus.*
- **l4vbus\_device\_handle\_t dev\_handle** () const  
*Access the device handle of this device.*
- **int device\_by\_hid** (Device \*child, char const \*hid, int depth=L4VBUS\_MAX\_DEPTH, l4vbus\_device\_t \*devinfo=0) const  
*Find a device by the hardware interface identifier (HID).*
- **int next\_device** (Device \*child, int depth=L4VBUS\_MAX\_DEPTH, l4vbus\_device\_t \*devinfo=0) const  
*Find next child following `child`.*
- **int device** (l4vbus\_device\_t \*devinfo) const  
*Obtain detailed information about a `Vbus` device.*
- **int get\_resource** (unsigned res\_idx, l4vbus\_resource\_t \*res) const  
*Obtain the resource description of an individual device resource.*
- **int is\_compatible** (char const \*cid) const  
*Check if the given device has a compatibility ID (CID) or HID that matches `cid`.*
- **bool operator==** (Device const &o) const  
*Test if two devices are the same `Vbus` device.*
- **bool operator!=** (Device const &o) const  
*Test if two `Vbus` devices are not the same.*

## Public Member Functions inherited from L4vbus::Pm< Device >

- **int pm\_suspend** () const  
*Suspend the device.*
- **int pm\_resume** () const  
*Resume the device.*

## Additional Inherited Members

## Protected Attributes inherited from L4vbus::Device

- **L4::Cap< Vbus > \_bus**  
*The `Vbus` capability where this device is located on.*
- **l4vbus\_device\_handle\_t \_dev**  
*The device handle for this device.*

### 15.370.1 Detailed Description

A PCI device.

Definition at line 95 of file `vbus_pci`.

## 15.370.2 Member Function Documentation

### 15.370.2.1 `cfg_read()`

```
int L4vbus::Pci_dev::cfg_read (
    14_uint32_t reg,
    14_uint32_t * value,
    14_uint32_t width ) const [inline]
```

Read from the device's vPCI configuration space.

#### Parameters

	<i>reg</i>	Register in configuration space to read
out	<i>value</i>	Value that has been read
	<i>width</i>	Width to read in bits (e.g. 8, 16, 32)

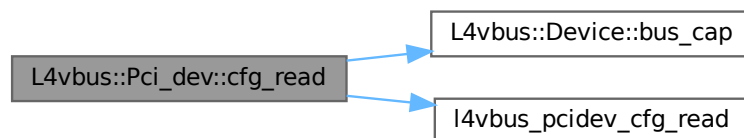
#### Returns

0 on success, else failure

Definition at line 107 of file `vbus_pci`.

References `L4vbus::Device::_dev`, `L4vbus::Device::bus_cap()`, and `l4vbus_pcidev_cfg_read()`.

Here is the call graph for this function:



### 15.370.2.2 `cfg_write()`

```
int L4vbus::Pci_dev::cfg_write (
    14_uint32_t reg,
    14_uint32_t value,
    14_uint32_t width ) const [inline]
```

Write to the device's vPCI configuration space.

#### Parameters

<i>reg</i>	Register in configuration space to write
<i>value</i>	Value to write
<i>width</i>	Width to write in bits (e.g. 8, 16, 32)

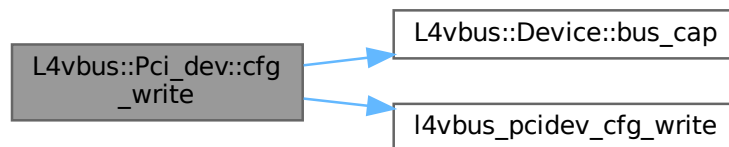
**Returns**

0 on success, else failure

Definition at line 123 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pcidev\\_cfg\\_write\(\)](#).

Here is the call graph for this function:

**15.370.2.3 irq\_enable()**

```
int L4vbus::Pci_dev::irq_enable (
    unsigned char * trigger,
    unsigned char * polarity ) const [inline]
```

Enable the device's PCI interrupt.

**Parameters**

out	<i>trigger</i>	False if interrupt is level-triggered
out	<i>polarity</i>	True if interrupt is of low polarity

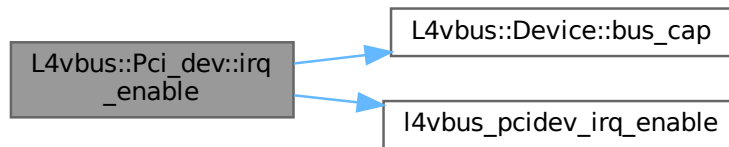
**Returns**

On success: Interrupt line to be used, else failure

Definition at line 139 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pcidev\\_irq\\_enable\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

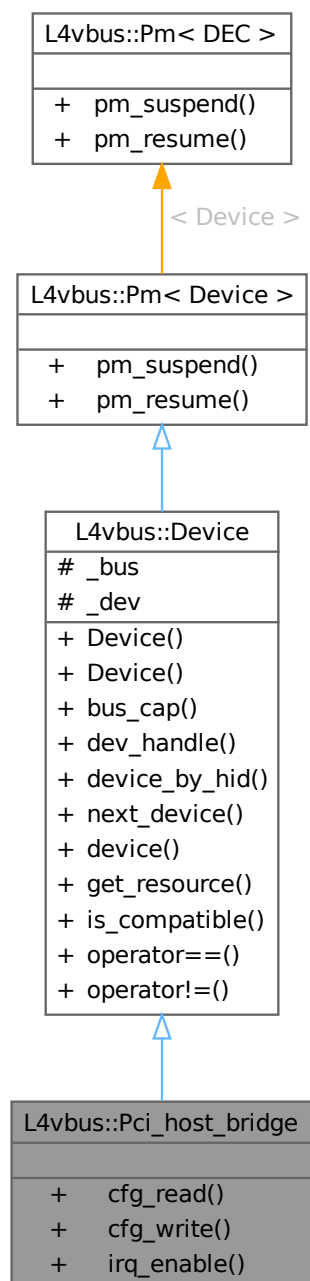
- `l4/vbus/vbus_pci`

## 15.371 L4vbus::Pci\_host\_bridge Class Reference

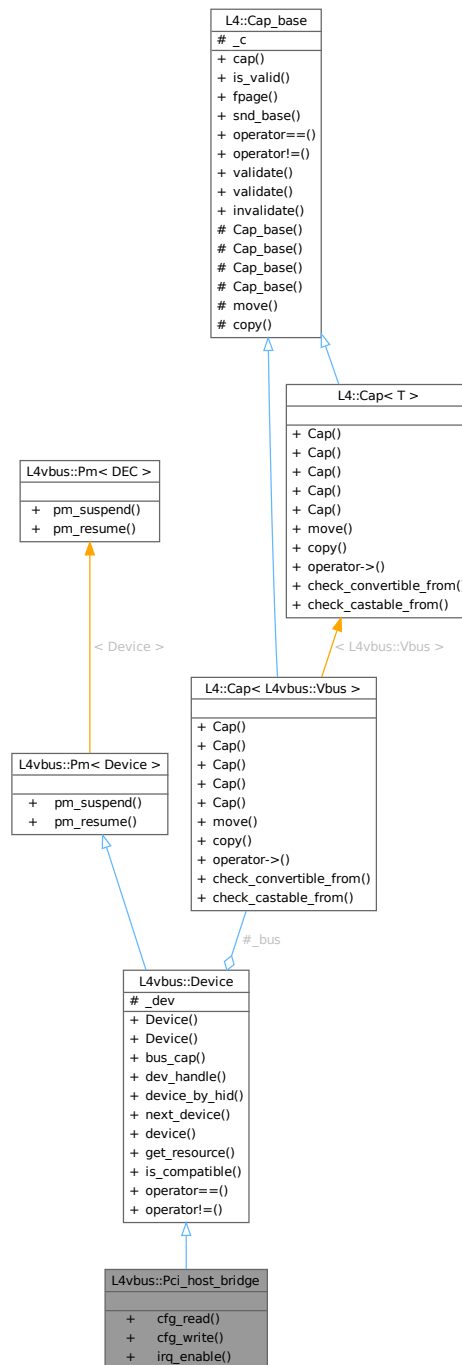
A Pci host bridge.

```
#include <vbus_pci>
```

Inheritance diagram for L4vbus::Pci\_host\_bridge:



Collaboration diagram for L4vbus::Pci\_host\_bridge:



## Public Member Functions

- `int cfg_read (l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width) const`  
Read from the vPCI configuration space using the PCI root bridge.
- `int cfg_write (l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width) const`  
Write to the vPCI configuration space using the PCI root bridge.

- int `irq_enable` (`l4_uint32_t` bus, `l4_uint32_t` devfn, int pin, unsigned char \*trigger, unsigned char \*polarity) const  
*Enable PCI interrupt for a specific device using the PCI root bridge.*

## Public Member Functions inherited from L4vbus::Device

- `Device` ()  
*Construct a new vbus device using the NULL device `L4VBUS_NULL`.*
- `Device` (`L4::Cap`< `Vbus` > bus, `l4vbus_device_handle_t` dev)  
*Construct a new vbus device using a device handle.*
- `L4::Cap`< `Vbus` > `bus_cap` () const  
*Access the `Vbus` capability of the underlying virtual bus.*
- `l4vbus_device_handle_t` `dev_handle` () const  
*Access the device handle of this device.*
- int `device_by_hid` (`Device` \*child, char const \*hid, int depth=L4VBUS\_MAX\_DEPTH, `l4vbus_device_t` \*devinfo=0) const  
*Find a device by the hardware interface identifier (HID).*
- int `next_device` (`Device` \*child, int depth=L4VBUS\_MAX\_DEPTH, `l4vbus_device_t` \*devinfo=0) const  
*Find next child following `child`.*
- int `device` (`l4vbus_device_t` \*devinfo) const  
*Obtain detailed information about a `Vbus` device.*
- int `get_resource` (unsigned res\_idx, `l4vbus_resource_t` \*res) const  
*Obtain the resource description of an individual device resource.*
- int `is_compatible` (char const \*cid) const  
*Check if the given device has a compatibility ID (CID) or HID that matches `cid`.*
- bool `operator==` (`Device` const &o) const  
*Test if two devices are the same `Vbus` device.*
- bool `operator!=` (`Device` const &o) const  
*Test if two `Vbus` devices are not the same.*

## Public Member Functions inherited from L4vbus::Pm< Device >

- int `pm_suspend` () const  
*Suspend the device.*
- int `pm_resume` () const  
*Resume the device.*

## Additional Inherited Members

## Protected Attributes inherited from L4vbus::Device

- `L4::Cap`< `Vbus` > `_bus`  
*The `Vbus` capability where this device is located on.*
- `l4vbus_device_handle_t` `_dev`  
*The device handle for this device.*

### 15.371.1 Detailed Description

A Pci host bridge.

Definition at line 27 of file [vbus\\_pci](#).

### 15.371.2 Member Function Documentation

#### 15.371.2.1 `cfg_read()`

```
int L4vbus::Pci_host_bridge::cfg_read (
    14_uint32_t bus,
    14_uint32_t devfn,
    14_uint32_t reg,
    14_uint32_t * value,
    14_uint32_t width ) const [inline]
```

Read from the vPCI configuration space using the PCI root bridge.

#### Parameters

	<i>bus</i>	Bus number
	<i>devfn</i>	<a href="#">Device</a> id (upper 16bit) and function (lower 16bit)
	<i>reg</i>	Register in configuration space to read
out	<i>value</i>	Value that has been read
	<i>width</i>	Width to read in bits (e.g. 8, 16, 32)

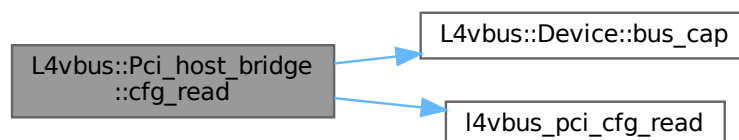
#### Returns

0 on success, else failure

Definition at line 41 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pci\\_cfg\\_read\(\)](#).

Here is the call graph for this function:





### 15.371.2.2 `cfg_write()`

```
int L4vbus::Pci_host_bridge::cfg_write (
    14_uint32_t bus,
    14_uint32_t devfn,
    14_uint32_t reg,
    14_uint32_t value,
    14_uint32_t width ) const [inline]
```

Write to the vPCI configuration space using the PCI root bridge.

#### Parameters

<i>bus</i>	Bus number
<i>devfn</i>	<a href="#">Device</a> id (upper 16bit) and function (lower 16bit)
<i>reg</i>	Register in configuration space to write
<i>value</i>	Value to write
<i>width</i>	Width to write in bits (e.g. 8, 16, 32)

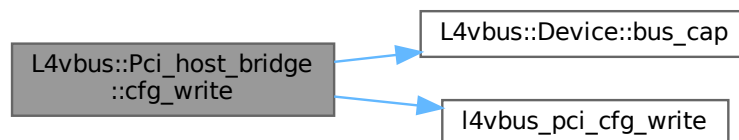
#### Returns

0 on success, else failure

Definition at line 60 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pci\\_cfg\\_write\(\)](#).

Here is the call graph for this function:



### 15.371.2.3 `irq_enable()`

```
int L4vbus::Pci_host_bridge::irq_enable (
    14_uint32_t bus,
    14_uint32_t devfn,
    int pin,
    unsigned char * trigger,
    unsigned char * polarity ) const [inline]
```

Enable PCI interrupt for a specific device using the PCI root bridge.

## Parameters

	<i>bus</i>	Bus number
	<i>devfn</i>	<a href="#">Device</a> id (upper 16bit) and function (lower 16bit)
	<i>pin</i>	Interrupt pin (normally as reported in configuration register INTR)
out	<i>trigger</i>	False if interrupt is level-triggered
out	<i>polarity</i>	True if interrupt is of low polarity

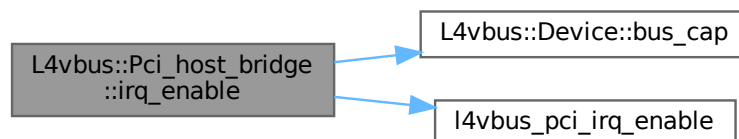
## Returns

On success: Interrupt line to be used, else failure

Definition at line 81 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pci\\_irq\\_enable\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

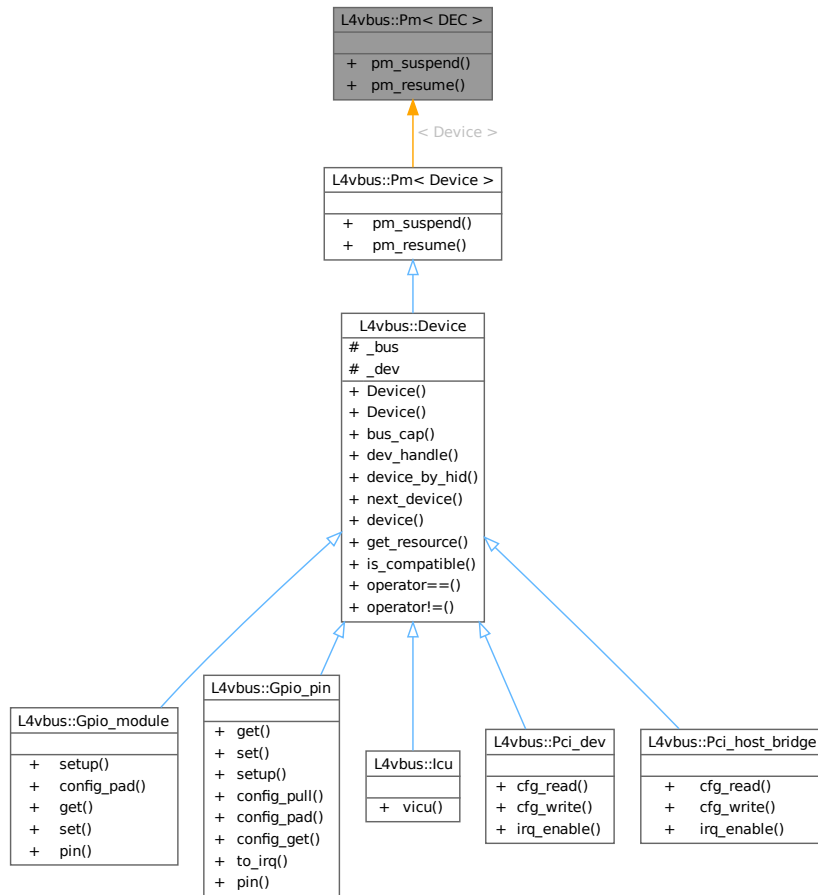
- `I4/vbus/vbus_pci`

## 15.372 L4vbus::Pm< DEC > Class Template Reference

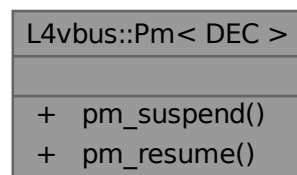
Power-management API mixin.

```
#include <vbus>
```

Inheritance diagram for L4vbus::Pm< DEC >:



Collaboration diagram for L4vbus::Pm< DEC >:



## Public Member Functions

- int [pm\\_suspend](#) () const  
*Suspend the device.*
- int [pm\\_resume](#) () const  
*Resume the device.*

### 15.372.1 Detailed Description

```
template<typename DEC>
class L4vbus::Pm< DEC >
```

Power-management API mixin.

Devices that inherit from this mixin provide an API to be suspended and resumed.

Definition at line 52 of file [vbus](#).

### 15.372.2 Member Function Documentation

#### 15.372.2.1 pm\_resume()

```
template<typename DEC >
int L4vbus::Pm< DEC >::pm_resume ( ) const [inline]
```

Resume the device.

Switches the device from low-power mode to normal operation and restores the saved state.

**Return values**

0	Success.
---	----------

Definition at line 76 of file [vbus](#).

References [l4vbus\\_pm\\_resume\(\)](#).

Here is the call graph for this function:



#### 15.372.2.2 pm\_suspend()

```
template<typename DEC >
int L4vbus::Pm< DEC >::pm_suspend ( ) const [inline]
```

Suspend the device.

Saves the state of the device and puts it into a low-power mode.

## Return values

0	Success.
---	----------

Definition at line 65 of file [vbus](#).

References [l4vbus\\_pm\\_suspend\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

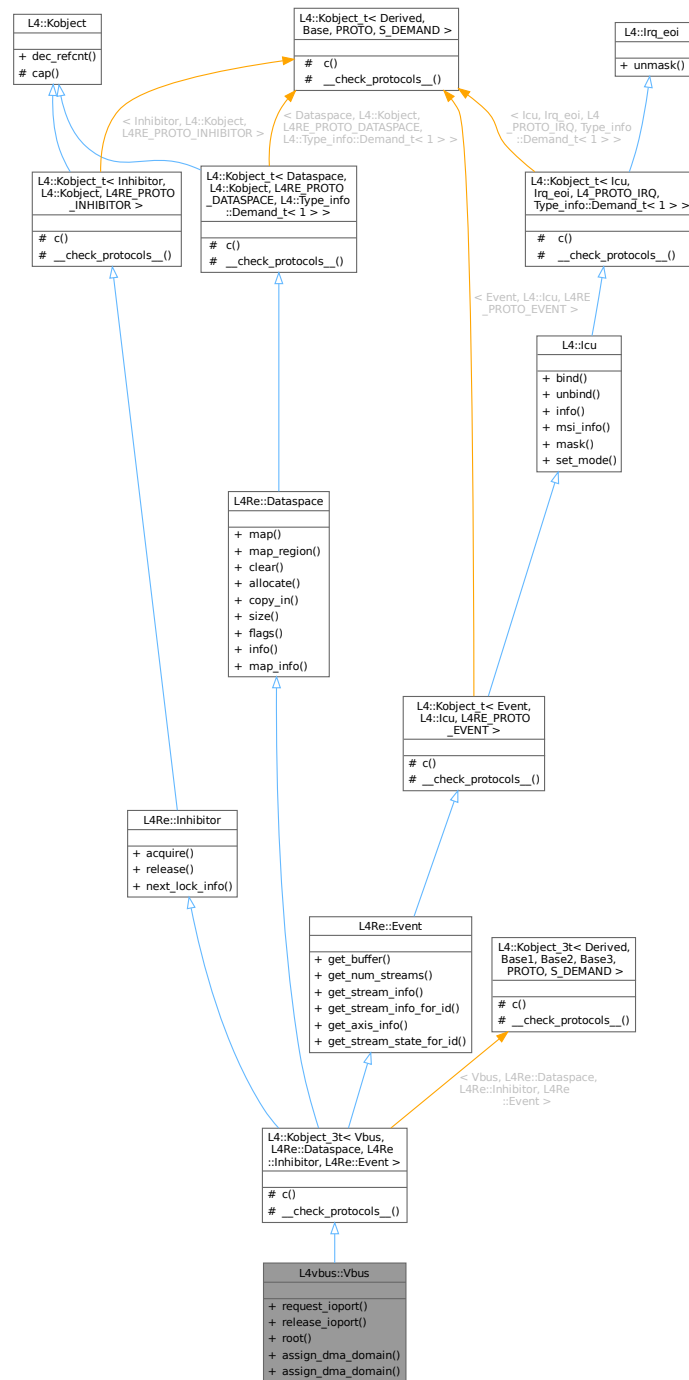
- `l4/vbus/vbus`

## 15.373 L4vbus::Vbus Class Reference

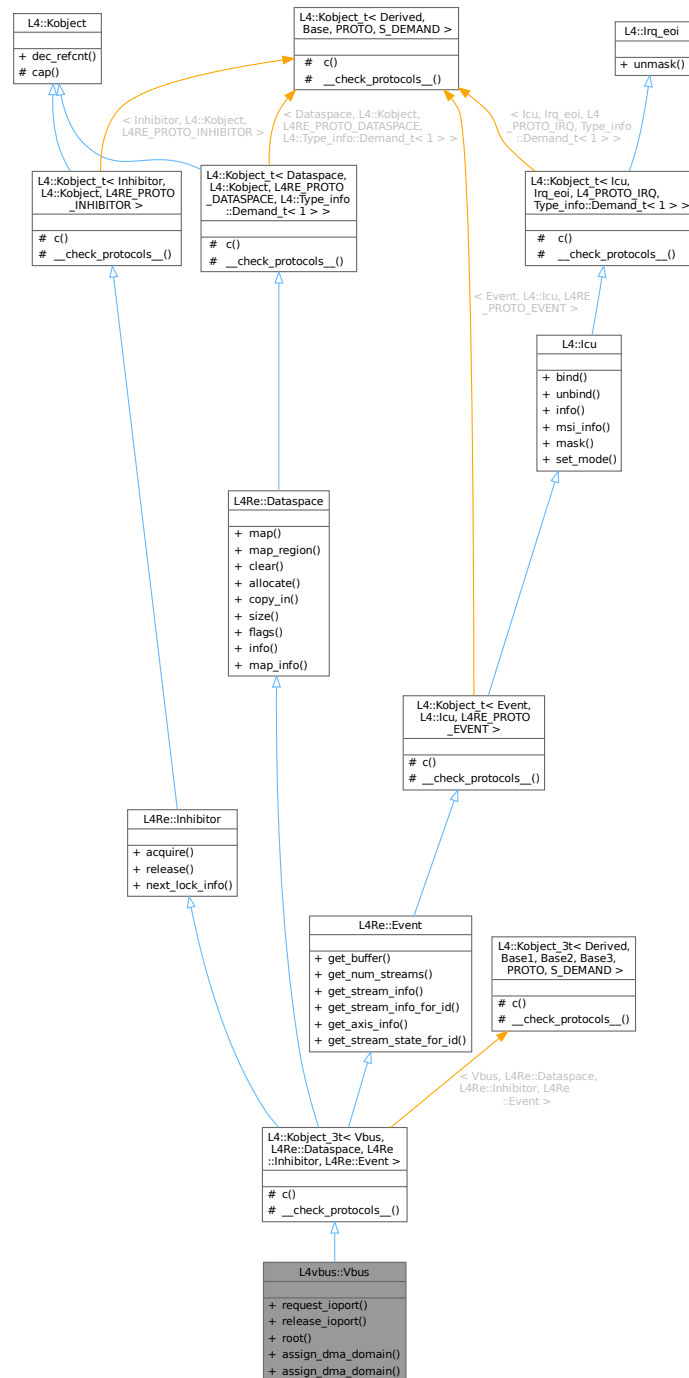
The virtual bus ([Vbus](#)) interface.

```
#include <vbus>
```

Inheritance diagram for L4vbus::Vbus:



Collaboration diagram for L4vbus::Vbus:



## Public Member Functions

- `int request_ioport (l4vbus_resource_t *res) const`  
Request the given IO port resource from the bus.
- `int release_ioport (l4vbus_resource_t *res) const`  
Release the given IO port resource from the bus.
- `Device root () const`

*Get the root device of the device tree of this bus.*

- int `assign_dma_domain` (unsigned domain\_id, unsigned flags, L4::Cap< L4Re::Dma\_space > dma\_space) const

*Bind or unbind an L4Re::Dma\_space to a DMA domain.*

- int `assign_dma_domain` (unsigned domain\_id, unsigned flags, L4::Cap< L4::Task > dma\_space) const

*Bind or unbind a kernel DMA space to a DMA domain.*

## Public Member Functions inherited from L4Re::Dataspace

- long `map` (Offset offset, Flags flags, Map\_addr local\_addr, Map\_addr min\_addr, Map\_addr max\_addr, L4::Cap< L4::Task > dst=L4::Cap< L4::Task >::Invalid) const noexcept

*Request a flex-page mapping from the dataspace.*

- long `map_region` (Offset offset, Flags flags, Map\_addr min\_addr, Map\_addr max\_addr, L4::Cap< L4::Task > dst=L4::Cap< L4::Task >::Invalid) const noexcept

*Map a part of a dataspace into a local memory area.*

- long `clear` (Offset offset, Size size)

*Clear parts of a dataspace.*

- long `allocate` (Offset offset, Size size)

*Allocate a range in the dataspace.*

- long `copy_in` (Offset dst\_offs, L4::lpc::Cap< Dataspace > src, Offset src\_offs, Size size)

*Copy contents from another dataspace.*

- Size `size` () const noexcept

*Get size of a dataspace.*

- Flags `flags` () const noexcept

*Get flags of the dataspace.*

- long `info` (Stats \*stats)

*Get information on the dataspace.*

- long `map_info` (l4\_addr\_t \*, l4\_addr\_t \*)

*Get mapping range of dataspace.*

## Public Member Functions inherited from L4::Kobject

- l4\_msgtag\_t `dec_refcnt` (l4\_mword\_t diff, l4\_utcb\_t \*utcb=l4\_utcb())

*Decrement the in kernel reference counter for the object.*

## Public Member Functions inherited from L4Re::Inhibitor

- long `acquire` (l4\_umword\_t id, L4::lpc::String<> reason)

*Acquire a specific inhibitor lock.*

- long `release` (l4\_umword\_t id)

*Release a specific inhibitor lock.*

- long `next_lock_info` (char \*name, unsigned len, l4\_mword\_t current\_id=-1, l4\_utcb\_t \*utcb=l4\_utcb())

*Get information for the next available inhibitor lock.*



## Public Member Functions inherited from L4Re::Event

- long `get_buffer` (L4::lpc::Out< L4::Cap< Dataspace > > ds)  
*Get event signal buffer.*
- long `get_num_streams` ()  
*Get number of event streams.*
- long `get_stream_info` (int idx, Event\_stream\_info \*info)  
*Get event stream infos.*
- long `get_stream_info_for_id` (l4\_umword\_t stream\_id, Event\_stream\_info \*info)  
*Get event stream infos.*
- long `get_axis_info` (l4\_umword\_t stream\_id, unsigned naxes, unsigned const \*axis, Event\_absinfo \*info) const noexcept  
*Get event stream axis infos.*
- long `get_stream_state_for_id` (l4\_umword\_t stream\_id, Event\_stream\_state \*state)  
*Get event stream state.*

## Public Member Functions inherited from L4::Icu

- `l4_msgtag_t bind` (unsigned irqnum, L4::Cap< Triggerable > irq, l4\_utcb\_t \*utcb=l4\_utcb()) noexcept  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- `l4_msgtag_t unbind` (unsigned irqnum, L4::Cap< Triggerable > irq, l4\_utcb\_t \*utcb=l4\_utcb()) noexcept  
*Remove binding of an interrupt line from the interrupt controller object.*
- `l4_msgtag_t info` (l4\_icu\_info\_t \*info, l4\_utcb\_t \*utcb=l4\_utcb()) noexcept  
*Get information about the ICU features.*
- `l4_msgtag_t msi_info` (l4\_umword\_t irqnum, l4\_uint64\_t source, l4\_icu\_msi\_info\_t \*msi\_info)  
*Get MSI info about IRQ.*
- `l4_msgtag_t mask` (unsigned irqnum, l4\_umword\_t \*label=0, l4\_timeout\_t to=L4\_IPC\_NEVER, l4\_utcb\_t \*utcb=l4\_utcb()) noexcept  
*Mask an IRQ line.*
- `l4_msgtag_t set_mode` (unsigned irqnum, l4\_umword\_t mode, l4\_utcb\_t \*utcb=l4\_utcb()) noexcept  
*Set interrupt mode.*

## Public Member Functions inherited from L4::Irq\_eoi

- `l4_msgtag_t unmask` (unsigned irqnum, l4\_umword\_t \*label=0, l4\_timeout\_t to=L4\_IPC\_NEVER, l4\_utcb\_t \*utcb=l4\_utcb()) noexcept  
*Unmask the given interrupt line.*

## Additional Inherited Members

## Public Types inherited from L4Re::Inhibitor

- enum { `Name_max` = 20 }

**Protected Types inherited from****L4::Kobject\_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >**

- typedef Vbus **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO\_ANY, Vbus > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, Typeid::Merge\_list< typename Base1::\_\_Iface\_list, Typeid::Merge\_list< typename Base2::\_\_Iface\_list, typename Base3::\_\_Iface\_list > > > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

**Protected Types inherited from****L4::Kobject\_t< Dataspace, L4::Kobject, L4RE\_PROTO\_DATASPACE, L4::Type\_info::Demand\_t< 1 > >**

- typedef Dataspace **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, Dataspace > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

**Protected Types inherited from****L4::Kobject\_t< Inhibitor, L4::Kobject, L4RE\_PROTO\_INHIBITOR >**

- typedef Inhibitor **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, Inhibitor > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

**Protected Types inherited from [L4::Kobject\\_t< Event, L4::lcu, L4RE\\_PROTO\\_EVENT >](#)**

- typedef Event **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, Event > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

**Protected Types inherited from****L4::Kobject\_t< lcu, Irq\_eoi, L4\_PROTO\_IRQ, Type\_info::Demand\_t< 1 > >**

- typedef lcu **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, lcu > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Base::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

**Protected Member Functions inherited from****L4::Kobject\_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >**

- **L4::Cap< Class > c ()** const noexcept  
*Get the capability to ourselves.*

**Protected Member Functions inherited from****L4::Kobject\_t< Dataspace, L4::Kobject, L4RE\_PROTO\_DATASPACE, L4::Type\_info::Demand\_t< 1 > >**

- **L4::Cap< Class > c ()** const noexcept  
*Get the capability to ourselves.*

**Protected Member Functions inherited from L4::Kobject**

- **l4\_cap\_idx\_t cap ()** const noexcept  
*Return capability selector.*

**Protected Member Functions inherited from****L4::Kobject\_t< Inhibitor, L4::Kobject, L4RE\_PROTO\_INHIBITOR >**

- **L4::Cap< Class > c ()** const noexcept  
*Get the capability to ourselves.*

**Protected Member Functions inherited from****L4::Kobject\_t< Event, L4::lcu, L4RE\_PROTO\_EVENT >**

- **L4::Cap< Class > c ()** const noexcept  
*Get the capability to ourselves.*

**Protected Member Functions inherited from****L4::Kobject\_t< lcu, Irq\_eoi, L4\_PROTO\_IRQ, Type\_info::Demand\_t< 1 > >**

- **L4::Cap< Class > c ()** const noexcept  
*Get the capability to ourselves.*

**Static Protected Member Functions inherited from****L4::Kobject\_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >**

- static void **\_\_check\_protocols\_\_ ()** noexcept  
*Helper to check for protocol conflicts.*

**Static Protected Member Functions inherited from****L4::Kobject\_t< Dataspace, L4::Kobject, L4RE\_PROTO\_DATASPACE, L4::Type\_info::Demand\_t< 1 > >**

- static void **\_\_check\_protocols\_\_ ()** noexcept  
*Helper to check for protocol conflicts.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< Inhibitor, L4::Kobject, L4RE\\_PROTO\\_INHIBITOR >](#)

- static void `__check_protocols__()` noexcept  
*Helper to check for protocol conflicts.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< Event, L4::lcu, L4RE\\_PROTO\\_EVENT >](#)

- static void `__check_protocols__()` noexcept  
*Helper to check for protocol conflicts.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< lcu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)

- static void `__check_protocols__()` noexcept  
*Helper to check for protocol conflicts.*

## 15.373.1 Detailed Description

The virtual bus ([Vbus](#)) interface.

See also

[L4Re Vbus API](#)

Definition at line [300](#) of file [vbus](#).

## 15.373.2 Member Function Documentation

### 15.373.2.1 `assign_dma_domain()` [1/2]

```
int L4vbus::Vbus::assign_dma_domain (
    unsigned domain_id,
    unsigned flags,
    L4::Cap< L4::Task > dma_space ) const [inline]
```

Bind or unbind a kernel [DMA space](#) to a DMA domain.

#### Parameters

<i>domain_id</i>	DMA domain ID (resource address of DMA domain found on the vBUS). If the value is ~0U the DMA space of the whole vBUS is used.
<i>flags</i>	A combination of <a href="#">L4vbus_dma_domain_assign_flags</a> .
<i>dma_space</i>	The DMA space capability to bind or unbind, this must be a kernel <a href="#">DMA space</a> ( <a href="#">L4::Task</a> created with <code>L4_PROTO_DMA_SPACE</code> )

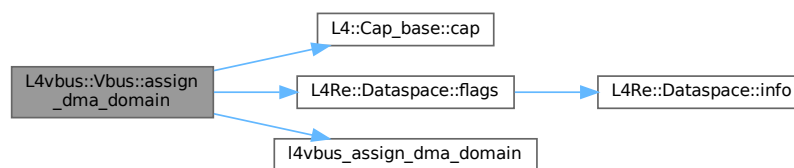
## Return values

0	Operation completed successfully.
-L4_ENOENT	The vbus does not support a global DMA domain or no DMA domain could be found.
-L4_EINVAL	Invalid argument used.
-L4_EBUSY	DMA domain is already active, this means another DMA space is already assigned.

Definition at line 384 of file [vbus](#).

References [L4::Cap\\_base::cap\(\)](#), [L4Re::Dataspace::flags\(\)](#), [l4vbus\\_assign\\_dma\\_domain\(\)](#), and [L4VBUS\\_DMAD\\_KERNEL\\_DMA\\_SP](#)

Here is the call graph for this function:



## 15.373.2.2 assign\_dma\_domain() [2/2]

```

int L4vbus::Vbus::assign_dma_domain (
    unsigned domain_id,
    unsigned flags,
    L4::Cap< L4Re::Dma_space > dma_space ) const [inline]

```

Bind or unbind an [L4Re::Dma\\_space](#) to a DMA domain.

## Parameters

<i>domain_id</i>	DMA domain ID (resource address of DMA domain found on the vBUS). If the value is ~0U the DMA space of the whole vBUS is used.
<i>flags</i>	A combination of <a href="#">L4vbus_dma_domain_assign_flags</a> .
<i>dma_space</i>	The DMA space capability to bind or unbind, this must be an <a href="#">L4Re::Dma_space</a>

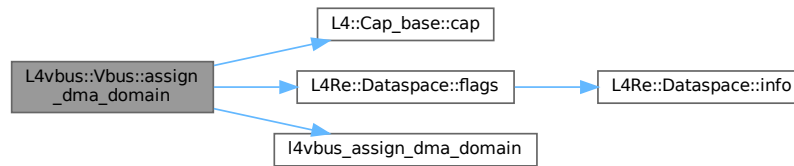
## Return values

0	Operation completed successfully.
-L4_ENOENT	The vbus does not support a global DMA domain or no DMA domain could be found.
-L4_EINVAL	Invalid argument used.
-L4_EBUSY	DMA domain is already active, this means another DMA space is already assigned.

Definition at line 359 of file [vbus](#).

References [L4::Cap\\_base::cap\(\)](#), [L4Re::Dataspace::flags\(\)](#), [l4vbus\\_assign\\_dma\\_domain\(\)](#), and [L4VBUS\\_DMAD\\_L4RE\\_DMA\\_SPAC](#)

Here is the call graph for this function:



### 15.373.2.3 release\_ioport()

```
int L4vbus::Vbus::release_ioport (
    l4vbus_resource_t * res ) const [inline]
```

Release the given IO port resource from the bus.

#### Parameters

<code>in</code>	<code>res</code>	The IO port resource to be released from the bus.
-----------------	------------------	---

#### Returns

$\geq 0$  on success,  $< 0$  on error.

Definition at line 325 of file `vbus`.

References `l4vbus_release_ioport()`.

Here is the call graph for this function:



### 15.373.2.4 request\_ioport()

```
int L4vbus::Vbus::request_ioport (
    l4vbus_resource_t * res ) const [inline]
```

Request the given IO port resource from the bus.

## Parameters

<i>in</i>	<i>res</i>	The IO port resource to be requested from the bus.
-----------	------------	--

## Return values

<i>0</i>	Success.
<i>-L4_EINVAL</i>	Resource is not an IO port resource.
<i>-L4_ENOENT</i>	No matching IO port resource found.

Definition at line 313 of file [vbus](#).

References [l4vbus\\_request\\_ioport\(\)](#).

Here is the call graph for this function:



### 15.373.2.5 root()

```
Device L4vbus::Vbus::root ( ) const [inline]
```

Get the root device of the device tree of this bus.

The root device is usually the starting point for iterating the bus, see [Device::next\\_device](#).

## Returns

A [Vbus](#) device representing the root of the device tree.

Definition at line 338 of file [vbus](#).

References [L4VBUS\\_ROOT\\_BUS](#).

The documentation for this class was generated from the following file:

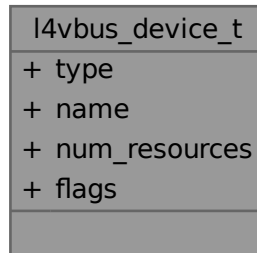
- `l4/vbus/vbus`

## 15.374 l4vbus\_device\_t Struct Reference

Detailed information about a vbus device.

```
#include <vbus_types.h>
```

Collaboration diagram for l4vbus\_device\_t:



### Data Fields

- [l4\\_uint32\\_t](#) **type**  
*Bitfield of supported sub-interfaces, see [l4vbus\\_iface\\_type\\_t](#).*
- char **name** [L4VBUS\_DEV\_NAME\_LEN]  
*Name.*
- unsigned **num\_resources**  
*Number of resources for this device.*
- unsigned **flags**  
*Flags, see [l4vbus\\_device\\_flags\\_t](#).*

### 15.374.1 Detailed Description

Detailed information about a vbus device.

Definition at line 70 of file [vbus\\_types.h](#).

The documentation for this struct was generated from the following file:

- [l4/vbus/vbus\\_types.h](#)



## 15.375 l4vbus\_resource\_t Struct Reference

Description of a single vbus resource.

```
#include <vbus_types.h>
```

Collaboration diagram for l4vbus\_resource\_t:

l4vbus_resource_t	
+	type
+	flags
+	start
+	end
+	provider
+	id

### Data Fields

- [l4\\_uint16\\_t type](#)  
*Resource type, see [l4vbus\\_resource\\_type\\_t](#).*
- [l4\\_uint16\\_t flags](#)  
*Flags.*
- [l4vbus\\_paddr\\_t start](#)  
*Start of resource range.*
- [l4vbus\\_paddr\\_t end](#)  
*End of resource range (inclusive)*
- [l4vbus\\_device\\_handle\\_t provider](#)  
*Device handle of the provider of the resource.*
- [l4\\_uint32\\_t id](#)  
*Resource ID (4 bytes), usually a 4 letter ASCII name is used.*

### 15.375.1 Detailed Description

Description of a single vbus resource.

Definition at line 25 of file [vbus\\_types.h](#).

The documentation for this struct was generated from the following file:

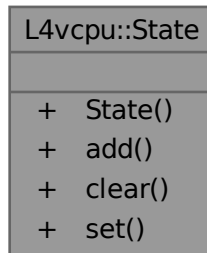
- [l4/vbus/vbus\\_types.h](#)

## 15.376 L4vcpu::State Class Reference

C++ implementation of state word in the vCPU area.

```
#include <vcpu>
```

Collaboration diagram for L4vcpu::State:



### Public Member Functions

- [State](#) (unsigned v)  
*Initialize state.*
- void [add](#) (unsigned bits) throw ()  
*Add flags.*
- void [clear](#) (unsigned bits) throw ()  
*Clear flags.*
- void [set](#) (unsigned v) throw ()  
*Set flags.*

### 15.376.1 Detailed Description

C++ implementation of state word in the vCPU area.

Definition at line [35](#) of file [vcpu](#).

### 15.376.2 Constructor & Destructor Documentation

#### 15.376.2.1 State()

```
L4vcpu::State::State (
    unsigned v ) [inline], [explicit]
```

Initialize state.

## Parameters

<i>v</i>	Initial state.
----------	----------------

Definition at line 45 of file [vcpu](#).

## 15.376.3 Member Function Documentation

### 15.376.3.1 add()

```
void L4vcpu::State::add (
    unsigned bits ) throw ( )    [inline]
```

Add flags.

## Parameters

<i>bits</i>	Bits to add to the word.
-------------	--------------------------

Definition at line 52 of file [vcpu](#).

### 15.376.3.2 clear()

```
void L4vcpu::State::clear (
    unsigned bits ) throw ( )    [inline]
```

Clear flags.

## Parameters

<i>bits</i>	Bits to clear in the word.
-------------	----------------------------

Definition at line 59 of file [vcpu](#).

### 15.376.3.3 set()

```
void L4vcpu::State::set (
    unsigned v ) throw ( )    [inline]
```

Set flags.

## Parameters

<i>v</i>	Set the word to the value of <i>v</i> .
----------	---

Definition at line 66 of file [vcpu](#).

The documentation for this class was generated from the following file:

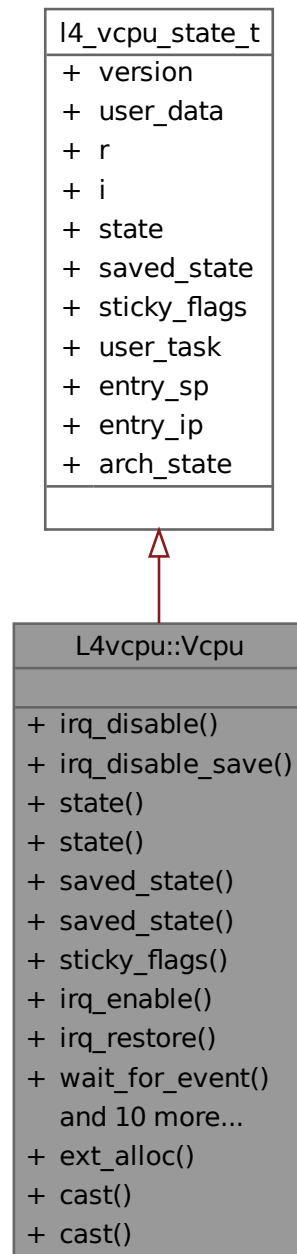
- [l4/vcpu/vcpu](#)

## 15.377 L4vcpu::Vcpu Class Reference

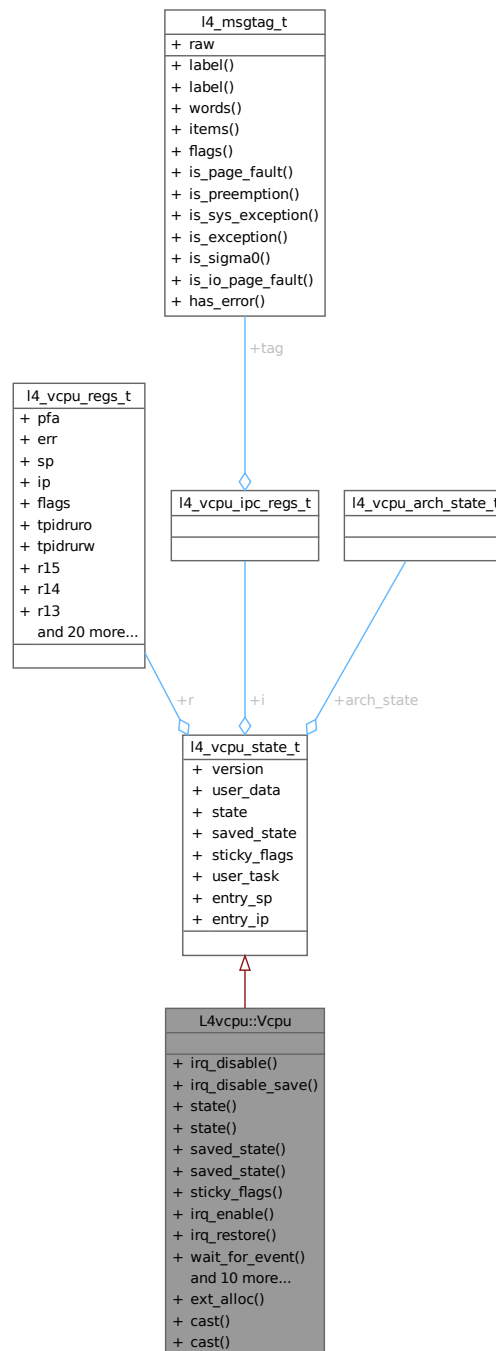
C++ implementation of the vCPU save state area.

```
#include <vcpu>
```

Inheritance diagram for L4vcpu::Vcpu:



Collaboration diagram for L4vcpu::Vcpu:



## Public Member Functions

- void **irq\_disable** () throw ()  
*Disable the vCPU for event delivery.*
- unsigned **irq\_disable\_save** () throw ()  
*Disable the vCPU for event delivery and return previous state.*
- **State** \* **state** () throw ()

- Get state word.*

  - `State state () const throw ()`
- Get state word.*

  - `State * saved_state () throw ()`
- Get saved\_state word.*

  - `State saved_state () const throw ()`
- Get saved\_state word.*

  - `l4_uint16_t sticky_flags () const throw ()`
- Get sticky flags.*

  - `void irq_enable (l4_utcb_t *utcb, l4vcpu_event_hndl_t do_event_work_cb, l4vcpu_setup_ipc_t setup_ipc) throw ()`
- Enable the vCPU for event delivery.*

  - `void irq_restore (unsigned s, l4_utcb_t *utcb, l4vcpu_event_hndl_t do_event_work_cb, l4vcpu_setup_ipc_t setup_ipc) throw ()`
- Restore a previously saved IRQ/event state.*

  - `void wait_for_event (l4_utcb_t *utcb, l4vcpu_event_hndl_t do_event_work_cb, l4vcpu_setup_ipc_t setup_ipc) throw ()`
- Wait for event.*

  - `void task (L4::Cap< L4::Task > const task=L4::Cap< L4::Task >::Invalid) throw ()`
- Set the task of the vCPU.*

  - `int is_page_fault_entry () const`
- Return whether the entry reason was a page fault.*

  - `int is_irq_entry () const`
- Return whether the entry reason was an IRQ/IPC message.*

  - `l4_vcpu_regs_t * r () throw ()`
- Return pointer to register state.*

  - `l4_vcpu_regs_t const * r () const throw ()`
- Return pointer to register state.*

  - `l4_vcpu_ipc_regs_t * i () throw ()`
- Return pointer to IPC state.*

  - `l4_vcpu_ipc_regs_t const * i () const throw ()`
- Return pointer to IPC state.*

  - `void entry_sp (l4_umword_t sp)`
- Set vCPU entry stack pointer.*

  - `void entry_ip (l4_umword_t ip)`
- Set vCPU entry instruction pointer.*

  - `void print_state (const char *prefix="") const throw ()`
- Print the state of the vCPU.*

### Static Public Member Functions

- `static int ext_alloc (Vcpu **vcpu, l4_addr_t *ext_state, L4::Cap< L4::Task > task=L4Re::Env::env() ->task(), L4::Cap< L4Re::Rm > rm=L4Re::Env::env() ->rm()) throw ()`

*Allocate state area for an extended vCPU.*
- `static Vcpu * cast (void *x) throw ()`

*Cast a void pointer to a class pointer.*
- `static Vcpu * cast (l4_addr_t x) throw ()`

*Cast an address to a class pointer.*

## 15.377.1 Detailed Description

C++ implementation of the vCPU save state area.

Definition at line 76 of file [vcpu](#).

## 15.377.2 Member Function Documentation

### 15.377.2.1 `cast()` [1/2]

```
static Vcpu * L4vcpu::Vcpu::cast (
    l4\_addr\_t x ) throw ( )    [inline], [static]
```

Cast an address to a class pointer.

#### Parameters

<code>x</code>	Pointer.
----------------	----------

#### Returns

Pointer to [Vcpu](#) class.

Definition at line 280 of file [vcpu](#).

### 15.377.2.2 `cast()` [2/2]

```
static Vcpu * L4vcpu::Vcpu::cast (
    void * x ) throw ( )    [inline], [static]
```

Cast a void pointer to a class pointer.

#### Parameters

<code>x</code>	Pointer.
----------------	----------

#### Returns

Pointer to [Vcpu](#) class.

Definition at line 270 of file [vcpu](#).

### 15.377.2.3 `entry_ip()`

```
void L4vcpu::Vcpu::entry_ip (
    l4\_umword\_t ip )    [inline]
```

Set vCPU entry instruction pointer.

## Parameters

<i>ip</i>	Instruction pointer address to set.
-----------	-------------------------------------

Definition at line 243 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::entry\\_ip](#).

**15.377.2.4 entry\_sp()**

```
void L4vcpu::Vcpu::entry_sp (
    l4_umword_t sp ) [inline]
```

Set vCPU entry stack pointer.

## Parameters

<i>sp</i>	Stack pointer address to set.
-----------	-------------------------------

## Note

The value is only used when entering from a user-task.

Definition at line 236 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::entry\\_sp](#).

**15.377.2.5 ext\_alloc()**

```
static int L4vcpu::Vcpu::ext_alloc (
    Vcpu ** vcpu,
    l4_addr_t * ext_state,
    L4::Cap< L4::Task > task = L4Re::Env::env() ->task(),
    L4::Cap< L4Re::Rm > rm = L4Re::Env::env() ->rm() ) throw ( ) [static]
```

Allocate state area for an extended vCPU.

## Parameters

out	<i>vcpu</i>	Allocated vcpu-state area.
out	<i>ext_state</i>	Allocated extended vcpu-state area.
	<i>task</i>	Task to use for allocation, defaults to own task.
	<i>rm</i>	Region manager to use for allocation defaults to standard region manager.

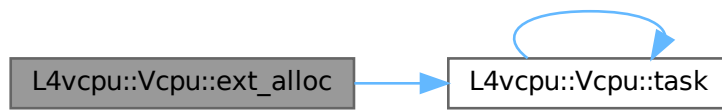
## Returns

0 for success, error code otherwise

References [task\(\)](#).



Here is the call graph for this function:



#### 15.377.2.6 `i()` [1/2]

```
l4_vcpu_ipc_regs_t * L4vcpu::Vcpu::i ( ) throw ( ) [inline]
```

Return pointer to IPC state.

##### Returns

Pointer to IPC state.

Definition at line 220 of file `vcpu`.

References `l4_vcpu_state_t::i`.

#### 15.377.2.7 `i()` [2/2]

```
l4_vcpu_ipc_regs_t const * L4vcpu::Vcpu::i ( ) const throw ( ) [inline]
```

Return pointer to IPC state.

##### Returns

Pointer to IPC state.

Definition at line 227 of file `vcpu`.

References `l4_vcpu_state_t::i`.

#### 15.377.2.8 `irq_disable_save()`

```
unsigned L4vcpu::Vcpu::irq_disable_save ( ) throw ( ) [inline]
```

Disable the vCPU for event delivery and return previous state.

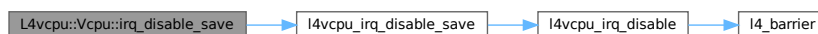
##### Returns

IRQ state before disabling IRQs.

Definition at line 89 of file `vcpu`.

References `l4vcpu_irq_disable_save()`.

Here is the call graph for this function:



### 15.377.2.9 irq\_enable()

```
void L4vcpu::Vcpu::irq_enable (
    l4_utcb_t * utcb,
    l4vcpu_event_hndl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc ) throw ( )    [inline]
```

Enable the vCPU for event delivery.

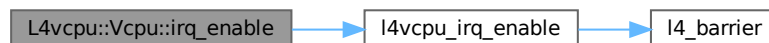
#### Parameters

<i>utcb</i>	The UTCB to use.
<i>do_event_work_cb</i>	Call-back function that is called in case an event (such as an interrupt) is pending.
<i>setup_ipc</i>	Call-back function that is called before an IPC operation is called, and before event delivery is enabled.

Definition at line 146 of file [vcpu](#).

References [l4vcpu\\_irq\\_enable\(\)](#).

Here is the call graph for this function:



### 15.377.2.10 irq\_restore()

```
void L4vcpu::Vcpu::irq_restore (
    unsigned s,
    l4_utcb_t * utcb,
    l4vcpu_event_hndl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc ) throw ( )    [inline]
```

Restore a previously saved IRQ/event state.

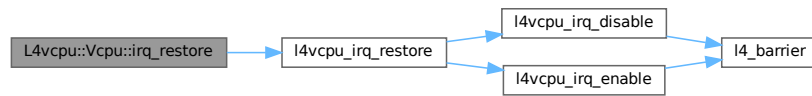
#### Parameters

<i>s</i>	IRQ state to be restored.
<i>utcb</i>	The UTCB to use.
<i>do_event_work_cb</i>	Call-back function that is called in case an event (such as an interrupt) is pending.
<i>setup_ipc</i>	Call-back function that is called before an IPC operation is called, and before event delivery is enabled.

Definition at line 161 of file [vcpu](#).

References [l4vcpu\\_irq\\_restore\(\)](#).

Here is the call graph for this function:



### 15.377.2.11 is\_irq\_entry()

```
int L4vcpu::Vcpu::is_irq_entry ( ) const [inline]
```

Return whether the entry reason was an IRQ/IPC message.

return 0 if not, !=0 otherwise.

Definition at line 199 of file [vcpu](#).

References [l4vcpu\\_is\\_irq\\_entry\(\)](#).

Here is the call graph for this function:



### 15.377.2.12 is\_page\_fault\_entry()

```
int L4vcpu::Vcpu::is_page_fault_entry ( ) const [inline]
```

Return whether the entry reason was a page fault.

return 0 if not, !=0 otherwise.

Definition at line 192 of file [vcpu](#).

References [l4vcpu\\_is\\_page\\_fault\\_entry\(\)](#).

Here is the call graph for this function:



**15.377.2.13 r()** [1/2]

```
l4_vcpu_regs_t * L4vcpu::Vcpu::r ( ) throw ( ) [inline]
```

Return pointer to register state.

**Returns**

Pointer to register state.

Definition at line 206 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::r](#).

**15.377.2.14 r()** [2/2]

```
l4_vcpu_regs_t const * L4vcpu::Vcpu::r ( ) const throw ( ) [inline]
```

Return pointer to register state.

**Returns**

Pointer to register state.

Definition at line 213 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::r](#).

**15.377.2.15 saved\_state()** [1/2]

```
State * L4vcpu::Vcpu::saved_state ( ) throw ( ) [inline]
```

Get saved\_state word.

**Returns**

Pointer to saved\_state word in the vCPU

Definition at line 117 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::saved\\_state](#).

**15.377.2.16 saved\_state()** [2/2]

```
State L4vcpu::Vcpu::saved_state ( ) const throw ( ) [inline]
```

Get saved\_state word.

**Returns**

Pointer to saved\_state word in the vCPU

Definition at line 127 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::saved\\_state](#).

**15.377.2.17 state()** [1/2]

```
State * L4vcpu::Vcpu::state ( ) throw ( ) [inline]
```

Get state word.

**Returns**

Pointer to state word in the vCPU

Definition at line 99 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::state](#).

**15.377.2.18 state()** [2/2]

```
State L4vcpu::Vcpu::state ( ) const throw ( ) [inline]
```

Get state word.

**Returns**

Pointer to state word in the vCPU

Definition at line 110 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::state](#).

**15.377.2.19 task()**

```
void L4vcpu::Vcpu::task (
    L4::Cap< L4::Task > const task = L4::Cap<L4::Task>::Invalid ) throw ( ) [inline]
```

Set the task of the vCPU.

**Parameters**

<i>task</i>	Task to set, defaults to invalid task.
-------------	--

Definition at line 185 of file [vcpu](#).

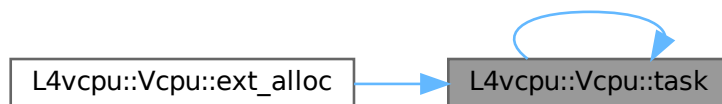
References [task\(\)](#), and [l4\\_vcpu\\_state\\_t::user\\_task](#).

Referenced by [ext\\_alloc\(\)](#), and [task\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.377.2.20 wait\_for\_event()

```

void L4vcpu::Vcpu::wait_for_event (
    l4_utcb_t * utcb,
    l4vcpu_event_hndl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc ) throw ( )    [inline]
  
```

Wait for event.

#### Parameters

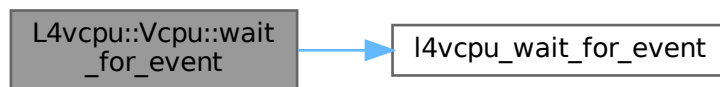
<i>utcb</i>	The UTCB to use.
<i>do_event_work_cb</i>	Call-back function that is called in case an event (such as an interrupt) is pending.
<i>setup_ipc</i>	Call-back function that is called before an IPC operation is called.

Note that event delivery remains disabled after this function returns.

Definition at line 177 of file `vcpu`.

References `l4vcpu_wait_for_event()`.

Here is the call graph for this function:



The documentation for this class was generated from the following file:

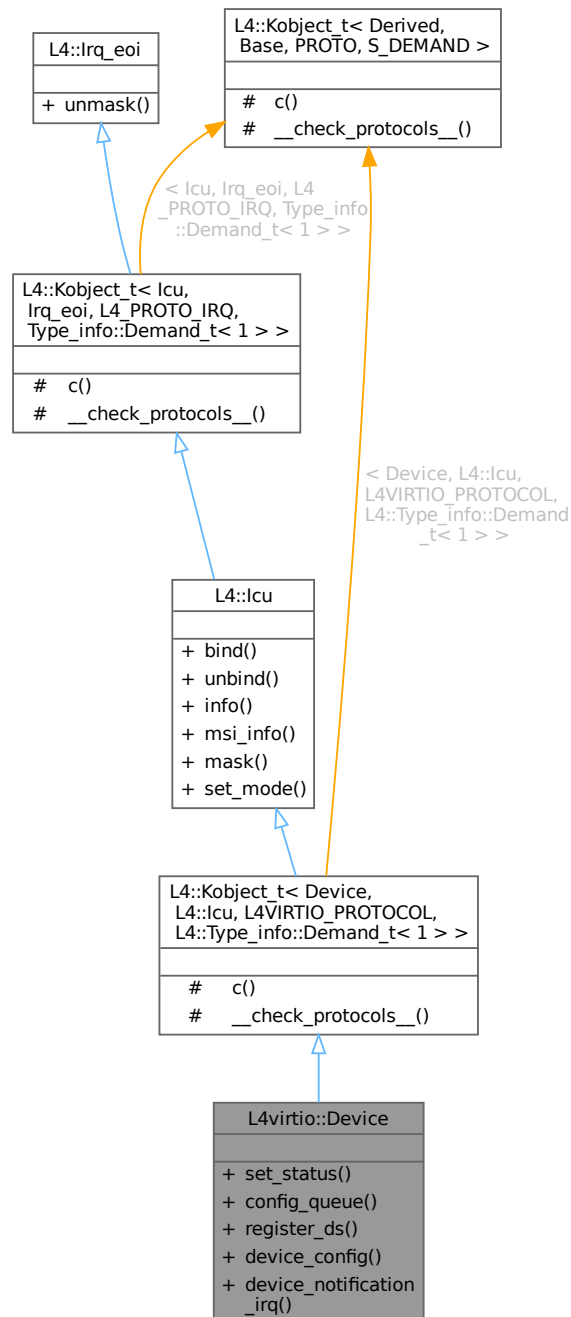
- [l4/vcpu/vcpu](#)

## 15.378 L4virtio::Device Class Reference

IPC interface for virtio over [L4](#) IPC.

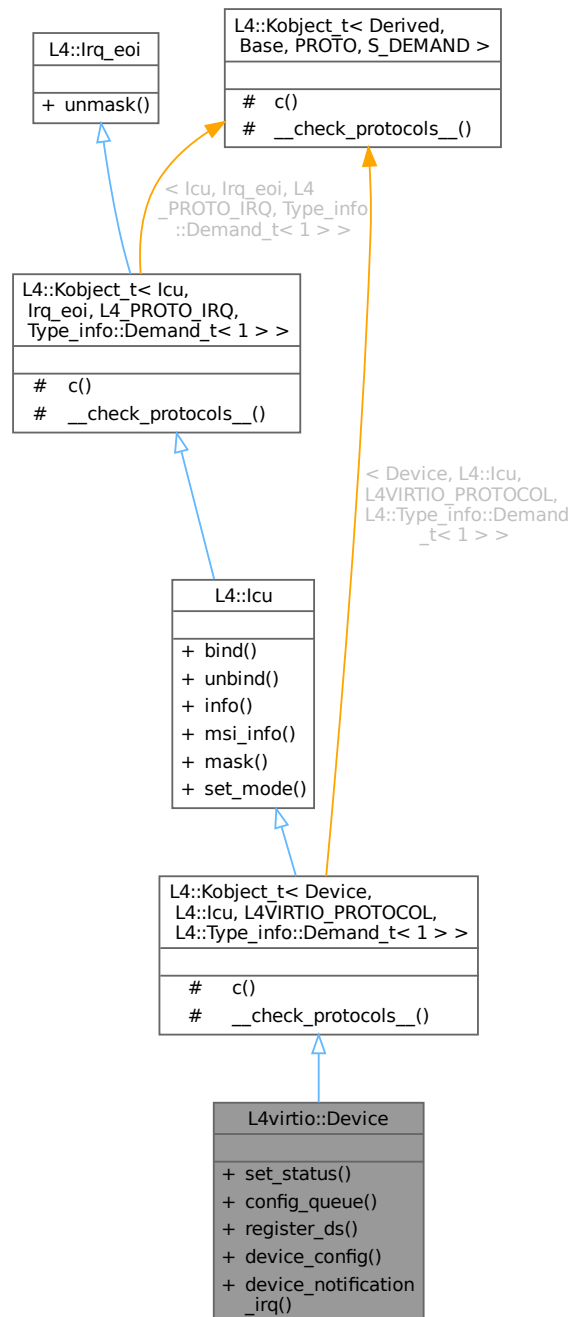
```
#include <l4virtio>
```

Inheritance diagram for L4virtio::Device:





Collaboration diagram for L4virtio::Device:



## Public Member Functions

- long `set_status` (unsigned status)  
*Write the VIRTIO status register.*
- long `config_queue` (unsigned queue)  
*Trigger queue configuration of the given queue.*

- long `register_ds` (`L4::lpc::Cap< L4Re::Dataspace >` ds\_cap, `l4_uint64_t` base, `l4_umword_t` offset, `l4_umword_t` size)  
*Register a shared data space with VIRTIO host.*
- long `device_config` (`L4::lpc::Out< L4::Cap< L4Re::Dataspace > >` config\_ds, `l4_addr_t` \*ds\_offset)  
*Get the dataspace with the [L4virtio](#) configuration page.*
- long `device_notification_irq` (unsigned index, `L4::lpc::Out< L4::Cap< L4::Triggerable > >` irq)  
*Get the notification interrupt corresponding to the given index.*

## Public Member Functions inherited from `L4::lcu`

- `l4_msgtag_t` `bind` (unsigned irqnum, `L4::Cap< Triggerable >` irq, `l4_utcb_t` \*utcb=`l4_utcb()`) noexcept  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- `l4_msgtag_t` `unbind` (unsigned irqnum, `L4::Cap< Triggerable >` irq, `l4_utcb_t` \*utcb=`l4_utcb()`) noexcept  
*Remove binding of an interrupt line from the interrupt controller object.*
- `l4_msgtag_t` `info` (`l4_icu_info_t` \*info, `l4_utcb_t` \*utcb=`l4_utcb()`) noexcept  
*Get information about the ICU features.*
- `l4_msgtag_t` `msi_info` (`l4_umword_t` irqnum, `l4_uint64_t` source, `l4_icu_msi_info_t` \*msi\_info)  
*Get MSI info about IRQ.*
- `l4_msgtag_t` `mask` (unsigned irqnum, `l4_umword_t` \*label=0, `l4_timeout_t` to=`L4_IPC_NEVER`, `l4_utcb_t` \*utcb=`l4_utcb()`) noexcept  
*Mask an IRQ line.*
- `l4_msgtag_t` `set_mode` (unsigned irqnum, `l4_umword_t` mode, `l4_utcb_t` \*utcb=`l4_utcb()`) noexcept  
*Set interrupt mode.*

## Public Member Functions inherited from `L4::lrq_eoi`

- `l4_msgtag_t` `unmask` (unsigned irqnum, `l4_umword_t` \*label=0, `l4_timeout_t` to=`L4_IPC_NEVER`, `l4_utcb_t` \*utcb=`l4_utcb()`) noexcept  
*Unmask the given interrupt line.*

## Additional Inherited Members

## Protected Types inherited from

`L4::Kobject_t< Device, L4::lcu, L4VIRTIO_PROTOCOL, L4::Type_info::Demand_t< 1 > >`

- typedef Device **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, Device > **\_\_iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_iface** >, typename Base::\_\_iface\_list > **\_\_iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Types inherited from

`L4::Kobject_t< lcu, lrq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >`

- typedef lcu **Class**  
*The target interface type (inheriting from [Kobject\\_t](#))*
- typedef Typeid::Iface< PROTO, lcu > **\_\_iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_iface** >, typename Base::\_\_iface\_list > **\_\_iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

**Protected Member Functions inherited from****L4::Kobject\_t< Device, L4::lcu, L4VIRTIO\_PROTOCOL, L4::Type\_info::Demand\_t< 1 > >**

- **L4::Cap< Class > c()** const noexcept  
*Get the capability to ourselves.*

**Protected Member Functions inherited from****L4::Kobject\_t< lcu, lrc\_eoi, L4\_PROTO\_IRQ, Type\_info::Demand\_t< 1 > >**

- **L4::Cap< Class > c()** const noexcept  
*Get the capability to ourselves.*

**Static Protected Member Functions inherited from****L4::Kobject\_t< Device, L4::lcu, L4VIRTIO\_PROTOCOL, L4::Type\_info::Demand\_t< 1 > >**

- static void **\_\_check\_protocols\_\_()** noexcept  
*Helper to check for protocol conflicts.*

**Static Protected Member Functions inherited from****L4::Kobject\_t< lcu, lrc\_eoi, L4\_PROTO\_IRQ, Type\_info::Demand\_t< 1 > >**

- static void **\_\_check\_protocols\_\_()** noexcept  
*Helper to check for protocol conflicts.*

**15.378.1 Detailed Description**

IPC interface for virtio over [L4 IPC](#).

The [L4virtio](#) protocol is an adaption of the mmio virtio transport 1.0(4). This interface allows to exchange the necessary resources: device configuration page, notification interrupts and dataspace for payload.

Notification interrupts can be configured independently for changes to the configuration space and each queue through special L4virtio-specific `notify_index` fields in the config page and queue configuration. The interface distinguishes between device-to-driver and driver-to-device notification interrupts.

Device-to-driver interrupts are configured via the ICU interface. The device announces the maximum number of supported interrupts via `lcu::info()`. The driver can then bind interrupts using `lcu::bind()`.

Driver-to-device interrupts must be requested from the device through [device\\_notification\\_irq\(\)](#).

Definition at line 39 of file [l4virtio](#).

**15.378.2 Member Function Documentation****15.378.2.1 config\_queue()**

```
long L4virtio::Device::config_queue (
    unsigned queue )
```

Trigger queue configuration of the given queue.

Usually all queues are configured when the status is written to running. However, in some cases queues shall be disabled or enabled dynamically, in this case this function triggers a reconfiguration from the shared memory register of the queue config.

## Parameters

<i>queue</i>	Queue index for the queue to be configured.
--------------	---

## Return values

<i>0</i>	on success.
<i>-L4_EIO</i>	The queue's status is invalid.
<i>-L4_ERANGE</i>	The queue index exceeds the number of queues.
<i>-L4_EINVAL</i>	Otherwise.

Referenced by [L4virtio::Driver::Device::config\\_queue\(\)](#).

Here is the caller graph for this function:



### 15.378.2.2 device\_config()

```

long L4virtio::Device::device_config (
    L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > config_ds,
    l4_addr_t * ds_offset )

```

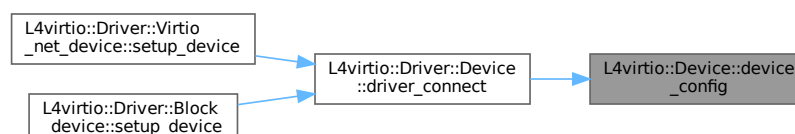
Get the dataspace with the [L4virtio](#) configuration page.

## Parameters

<i>config_ds</i>	Capability for receiving the dataspace capability for the shared L4-VIRTIO config data space.
<i>ds_offset</i>	Offset into the dataspace where the device configuration structure starts.

Referenced by [L4virtio::Driver::Device::driver\\_connect\(\)](#).

Here is the caller graph for this function:



### 15.378.2.3 device\_notification\_irq()

```
long L4virtio::Device::device_notification_irq (
    unsigned index,
    L4::Ipc::Out< L4::Cap< L4::Triggerable > > irq )
```

Get the notification interrupt corresponding to the given index.

#### Parameters

	<i>index</i>	Index of the interrupt.
out	<i>irq</i>	Triggerable for the given index.

#### Return values

<i>L4_EOK</i>	Success.
<i>L4_ENOSYS</i>	IRQ notification not supported by device.
<i>&lt;0</i>	Other error.

An index is only guaranteed to return an IRQ object when the index is set in one of the device notify index fields. The device must return the same interrupt for a given index as long as the index is in use. If an index disappears as a result of a configuration change and then is reused later, the interrupt is not guaranteed to be the same.

Interrupts must always be rerequested after a device reset.

Referenced by [L4virtio::Driver::Device::driver\\_connect\(\)](#).

Here is the caller graph for this function:



### 15.378.2.4 register\_ds()

```
long L4virtio::Device::register_ds (
    L4::Ipc::Cap< L4Re::Dataspace > ds_cap,
    l4_uint64_t base,
    l4_umword_t offset,
    l4_umword_t size )
```

Register a shared data space with VIRTIO host.

## Parameters

<i>ds_cap</i>	Dataspace capability to register. The lower 8 bits determine the rights mask with which the guest's rights are masked during the registration of the dataspace at the VIRTIO host.
<i>base</i>	VIRTIO guest physical start address of shared memory region
<i>offset</i>	Offset within the data space that is attached to the given <i>base</i> in the guest physical memory.
<i>size</i>	Size of the memory region in the guest

## Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EINVAL</i>	The <i>ds_cap</i> capability is invalid, does not refer to a valid dataspace, is not a trusted dataspace if trusted dataspace validation is enabled, or <i>size</i> and <i>offset</i> specify an invalid region.
<i>-L4_ENOMEM</i>	The limit of dataspaces that can be registered has been reached or no capability slot could be allocated.
<i>-L4_ERANGE</i>	<i>offset</i> is larger than the size of the dataspace.
<i>&lt;0</i>	Any error returned by the dataspace when queried for information during setup or any error returned by the region manager from attaching the dataspace.

Referenced by [L4virtio::Driver::Device::register\\_ds\(\)](#).

Here is the caller graph for this function:

15.378.2.5 `set_status()`

```
long L4virtio::Device::set_status (
    unsigned status )
```

Write the VIRTIO status register.

## Parameters

<i>status</i>	Status word to write to the VIRTIO status.
---------------	--

## Return values

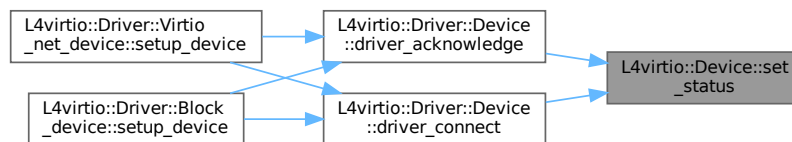
<i>0</i>	on success.
----------	-------------

**Note**

All other registers are accessed via shared memory.

Referenced by [L4virtio::Driver::Device::driver\\_acknowledge\(\)](#), and [L4virtio::Driver::Device::driver\\_connect\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

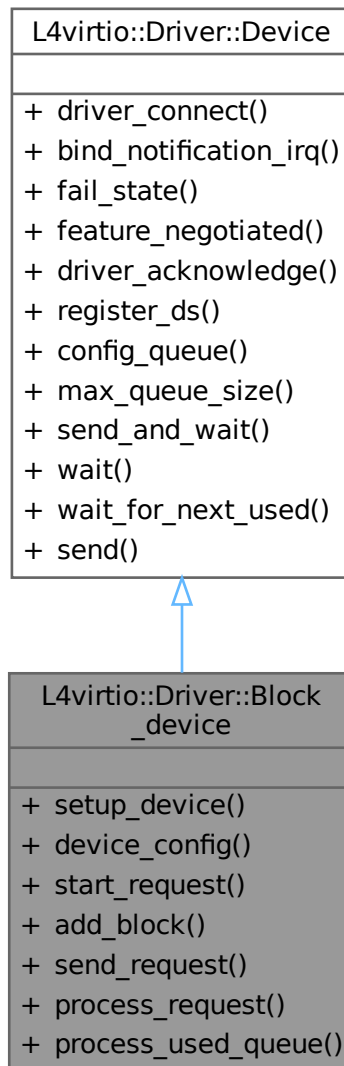
- `l4/l4virtio/l4virtio`

## 15.379 L4virtio::Driver::Block\_device Class Reference

Simple class for accessing a virtio block device synchronously.

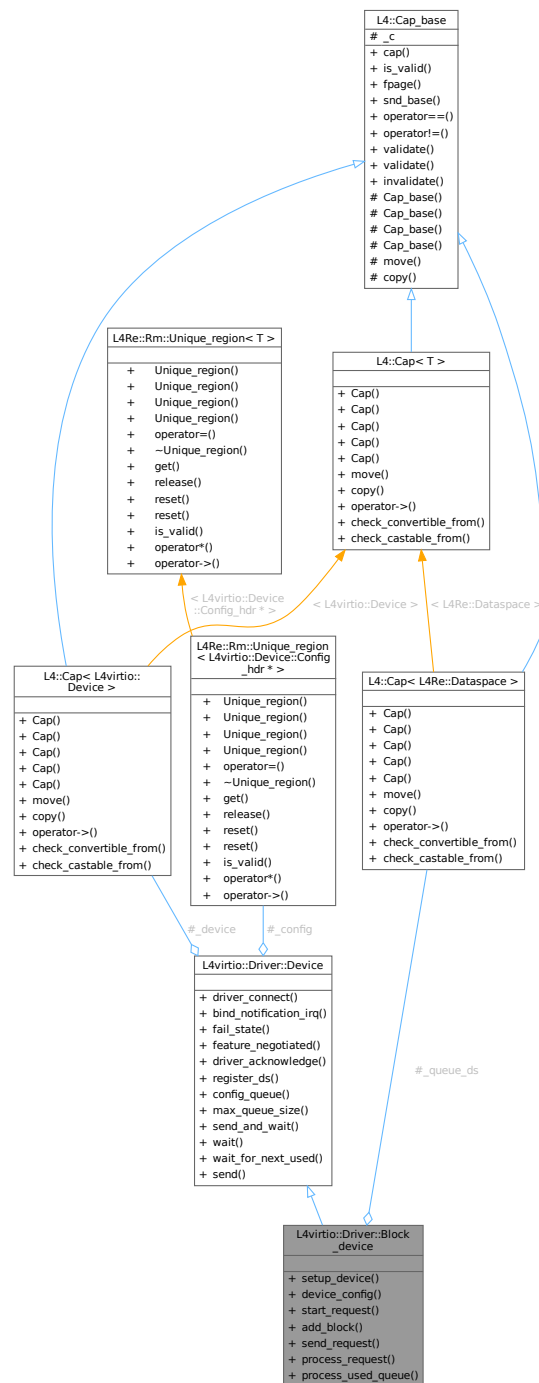
```
#include <virtio-block>
```

Inheritance diagram for L4virtio::Driver::Block\_device:





Collaboration diagram for L4virtio::Driver::Block\_device:



## Data Structures

- class [Handle](#)  
*Handle to an ongoing request.*

## Public Member Functions

- void `setup_device` (`L4::Cap`< `L4virtio::Device` > `srvcap`, `l4_size_t` `usermem`, void `**userdata`, `Ptr`< void > `&user_devaddr`, `L4::Cap`< `L4Re::Dataspace` > `qds=L4::Cap`< `L4Re::Dataspace` >(), `l4_uint32_t` `fmask0=-1U`, `l4_uint32_t` `fmask1=-1U`)  
*Establish a connection to the device and set up shared memory.*
- `l4virtio_block_config_t` const & `device_config` () const  
*Return a reference to the device configuration.*
- `Handle` `start_request` (`l4_uint64_t` `sector`, `l4_uint32_t` `type`, `Callback` `callback`)  
*Start the setup of a new request.*
- int `add_block` (`Handle` `handle`, `Ptr`< void > `addr`, `l4_uint32_t` `size`)  
*Add a data block to a request that has already been set up.*
- int `send_request` (`Handle` `handle`)  
*Process request asynchronously.*
- int `process_request` (`Handle` `handle`)  
*Process request synchronously.*
- void `process_used_queue` ()  
*Process and free all items in the used queue.*

## Public Member Functions inherited from `L4virtio::Driver::Device`

- void `driver_connect` (`L4::Cap`< `L4virtio::Device` > `srvcap`, bool `manage_notify=true`)  
*Contacts the device and starts the initial handshake.*
- int `bind_notification_irq` (unsigned index, `L4::Cap`< `L4::Triggerable` > `irq`) const  
*Register a triggerable to receive notifications from the device.*
- bool `fail_state` () const  
*Return true if the device is in a fail state.*
- bool `feature_negotiated` (unsigned int `feat`) const  
*Check if a particular feature bit was negotiated with the device.*
- int `driver_acknowledge` ()  
*Finalize handshake with the device.*
- int `register_ds` (`L4::Cap`< `L4Re::Dataspace` > `ds`, `l4_umword_t` `offset`, `l4_umword_t` `size`, `l4_uint64_t` `*devaddr`)  
*Share a dataspace with the device.*
- int `config_queue` (int `num`, unsigned `size`, `l4_uint64_t` `desc_addr`, `l4_uint64_t` `avail_addr`, `l4_uint64_t` `used_↔addr`)  
*Send the virtqueue configuration to the device.*
- int `max_queue_size` (int `num`) const  
*Maximum queue size allowed by the device.*
- int `send_and_wait` (`Virtqueue` &`queue`, `l4_uint16_t` `descno`)  
*Send a request to the device and wait for it to be processed.*
- int `wait` (int index) const  
*Wait for a notification from the device.*
- int `wait_for_next_used` (`Virtqueue` &`queue`, `l4_uint32_t` `*len=nullptr`) const  
*Wait for the next item to arrive in the used queue and return it.*
- void `send` (`Virtqueue` &`queue`, `l4_uint16_t` `descno`)  
*Send a request to the device.*

### 15.379.1 Detailed Description

Simple class for accessing a virtio block device synchronously.

Definition at line 36 of file [virtio-block](#).

### 15.379.2 Member Function Documentation

#### 15.379.2.1 add\_block()

```
int L4virtio::Driver::Block_device::add_block (
    Handle handle,
    Ptr< void > addr,
    l4_uint32_t size ) [inline]
```

Add a data block to a request that has already been set up.

##### Parameters

<i>handle</i>	<a href="#">Handle</a> to request previously set up with <a href="#">start_request()</a> .
<i>addr</i>	Address of data block in device address space.
<i>size</i>	Size of data block.

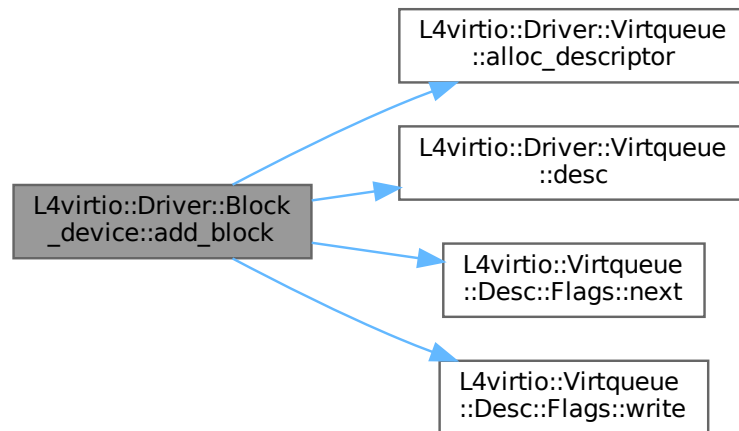
##### Return values

<i>L4_OK</i>	Block was successfully added.
<i>-L4_EAGAIN</i>	No descriptors available. Try again later.

Definition at line 229 of file [virtio-block](#).

References [L4virtio::Virtqueue::Desc::addr](#), [L4virtio::Driver::Virtqueue::alloc\\_descriptor\(\)](#), [L4virtio::Driver::Virtqueue::desc\(\)](#), [L4virtio::Virtqueue::Desc::flags](#), [L4\\_EAGAIN](#), [L4\\_EOK](#), [L4virtio::Virtqueue::Desc::len](#), [L4virtio::Virtqueue::Desc::Flags::next\(\)](#), [L4virtio::Virtqueue::Desc::next](#), [L4virtio::Virtqueue::Desc::Flags::raw](#), [l4virtio\\_block\\_header\\_t::type](#), and [L4virtio::Virtqueue::Desc::Fla](#)

Here is the call graph for this function:



### 15.379.2.2 process\_request()

```
int L4virtio::Driver::Block_device::process_request (
    Handle handle ) [inline]
```

Process request synchronously.

#### Parameters

<i>handle</i>	<a href="#">Handle</a> to request to process.
---------------	---

#### Return values

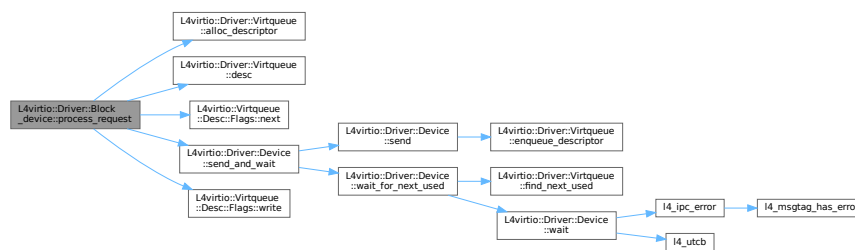
<i>L4_EOK</i>	Request processed successfully.
<i>-L4_EAGAIN</i>	No descriptors available. Try again later.
<i>-L4_EIO</i>	IO error during request processing.
<i>-L4_ENOSYS</i>	Unsupported request.
<i>&lt;0</i>	Another unspecified error occurred.

Sends a request to the driver that was previously set up with [start\\_request\(\)](#) and [add\\_block\(\)](#) and wait for it to be executed.

Definition at line 306 of file [virtio-block](#).

References [L4virtio::Virtqueue::Desc::addr](#), [L4virtio::Driver::Virtqueue::alloc\\_descriptor\(\)](#), [L4virtio::Driver::Virtqueue::desc\(\)](#), [L4virtio::Virtqueue::Desc::flags](#), [L4\\_EAGAIN](#), [L4\\_EINVAL](#), [L4\\_EIO](#), [L4\\_ENOSYS](#), [L4\\_EOK](#), [L4VIRTIO\\_BLOCK\\_S\\_IOERR](#), [L4VIRTIO\\_BLOCK\\_S\\_OK](#), [L4VIRTIO\\_BLOCK\\_S\\_UNSUPP](#), [L4virtio::Virtqueue::Desc::len](#), [L4virtio::Virtqueue::Desc::Flags::next\(\)](#), [L4virtio::Virtqueue::Desc::next](#), [L4virtio::Virtqueue::Desc::Flags::raw](#), [L4virtio::Driver::Device::send\\_and\\_wait\(\)](#), and [L4virtio::Virtqueue::Desc::Flags::write\(\)](#).

Here is the call graph for this function:



### 15.379.2.3 process\_used\_queue()

```
void L4virtio::Driver::Block_device::process_used_queue ( ) [inline]
```

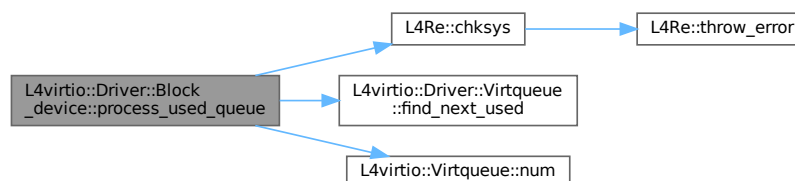
Process and free all items in the used queue.

If the request has a callback registered it is called after the item has been removed from the queue.

Definition at line 357 of file [virtio-block](#).

References [L4Re::chksys\(\)](#), [L4virtio::Driver::Virtqueue::find\\_next\\_used\(\)](#), [L4\\_ENOSYS](#), and [L4virtio::Virtqueue::num\(\)](#).

Here is the call graph for this function:



### 15.379.2.4 send\_request()

```
int L4virtio::Driver::Block_device::send_request (
    Handle handle ) [inline]
```

Process request asynchronously.

Parameters

<i>handle</i>	<a href="#">Handle</a> to request to send to the device
---------------	---

## Return values

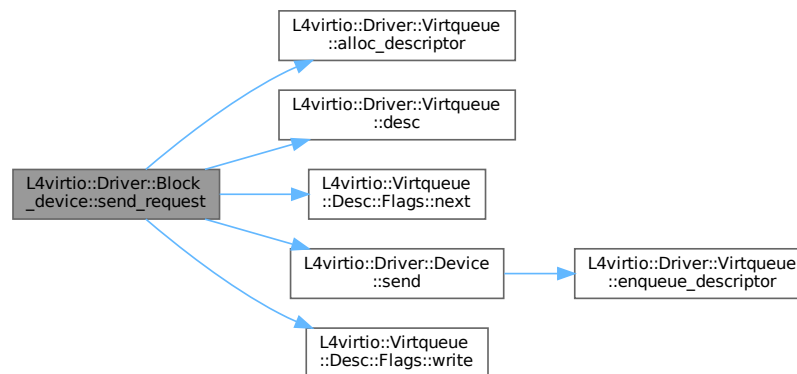
<code>L4_OK</code>	Request was successfully scheduled.
<code>-L4_EAGAIN</code>	No descriptors available. Try again later.

Sends a request to the driver that was previously set up with [start\\_request\(\)](#) and [add\\_block\(\)](#) and wait for it to be executed.

Definition at line 265 of file [virtio-block](#).

References [L4virtio::Virtqueue::Desc::addr](#), [L4virtio::Driver::Virtqueue::alloc\\_descriptor\(\)](#), [L4virtio::Driver::Virtqueue::desc\(\)](#), [L4virtio::Virtqueue::Desc::flags](#), [L4\\_EAGAIN](#), [L4\\_EOK](#), [L4virtio::Virtqueue::Desc::len](#), [L4virtio::Virtqueue::Desc::Flags::next\(\)](#), [L4virtio::Virtqueue::Desc::next](#), [L4virtio::Virtqueue::Desc::Flags::raw](#), [L4virtio::Driver::Device::send\(\)](#), and [L4virtio::Virtqueue::Desc::Flags::write\(\)](#).

Here is the call graph for this function:



### 15.379.2.5 setup\_device()

```

void L4virtio::Driver::Block_device::setup_device (
    L4::Cap< L4virtio::Device > srvcap,
    l4_size_t usermem,
    void ** userdata,
    Ptr< void > & user_devaddr,
    L4::Cap< L4Re::Dataspace > qds = L4::Cap<L4Re::Dataspace>(),
    l4_uint32_t fmask0 = -1U,
    l4_uint32_t fmask1 = -1U ) [inline]

```

Establish a connection to the device and set up shared memory.

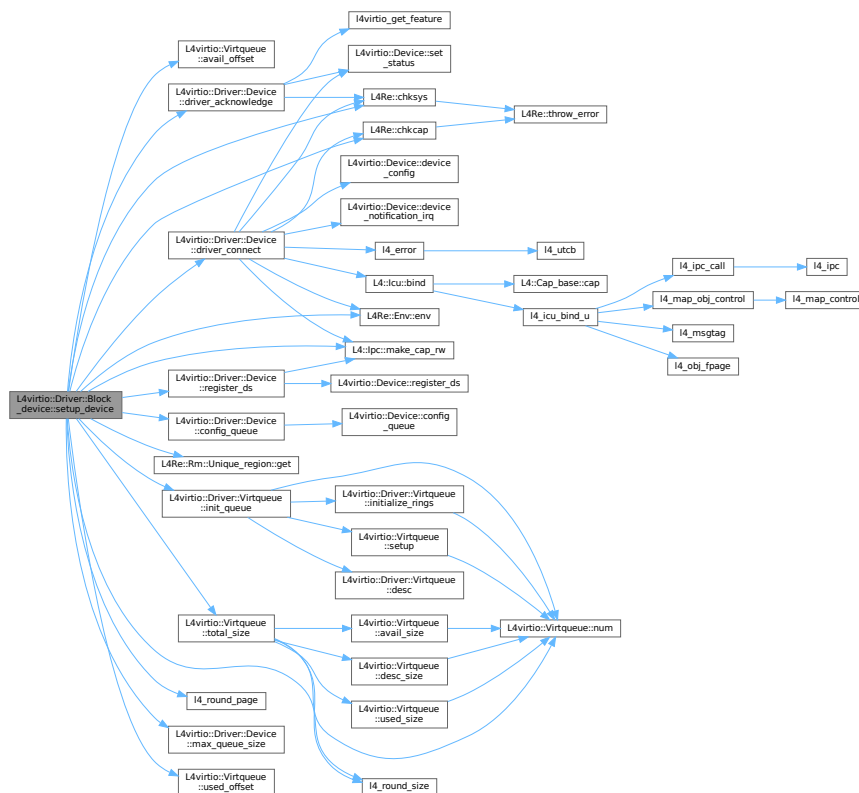
## Parameters

	<code>srvcap</code>	IPC capability of the channel to the server.
	<code>usermem</code>	Size of additional memory to share with device.
out	<code>userdata</code>	Pointer to the region of user-usable memory.
out	<code>user_devaddr</code>	Address of user-usable memory in device address space.

	<i>qds</i>	External queue dataspace. If this capability is invalid, the function will attempt to allocate a dataspace on its own. Note that the external queue dataspace must be large enough.
	<i>fmask0</i>	Feature bits 0..31 that the driver supports.
	<i>fmask1</i>	Feature bits 32..63 that the driver supports.

Definition at line 92 of file virtio-block.

Here is the call graph for this function:



### 15.379.2.6 start\_request()

```
Handle L4virtio::Driver::Block_device::start_request (
    l4_uint64_t sector,
    l4_uint32_t type,
    Callback callback ) [inline]
```

Start the setup of a new request.

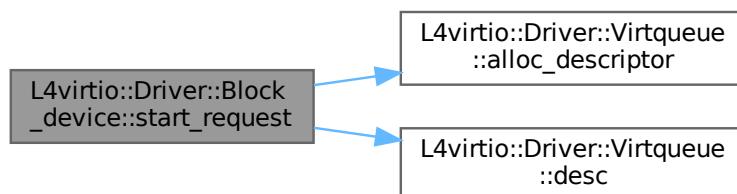
#### Parameters

<i>sector</i>	First sector to write to/read from.
<i>type</i>	Request type.
<i>callback</i>	Function to call, when the request is finished. May be 0 for synchronous requests.

Definition at line 191 of file [virtio-block](#).

References [L4virtio::Virtqueue::Desc::addr](#), [L4virtio::Driver::Virtqueue::alloc\\_descriptor\(\)](#), [L4virtio::Driver::Virtqueue::desc\(\)](#), [L4virtio::Virtqueue::Desc::flags](#), [l4virtio\\_block\\_header\\_t::ioprio](#), [L4virtio::Virtqueue::Desc::len](#), [L4virtio::Virtqueue::Desc::Flags::raw](#), [l4virtio\\_block\\_header\\_t::sector](#), and [l4virtio\\_block\\_header\\_t::type](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- [l4/l4virtio/client/virtio-block](#)

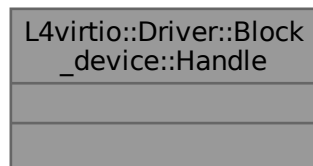
## 15.380 L4virtio::Driver::Block\_device::Handle Class Reference

[Handle](#) to an ongoing request.

```
#include <virtio-block>
```



Collaboration diagram for L4virtio::Driver::Block\_device::Handle:



### 15.380.1 Detailed Description

[Handle](#) to an ongoing request.

Definition at line [56](#) of file [virtio-block](#).

The documentation for this class was generated from the following file:

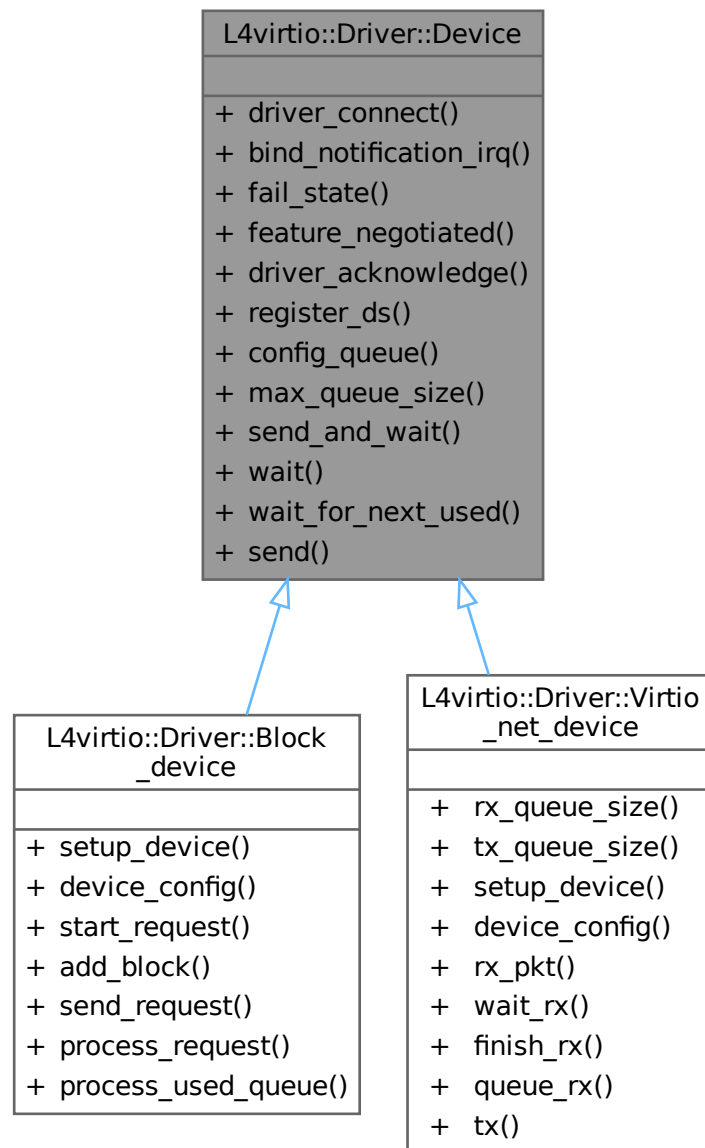
- [l4/l4virtio/client/virtio-block](#)

## 15.381 L4virtio::Driver::Device Class Reference

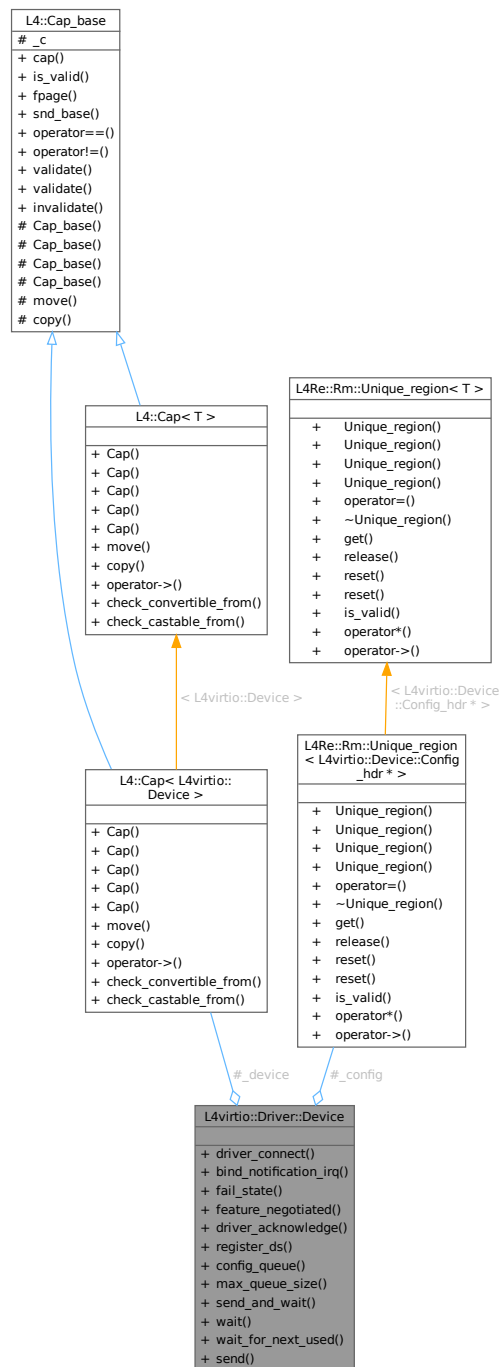
Client-side implementation for a general virtio device.

```
#include <l4virtio>
```

Inheritance diagram for L4virtio::Driver::Device:



Collaboration diagram for L4virtio::Driver::Device:



## Public Member Functions

- void [driver\\_connect](#) (L4::Cap< L4virtio::Device > srvcap, bool manage\_notify=true)  
Contacts the device and starts the initial handshake.
- int [bind\\_notification\\_irq](#) (unsigned index, L4::Cap< L4::Triggerable > irq) const  
Register a triggerable to receive notifications from the device.
- bool [fail\\_state](#) () const

- Return true if the device is in a fail state.*

  - bool `feature_negotiated` (unsigned int feat) const

*Check if a particular feature bit was negotiated with the device.*
- int `driver_acknowledge` ()

*Finalize handshake with the device.*
- int `register_ds` (L4::Cap< L4Re::Dataspace > ds, l4\_umword\_t offset, l4\_umword\_t size, l4\_uint64\_t \*devaddr)

*Share a dataspace with the device.*
- int `config_queue` (int num, unsigned size, l4\_uint64\_t desc\_addr, l4\_uint64\_t avail\_addr, l4\_uint64\_t used\_addr)

*Send the virtqueue configuration to the device.*
- int `max_queue_size` (int num) const

*Maximum queue size allowed by the device.*
- int `send_and_wait` (Virtqueue &queue, l4\_uint16\_t descno)

*Send a request to the device and wait for it to be processed.*
- int `wait` (int index) const

*Wait for a notification from the device.*
- int `wait_for_next_used` (Virtqueue &queue, l4\_uint32\_t \*len=NULLPTR) const

*Wait for the next item to arrive in the used queue and return it.*
- void `send` (Virtqueue &queue, l4\_uint16\_t descno)

*Send a request to the device.*

## 15.381.1 Detailed Description

Client-side implementation for a general virtio device.

Definition at line 31 of file `l4virtio`.

## 15.381.2 Member Function Documentation

### 15.381.2.1 bind\_notification\_irq()

```
int L4virtio::Driver::Device::bind_notification_irq (
    unsigned index,
    L4::Cap< L4::Triggerable > irq ) const [inline]
```

Register a triggerable to receive notifications from the device.

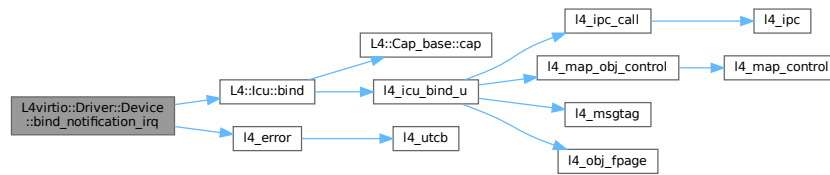
#### Parameters

	<i>index</i>	Index of the interrupt.
out	<i>irq</i>	Triggerable to register for notifications.

Definition at line 129 of file `l4virtio`.

References `L4::lcu::bind()`, and `l4_error()`.

Here is the call graph for this function:



### 15.381.2.2 config\_queue()

```

int L4virtio::Driver::Device::config_queue (
    int num,
    unsigned size,
    l4_uint64_t desc_addr,
    l4_uint64_t avail_addr,
    l4_uint64_t used_addr ) [inline]

```

Send the virtqueue configuration to the device.

#### Parameters

<i>num</i>	Number of queue to configure.
<i>size</i>	Size of rings in the queue, must be a power of 2)
<i>desc_addr</i>	Address of descriptor table (device address)
<i>avail_addr</i>	Address of available ring (device address)
<i>used_addr</i>	Address of used ring (device address)

Definition at line 212 of file [l4virtio](#).

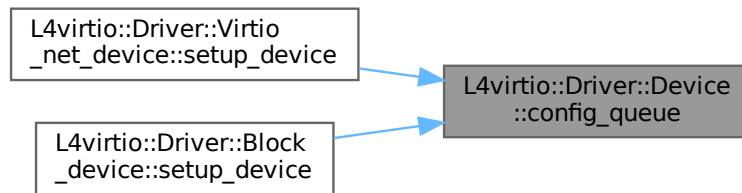
References [L4virtio::Device::config\\_queue\(\)](#).

Referenced by [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#), and [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.381.2.3 driver\_acknowledge()

```
int L4virtio::Driver::Device::driver_acknowledge ( ) [inline]
```

Finalize handshake with the device.

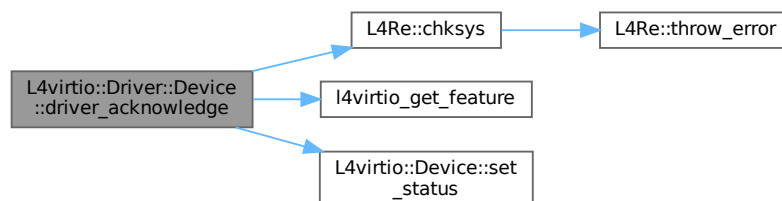
Must be called after all queues have been set up and before the first request is sent. It is still possible to add more shared dataspace after the handshake has been finished.

Definition at line 156 of file [l4virtio](#).

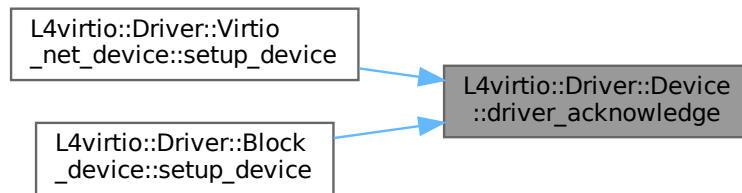
References [L4Re::chksys\(\)](#), [L4\\_EINVAL](#), [L4\\_EIO](#), [L4\\_ENODEV](#), [L4\\_EOK](#), [L4VIRTIO\\_FEATURE\\_VERSION\\_1](#), [l4virtio\\_get\\_feature\(\)](#), [L4VIRTIO\\_STATUS\\_DRIVER\\_OK](#), [L4VIRTIO\\_STATUS\\_FEATURES\\_OK](#), and [L4virtio::Device::set\\_status\(\)](#).

Referenced by [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#), and [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 15.381.2.4 driver\_connect()

```
void L4virtio::Driver::Device::driver_connect (
    L4::Cap< L4virtio::Device > srvcap,
    bool manage_notify = true ) [inline]
```

Contacts the device and starts the initial handshake.

##### Parameters

<i>srvcap</i>	Capability for device communication.
<i>manage_notify</i>	Set up a semaphore for notifications from the device. See below.

##### Exceptions

<a href="#">L4::Runtime_error</a>	if the initialisation fails
-----------------------------------	-----------------------------

This function contacts the server, sets up the notification channels and the configuration dataspace. After this is done, the caller can set up any dataspace it needs. The initialisation then needs to be finished by calling [driver\\_acknowledge\(\)](#).

Per default this function creates and registers a semaphore for receiving notification from the device. This semaphore is used in the blocking functions [send\\_and\\_wait\(\)](#), [wait\(\)](#) and [next\\_used\(\)](#).

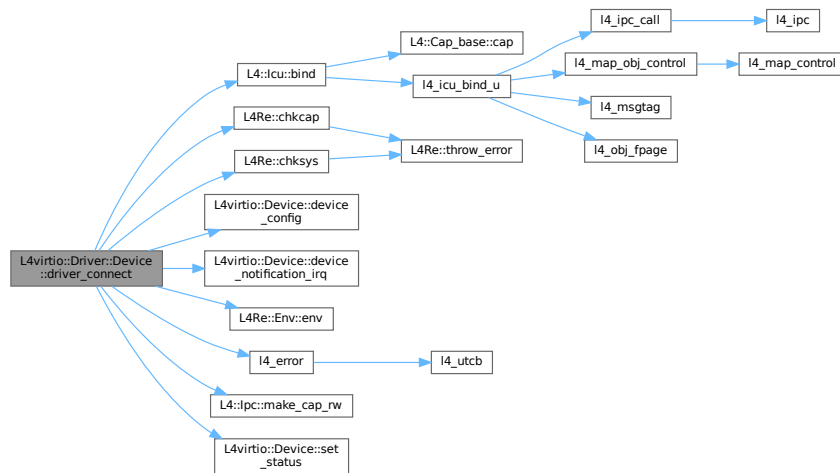
When `manage_notify` is false, then the caller may manually register and handle notification interrupts from the device. This is for example useful, when the client runs in an application with a server loop.

Definition at line 56 of file [l4virtio](#).

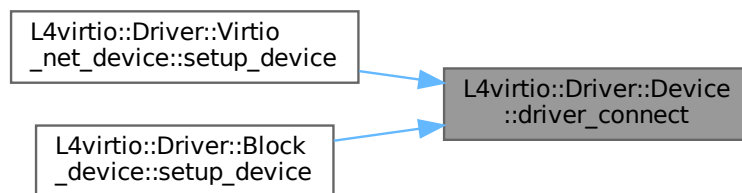
References [L4::lcu::bind\(\)](#), [L4Re::chkcap\(\)](#), [L4Re::chksys\(\)](#), [L4virtio::Device::device\\_config\(\)](#), [L4virtio::Device::device\\_notification\\_irq\(\)](#), [L4Re::Env::env\(\)](#), [L4\\_EINVAL](#), [L4\\_EIO](#), [L4\\_ENODEV](#), [l4\\_error\(\)](#), [L4\\_PAGEMASK](#), [L4\\_PAGESHIFT](#), [L4\\_PAGESIZE](#), [L4\\_SUPERPAGESIZE](#), [L4VIRTIO\\_STATUS\\_ACKNOWLEDGE](#), [L4VIRTIO\\_STATUS\\_DRIVER](#), [L4::lpc::make\\_cap\\_rw\(\)](#), [L4Re::Rm::F::RW](#), [L4Re::Rm::F::Search\\_addr](#), and [L4virtio::Device::set\\_status\(\)](#).

Referenced by [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#), and [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.381.2.5 feature\_negotiated()

```
bool L4virtio::Driver::Device::feature_negotiated (
    unsigned int feat ) const [inline]
```

Check if a particular feature bit was negotiated with the device.

The result is only valid after [driver\\_acknowledge\(\)](#) was called (when the handshake with the device was completed).

#### Parameters

<i>feat</i>	The feature bit.
-------------	------------------

#### Return values

<i>true</i>	The feature is supported by both driver and device.
-------------	---



## Return values

<i>false</i>	The feature is not supported by the driver and/or device.
--------------	---

Definition at line 145 of file [l4virtio](#).

References [l4virtio\\_get\\_feature\(\)](#).

Here is the call graph for this function:



## 15.381.2.6 max\_queue\_size()

```
int L4virtio::Driver::Device::max_queue_size (
    int num ) const [inline]
```

Maximum queue size allowed by the device.

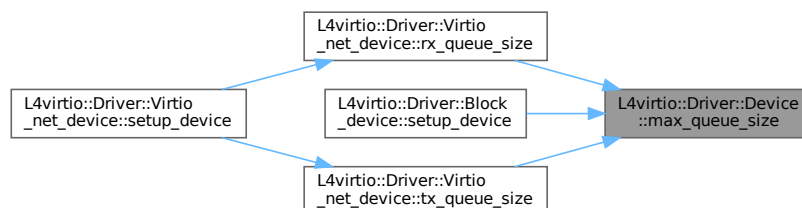
## Parameters

<i>num</i>	Number of queue for which to determine the maximum size.
------------	--

Definition at line 230 of file [l4virtio](#).

Referenced by [L4virtio::Driver::Virtio\\_net\\_device::rx\\_queue\\_size\(\)](#), [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#), and [L4virtio::Driver::Virtio\\_net\\_device::tx\\_queue\\_size\(\)](#).

Here is the caller graph for this function:



### 15.381.2.7 register\_ds()

```
int L4virtio::Driver::Device::register_ds (
    L4::Cap< L4Re::Dataspace > ds,
    l4_umword_t offset,
    l4_umword_t size,
    l4_uint64_t * devaddr ) [inline]
```

Share a dataspace with the device.

#### Parameters

<i>ds</i>	Dataspace to share with the device.
<i>offset</i>	Offset in dataspace where the shared part starts.
<i>size</i>	Total size in bytes of the shared space.
<i>devaddr</i>	Start of shared space in the device address space.

Although this function allows to share only a part of the given dataspace for convenience, the granularity of sharing is always the dataspace level. Thus, the remainder of the dataspace is not protected from the device.

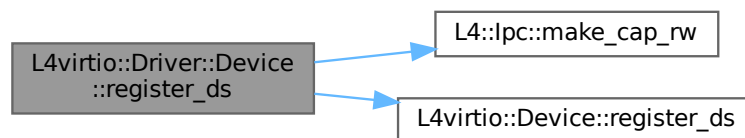
When communicating with the device, addresses must be given with respect to the device address space. This is not the same as the virtual address space of the client in order to not leak information about the address space layout.

Definition at line 196 of file [l4virtio](#).

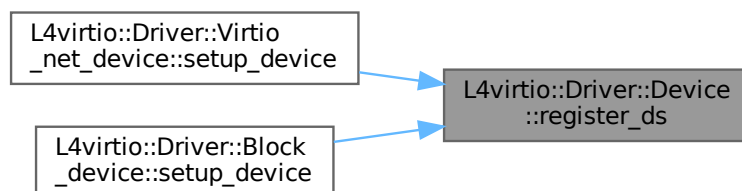
References [L4::lpc::make\\_cap\\_rw\(\)](#), and [L4virtio::Device::register\\_ds\(\)](#).

Referenced by [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#), and [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.381.2.8 send()

```
void L4virtio::Driver::Device::send (
    Virtqueue & queue,
    l4_uint16_t descno ) [inline]
```

Send a request to the device.

#### Parameters

<i>queue</i>	Queue that contains the request in its descriptor table
<i>descno</i>	Index of first entry in descriptor table where

Definition at line 312 of file [l4virtio](#).

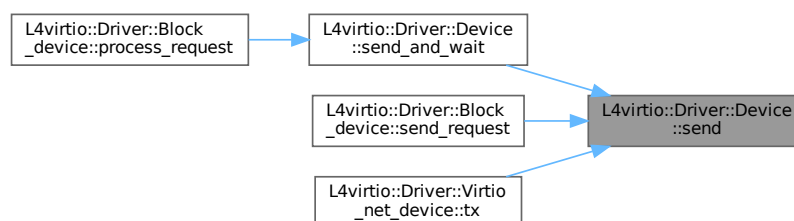
References [L4virtio::Driver::Virtqueue::enqueue\\_descriptor\(\)](#).

Referenced by [send\\_and\\_wait\(\)](#), [L4virtio::Driver::Block\\_device::send\\_request\(\)](#), and [L4virtio::Driver::Virtio\\_net\\_device::tx\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.381.2.9 send\_and\_wait()

```
int L4virtio::Driver::Device::send_and_wait (
    Virtqueue & queue,
    l4_uint16_t descno ) [inline]
```

Send a request to the device and wait for it to be processed.

## Parameters

<i>queue</i>	Queue that contains the request in its descriptor table
<i>descno</i>	Index of first entry in descriptor table where

This function provides a simple mechanism to send requests synchronously. It must not be used with other requests at the same time as it directly waits for a notification on the device irq cap.

## Precondition

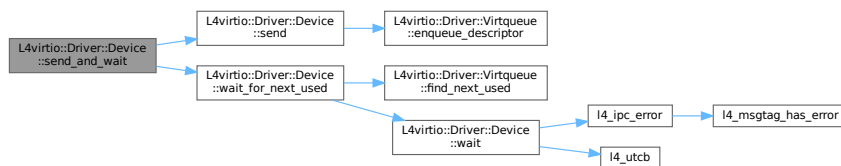
[driver\\_connect\(\)](#) was called with `manage_notify`.

Definition at line 247 of file [l4virtio](#).

References [L4\\_EINVAL](#), [L4\\_EOK](#), [send\(\)](#), and [wait\\_for\\_next\\_used\(\)](#).

Referenced by [L4virtio::Driver::Block\\_device::process\\_request\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 15.381.2.10 wait()

```
int L4virtio::Driver::Device::wait (
    int index ) const [inline]
```

Wait for a notification from the device.

## Parameters

<i>index</i>	Notification slot to wait for.
--------------	--------------------------------

**Precondition**

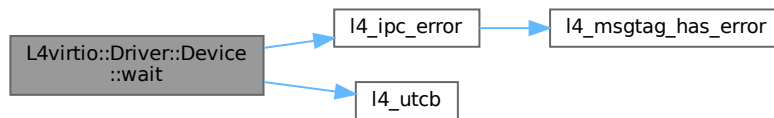
`driver_connect()` was called with `manage_notify`.

Definition at line 268 of file `l4virtio`.

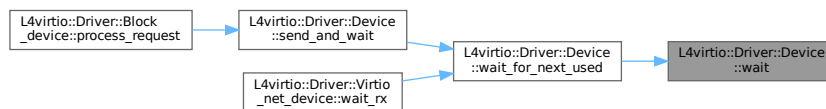
References `L4_EEXIST`, `l4_ipc_error()`, and `l4_utcb()`.

Referenced by `wait_for_next_used()`.

Here is the call graph for this function:



Here is the caller graph for this function:

**15.381.2.11 wait\_for\_next\_used()**

```

int L4virtio::Driver::Device::wait_for_next_used (
    Virtqueue & queue,
    l4_uint32_t * len = nullptr ) const [inline]
  
```

Wait for the next item to arrive in the used queue and return it.

**Parameters**

	<i>queue</i>	A queue.
out	<i>len</i>	(optional) Size of valid data in finished block. Note that this is the value reported by the device, which may set it to a value that is larger than the original buffer size.

**Return values**

$\geq 0$	Descriptor number of item removed from used queue.
$< 0$	IPC error while waiting for notification.

The call blocks until the next item is available in the used queue.

#### Precondition

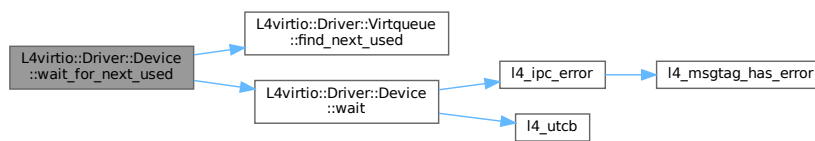
[driver\\_connect\(\)](#) was called with `manage_notify`.

Definition at line 291 of file [l4virtio](#).

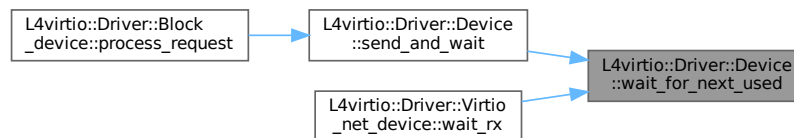
References [L4virtio::Driver::Virtqueue::find\\_next\\_used\(\)](#), and [wait\(\)](#).

Referenced by [send\\_and\\_wait\(\)](#), and [L4virtio::Driver::Virtio\\_net\\_device::wait\\_rx\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

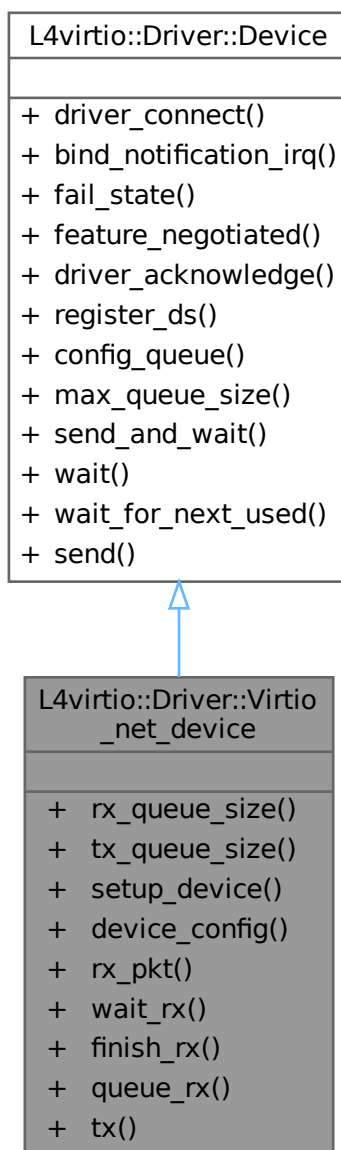
- `I4/I4virtio/client/I4virtio`

## 15.382 L4virtio::Driver::Virtio\_net\_device Class Reference

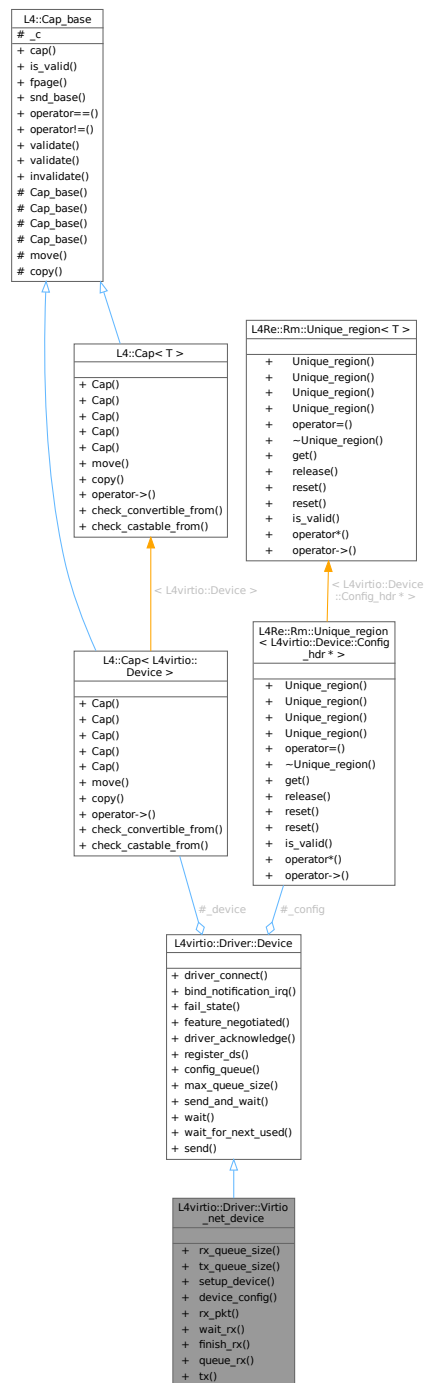
Simple class for accessing a virtio net device.

```
#include <virtio-net>
```

Inheritance diagram for L4virtio::Driver::Virtio\_net\_device:



Collaboration diagram for L4virtio::Driver::Virtio\_net\_device:



## Data Structures

- struct [Packet](#)

Structure for a network packet (header including data) with maximum size, assuming that no extra features have been negotiated.



## Public Member Functions

- int [rx\\_queue\\_size](#) () const  
*Return the maximum receive queue size allowed by the device.*
- int [tx\\_queue\\_size](#) () const  
*Return the maximum transmit queue size allowed by the device.*
- void [setup\\_device](#) (L4::Cap< L4virtio::Device > srvcap)  
*Establish a connection to the device and set up shared memory.*
- [l4virtio\\_net\\_config\\_t](#) const & [device\\_config](#) () const  
*Return a reference to the device configuration.*
- [Packet](#) & [rx\\_pkt](#) (l4\_uint16\_t descno)  
*Return a reference to the RX packet buffer of the specified descriptor, e.g.*
- [l4\\_uint16\\_t](#) [wait\\_rx](#) (l4\_uint32\_t \*len=nullptr)  
*Block until a network packet has been received from the device and return the descriptor number.*
- void [finish\\_rx](#) (l4\_uint16\_t descno)  
*Free an RX descriptor number to make it available for the RX queue again.*
- void [queue\\_rx](#) ()  
*Queue new available descriptors in the RX queue.*
- bool [tx](#) (std::function< l4\_uint32\_t(Packet &)> prepare)  
*Attempt to allocate a descriptor in the TX queue and transmit the packet, after calling the prepare callback.*

## Public Member Functions inherited from L4virtio::Driver::Device

- void [driver\\_connect](#) (L4::Cap< L4virtio::Device > srvcap, bool manage\_notify=true)  
*Contacts the device and starts the initial handshake.*
- int [bind\\_notification\\_irq](#) (unsigned index, L4::Cap< L4::Triggerable > irq) const  
*Register a triggerable to receive notifications from the device.*
- bool [fail\\_state](#) () const  
*Return true if the device is in a fail state.*
- bool [feature\\_negotiated](#) (unsigned int feat) const  
*Check if a particular feature bit was negotiated with the device.*
- int [driver\\_acknowledge](#) ()  
*Finalize handshake with the device.*
- int [register\\_ds](#) (L4::Cap< L4Re::Dataspace > ds, l4\_umword\_t offset, l4\_umword\_t size, l4\_uint64\_t \*devaddr)  
*Share a dataspace with the device.*
- int [config\\_queue](#) (int num, unsigned size, l4\_uint64\_t desc\_addr, l4\_uint64\_t avail\_addr, l4\_uint64\_t used\_addr)  
*Send the virtqueue configuration to the device.*
- int [max\\_queue\\_size](#) (int num) const  
*Maximum queue size allowed by the device.*
- int [send\\_and\\_wait](#) (Virtqueue &queue, l4\_uint16\_t descno)  
*Send a request to the device and wait for it to be processed.*
- int [wait](#) (int index) const  
*Wait for a notification from the device.*
- int [wait\\_for\\_next\\_used](#) (Virtqueue &queue, l4\_uint32\_t \*len=nullptr) const  
*Wait for the next item to arrive in the used queue and return it.*
- void [send](#) (Virtqueue &queue, l4\_uint16\_t descno)  
*Send a request to the device.*

### 15.382.1 Detailed Description

Simple class for accessing a virtio net device.

Definition at line 30 of file [virtio-net](#).

### 15.382.2 Member Function Documentation

#### 15.382.2.1 finish\_rx()

```
void L4virtio::Driver::Virtio_net_device::finish_rx (
    l4_uint16_t descno ) [inline]
```

Free an RX descriptor number to make it available for the RX queue again.

##### Parameters

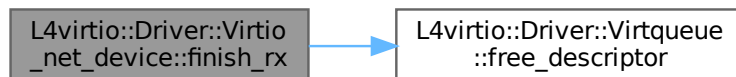
<i>descno</i>	Descriptor number in the virtio queue.
---------------	--

Usually [queue\\_rx\(\)](#) should be called afterwards to queue the freed descriptor(s).

Definition at line 194 of file [virtio-net](#).

References [L4virtio::Driver::Virtqueue::free\\_descriptor\(\)](#).

Here is the call graph for this function:



#### 15.382.2.2 rx\_pkt()

```
Packet & L4virtio::Driver::Virtio_net_device::rx_pkt (
    l4_uint16_t descno ) [inline]
```

Return a reference to the RX packet buffer of the specified descriptor, e.g.

from [wait\\_rx\(\)](#).

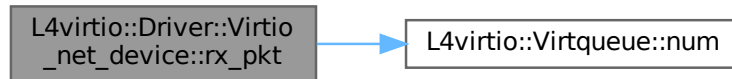
##### Parameters

<i>descno</i>	Descriptor number in the virtio queue.
---------------	--

Definition at line 158 of file [virtio-net](#).

References [L4virtio::Virtqueue::num\(\)](#).

Here is the call graph for this function:



### 15.382.2.3 rx\_queue\_size()

```
int L4virtio::Driver::Virtio_net_device::rx_queue_size ( ) const [inline]
```

Return the maximum receive queue size allowed by the device.

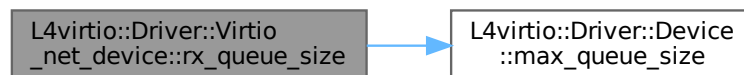
[wait\\_rx\(\)](#) will return a descriptor number that is smaller than this size.

Definition at line 47 of file [virtio-net](#).

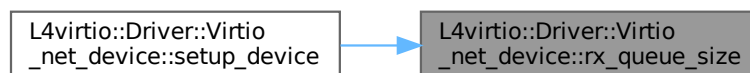
References [L4virtio::Driver::Device::max\\_queue\\_size\(\)](#).

Referenced by [setup\\_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.382.2.4 setup\_device()

```
void L4virtio::Driver::Virtio_net_device::setup_device (
    L4::Cap< L4virtio::Device > srvcap ) [inline]
```

Establish a connection to the device and set up shared memory.

## Parameters

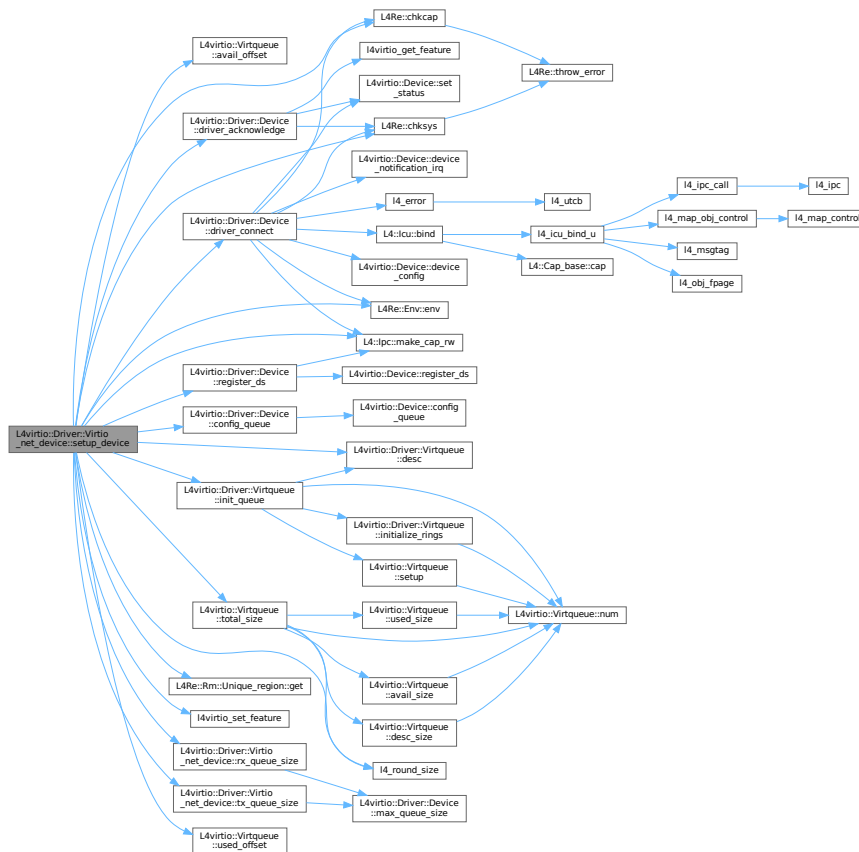
<code>srcvcap</code>	IPC capability of the channel to the server.
----------------------	--

This function starts a handshake with the device and sets up the virtqueues for communication and the additional data structures for the network device.

Definition at line 66 of file [virtio-net](#).

References [L4virtio::Virtqueue::Desc::addr](#), [L4virtio::Virtqueue::avail\\_offset\(\)](#), [L4Re::chkcap\(\)](#), [L4Re::chksys\(\)](#), [L4virtio::Driver::Device::config\\_queue\(\)](#), [L4Re::Mem\\_alloc::Continuous](#), [L4virtio::Driver::Virtqueue::desc\(\)](#), [L4virtio::Driver::Device::driver\\_acknowledge\(\)](#), [L4virtio::Driver::Device::driver\\_connect\(\)](#), [L4Re::Env::env\(\)](#), [L4Re::Rm::Unique\\_region< T >::get\(\)](#), [L4virtio::Driver::Virtqueue::init\\_queue\(\)](#), [L4\\_EINVAL](#), [L4\\_ENODEV](#), [L4\\_PAGESHIFT](#), [I4\\_round\\_size\(\)](#), [L4VIRTIO\\_FEATURE\\_VERSION\\_1](#), [L4VIRTIO\\_ID\\_NET](#), [l4virtio\\_set\\_feature\(\)](#), [L4::lpc::make\\_cap\\_rw\(\)](#), [L4Re::Mem\\_alloc::Pinned](#), [L4virtio::Driver::Device::register\\_ds\(\)](#), [L4Re::Rm::F::RW](#), [rx\\_queue\\_size\(\)](#), [L4Re::Rm::F::Search\\_addr](#), [L4virtio::Virtqueue::total\\_size\(\)](#), [tx\\_queue\\_size\(\)](#), and [L4virtio::Virtqueue::used\\_offset\(\)](#).

Here is the call graph for this function:



### 15.382.2.5 tx()

```
bool L4virtio::Driver::Virtio_net_device::tx (
    std::function< I4_uint32_t(Packet &)> prepare ) [inline]
```

Attempt to allocate a descriptor in the TX queue and transmit the packet, after calling the prepare callback.

## Parameters

<i>prepare</i>	Function that fills the packet with data, should return the length of the data copied to the packet.
----------------	--

## Return values

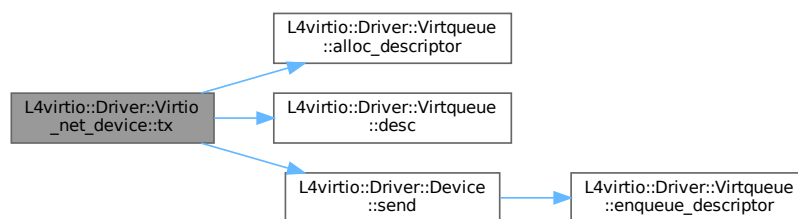
<i>true</i>	The packet was queued.
<i>false</i>	TX queue is full.

The prepare callback should fill the packet with data and return the length of the packet data (without the size of the virtio-net packet header).

Definition at line 224 of file [virtio-net](#).

References [L4virtio::Driver::Virtqueue::alloc\\_descriptor\(\)](#), [L4virtio::Driver::Virtqueue::desc\(\)](#), [L4virtio::Virtqueue::Desc::len](#), and [L4virtio::Driver::Device::send\(\)](#).

Here is the call graph for this function:



## 15.382.2.6 tx\_queue\_size()

```
int L4virtio::Driver::Virtio_net_device::tx_queue_size ( ) const [inline]
```

Return the maximum transmit queue size allowed by the device.

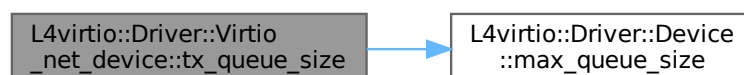
[tx\(\)](#) will fail if the amount of queued packets exceeds this size.

Definition at line 54 of file [virtio-net](#).

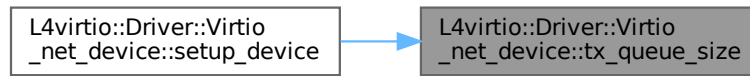
References [L4virtio::Driver::Device::max\\_queue\\_size\(\)](#).

Referenced by [setup\\_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.382.2.7 wait\_rx()

```
l4_uint16_t L4virtio::Driver::Virtio_net_device::wait_rx (
    l4_uint32_t * len = nullptr ) [inline]
```

Block until a network packet has been received from the device and return the descriptor number.

#### Parameters

out	len	(optional) Length of valid data in RX packet.
-----	-----	---

#### Returns

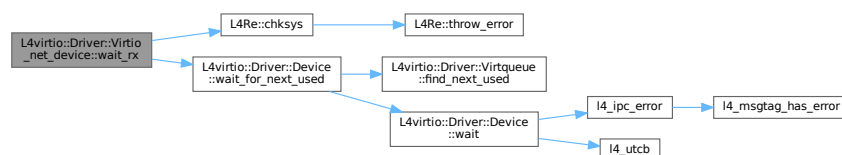
Descriptor number of received packet.

The packet data can be obtained with [rx\\_pkt\(\)](#). [finish\\_rx\(\)](#) should be called after the packet buffer can be returned to the RX queue.

Definition at line 176 of file [virtio-net](#).

References [L4Re::chksys\(\)](#), and [L4virtio::Driver::Device::wait\\_for\\_next\\_used\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

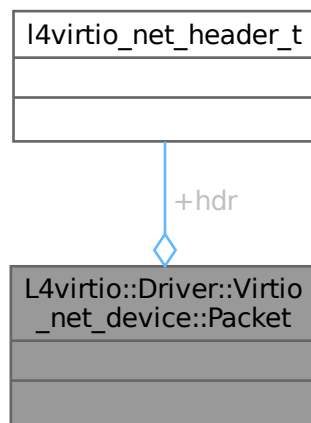
- [I4/I4virtio/client/virtio-net](#)

## 15.383 L4virtio::Driver::Virtio\_net\_device::Packet Struct Reference

Structure for a network packet (header including data) with maximum size, assuming that no extra features have been negotiated.

```
#include <virtio-net>
```

Collaboration diagram for L4virtio::Driver::Virtio\_net\_device::Packet:



### 15.383.1 Detailed Description

Structure for a network packet (header including data) with maximum size, assuming that no extra features have been negotiated.

Definition at line 37 of file [virtio-net](#).

The documentation for this struct was generated from the following file:

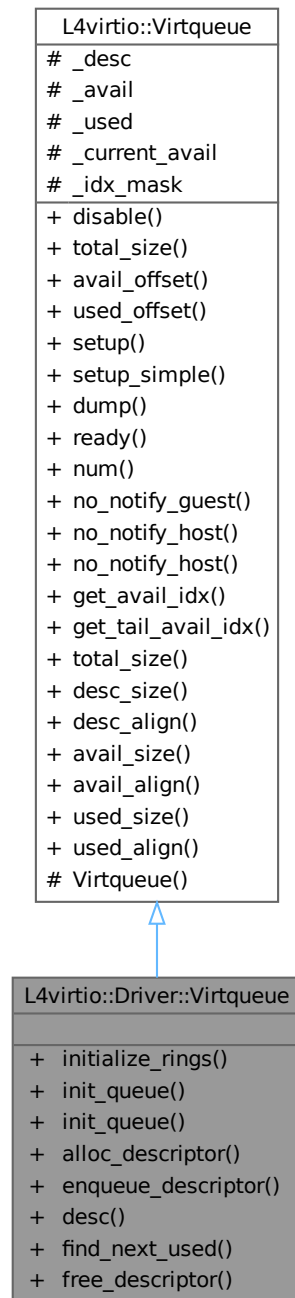
- I4/I4virtio/client/virtio-net

## 15.384 L4virtio::Driver::Virtqueue Class Reference

Driver-side implementation of a [Virtqueue](#).

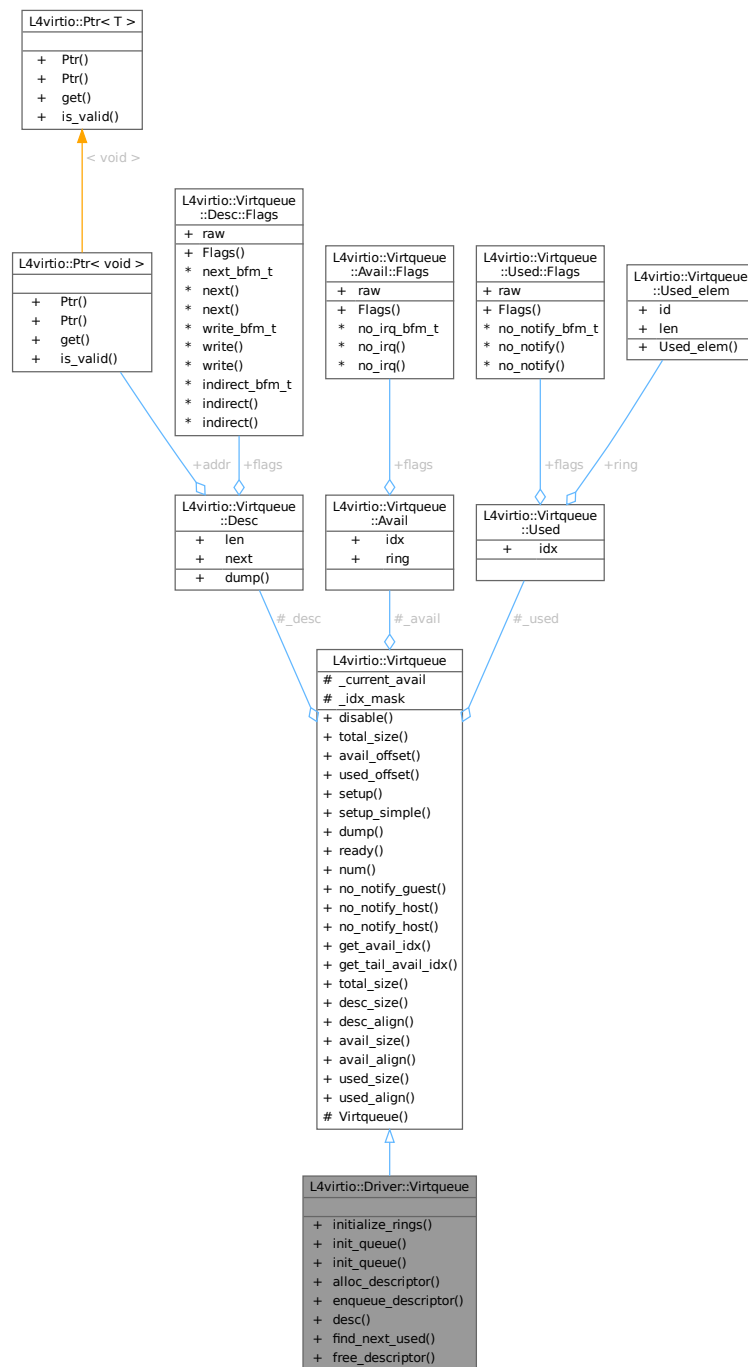
```
#include <virtqueue>
```

Inheritance diagram for L4virtio::Driver::Virtqueue:





Collaboration diagram for L4virtio::Driver::Virtqueue:



## Public Member Functions

- void `initialize_rings` (unsigned `num`)  
*Initialize the descriptor table and the index structures of this queue.*
- void `init_queue` (unsigned `num`, void `*desc`, void `*avail`, void `*used`)  
*Initialize this virtqueue.*
- void `init_queue` (unsigned `num`, void `*base`)

- Initialize this virtqueue.*
  - `l4_uint16_t alloc_descriptor ()`  
*Allocate and return an unused descriptor from the descriptor table.*
  - `void enqueue_descriptor (l4_uint16_t descno)`  
*Enqueue a descriptor in the available ring.*
  - `Desc & desc (l4_uint16_t descno)`  
*Return a reference to a descriptor in the descriptor table.*
  - `l4_uint16_t find_next_used (l4_uint32_t *len=NULLPTR)`  
*Return the next finished block.*
  - `void free_descriptor (l4_uint16_t head, l4_uint16_t tail)`  
*Free a chained list of descriptors in the descriptor queue.*

## Public Member Functions inherited from `L4virtio::Virtqueue`

- `void disable ()`  
*Completely disable the queue.*
- `unsigned long total_size () const`  
*Calculate the total size of this virtqueue.*
- `unsigned long avail_offset () const`  
*Get the offset of the available ring from the descriptor table.*
- `unsigned long used_offset () const`  
*Get the offset of the used ring from the descriptor table.*
- `void setup (unsigned num, void *desc, void *avail, void *used)`  
*Enable this queue.*
- `void setup_simple (unsigned num, void *ring)`  
*Enable this queue.*
- `void dump (Desc const *d) const`  
*Dump descriptors for this queue.*
- `bool ready () const`  
*Test if this queue is in working state.*
- `unsigned num () const`
- `bool no_notify_guest () const`  
*Get the no IRQ flag of this queue.*
- `bool no_notify_host () const`  
*Get the no notify flag of this queue.*
- `void no_notify_host (bool value)`  
*Set the no-notify flag for this queue.*
- `l4_uint16_t get_avail_idx () const`  
*Get available index from available ring (for debugging).*
- `l4_uint16_t get_tail_avail_idx () const`  
*Get tail-available index stored in local state (for debugging).*

## Additional Inherited Members

## Public Types inherited from `L4virtio::Virtqueue`

- `enum`  
*Fixed alignment values for different parts of a virtqueue.*

## Static Public Member Functions inherited from L4virtio::Virtqueue

- static unsigned long [total\\_size](#) (unsigned [num](#))  
*Calculate the total size for a virtqueue of the given dimensions.*
- static unsigned long [desc\\_size](#) (unsigned [num](#))  
*Calculate the size of the descriptor table for *num* entries.*
- static unsigned long [desc\\_align](#) ()  
*Get the alignment in zero LSBs needed for the descriptor table.*
- static unsigned long [avail\\_size](#) (unsigned [num](#))  
*Calculate the size of the available ring for *num* entries.*
- static unsigned long [avail\\_align](#) ()  
*Get the alignment in zero LSBs needed for the available ring.*
- static unsigned long [used\\_size](#) (unsigned [num](#))  
*Calculate the size of the used ring for *num* entries.*
- static unsigned long [used\\_align](#) ()  
*Get the alignment in zero LSBs needed for the used ring.*

## Protected Member Functions inherited from L4virtio::Virtqueue

- [Virtqueue](#) ()=default  
*Create a disabled virtqueue.*

## Protected Attributes inherited from L4virtio::Virtqueue

- [Desc](#) \* [\\_desc](#) = nullptr  
*pointer to descriptor table, NULL if queue is off.*
- [Avail](#) \* [\\_avail](#) = nullptr  
*pointer to available ring.*
- [Used](#) \* [\\_used](#) = nullptr  
*pointer to used ring.*
- [l4\\_uint16\\_t](#) [\\_current\\_avail](#) = 0  
*The life counter for the queue.*
- [l4\\_uint16\\_t](#) [\\_idx\\_mask](#) = 0  
*mask used for indexing into the descriptor table and the rings.*

### 15.384.1 Detailed Description

Driver-side implementation of a [Virtqueue](#).

Adds function for managing the descriptor list, enqueueing new and dequeueing finished requests.

#### Note

The [Virtqueue](#) implementation is not thread-safe.

Definition at line 476 of file [virtqueue](#).

## 15.384.2 Member Function Documentation

### 15.384.2.1 alloc\_descriptor()

```
l4_uint16_t L4virtio::Driver::Virtqueue::alloc_descriptor ( ) [inline]
```

Allocate and return an unused descriptor from the descriptor table.

The descriptor will be removed from the free list, the content should be considered undefined. After use, it needs to be freed using [free\\_descriptor\(\)](#).

#### Returns

The index of the reserved descriptor or Virtqueue::Eoq if no free descriptor is available.

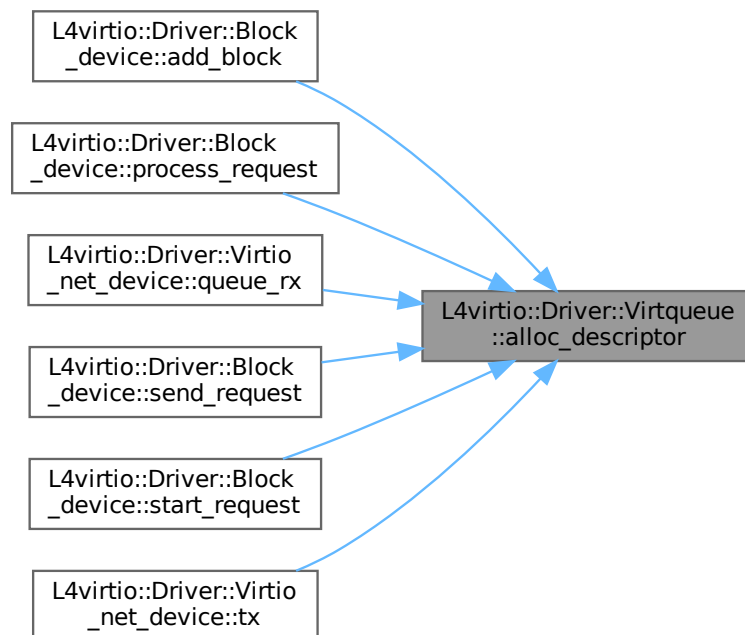
Note: the implementation uses  $(2^{16} - 1)$  as the end of queue marker. That means that the final entry in the queue can not be allocated iff the queue size is  $2^{16}$ .

Definition at line 564 of file [virtqueue](#).

References [L4virtio::Virtqueue::\\_desc](#), and [L4virtio::Virtqueue::Desc::next](#).

Referenced by [L4virtio::Driver::Block\\_device::add\\_block\(\)](#), [L4virtio::Driver::Block\\_device::process\\_request\(\)](#), [L4virtio::Driver::Virtio\\_net\\_device::queue\\_rx\(\)](#), [L4virtio::Driver::Block\\_device::send\\_request\(\)](#), [L4virtio::Driver::Block\\_device::start\\_request\(\)](#) and [L4virtio::Driver::Virtio\\_net\\_device::tx\(\)](#).

Here is the caller graph for this function:



### 15.384.2.2 desc()

```
Desc & L4virtio::Driver::Virtqueue::desc (
    l4_uint16_t descno ) [inline]
```

Return a reference to a descriptor in the descriptor table.

## Parameters

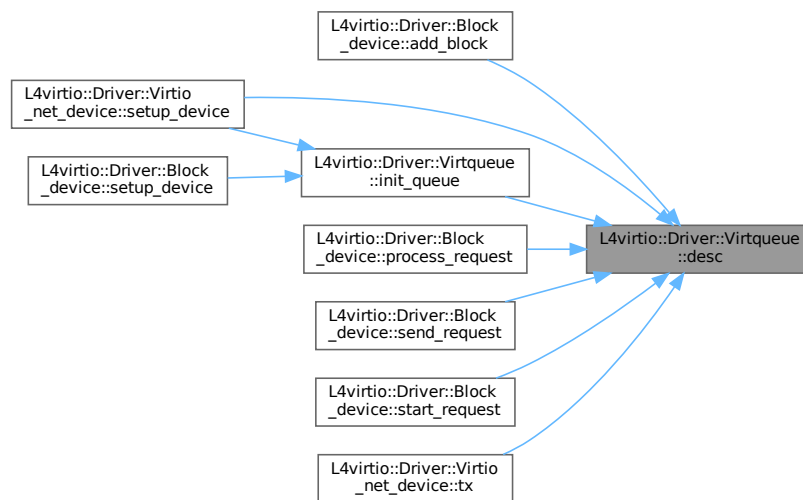
<i>descno</i>	Index of the descriptor, expected to be in correct range.
---------------	---

Definition at line 596 of file [virtqueue](#).

References [L4virtio::Virtqueue::\\_desc](#), and [L4virtio::Virtqueue::\\_idx\\_mask](#).

Referenced by [L4virtio::Driver::Block\\_device::add\\_block\(\)](#), [init\\_queue\(\)](#), [L4virtio::Driver::Block\\_device::process\\_request\(\)](#), [L4virtio::Driver::Block\\_device::send\\_request\(\)](#), [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#), [L4virtio::Driver::Block\\_device::start\\_request\(\)](#), and [L4virtio::Driver::Virtio\\_net\\_device::tx\(\)](#).

Here is the caller graph for this function:



### 15.384.2.3 enqueue\_descriptor()

```
void L4virtio::Driver::Virtqueue::enqueue_descriptor (
    14_uint16_t descno ) [inline]
```

Enqueue a descriptor in the available ring.

## Parameters

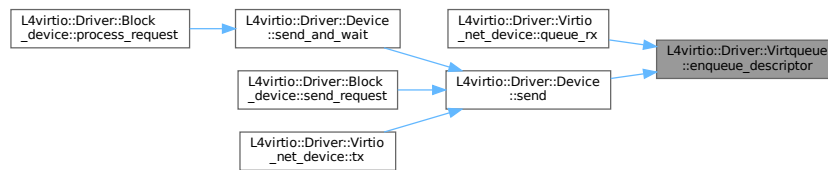
<i>descno</i>	Index of the head descriptor to enqueue.
---------------	--

Definition at line 580 of file [virtqueue](#).

References [L4virtio::Virtqueue::\\_avail](#), [L4virtio::Virtqueue::\\_idx\\_mask](#), [L4virtio::Virtqueue::Avail::idx](#), and [L4virtio::Virtqueue::Avail::ring](#).

Referenced by [L4virtio::Driver::Virtio\\_net\\_device::queue\\_rx\(\)](#), and [L4virtio::Driver::Device::send\(\)](#).

Here is the caller graph for this function:



#### 15.384.2.4 find\_next\_used()

```

14_uint16_t L4virtio::Driver::Virtqueue::find_next_used (
    14_uint32_t * len = nullptr ) [inline]
  
```

Return the next finished block.

##### Parameters

out	len	(optional) Size of valid data in finished block. Note that this is the value reported by the device, which may set it to a value that is larger than the original buffer size.
-----	-----	--

##### Returns

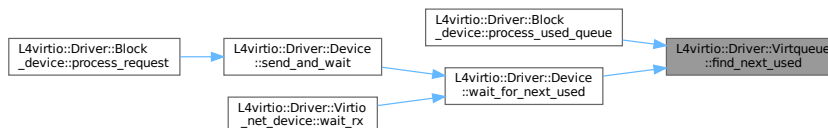
Index of the head or Virtqueue::Eoq if no used element is currently available.

Definition at line 615 of file [virtqueue](#).

References [L4virtio::Virtqueue::\\_current\\_avail](#), [L4virtio::Virtqueue::\\_idx\\_mask](#), [L4virtio::Virtqueue::\\_used](#), [L4virtio::Virtqueue::Used::idx](#), [L4virtio::Virtqueue::Used::elem::len](#), and [L4virtio::Virtqueue::Used::ring](#).

Referenced by [L4virtio::Driver::Block\\_device::process\\_used\\_queue\(\)](#), and [L4virtio::Driver::Device::wait\\_for\\_next\\_used\(\)](#).

Here is the caller graph for this function:



#### 15.384.2.5 free\_descriptor()

```

void L4virtio::Driver::Virtqueue::free_descriptor (
    14_uint16_t head,
    14_uint16_t tail ) [inline]
  
```

Free a chained list of descriptors in the descriptor queue.

## Parameters

<i>head</i>	Index of the first element in the descriptor chain.
<i>tail</i>	Index of the last element in the descriptor chain.

Simply takes the descriptor chain and prepends it to the beginning of the free list. Assumes that the list has been correctly chained.

Definition at line 637 of file [virtqueue](#).

References [L4virtio::Virtqueue::\\_desc](#), [L4virtio::Virtqueue::\\_idx\\_mask](#), and [L4virtio::Virtqueue::Desc::next](#).

Referenced by [L4virtio::Driver::Virtio\\_net\\_device::finish\\_rx\(\)](#).

Here is the caller graph for this function:



### 15.384.2.6 init\_queue() [1/2]

```
void L4virtio::Driver::Virtqueue::init_queue (
    unsigned num,
    void * base ) [inline]
```

Initialize this virtqueue.

## Parameters

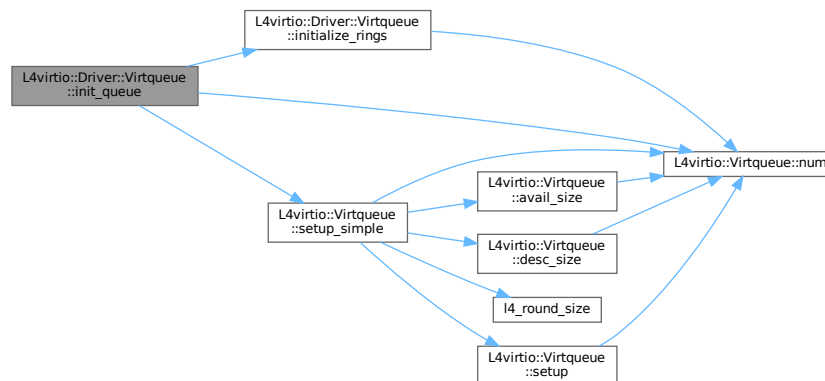
<i>num</i>	The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).
<i>base</i>	The base address for the queue data structure.

This function sets up the memory and initializes the freelist.

Definition at line 543 of file [virtqueue](#).

References [initialize\\_rings\(\)](#), [L4virtio::Virtqueue::num\(\)](#), and [L4virtio::Virtqueue::setup\\_simple\(\)](#).

Here is the call graph for this function:



### 15.384.2.7 init\_queue() [2/2]

```
void L4virtio::Driver::Virtqueue::init_queue (
    unsigned num,
    void * desc,
    void * avail,
    void * used ) [inline]
```

Initialize this virtqueue.

#### Parameters

<i>num</i>	The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).
<i>desc</i>	The address of the descriptor table. (Must be Desc_align aligned and at least desc_size(num) bytes in size.)
<i>avail</i>	The address of the available ring. (Must be Avail_align aligned and at least avail_size(num) bytes in size.)
<i>used</i>	The address of the used ring. (Must be Used_align aligned and at least used_size(num) bytes in size.)

This function sets up the memory and initializes the freelist.

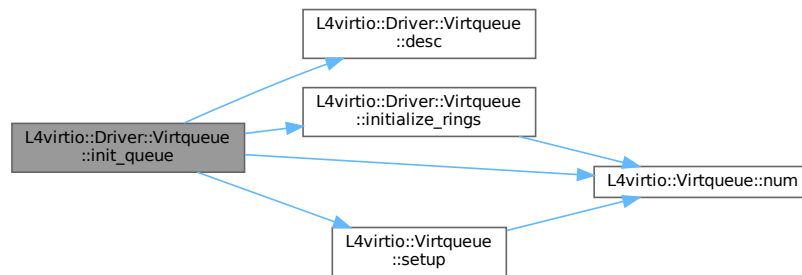
Definition at line 528 of file [virtqueue](#).

References [desc\(\)](#), [initialize\\_rings\(\)](#), [L4virtio::Virtqueue::num\(\)](#), and [L4virtio::Virtqueue::setup\(\)](#).

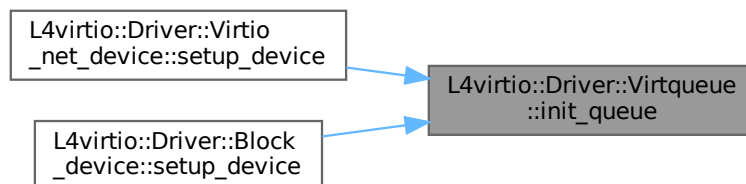
Referenced by [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#), and [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#).



Here is the call graph for this function:



Here is the caller graph for this function:



### 15.384.2.8 initialize\_rings()

```
void L4virtio::Driver::Virtqueue::initialize_rings (
    unsigned num ) [inline]
```

Initialize the descriptor table and the index structures of this queue.

#### Parameters

<i>num</i>	The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).
------------	--

#### Precondition

The queue must be set up correctly with [setup\(\)](#) or [setup\\_simple\(\)](#).

Definition at line 500 of file [virtqueue](#).

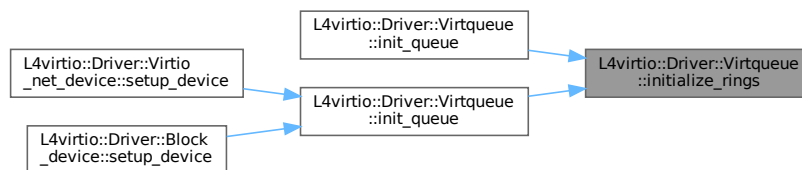
References [L4virtio::Virtqueue::\\_avail](#), [L4virtio::Virtqueue::\\_desc](#), [L4virtio::Virtqueue::\\_used](#), [L4virtio::Virtqueue::Avail::idx](#), [L4virtio::Virtqueue::Used::idx](#), [L4virtio::Virtqueue::Desc::next](#), and [L4virtio::Virtqueue::num\(\)](#).

Referenced by [init\\_queue\(\)](#), and [init\\_queue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

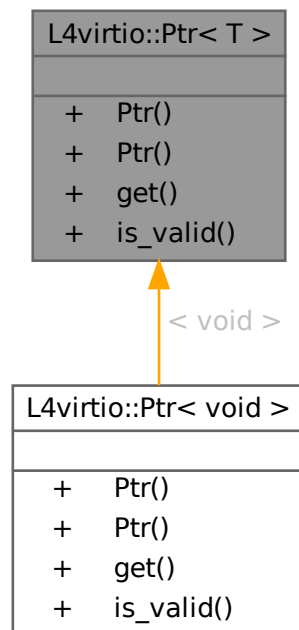
- `I4/I4virtio/virtqueue`

## 15.385 L4virtio::Ptr< T > Class Template Reference

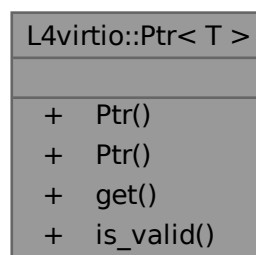
Pointer used in virtio descriptors.

```
#include <virtqueue>
```

Inheritance diagram for L4virtio::Ptr< T >:



Collaboration diagram for L4virtio::Ptr< T >:



## Public Types

- enum `Invalid_type` { `Invalid` }  
Type for making an invalid (NULL) `Ptr`.

## Public Member Functions

- **Ptr** ([Invalid\\_type](#))  
*Make and invalid [Ptr](#).*
- **Ptr** ([l4\\_uint64\\_t](#) vm\_addr)  
*Make a [Ptr](#) from a raw 64bit address.*
- [l4\\_uint64\\_t](#) **get** () const
- bool [is\\_valid](#) () const

### 15.385.1 Detailed Description

```
template<typename T>
class L4virtio::Ptr< T >
```

Pointer used in virtio descriptors.

As the descriptor contain guest addresses these pointers cannot be dereferenced directly.

Definition at line [54](#) of file [virtqueue](#).

### 15.385.2 Member Enumeration Documentation

#### 15.385.2.1 Invalid\_type

```
template<typename T >
enum L4virtio::Ptr::Invalid_type
```

Type for making an invalid (NULL) [Ptr](#).

Enumerator

Invalid	Use to set a <a href="#">Ptr</a> to invalid (NULL)
---------	--

Definition at line [58](#) of file [virtqueue](#).

### 15.385.3 Member Function Documentation

#### 15.385.3.1 get()

```
template<typename T >
l4_uint64_t L4virtio::Ptr< T >::get ( ) const [inline]
```

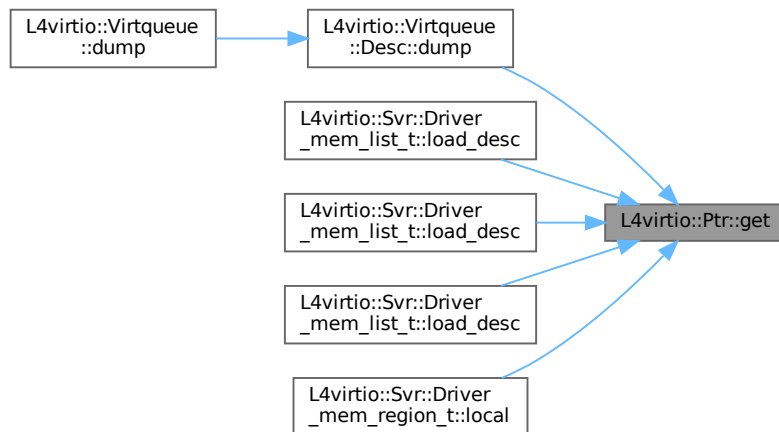
**Returns**

The raw 64bit address of the stored pointer.

Definition at line 69 of file [virtqueue](#).

Referenced by [L4virtio::Virtqueue::Desc::dump\(\)](#), [L4virtio::Svr::Driver\\_mem\\_list\\_t< DATA >::load\\_desc\(\)](#), [L4virtio::Svr::Driver\\_mem\\_list\\_t< DATA >::load\\_desc\(\)](#), [L4virtio::Svr::Driver\\_mem\\_list\\_t< DATA >::load\\_desc\(\)](#), and [L4virtio::Svr::Driver\\_mem\\_region\\_t< DATA >::local\(\)](#).

Here is the caller graph for this function:

**15.385.3.2 is\_valid()**

```
template<typename T >
bool L4virtio::Ptr< T >::is_valid ( ) const [inline]
```

**Returns**

true if the stored pointer is valid (not NULL).

Definition at line 72 of file [virtqueue](#).

The documentation for this class was generated from the following file:

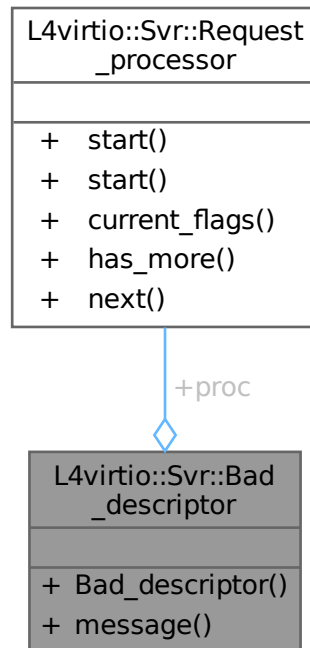
- `l4/l4virtio/virtqueue`

## 15.386 L4virtio::Svr::Bad\_descriptor Struct Reference

Exception used by Queue to indicate descriptor errors.

```
#include <virtio>
```

Collaboration diagram for L4virtio::Svr::Bad\_descriptor:



### Public Types

- enum [Error](#) {  
[Bad\\_address](#) , [Bad\\_rights](#) , [Bad\\_flags](#) , [Bad\\_next](#) ,  
[Bad\\_size](#) }

*The error code.*

### Public Member Functions

- [Bad\\_descriptor](#) ([Request\\_processor](#) const \*[proc](#), [Error](#) e)  
*Make a bad descriptor exception.*
- char const \* [message](#) () const  
*Get a human readable description of the error code.*

### Data Fields

- [Request\\_processor](#) const \* **proc**  
*The processor that triggered the exception.*

## 15.386.1 Detailed Description

Exception used by Queue to indicate descriptor errors.

Definition at line 378 of file [virtio](#).

## 15.386.2 Member Enumeration Documentation

### 15.386.2.1 Error

```
enum L4virtio::Svr::Bad_descriptor::Error
```

The error code.

Enumerator

Bad_address	Address cannot be translated.
Bad_rights	Missing access rights on memory.
Bad_flags	Invalid combination of descriptor flags.
Bad_next	Invalid next index.
Bad_size	Invalid size of memory block.

Definition at line 381 of file [virtio](#).

## 15.386.3 Constructor & Destructor Documentation

### 15.386.3.1 Bad\_descriptor()

```
L4virtio::Svr::Bad_descriptor::Bad_descriptor (
    Request_processor const * proc,
    Error e ) [inline]
```

Make a bad descriptor exception.

Parameters

<i>proc</i>	The request processor causing the exception
<i>e</i>	The error code.

Definition at line 402 of file [virtio](#).

## 15.386.4 Member Function Documentation

### 15.386.4.1 message()

```
char const * L4virtio::Svr::Bad_descriptor::message ( ) const [inline]
```

Get a human readable description of the error code.

**Returns**

Message describing the error.

Definition at line 411 of file [virtio](#).

References [Bad\\_address](#), [Bad\\_flags](#), [Bad\\_next](#), [Bad\\_rights](#), and [Bad\\_size](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/server/virtio

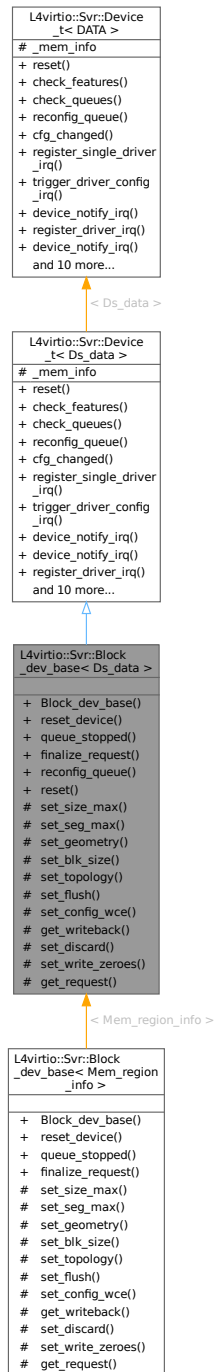
## 15.387 L4virtio::Svr::Block\_dev\_base< Ds\_data > Class Template Reference

Base class for virtio block devices.

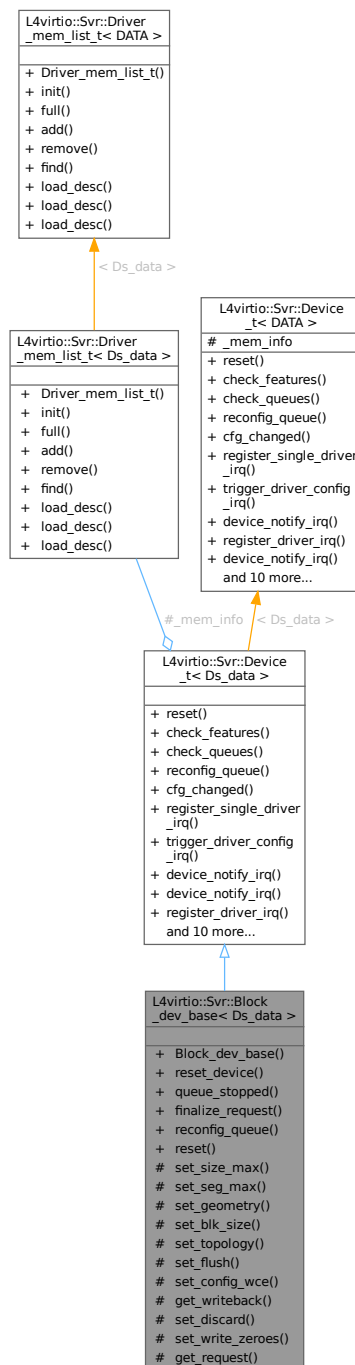
```
#include <virtio-block>
```



Inheritance diagram for L4virtio::Svr::Block\_dev\_base< Ds\_data >:



Collaboration diagram for L4virtio::Svr::Block\_dev\_base< Ds\_data >:



## Public Member Functions

- **Block\_dev\_base** (l4\_uint32\_t vendor, unsigned queue\_size, l4\_uint64\_t capacity, bool read\_only)  
Create a new virtio block device.
- virtual void **reset\_device** ()=0  
Reset the actual hardware device.
- virtual bool **queue\_stopped** ()=0

- *Return true, if the queues should not be processed further.*
- void **finalize\_request** (cxx::unique\_ptr< Request > req, unsigned sz, [l4\\_uint8\\_t](#) status=L4VIRTIO\_BLOCK\_S\_OK)  
*Releases resources related to a request and notifies the client.*
- int **reconfig\_queue** (unsigned idx) override  
*callback for client queue-config request*
- void **reset** () override  
*reset callback, called for doing a device reset*

## Public Member Functions inherited from [L4virtio::Svr::Device\\_t< Ds\\_data >](#)

- virtual bool **check\_features** ()  
*callback for checking the subset of accepted features*
- virtual void **cfg\_changed** (unsigned)  
*callback for client device configuration changes*
- virtual [L4::Cap](#)< [L4::Irq](#) > **device\_notify\_irq** () const  
*callback to gather the device notification IRQ (old-style)*
- virtual [L4::Cap](#)< [L4::Irq](#) > **device\_notify\_irq** (unsigned idx)  
*Callback to gather the device notification IRQ (multi IRQ).*
- virtual void **register\_driver\_irq** (unsigned idx)  
*Callback for registering an notification IRQ (multi IRQ).*
- virtual unsigned **num\_events\_supported** () const  
*Return the highest notification index supported.*
- [Device\\_t](#) ([Dev\\_config](#) \*dev\_config)  
*Make a device for the given config.*
- [Mem\\_list](#) const \* **mem\_info** () const  
*Get the memory region list used for this device.*
- void **reset\_queue\_config** (unsigned idx, unsigned num\_max, bool inc\_generation=false)  
*Trigger reset for the configuration space for queue idx.*
- void **init\_mem\_info** (unsigned num)  
*Initialize the memory region list to the given maximum.*
- void **device\_error** ()  
*Transition device into DEVICE\_NEEDS\_RESET state.*
- bool **setup\_queue** ([Virtqueue](#) \*q, unsigned qn, unsigned num\_max)  
*Enable/disable the specified queue.*
- bool **handle\_mem\_cmd\_write** ()  
*Check for a value in the cmd register and handle a write.*
- void **enable\_trusted\_ds\_validation** ()  
*Enable trusted dataspace validation.*
- void **add\_trusted\_dataspaces** (std::shared\_ptr< [Ds\\_vector](#) const > ds)  
*Provide a list of trusted dataspaces that can be used for validation.*

## Protected Member Functions

- void **set\_size\_max** ([l4\\_uint32\\_t](#) sz)  
*Sets the maximum size of any single segment reported to client.*
- void **set\_seg\_max** ([l4\\_uint32\\_t](#) sz)  
*Sets the maximum number of segments in a request that is reported to client.*
- void **set\_geometry** ([l4\\_uint16\\_t](#) cylinders, [l4\\_uint8\\_t](#) heads, [l4\\_uint8\\_t](#) sectors)  
*Set disk geometry that is reported to the client.*

- void `set_blk_size` (`l4_uint32_t` sz)  
*Sets block disk size to be reported to the client.*
- void `set_topology` (`l4_uint8_t` physical\_block\_exp, `l4_uint8_t` alignment\_offset, `l4_uint32_t` min\_io\_size, `l4_uint32_t` opt\_io\_size)  
*Sets the I/O alignment information reported back to the client.*
- void `set_flush` ()  
*Enables the flush command.*
- void `set_config_wce` (`l4_uint8_t` writeback)  
*Sets cache mode and enables the writeback toggle.*
- `l4_uint8_t` `get_writeback` ()  
*Get the writeback field from the configuration space.*
- void `set_discard` (`l4_uint32_t` max\_discard\_sectors, `l4_uint32_t` max\_discard\_seg, `l4_uint32_t` discard\_↔ sector\_alignment)  
*Sets constraints for and enables the discard command.*
- void `set_write_zeroes` (`l4_uint32_t` max\_write\_zeroes\_sectors, `l4_uint32_t` max\_write\_zeroes\_seg, `l4_uint8_t` write\_zeroes\_may\_unmap)  
*Sets constraints for and enables the write zeroes command.*
- `cxx::unique_ptr< Request >` `get_request` ()  
*Return one request if available.*

### Additional Inherited Members

### Protected Attributes inherited from `L4virtio::Svr::Device_t< Ds_data >`

- `Mem_list _mem_info`  
*Memory region list.*

## 15.387.1 Detailed Description

```
template<typename Ds_data>
class L4virtio::Svr::Block_dev_base< Ds_data >
```

Base class for virtio block devices.

Use this class as a base to implement your own specific block device.

Definition at line 257 of file [virtio-block](#).

## 15.387.2 Constructor & Destructor Documentation

### 15.387.2.1 `Block_dev_base()`

```
template<typename Ds_data >
L4virtio::Svr::Block_dev_base< Ds_data >::Block_dev_base (
    l4_uint32_t vendor,
    unsigned queue_size,
    l4_uint64_t capacity,
    bool read_only ) [inline]
```

Create a new virtio block device.

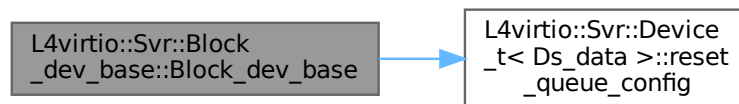
## Parameters

<i>vendor</i>	Vendor ID
<i>queue_size</i>	Number of entries to provide in avail and used queue.
<i>capacity</i>	Size of the device in 512-byte sectors.
<i>read_only</i>	True, if the device should not be writable.

Definition at line 442 of file [virtio-block](#).

References [L4VIRTIO\\_FEATURE\\_VERSION\\_1](#), and [L4virtio::Svr::Device\\_t< Ds\\_data >::reset\\_queue\\_config\(\)](#).

Here is the call graph for this function:



## 15.387.3 Member Function Documentation

### 15.387.3.1 finalize\_request()

```

template<typename Ds_data >
void L4virtio::Svr::Block_dev_base< Ds_data >::finalize_request (
    cxx::unique_ptr< Request > req,
    unsigned sz,
    l4_uint8_t status = L4VIRTIO_BLOCK_S_OK ) [inline]
  
```

Releases resources related to a request and notifies the client.

## Parameters

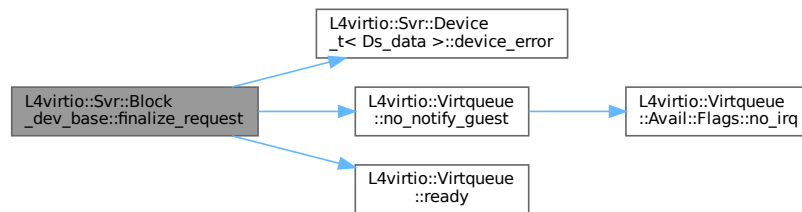
<i>req</i>	Pointer to request that has finished.
<i>sz</i>	Number of bytes consumed.
<i>status</i>	Status of request (see <a href="#">L4virtio_block_status</a> ).

This function must be called when an asynchronous request finishes, either successfully or with an error. The status byte in the request must have been set prior to calling it.

Definition at line 481 of file [virtio-block](#).

References [L4virtio::Svr::Device\\_t< Ds\\_data >::device\\_error\(\)](#), [L4VIRTIO\\_IRQ\\_STATUS\\_VRING](#), [L4virtio::Virtqueue::no\\_notify\\_queue](#) and [L4virtio::Virtqueue::ready\(\)](#).

Here is the call graph for this function:



### 15.387.3.2 get\_writeback()

```
template<typename Ds_data >
l4_uint8_t L4virtio::Svr::Block_dev_base< Ds_data >::get_writeback ( ) [inline], [protected]
```

Get the writeback field from the configuration space.

#### Returns

Value of the writeback field.

Definition at line 386 of file [virtio-block](#).

### 15.387.3.3 set\_blk\_size()

```
template<typename Ds_data >
void L4virtio::Svr::Block_dev_base< Ds_data >::set_blk_size (
    l4_uint32_t sz ) [inline], [protected]
```

Sets block disk size to be reported to the client.

Setting this does not change the logical sector size used for addressing the device.

Definition at line 330 of file [virtio-block](#).

### 15.387.3.4 set\_config\_wce()

```
template<typename Ds_data >
void L4virtio::Svr::Block_dev_base< Ds_data >::set_config_wce (
    l4_uint8_t writeback ) [inline], [protected]
```

Sets cache mode and enables the writeback toggle.

#### Parameters

<i>writeback</i>	Mode of the cache (0 for writethrough, 1 for writeback).
------------------	--

Definition at line 373 of file [virtio-block](#).

### 15.387.3.5 set\_discard()

```
template<typename Ds_data >
void L4virtio::Svr::Block_dev_base< Ds_data >::set_discard (
    14_uint32_t max_discard_sectors,
    14_uint32_t max_discard_seg,
    14_uint32_t discard_sector_alignment ) [inline], [protected]
```

Sets constraints for and enables the discard command.

#### Parameters

<i>max_discard_sectors</i>	Maximum discard sectors size.
<i>max_discard_seg</i>	Maximum discard segment number.
<i>discard_sector_alignment</i>	Can be used by the driver when splitting a request based on alignment.

Definition at line 400 of file [virtio-block](#).

### 15.387.3.6 set\_size\_max()

```
template<typename Ds_data >
void L4virtio::Svr::Block_dev_base< Ds_data >::set_size_max (
    14_uint32_t sz ) [inline], [protected]
```

Sets the maximum size of any single segment reported to client.

The limit is also applied to any incoming requests. Requests with larger segments result in an IO error being reported to the client. That means that `process_request()` can safely make the assumption that all segments in the received request are smaller.

Definition at line 288 of file [virtio-block](#).

### 15.387.3.7 set\_topology()

```
template<typename Ds_data >
void L4virtio::Svr::Block_dev_base< Ds_data >::set_topology (
    14_uint8_t physical_block_exp,
    14_uint8_t alignment_offset,
    14_uint32_t min_io_size,
    14_uint32_t opt_io_size ) [inline], [protected]
```

Sets the I/O alignment information reported back to the client.

#### Parameters

<i>physical_block_exp</i>	Number of logical blocks per physical block(log2)
<i>alignment_offset</i>	Offset of the first aligned logical block
<i>min_io_size</i>	Suggested minimum I/O size in blocks
<i>opt_io_size</i>	Optimal I/O size in blocks

Definition at line 346 of file [virtio-block](#).

### 15.387.3.8 set\_write\_zeroes()

```
template<typename Ds_data >
void L4virtio::Svr::Block_dev_base< Ds_data >::set_write_zeroes (
    14_uint32_t max_write_zeroes_sectors,
    14_uint32_t max_write_zeroes_seg,
    14_uint8_t write_zeroes_may_unmap ) [inline], [protected]
```

Sets constraints for and enables the write zeroes command.

#### Parameters

<i>max_write_zeroes_sectors</i>	Maximum write zeroes sectors size.
<i>max_write_zeroes_seg</i>	maximum write zeroes segment number.
<i>write_zeroes_may_unmap</i>	Set if a write zeroes request can result in deallocating one or more sectors.

Definition at line 420 of file [virtio-block](#).

The documentation for this class was generated from the following file:

- l4/l4virtio/server/virtio-block

## 15.388 L4virtio::Svr::Block\_request< Ds\_data > Class Template Reference

A request to read or write data.

```
#include <virtio-block>
```

Collaboration diagram for L4virtio::Svr::Block\_request< Ds\_data >:

L4virtio::Svr::Block_request< Ds_data >
<ul style="list-style-type: none"> <li>+ data_size()</li> <li>+ has_more()</li> <li>+ next_block()</li> <li>+ header()</li> </ul>



## Public Member Functions

- unsigned [data\\_size](#) () const  
*Compute the total size of the data in the request.*
- bool **has\_more** ()  
*Check if the request contains more data blocks.*
- Data\_block [next\\_block](#) ()  
*Return next block in scatter-gather list.*
- [l4virtio\\_block\\_header\\_t](#) const & **header** () const  
*Return the block request header.*

## 15.388.1 Detailed Description

```
template<typename Ds_data>
class L4virtio::Svr::Block_request< Ds_data >
```

A request to read or write data.

Definition at line 28 of file [virtio-block](#).

## 15.388.2 Member Function Documentation

### 15.388.2.1 data\_size()

```
template<typename Ds_data >
unsigned L4virtio::Svr::Block_request< Ds_data >::data_size ( ) const [inline]
```

Compute the total size of the data in the request.

#### Return values

<i>Size</i>	in bytes or 0 if there was an error.
-------------	--------------------------------------

#### Exceptions

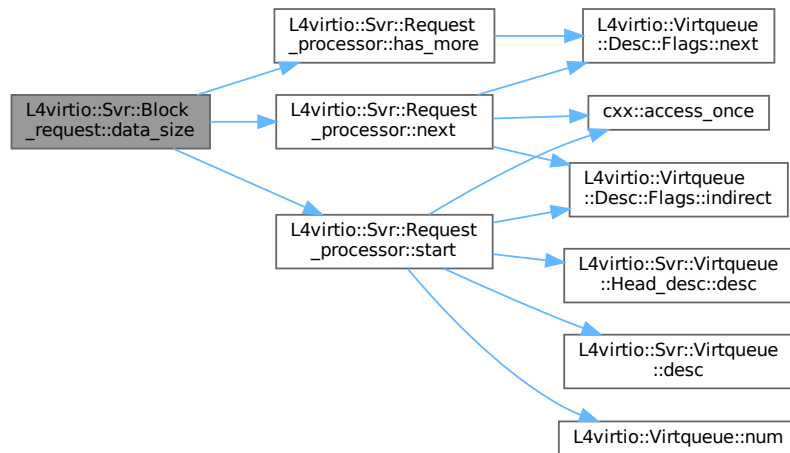
<a href="#">L4::Runtime_error(-L4_EIO)</a>	Request has a bad format.
--	---------------------------

Note that this operation is relatively expensive as it has to iterate over the complete list of blocks.

Definition at line 63 of file [virtio-block](#).

References [L4virtio::Svr::Request\\_processor::has\\_more\(\)](#), [L4\\_EIO](#), [L4virtio::Svr::Request\\_processor::next\(\)](#), and [L4virtio::Svr::Request\\_processor::start\(\)](#).

Here is the call graph for this function:



### 15.388.2.2 next\_block()

```
template<typename Ds_data >
Data_block L4virtio::Svr::Block_request< Ds_data >::next_block ( ) [inline]
```

Return next block in scatter-gather list.

#### Returns

Information about the next data block.

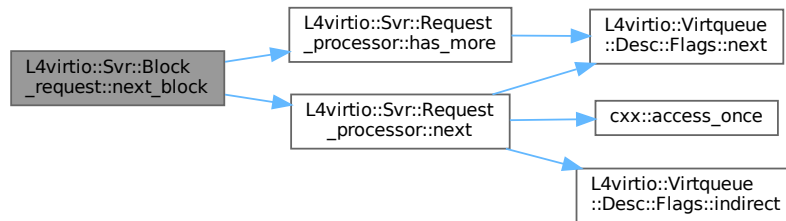
#### Exceptions

<a href="#"><i>L4::Runtime_error</i></a>	No more data block is available.
<a href="#"><i>Bad_descriptor</i></a>	Virtio request is corrupted.

Definition at line 113 of file [virtio-block](#).

References [L4virtio::Svr::Bad\\_descriptor::Bad\\_size](#), [L4virtio::Svr::Request\\_processor::has\\_more\(\)](#), [L4\\_EEXIST](#), and [L4virtio::Svr::Request\\_processor::next\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

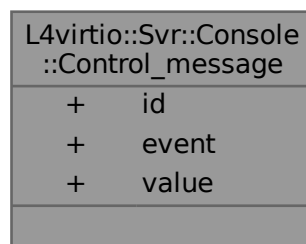
- l4/l4virtio/server/virtio-block

## 15.389 L4virtio::Svr::Console::Control\_message Struct Reference

Virtio console control message.

```
#include <virtio-console>
```

Collaboration diagram for L4virtio::Svr::Console::Control\_message:



### Public Types

- enum [Events](#) {  
[Device\\_ready](#) = 0 , [Device\\_add](#) = 1 , [Device\\_remove](#) = 2 , [Port\\_ready](#) = 3 ,  
[Console\\_port](#) = 4 , [Resize](#) = 5 , [Port\\_open](#) = 6 , [Port\\_name](#) = 7 }

*Possible control events.*

## Data Fields

- [l4\\_uint32\\_t id](#)  
*Port number.*
- [l4\\_uint16\\_t event](#)  
*Control event, see [Events](#).*
- [l4\\_uint16\\_t value](#)  
*Extra information.*

### 15.389.1 Detailed Description

Virtio console control message.

Definition at line 31 of file [virtio-console](#).

### 15.389.2 Member Enumeration Documentation

#### 15.389.2.1 Events

```
enum L4virtio::Svr::Console::Control_message::Events
```

Possible control events.

#### Enumerator

Device_ready	Sent by driver at initialization.
Device_add	Sent by device to create new ports.
Device_remove	Sent by device to remove added ports.
Port_ready	Sent by driver as response to Device_add.
Console_port	Sent by device to nominate port as console port.
Resize	Sent by device to indicate a console size change.
Port_open	Sent by device and driver to indicate whether a port is open.
Port_name	Sent by device to tag a port.

Definition at line 34 of file [virtio-console](#).

The documentation for this struct was generated from the following file:

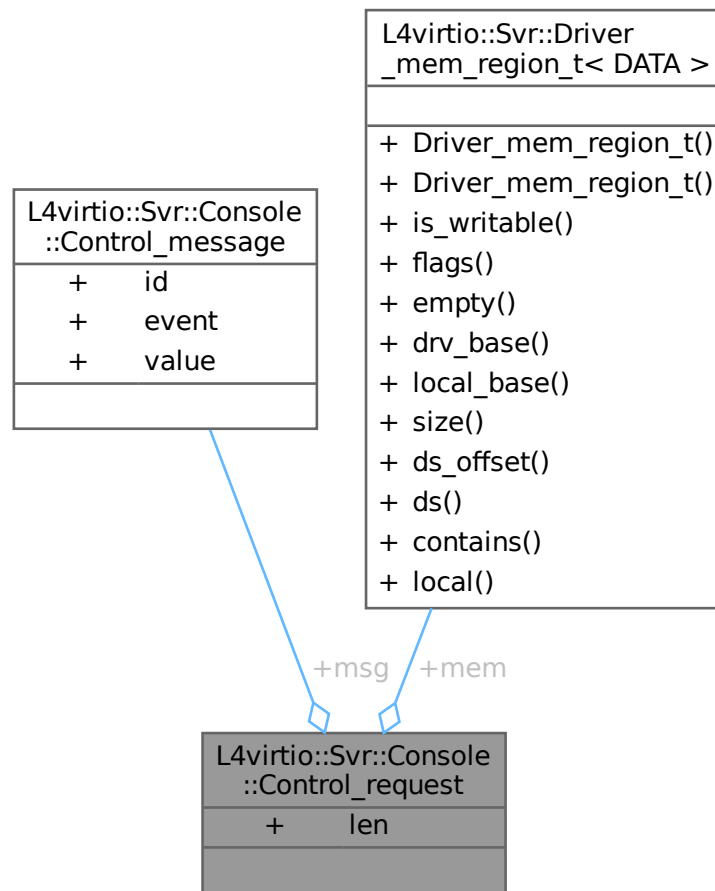
- [l4/l4virtio/server/virtio-console](#)

## 15.390 L4virtio::Svr::Console::Control\_request Struct Reference

Specialised `Virtqueue::Request` providing access to control message payload.

```
#include <virtio-console>
```

Collaboration diagram for L4virtio::Svr::Console::Control\_request:



## Data Fields

- [Control\\_message](#) \* **msg**  
*Virtual address of the data block (in device space).*
- [l4\\_uint32\\_t](#) **len**  
*Length of datablock in bytes.*
- [Driver\\_mem\\_region](#) \* **mem**  
*Pointer to driver memory region.*

## 15.390.1 Detailed Description

Specialised `Virtqueue::Request` providing access to control message payload.

Definition at line 65 of file [virtio-console](#).

The documentation for this struct was generated from the following file:

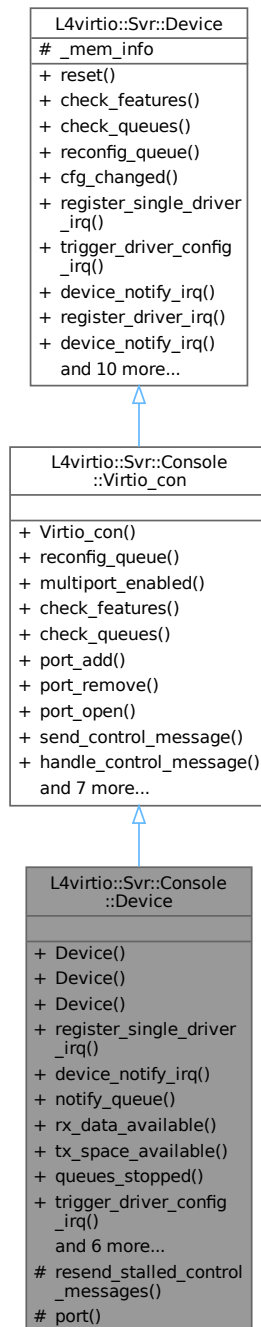
- `l4/l4virtio/server/virtio-console`

## 15.391 L4virtio::Svr::Console::Device Class Reference

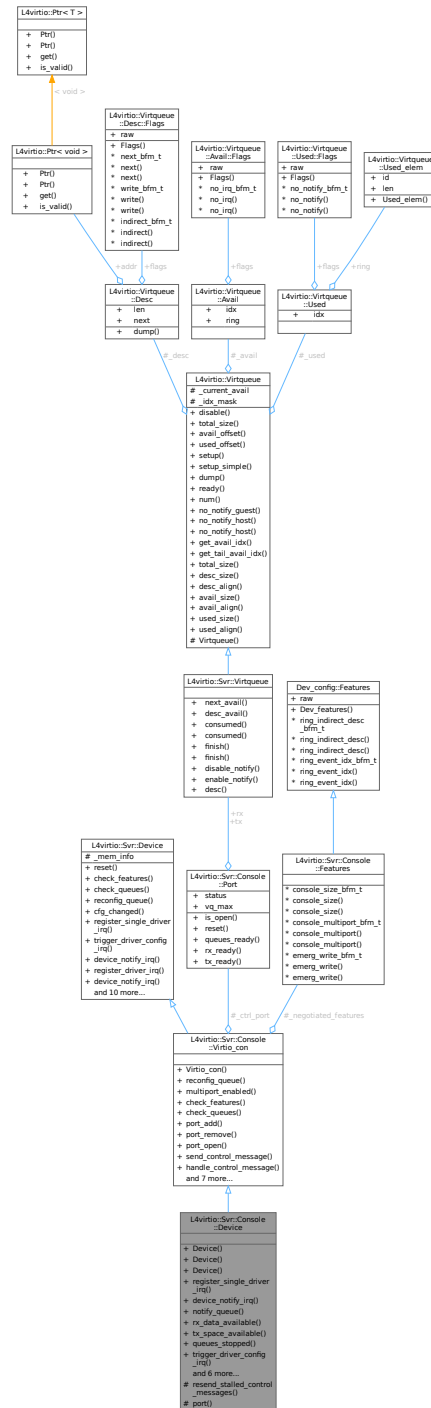
Base class implementing a virtio console device with L4Re-based notification handling.

```
#include <virtio-console-device>
```

Inheritance diagram for L4virtio::Svr::Console::Device:



Collaboration diagram for L4virtio::Svr::Console::Device:



## Public Member Functions

- **Device** (unsigned vq\_max)  
*Create a new console device.*
- **Device** (unsigned vq\_max, unsigned ports)  
*Create a new console device.*
- **Device** (cxx::static\_vector< unsigned > const &vq\_max\_nums)

- Create a new console [Device](#).
- void **register\_single\_driver\_irq** () override  
callback for registering a single guest IRQ for all queues (old-style)
- [L4::Cap](#)< [L4::Irq](#) > **device\_notify\_irq** () const override  
callback to gather the device notification IRQ (old-style)
- void **notify\_queue** ([Virtqueue](#) \*queue) override  
Notify queue of available data.
- virtual void **rx\_data\_available** (unsigned [port](#))=0  
Callback to notify that new data is available to be read from port.
- virtual void **tx\_space\_available** (unsigned [port](#))=0  
Callback to notify that data can be written to port.
- virtual bool **queues\_stopped** ()  
Return true, if the queues should not be processed further.
- void **trigger\_driver\_config\_irq** () override  
callback for triggering configuration change notification IRQ
- unsigned **port\_read** (char \*buf, unsigned len, unsigned [port](#)=0)  
Read data from port.
- unsigned **port\_write** (char const \*buf, unsigned len, unsigned [port](#)=0)  
Write data to port.
- int **do\_control\_work** ()  
Receive new control messages and resend stalled ones.
- void **process\_device\_ready** ([l4\\_uint16\\_t](#) value) override  
Callback called on `DEVICE_READY` event.
- void **process\_port\_ready** ([l4\\_uint32\\_t](#) id, [l4\\_uint16\\_t](#) value) override  
Callback called on `PORT_READY` event.
- void **process\_port\_open** ([l4\\_uint32\\_t](#) id, [l4\\_uint16\\_t](#) value) override  
Callback called on `PORT_OPEN` event.

## Public Member Functions inherited from [L4virtio::Svr::Console::Virtio\\_con](#)

- [Virtio\\_con](#) (unsigned max\_ports, bool enable\_multiport)  
Create a new multiport console device.
- int **reconfig\_queue** (unsigned index) override  
callback for client queue-config request
- bool **multiport\_enabled** () const  
Return true if the multiport feature is enabled and control queues are available.
- bool **check\_features** (void) override  
callback for checking the subset of accepted features
- bool **check\_queues** () override  
callback for checking if the queues at `DRIVER_OK` transition
- int **port\_add** (unsigned idx)  
Send a `DEVICE_ADD` message and update the internal state.
- int **port\_remove** (unsigned idx)  
Send a `DEVICE_REMOVE` message and update the internal state.
- int **port\_open** (unsigned idx, bool open)  
Send a `PORT_OPEN` message and update the internal state.
- int **send\_control\_message** ([l4\\_uint32\\_t](#) idx, [l4\\_uint16\\_t](#) event, [l4\\_uint16\\_t](#) value=0, const char \*name=0)  
Send control message to driver.
- int **handle\_control\_message** ()  
Handle control message received from the driver.
- void **reset** () override  
reset callback, called for doing a device reset
- virtual void **reset\_device** ()  
Reset the state of the actual console device.



## Public Member Functions inherited from L4virtio::Svr::Device\_t< DATA >

- virtual void **cfg\_changed** (unsigned)  
*callback for client device configuration changes*
- virtual void **register\_driver\_irq** (unsigned idx)  
*Callback for registering an notification IRQ (multi IRQ).*
- virtual L4::Cap< L4::Irq > **device\_notify\_irq** (unsigned idx)  
*Callback to gather the device notification IRQ (multi IRQ).*
- virtual unsigned **num\_events\_supported** () const  
*Return the highest notification index supported.*
- **Device\_t** (Dev\_config \*dev\_config)  
*Make a device for the given config.*
- Mem\_list const \* **mem\_info** () const  
*Get the memory region list used for this device.*
- void **reset\_queue\_config** (unsigned idx, unsigned num\_max, bool inc\_generation=false)  
*Trigger reset for the configuration space for queue idx.*
- void **init\_mem\_info** (unsigned num)  
*Initialize the memory region list to the given maximum.*
- void **device\_error** ()  
*Transition device into DEVICE\_NEEDS\_RESET state.*
- bool **setup\_queue** (Virtqueue \*q, unsigned qn, unsigned num\_max)  
*Enable/disable the specified queue.*
- bool **handle\_mem\_cmd\_write** ()  
*Check for a value in the cmd register and handle a write.*
- void **enable\_trusted\_ds\_validation** ()  
*Enable trusted dataspace validation.*
- void **add\_trusted\_dataspaces** (std::shared\_ptr< Ds\_vector const > ds)  
*Provide a list of trusted dataspaces that can be used for validation.*

## Protected Member Functions

- void **resend\_stalled\_control\_messages** ()  
*Try to resend a stalled control message and update the port and global state accordingly.*
- Port \* **port** (unsigned idx) override  
*Return the specified port.*

## Additional Inherited Members

## Protected Attributes inherited from L4virtio::Svr::Device\_t< DATA >

- Mem\_list **\_mem\_info**  
*Memory region list.*

### 15.391.1 Detailed Description

Base class implementing a virtio console device with L4Re-based notification handling.

This console device is derived from [Virtio\\_con](#) and already includes functionality to handle interrupts and notify drivers. If an interrupt is received, all the necessary interaction with the virtqueues is performed and only the actual data processing has to be done by the derived class. By default all available ports are added and an "open"-request of a port by the driver is automatically acknowledged. The derived class can optionally change this behaviour by overriding [process\\_device\\_ready\(\)](#), [process\\_port\\_ready\(\)](#) and [process\\_port\\_open\(\)](#).

This class provides a stream-based interface to access the port data with edge-triggered notification callbacks. If a port receives data from the driver the derived class is notified with the [rx\\_data\\_available\(\)](#) callback. The actual data can be retrieved by [port\\_read\(\)](#). If there was not enough data to be read, the call will return the available partial data. Only then will the [rx\\_data\\_available\(\)](#) callback be triggered again.

Data on a port may be transmitted by [port\\_write\(\)](#). If there were not enough buffers available, only a part of the data will be transmitted. Once there are new buffers available, the [tx\\_space\\_available\(\)](#) callback will be invoked. This callback will be called again only after a previous [port\\_write\(\)](#) was not able to send all requested data.

Use this class as a base to provide your own high-level console device. You must derive from this class as well as `L4::Epiface_t<..., L4virtio::Device>`. For a working device the `irq_iface()` must be registered too. A typical implementation might look like the following:

```
class My_console
: public L4virtio::Svr::Console::Device,
  public L4::Epiface_t<My_console, L4virtio::Device>
{
public:
    My_console(L4Re::Util::Object_registry *r)
        : L4virtio::Svr::Console::Device(0x100)
    {
        init_mem_info(4);
        L4Re::chkcap(r->register_irq_obj(irq_iface()), "virtio notification IRQ");
    }

    void rx_data_available(unsigned port) override
    {
        // call port_read() to fetch available data
    }

    void tx_space_available(unsigned port) override
    {
        // can call port_write() to send (pending) data
    }
};

My_console console(registry);
registry->register_obj(&console, ...);
```

The maximum number of memory regions ([init\\_mem\\_info\(\)](#)) should correlate with the number of supported ports.

Definition at line 116 of file [virtio-console-device](#).

### 15.391.2 Constructor & Destructor Documentation

#### 15.391.2.1 Device() [1/3]

```
L4virtio::Svr::Console::Device::Device (
    unsigned vq_max ) [inline], [explicit]
```

Create a new console device.

## Parameters

<i>vq_max</i>	Maximum number of buffers in data queues.
---------------	---

Create a console device with no multiport support, i.e. control queues are disabled.

Definition at line 143 of file [virtio-console-device](#).

**15.391.2.2 Device() [2/3]**

```
L4virtio::Svr::Console::Device::Device (
    unsigned vq_max,
    unsigned ports ) [inline], [explicit]
```

Create a new console device.

## Parameters

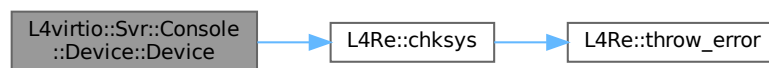
<i>vq_max</i>	Maximum number of buffers in data queues.
<i>ports</i>	Number of ports (maximum 32).

Create a console device with multiport support, i.e. control queues are enabled.

Definition at line 162 of file [virtio-console-device](#).

References [L4Re::chksys\(\)](#), and [L4\\_ENOMEM](#).

Here is the call graph for this function:

**15.391.2.3 Device() [3/3]**

```
L4virtio::Svr::Console::Device::Device (
    cxx::static_vector< unsigned > const & vq_max_nums ) [inline], [explicit]
```

Create a new console [Device](#).

## Parameters

<i>vq_max_nums</i>	Maximum number of buffers in data queues, given as a <a href="#">cxx::static_vector</a> with one entry per port.
--------------------	--

Create a console device with multiport support, i.e. control queues are enabled.

Definition at line 184 of file [virtio-console-device](#).

References [L4Re::chksys\(\)](#), and [L4\\_ENOMEM](#).

Here is the call graph for this function:



### 15.391.3 Member Function Documentation

#### 15.391.3.1 do\_control\_work()

```
int L4virtio::Svr::Console::Device::do_control_work ( ) [inline]
```

Receive new control messages and resend stalled ones.

Return values

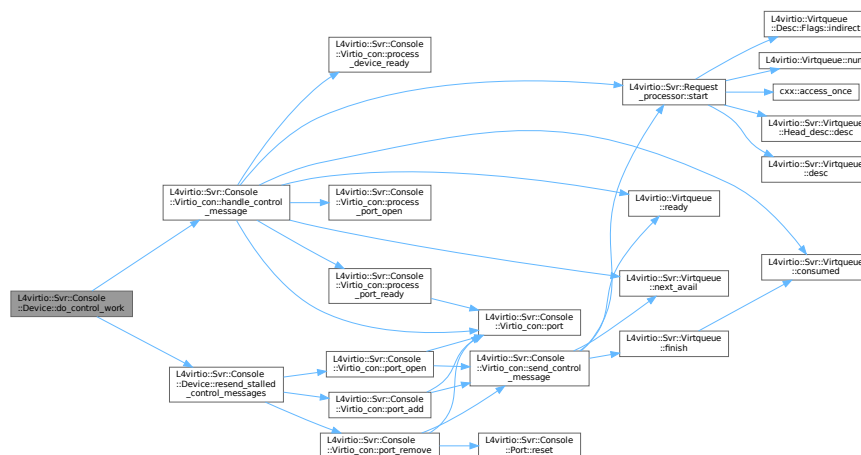
see	<a href="#">handle_control_message()</a>
-----	--

As it may happen that the driver is not able to receive a control message, there may be some stalled control message. This function checks if such messages exist and tries to resend them once more. Afterwards it checks for new messages from the driver, see [handle\\_control\\_message\(\)](#).

Definition at line 400 of file [virtio-console-device](#).

References [L4virtio::Svr::Console::Virtio\\_con::handle\\_control\\_message\(\)](#), and [resend\\_stalled\\_control\\_messages\(\)](#).

Here is the call graph for this function:



### 15.391.3.2 notify\_queue()

```
void L4virtio::Svr::Console::Device::notify_queue (
    Virtqueue * queue ) [inline], [override], [virtual]
```

Notify queue of available data.

#### Parameters

<i>queue</i>	Virtqueue to notify.
--------------	----------------------

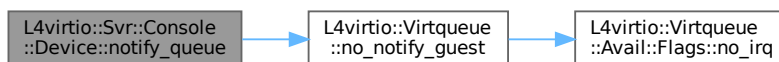
This callback is called whenever data is sent to `queue`. It is the responsibility of the derived class to perform all necessary notification actions, e.g. triggering guest interrupts.

Implements [L4virtio::Svr::Console::Virtio\\_con](#).

Definition at line 208 of file [virtio-console-device](#).

References [L4VIRTIO\\_IRQ\\_STATUS\\_VRING](#), and [L4virtio::Virtqueue::no\\_notify\\_guest\(\)](#).

Here is the call graph for this function:



### 15.391.3.3 port()

```
Port * L4virtio::Svr::Console::Device::port (
    unsigned port ) [inline], [override], [protected], [virtual]
```

Return the specified port.

#### Parameters

<i>port</i>	Port number.
-------------	--------------

#### Precondition

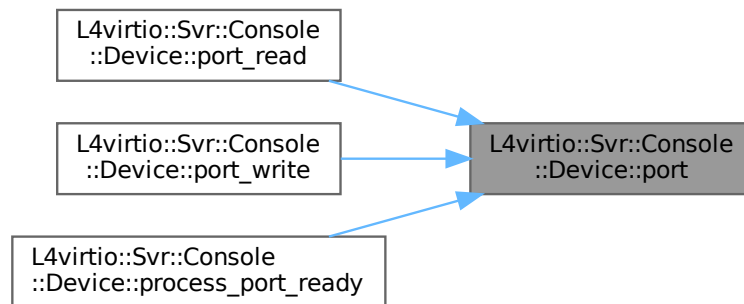
`Port` number must be lower than the configured maximum number of ports.

Implements [L4virtio::Svr::Console::Virtio\\_con](#).

Definition at line 505 of file [virtio-console-device](#).

Referenced by [port\\_read\(\)](#), [port\\_write\(\)](#), and [process\\_port\\_ready\(\)](#).

Here is the caller graph for this function:



### 15.391.3.4 port\_read()

```

unsigned L4virtio::Svr::Console::Device::port_read (
    char * buf,
    unsigned len,
    unsigned port = 0 ) [inline]
  
```

Read data from port.

Will read up to *len* bytes from *port* into *buf*. Returns the number of bytes read, which may be less if not enough data was available. If all data was read, the [rx\\_data\\_available\(\)](#) callback will be invoked the next time the driver queues new data for the port. The callback won't be called again until all data was consumed again.

#### Parameters

<i>buf</i>	The destination buffer
<i>len</i>	Size of the buffer
<i>port</i>	<a href="#">Port</a> index to read data from

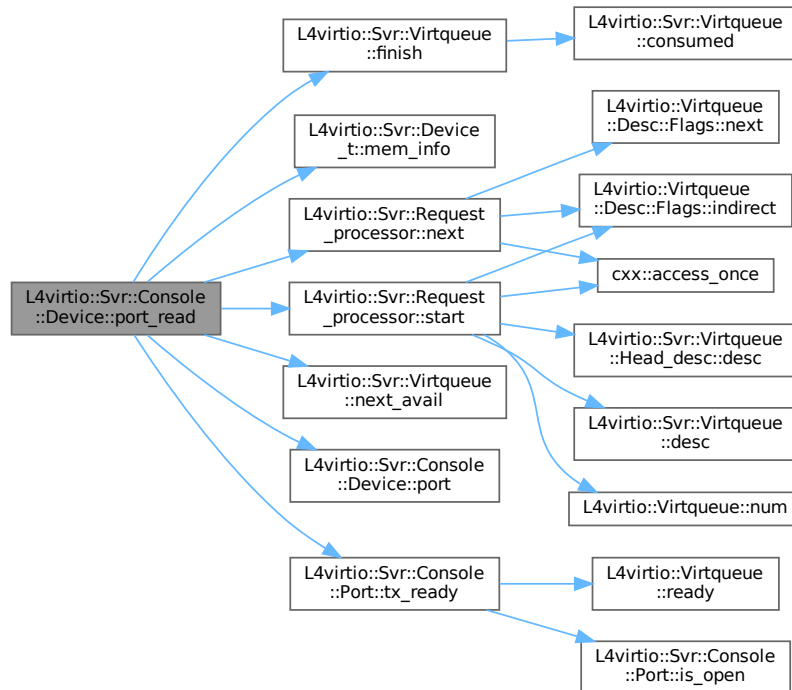
#### Returns

Number of bytes read

Definition at line 278 of file [virtio-console-device](#).

References [L4virtio::Svr::Virtqueue::finish\(\)](#), [L4virtio::Svr::Data\\_buffer::left](#), [L4virtio::Svr::Device\\_t< DATA >::mem\\_info\(\)](#), [L4virtio::Svr::Request\\_processor::next\(\)](#), [L4virtio::Svr::Virtqueue::next\\_avail\(\)](#), [port\(\)](#), [L4virtio::Svr::Data\\_buffer::pos](#), [L4virtio::Svr::Console::Device\\_port::request](#), [L4virtio::Svr::Console::Device\\_port::rp](#), [L4virtio::Svr::Console::Device\\_port::src](#), [L4virtio::Svr::Request\\_processor::start\(\)](#), [L4virtio::Svr::Console::Port::tx](#), and [L4virtio::Svr::Console::Port::tx\\_ready\(\)](#).

Here is the call graph for this function:



### 15.391.3.5 port\_write()

```

unsigned L4virtio::Svr::Console::Device::port_write (
    char const * buf,
    unsigned len,
    unsigned port = 0 ) [inline]

```

Write data to port.

Will write up to *len* bytes to *port* from *buf*. Returns the number of bytes written, which may be less if not enough virtio buffers were available. If not all data could be written, the [tx\\_space\\_available\(\)](#) callback will be invoked the next time the driver queues new receive buffers for the port. The callback won't be called again until all receive buffers were filled again.

#### Parameters

<i>buf</i>	The source buffer
<i>len</i>	Size of the buffer
<i>port</i>	<a href="#">Port</a> index to write data to

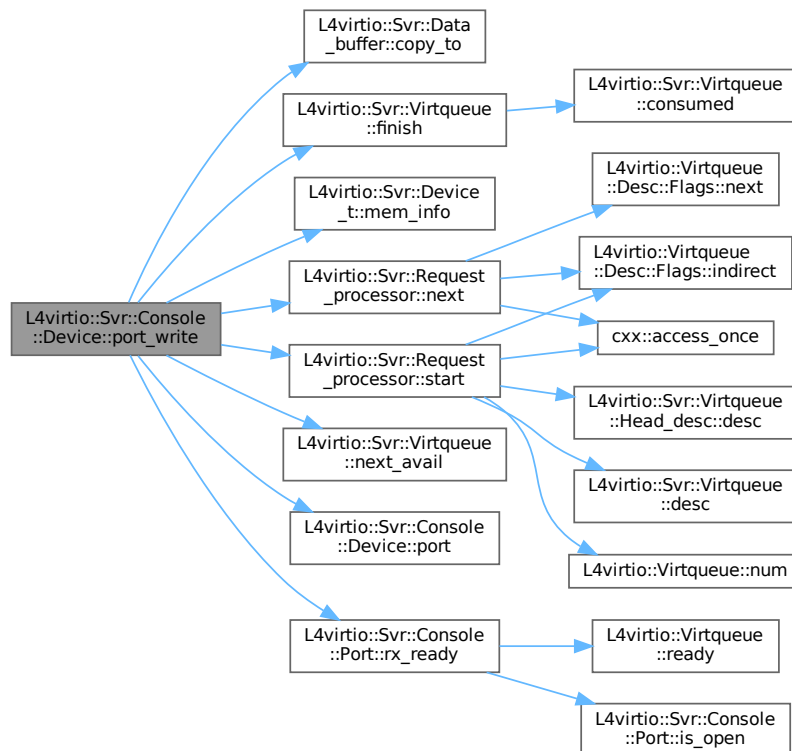
#### Returns

Number of bytes written

Definition at line 345 of file [virtio-console-device](#).

References [L4virtio::Svr::Data\\_buffer::copy\\_to\(\)](#), [L4virtio::Svr::Virtqueue::finish\(\)](#), [L4virtio::Svr::Data\\_buffer::left](#), [L4virtio::Svr::Device\\_t< DATA >::mem\\_info\(\)](#), [L4virtio::Svr::Request\\_processor::next\(\)](#), [L4virtio::Svr::Virtqueue::next\\_avail\(\)](#), [port\(\)](#), [L4virtio::Svr::Data\\_buffer::pos](#), [L4virtio::Svr::Console::Port::rx](#), [L4virtio::Svr::Console::Port::rx\\_ready\(\)](#), and [L4virtio::Svr::Request\\_processor::start\(\)](#).

Here is the call graph for this function:



### 15.391.3.6 process\_device\_ready()

```
void L4virtio::Svr::Console::Device::process_device_ready (
    14_uint16_t value ) [inline], [override], [virtual]
```

Callback called on DEVICE\_READY event.

#### Parameters

<i>value</i>	The value field of the control message, indicating if the initialization was successful.
--------------	--

By default, this function adds all ports if the driver indicates successful initialization. Override this function to perform custom actions for a DEVICE\_READY event, *after* the generic management of the base class. It is then your responsibility to inform the driver about usable ports, see [port\\_add\(\)](#).

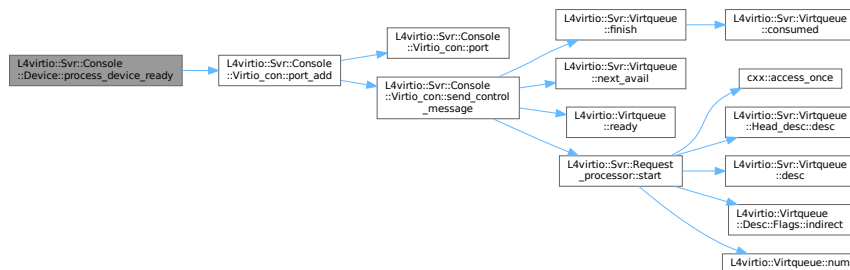
Implements [L4virtio::Svr::Console::Virtio\\_con](#).



Definition at line 426 of file [virtio-console-device](#).

References [L4virtio::Svr::Console::Virtio\\_con::port\\_add\(\)](#).

Here is the call graph for this function:



### 15.391.3.7 process\_port\_open()

```
void L4virtio::Svr::Console::Device::process_port_open (
    14_uint32_t id,
    14_uint16_t value ) [inline], [override], [virtual]
```

Callback called on PORT\_OPEN event.

#### Parameters

<i>id</i>	The id field of the control message, i.e. the port number.
<i>value</i>	The value field of the control message, indicating if the port was opened or closed.

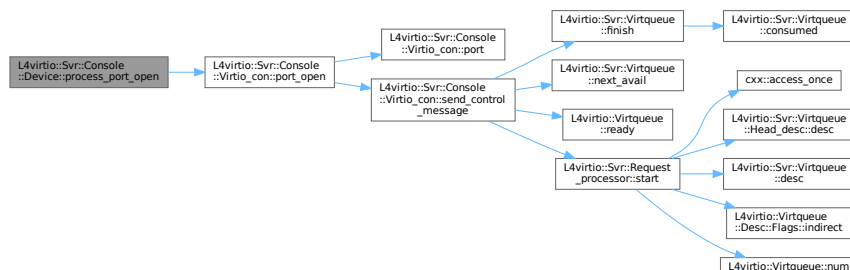
By default, this function acknowledges the message from the driver by sending back the same message. Override this function to perform custom actions for a PORT\_OPEN event, *after* the generic management of the base class. It is then your responsibility to update the internal port state and answer the driver's messages, see [port\\_open\(\)](#).

Implements [L4virtio::Svr::Console::Virtio\\_con](#).

Definition at line 470 of file [virtio-console-device](#).

References [L4virtio::Svr::Console::Virtio\\_con::port\\_open\(\)](#).

Here is the call graph for this function:



### 15.391.3.8 process\_port\_ready()

```
void L4virtio::Svr::Console::Device::process_port_ready (
    14_uint32_t id,
    14_uint16_t value ) [inline], [override], [virtual]
```

Callback called on PORT\_READY event.

#### Parameters

<i>id</i>	The id field of the control message, i.e. the port number.
<i>value</i>	The value field of the control message, indicating if the initialization was successful.

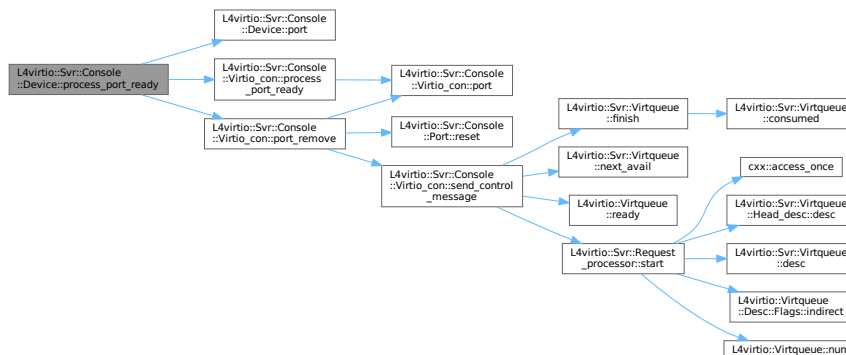
By default, this function removes the port, if the driver failed to set it up correctly. Override this function to perform custom actions for a PORT\_READY event, *after* the generic management of the base class. It is then your responsibility to inform the driver about unusable ports, see [port\\_remove\(\)](#).

Reimplemented from [L4virtio::Svr::Console::Virtio\\_con](#).

Definition at line 448 of file [virtio-console-device](#).

References [port\(\)](#), [L4virtio::Svr::Console::Port::Port\\_failed](#), [L4virtio::Svr::Console::Virtio\\_con::port\\_remove\(\)](#), [L4virtio::Svr::Console::Virtio\\_con::process\\_port\\_ready\(\)](#), and [L4virtio::Svr::Console::Port::status](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

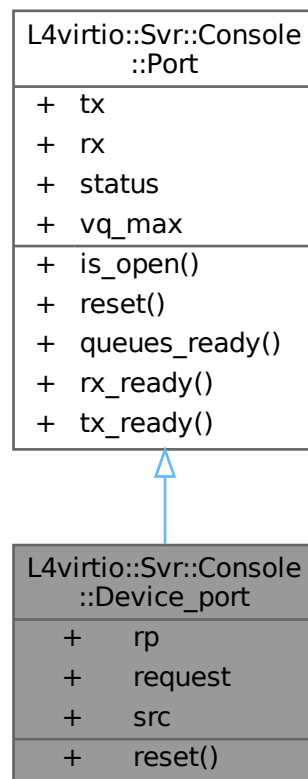
- [l4/l4virtio/server/virtio-console-device](#)

## 15.392 L4virtio::Svr::Console::Device\_port Struct Reference

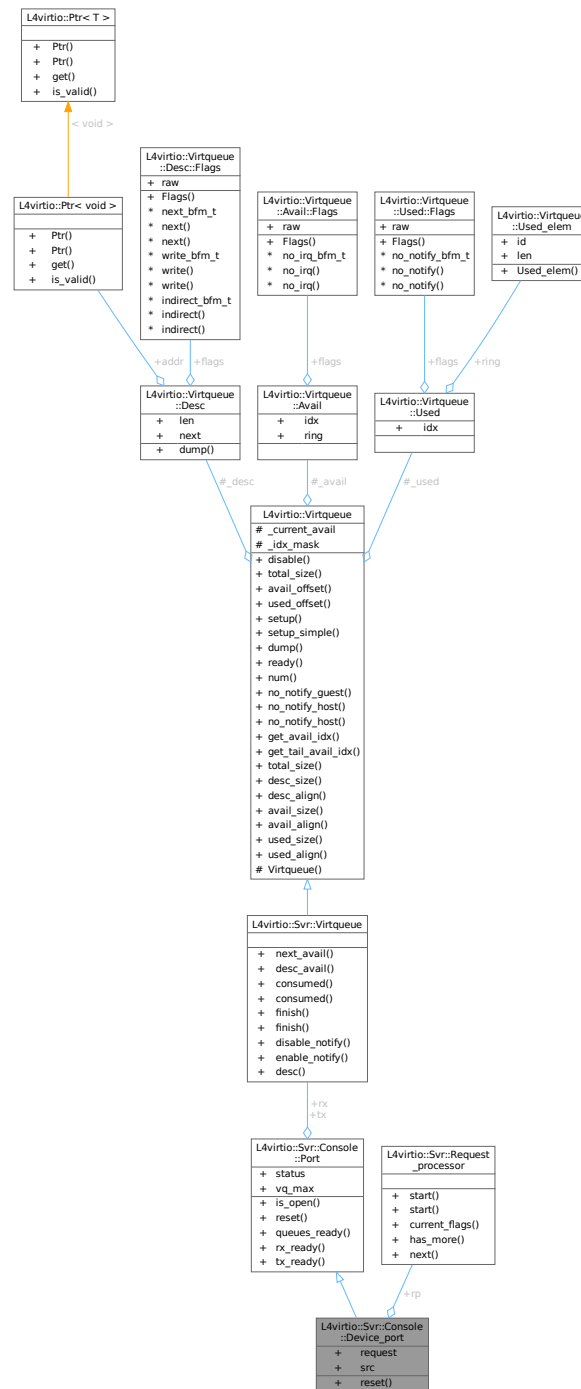
A console port with associated read/write state.

```
#include <virtio-console-device>
```

Inheritance diagram for L4virtio::Svr::Console::Device\_port:



Collaboration diagram for L4virtio::Svr::Console::Device\_port:



## Public Member Functions

- void **reset** () override  
*Reset the port to the initial state and disable its virtqueues.*

## Public Member Functions inherited from L4virtio::Svr::Console::Port

- bool **is\_open** () const

- Check that the port is open.*
- bool **queues\_ready** () const  
*Check that both virtqueues are set up correctly.*
- bool **rx\_ready** () const  
*Check that device implementation may write to receive queues.*
- bool **tx\_ready** () const  
*Check that device implementation may read from transmit queues.*

## Data Fields

- [Request\\_processor](#) **rp**  
*Request processor associated with current request.*
- Virtqueue::Request **request**  
*Current virtio tx queue request.*
- Buffer **src**  
*Source data block to process.*

## Data Fields inherited from [L4virtio::Svr::Console::Port](#)

- [Virtqueue](#) **tx**  
*Receiveq of the port.*
- [Virtqueue](#) **rx**  
*Transmitq of the port.*
- [Port\\_status](#) **status**  
*State the port is in.*
- unsigned **vq\_max**  
*Maximum queue sizes for this port.*

## Additional Inherited Members

## Public Types inherited from [L4virtio::Svr::Console::Port](#)

- enum [Port\\_status](#) {  
  [Port\\_disabled](#) = 0 , [Port\\_added](#) , [Port\\_ready](#) , [Port\\_open](#) ,  
  [Port\\_failed](#) }
- Possible states of a virtio console port.*
- enum  
*Size of control queues, also used as default size.*

### 15.392.1 Detailed Description

A console port with associated read/write state.

Tracks the notification of the device implementation and holds the state when receiving data from the driver.

Definition at line 26 of file [virtio-console-device](#).

The documentation for this struct was generated from the following file:

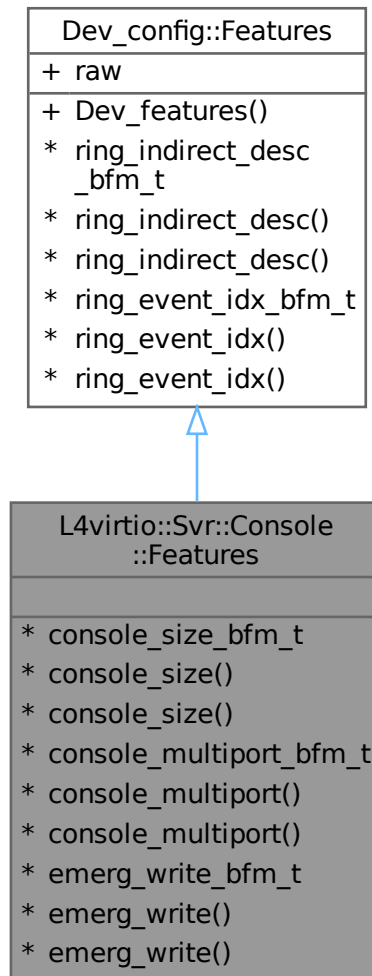
- l4/l4virtio/server/virtio-console-device

## 15.393 L4virtio::Svr::Console::Features Struct Reference

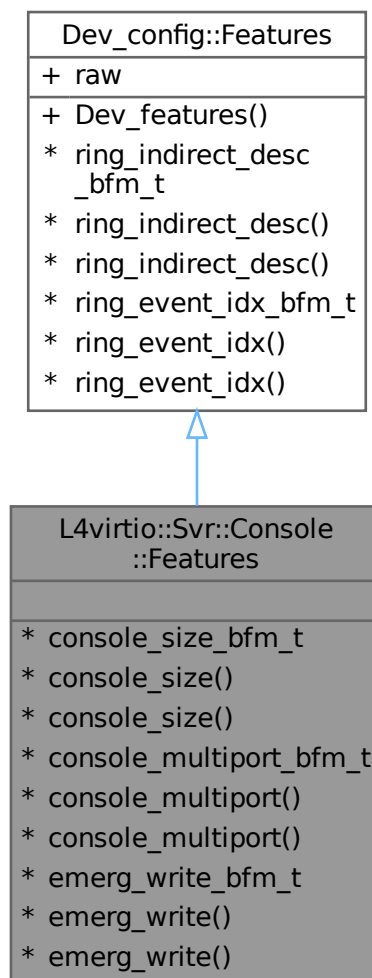
Virtio console specific feature bits.

```
#include <virtio-console>
```

Inheritance diagram for L4virtio::Svr::Console::Features:



Collaboration diagram for L4virtio::Svr::Console::Features:



#### Additional Inherited Members

#### Public Types inherited from [L4virtio::Svr::Dev\\_features](#)

#### Public Member Functions inherited from [L4virtio::Svr::Dev\\_features](#)

- **Dev\_features** ([l4\\_uint32\\_t](#) v)  
*Make Features from a raw bitmap.*

#### Data Fields inherited from [L4virtio::Svr::Dev\\_features](#)

- [l4\\_uint32\\_t](#) **raw**  
*The raw value of the features bitmap.*

### 15.393.1 Detailed Description

Virtio console specific feature bits.

Definition at line 18 of file [virtio-console](#).

### 15.393.2 Member Typedef Documentation

#### 15.393.2.1 console\_multiport\_bfm\_t

```
typedef cxx::Bitfield<decltype( raw ), 1, 1> L4virtio::Svr::Console::Features::console_multiport_bfm_t
```

[Device](#) has support for multiple ports.

Type to access the `console_multiport` bits ( 1 to 1 ) of `raw`.

Definition at line 25 of file [virtio-console](#).

#### 15.393.2.2 console\_size\_bfm\_t

```
typedef cxx::Bitfield<decltype( raw ), 0, 0> L4virtio::Svr::Console::Features::console_size_bfm_t
```

Configuration `cols` and `rows` are valid.

Type to access the `console_size` bits ( 0 to 0 ) of `raw`.

Definition at line 23 of file [virtio-console](#).

#### 15.393.2.3 emerg\_write\_bfm\_t

```
typedef cxx::Bitfield<decltype( raw ), 2, 2> L4virtio::Svr::Console::Features::emerg_write_bfm_t
```

[Device](#) has support for emergency write.

Type to access the `emerg_write` bits ( 2 to 2 ) of `raw`.

Definition at line 27 of file [virtio-console](#).

The documentation for this struct was generated from the following file:

- `l4/l4virtio/server/virtio-console`

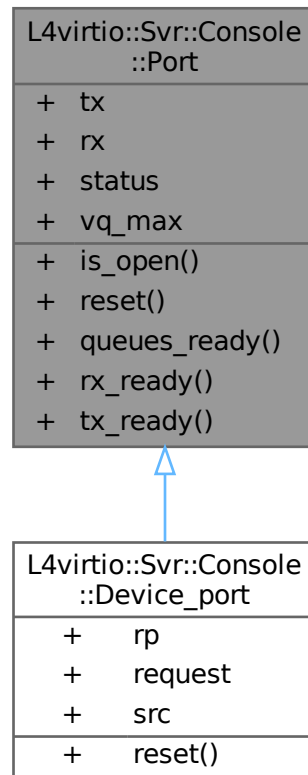


## 15.394 L4virtio::Svr::Console::Port Struct Reference

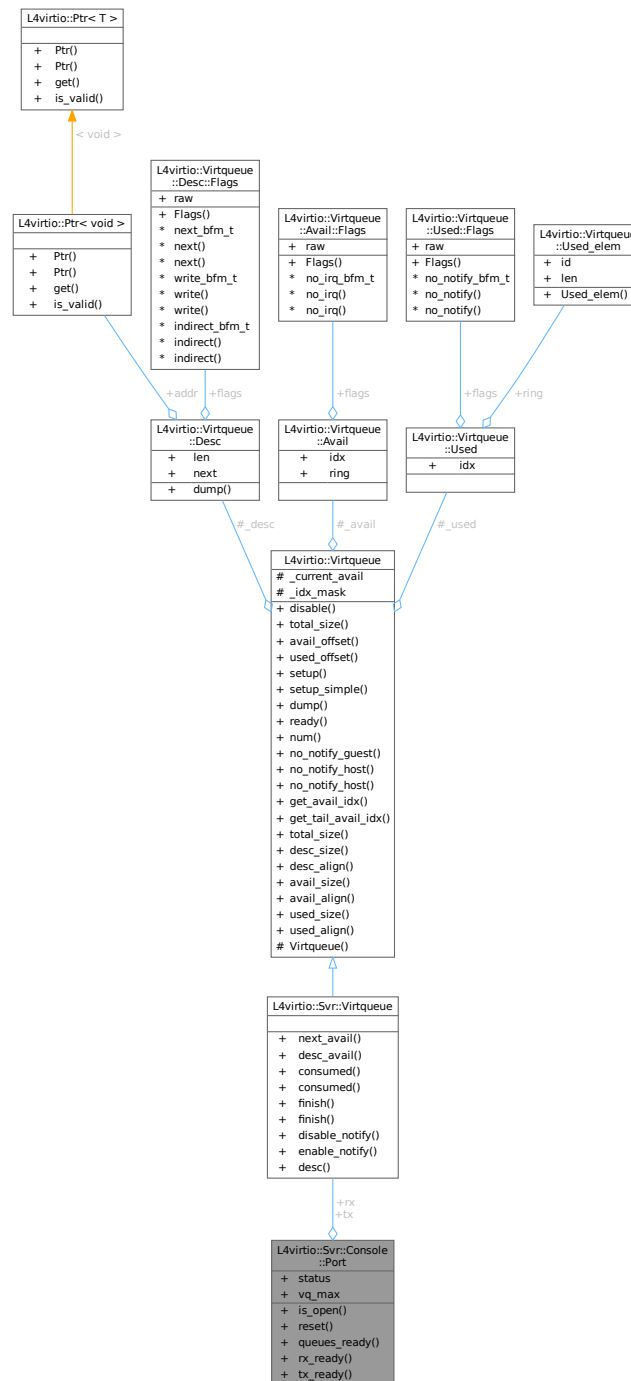
Representation of a Virtio console port.

```
#include <virtio-console>
```

Inheritance diagram for L4virtio::Svr::Console::Port:



Collaboration diagram for L4virtio::Svr::Console::Port:



## Public Types

- enum **Port\_status** {  
**Port\_disabled** = 0 , **Port\_added** , **Port\_ready** , **Port\_open** ,  
**Port\_failed** }

*Possible states of a virtio console port.*

- enum  
*Size of control queues, also used as default size.*

- bool **is\_open** () const  
*Check that the port is open.*
- virtual void **reset** ()  
*Reset the port to the initial state and disable its virtqueues.*
- bool **queues\_ready** () const  
*Check that both virtqueues are set up correctly.*
- bool **rx\_ready** () const  
*Check that device implementation may write to receive queues.*
- bool **tx\_ready** () const  
*Check that device implementation may read from transmit queues.*

- **Virtqueue tx**  
*Receiveq of the port.*
- **Virtqueue rx**  
*Transmitq of the port.*
- **Port\_status status**  
*State the port is in.*
- **unsigned vq\_max**  
*Maximum queue sizes for this port.*

```

+-----+ port_remove()
| DISABLED |<----- [all]
+-----+
|
| port_add()
v
+-----+ process_port_ready(0)
| ADDED + -----+
+-----+ | +-----+
| | +----->| |
| | process_port_ready(1) | |
v | | FAILED |
+-----+ process_port_ready(0) | |
| READY |----->| |
+-----+<-----+ +-----+
| | ^
| | port_open(true) | port_open(false) |
v | | process_port_ready(0)
+-----+ |
| OPEN |-----+
+-----+

```

**Enumerator**

Port_disabled	Reset state, waiting for port to be added.
Port_added	<a href="#">Port</a> has been added by device, waiting for ready message.
Port_ready	<a href="#">Port</a> is ready but still closed.
Port_open	<a href="#">Port</a> is in a working state.
Port_failed	<a href="#">Device</a> failure, port unusable.

Definition at line 112 of file [virtio-console](#).

The documentation for this struct was generated from the following file:

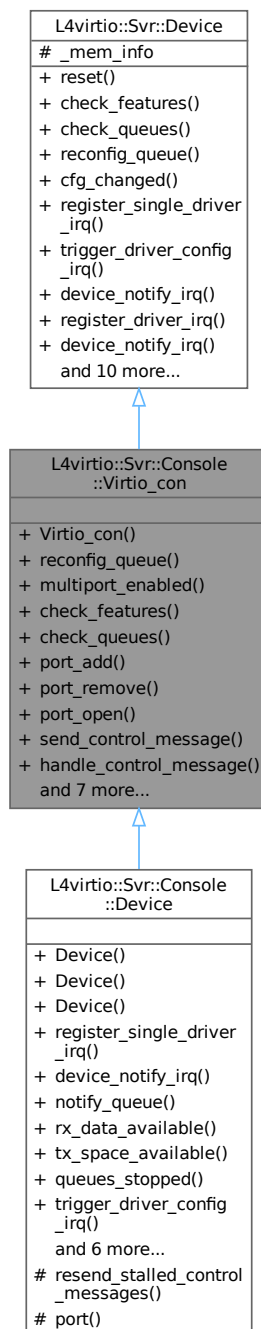
- l4/l4virtio/server/virtio-console

## 15.395 L4virtio::Svr::Console::Virtio\_con Class Reference

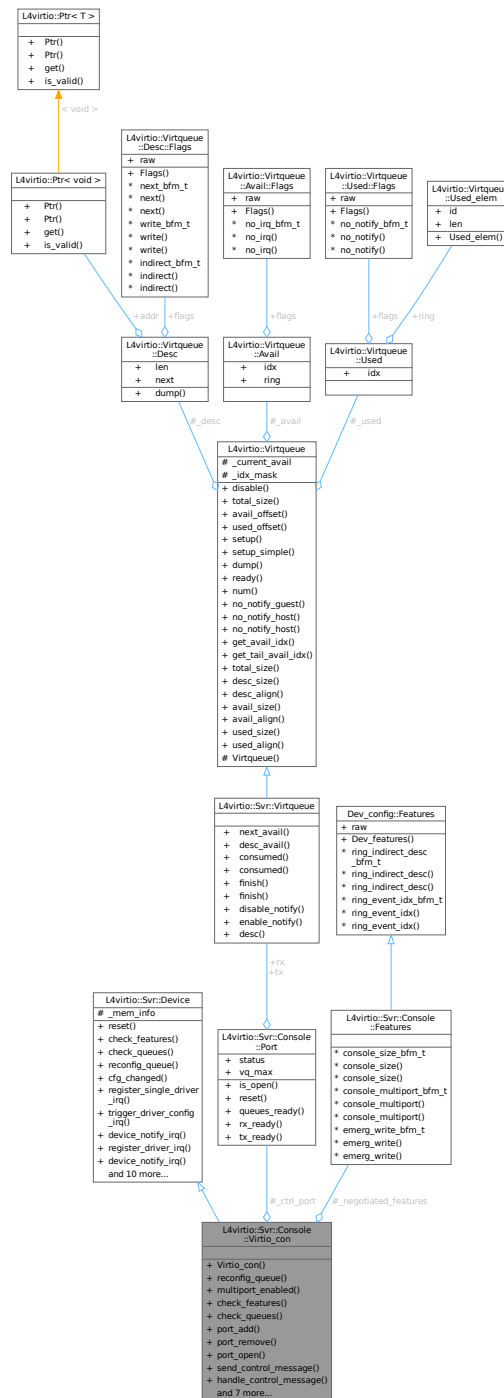
Base class implementing a virtio console functionality.

```
#include <virtio-console>
```

Inheritance diagram for L4virtio::Svr::Console::Virtio\_con:



Collaboration diagram for L4virtio::Svr::Console::Virtio\_con:



## Public Member Functions

- **Virtio\_con** (unsigned max\_ports, bool enable\_multiport)  
Create a new multiport console device.
- int **reconfig\_queue** (unsigned index) override  
callback for client queue-config request
- bool **multiport\_enabled** () const

- Return true if the multiport feature is enabled and control queues are available.*

  - bool **check\_features** (void) override  
*callback for checking the subset of accepted features*
  - bool **check\_queues** () override  
*callback for checking if the queues at DRIVER\_OK transition*
  - int **port\_add** (unsigned idx)  
*Send a DEVICE\_ADD message and update the internal state.*
  - int **port\_remove** (unsigned idx)  
*Send a DEVICE\_REMOVE message and update the internal state.*
  - int **port\_open** (unsigned idx, bool open)  
*Send a PORT\_OPEN message and update the internal state.*
  - int **send\_control\_message** (l4\_uint32\_t idx, l4\_uint16\_t event, l4\_uint16\_t value=0, const char \*name=0)  
*Send control message to driver.*
  - int **handle\_control\_message** ()  
*Handle control message received from the driver.*
  - void **reset** () override  
*reset callback, called for doing a device reset*
  - virtual void **reset\_device** ()  
*Reset the state of the actual console device.*
  - virtual void **notify\_queue** (Virtqueue \*queue)=0  
*Notify queue of available data.*
  - virtual Port \* **port** (unsigned port)=0  
*Return the specified port.*
  - virtual void **process\_device\_ready** (l4\_uint16\_t value)=0  
*Callback called on DEVICE\_READY event.*
  - virtual void **process\_port\_ready** (l4\_uint32\_t id, l4\_uint16\_t value)  
*Callback called on PORT\_READY event.*
  - virtual void **process\_port\_open** (l4\_uint32\_t id, l4\_uint16\_t value)=0  
*Callback called on PORT\_OPEN event.*

## Public Member Functions inherited from L4virtio::Svr::Device\_t< DATA >

- virtual void **cfg\_changed** (unsigned)  
*callback for client device configuration changes*
- virtual void **register\_single\_driver\_irq** ()  
*callback for registering a single guest IRQ for all queues (old-style)*
- virtual void **trigger\_driver\_config\_irq** ()=0  
*callback for triggering configuration change notification IRQ*
- virtual L4::Cap< L4::Irq > **device\_notify\_irq** () const  
*callback to gather the device notification IRQ (old-style)*
- virtual void **register\_driver\_irq** (unsigned idx)  
*Callback for registering an notification IRQ (multi IRQ).*
- virtual L4::Cap< L4::Irq > **device\_notify\_irq** (unsigned idx)  
*Callback to gather the device notification IRQ (multi IRQ).*
- virtual unsigned **num\_events\_supported** () const  
*Return the highest notification index supported.*
- **Device\_t** (Dev\_config \*dev\_config)  
*Make a device for the given config.*
- Mem\_list const \* **mem\_info** () const  
*Get the memory region list used for this device.*

- void [reset\\_queue\\_config](#) (unsigned idx, unsigned num\_max, bool inc\_generation=false)  
*Trigger reset for the configuration space for queue idx.*
- void [init\\_mem\\_info](#) (unsigned num)  
*Initialize the memory region list to the given maximum.*
- void [device\\_error](#) ()  
*Transition device into DEVICE\_NEEDS\_RESET state.*
- bool [setup\\_queue](#) ([Virtqueue](#) \*q, unsigned qn, unsigned num\_max)  
*Enable/disable the specified queue.*
- bool [handle\\_mem\\_cmd\\_write](#) ()  
*Check for a value in the cmd register and handle a write.*
- void [enable\\_trusted\\_ds\\_validation](#) ()  
*Enable trusted dataspace validation.*
- void [add\\_trusted\\_dataspaces](#) (std::shared\_ptr< Ds\_vector const > ds)  
*Provide a list of trusted dataspaces that can be used for validation.*

### Additional Inherited Members

### Protected Attributes inherited from [L4virtio::Svr::Device\\_t< DATA >](#)

- [Mem\\_list\\_mem\\_info](#)  
*Memory region list.*

## 15.395.1 Detailed Description

Base class implementing a virtio console functionality.

It is possible to activate the MULTIPORT feature, in which case incoming control messages need to be dispatched by calling [handle\\_control\\_message\(\)](#). The derived class must additionally override [process\\_device\\_ready\(\)](#), [process\\_port\\_ready\(\)](#) and [process\\_port\\_open\(\)](#) to implement the actual behaviour. The derived class has the following responsibilities:

- inform the driver about usable ports once the device is ready as signaled in [process\\_device\\_ready\(\)](#), see the wrapper [port\\_add\(\)](#).
- inform the driver about unusable ports, see the wrapper [port\\_remove\(\)](#).
- react to open/close events, see the wrapper [port\\_open\(\)](#).

This implementation provides no means to handle interrupts or notify guests, therefore derived classes have to provide this functionality, see [notify\\_queue\(\)](#) and [handle\\_control\\_message\(\)](#). Similarly, all interaction with data queues has to be implemented. Memory for port structures must be managed by the implementor as well.

Use this class as a base to implement your own specific console device.

Definition at line 185 of file [virtio-console](#).

## 15.395.2 Constructor & Destructor Documentation

### 15.395.2.1 Virtio\_con()

```
L4virtio::Svr::Console::Virtio_con::Virtio_con (
    unsigned max_ports,
    bool enable_multiport ) [inline], [explicit]
```

Create a new multiport console device.



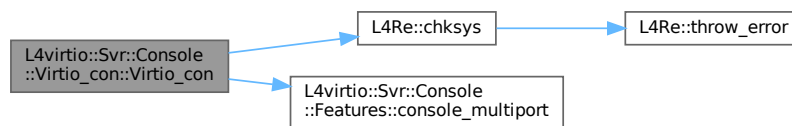
## Parameters

<i>max_ports</i>	Maximum number of ports the device should be able to handle (ignored when <code>enable_multiport</code> is false).
<i>enable_multiport</i>	Enable the control queue for dynamic handling of ports.

Definition at line 211 of file [virtio-console](#).

References [L4Re::chksys\(\)](#), [L4virtio::Svr::Console::Features::console\\_multiport\(\)](#), [L4\\_EINVAL](#), and [L4virtio::Svr::Dev\\_features::raw](#).

Here is the call graph for this function:



## 15.395.3 Member Function Documentation

### 15.395.3.1 handle\_control\_message()

```
int L4virtio::Svr::Console::Virtio_con::handle_control_message ( ) [inline]
```

Handle control message received from the driver.

## Return values

<i>L4_EOK</i>	Message has been handled.
<i>-L4_ENODEV</i>	Control queue is not ready.
<i>-L4_EBUSY</i>	Currently no descriptor available in the control queue.
<i>-L4_EINVAL</i>	Received an unexpected control event.

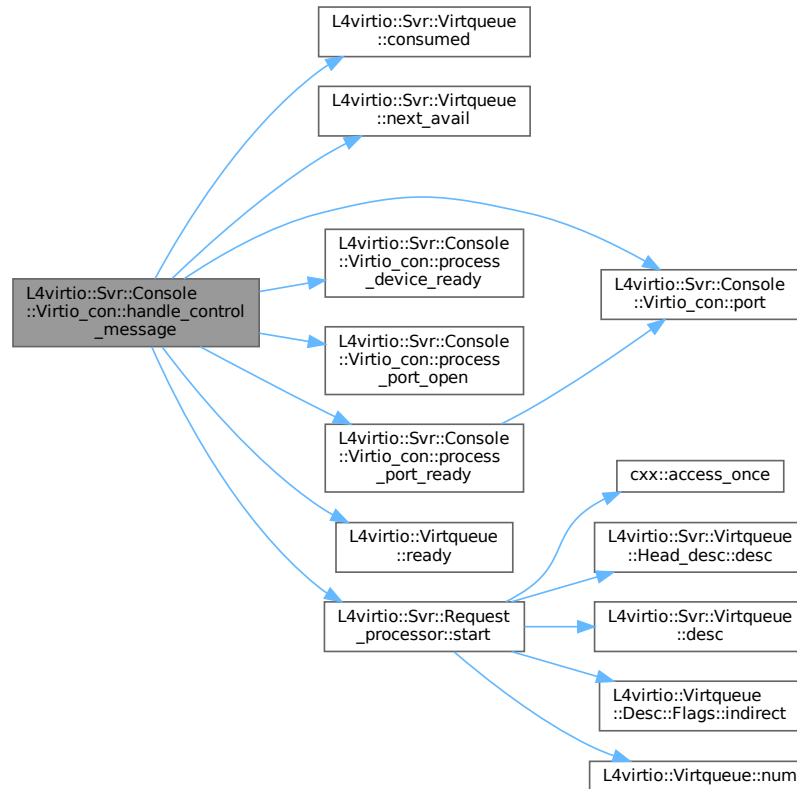
This function performs the basic handling of control messages from the driver. It does all necessary work with the control queues and performs some sanity checks. All other work is deferred to the derived class, see [process\\_device\\_ready\(\)](#), [process\\_port\\_ready\(\)](#) and [process\\_port\\_open\(\)](#).

Definition at line 443 of file [virtio-console](#).

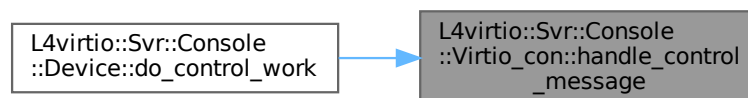
References [L4virtio::Svr::Virtqueue::consumed\(\)](#), [L4virtio::Svr::Console::Control\\_message::Device\\_ready](#), [L4virtio::Svr::Console::Control\\_message::event](#), [L4virtio::Svr::Console::Control\\_message::id](#), [L4\\_EINVAL](#), [L4\\_ENODEV](#), [L4\\_EOK](#), [L4virtio::Svr::Console::Control\\_request::len](#), [L4virtio::Svr::Console::Control\\_request::msg](#), [L4virtio::Svr::Virtqueue::next\\_avail\(\)](#), [port\(\)](#), [L4virtio::Svr::Console::Port::Port\\_added](#), [L4virtio::Svr::Console::Port::Port\\_disabled](#), [L4virtio::Svr::Console::Control\\_message::Port\\_open](#), [L4virtio::Svr::Console::Port::Port\\_open](#), [L4virtio::Svr::Console::Control\\_message::Port\\_ready](#), [process\\_device\\_ready\(\)](#), [process\\_port\\_open\(\)](#), [process\\_port\\_ready\(\)](#), [L4virtio::Virtqueue::ready\(\)](#), [L4virtio::Svr::Request\\_processor::start\(\)](#), [L4virtio::Svr::Console::Port::status](#), [L4virtio::Svr::Console::Port::value](#), and [L4virtio::Svr::Console::Control\\_message::value](#).

Referenced by [L4virtio::Svr::Console::Device::do\\_control\\_work\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.395.3.2 notify\_queue()

```
virtual void L4virtio::Svr::Console::Virtio_con::notify_queue (
    Virtqueue * queue ) [pure virtual]
```

Notify queue of available data.

## Parameters

<i>queue</i>	Virtqueue to notify.
--------------	----------------------

This callback is called whenever data is sent to `queue`. It is the responsibility of the derived class to perform all necessary notification actions, e.g. triggering guest interrupts.

Implemented in `L4virtio::Svr::Console::Device`.

### 15.395.3.3 port()

```
virtual Port * L4virtio::Svr::Console::Virtio_con::port (
    unsigned port ) [pure virtual]
```

Return the specified port.

## Parameters

<i>port</i>	Port number.
-------------	--------------

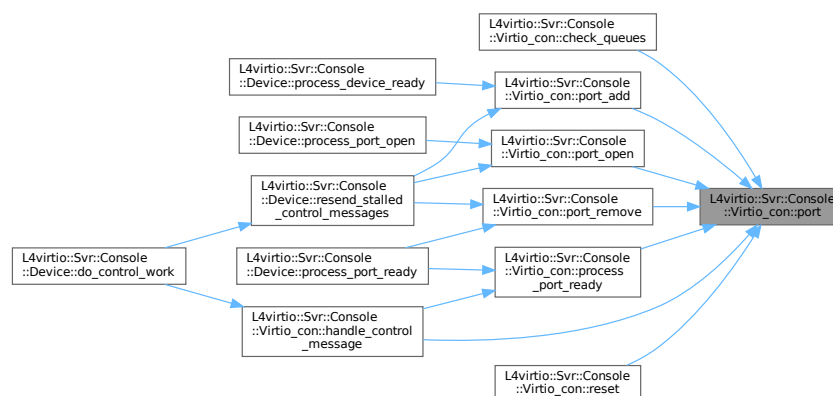
### Precondition

Port number must be lower than the configured maximum number of ports.

Implemented in `L4virtio::Svr::Console::Device`.

Referenced by [check\\_queues\(\)](#), [handle\\_control\\_message\(\)](#), [port\\_add\(\)](#), [port\\_open\(\)](#), [port\\_remove\(\)](#), [process\\_port\\_ready\(\)](#), and [reset\(\)](#).

Here is the caller graph for this function:



#### 15.395.3.4 port\_add()

```
int L4virtio::Svr::Console::Virtio_con::port_add (
    unsigned idx ) [inline]
```

Send a `DEVICE_ADD` message and update the internal state.

## Parameters

<i>idx</i>	Port that should be added.
------------	----------------------------

## Return values

<i>L4_EOK</i>	Message has been sent.
<i>-L4_EPERM</i>	Control message is not allowed in the current state.

## Returns

Errors from [send\\_control\\_message\(\)](#)

## Precondition

`port` must be smaller than the configured number of ports.

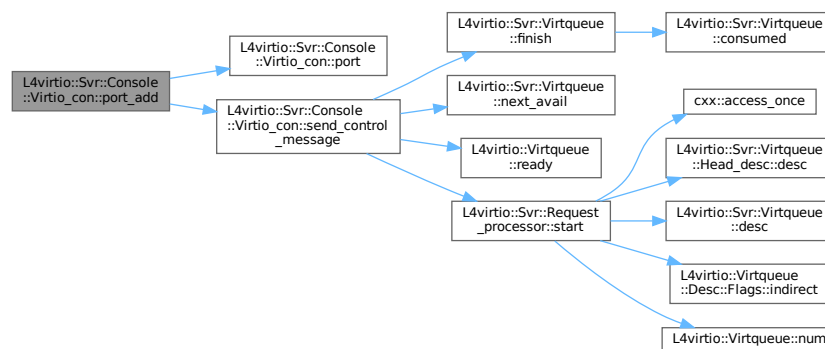
Port must not already exist.

Definition at line 298 of file [virtio-console](#).

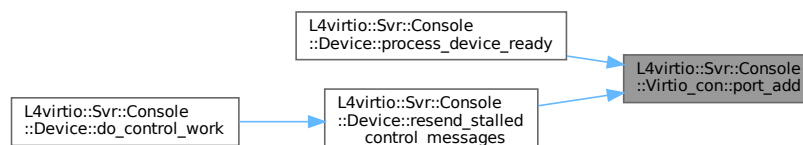
References [L4virtio::Svr::Console::Control\\_message::Device\\_add](#), [L4\\_EOK](#), [L4\\_EPERM](#), [port\(\)](#), [L4virtio::Svr::Console::Port::Port\\_add](#), [L4virtio::Svr::Console::Port::Port\\_disabled](#), [send\\_control\\_message\(\)](#), and [L4virtio::Svr::Console::Port::status](#).

Referenced by [L4virtio::Svr::Console::Device::process\\_device\\_ready\(\)](#), and [L4virtio::Svr::Console::Device::resend\\_stalled\\_control\\_m](#)

Here is the call graph for this function:



Here is the caller graph for this function:



## 15.395.3.5 port\_open()

```
int L4virtio::Svr::Console::Virtio_con::port_open (
    unsigned idx,
    bool open ) [inline]
```

Send a PORT\_OPEN message and update the internal state.

## Parameters

<i>idx</i>	Port that should be opened or closed.
<i>open</i>	Open or close port.

## Return values

<i>L4_EOK</i>	Message has been sent.
<i>-L4_EPERM</i>	Control message is not allowed in the current state.

## Returns

Errors from [send\\_control\\_message\(\)](#)

## Precondition

`port` must be smaller than the configured number of ports.

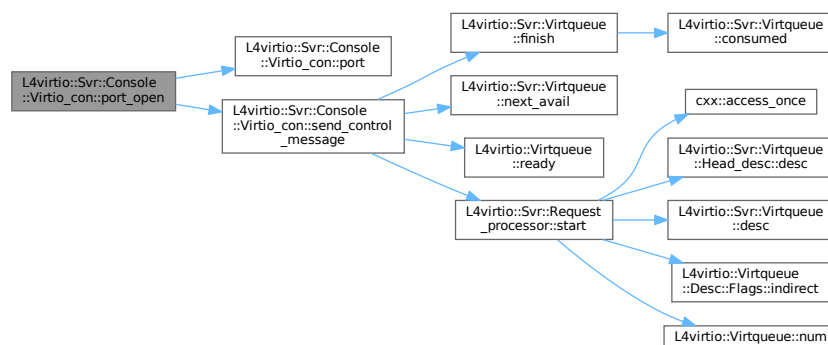
Port must already exist.

Definition at line 355 of file [virtio-console](#).

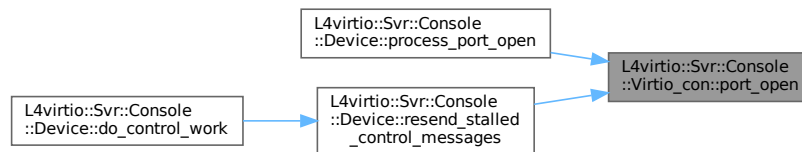
References [L4\\_EOK](#), [L4\\_EPERM](#), [port\(\)](#), [L4virtio::Svr::Console::Control\\_message::Port\\_open](#), [L4virtio::Svr::Console::Port::Port\\_open](#), [L4virtio::Svr::Console::Port::Port\\_ready](#), [send\\_control\\_message\(\)](#), and [L4virtio::Svr::Console::Port::status](#).

Referenced by [L4virtio::Svr::Console::Device::process\\_port\\_open\(\)](#), and [L4virtio::Svr::Console::Device::resend\\_stalled\\_control\\_mess](#)

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.395.3.6 port\_remove()

```
int L4virtio::Svr::Console::Virtio_con::port_remove (
    unsigned idx ) [inline]
```

Send a DEVICE\_REMOVE message and update the internal state.

#### Parameters

<i>idx</i>	Port that should be removed.
------------	------------------------------

#### Return values

<i>L4_EOK</i>	Message has been sent.
<i>-L4_EPERM</i>	Control message is not allowed in the current state.

#### Returns

Errors from [send\\_control\\_message\(\)](#)

#### Precondition

`port` must be smaller than the configured number of ports.

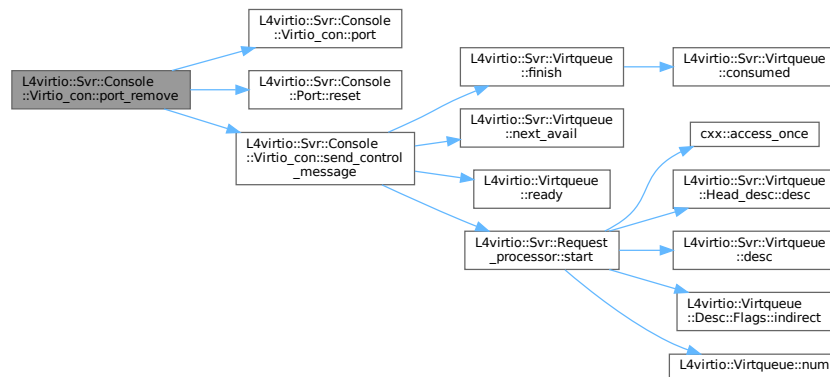
`Port` must already exist.

Definition at line 325 of file [virtio-console](#).

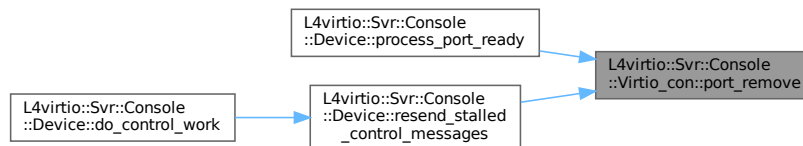
References [L4virtio::Svr::Console::Control\\_message::Device\\_remove](#), [L4\\_EOK](#), [L4\\_EPERM](#), [port\(\)](#), [L4virtio::Svr::Console::Port::Port\\_open](#), [L4virtio::Svr::Console::Port::Port\\_ready](#), [L4virtio::Svr::Console::Port::reset\(\)](#), [send\\_control\\_message\(\)](#), and [L4virtio::Svr::Console::Port::status](#).

Referenced by [L4virtio::Svr::Console::Device::process\\_port\\_ready\(\)](#), and [L4virtio::Svr::Console::Device::resend\\_stalled\\_control\\_mes](#)

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.395.3.7 process\_device\_ready()

```
virtual void L4virtio::Svr::Console::Virtio_con::process_device_ready (
    uint16_t value ) [pure virtual]
```

Callback called on DEVICE\_READY event.

#### Parameters

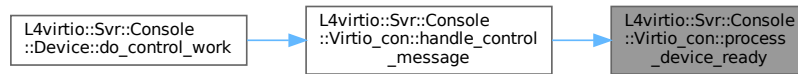
<i>value</i>	The value field of the control message, indicating if the initialization was successful.
--------------	--

Needs to be overridden by the derived class if the MULTIPORT feature is enabled. Control messages may be sent only after the driver has successfully initialized the device.

Implemented in [L4virtio::Svr::Console::Device](#).

Referenced by [handle\\_control\\_message\(\)](#).

Here is the caller graph for this function:



### 15.395.3.8 process\_port\_open()

```
virtual void L4virtio::Svr::Console::Virtio_con::process_port_open (
    14_uint32_t id,
    14_uint16_t value ) [pure virtual]
```

Callback called on PORT\_OPEN event.

#### Parameters

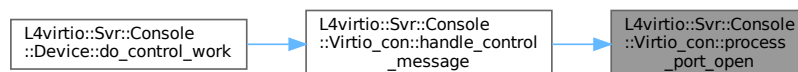
<i>id</i>	The id field of the control message, i.e. the port number.
<i>value</i>	The value field of the control message, indicating if the port was opened or closed.

Needs to be overridden by the derived class if the MULTIPORT feature is enabled. The device must acknowledge the message by calling `port_open()`.

Implemented in [L4virtio::Svr::Console::Device](#).

Referenced by [handle\\_control\\_message\(\)](#).

Here is the caller graph for this function:



### 15.395.3.9 process\_port\_ready()

```
virtual void L4virtio::Svr::Console::Virtio_con::process_port_ready (
    14_uint32_t id,
    14_uint16_t value ) [inline], [virtual]
```

Callback called on PORT\_READY event.

#### Parameters

<i>id</i>	The id field of the control message, i.e. the port number.
<i>value</i>	The value field of the control message, indicating if the initialization was successful.



May be overridden by the derived class if the MULTIPORT feature is enabled. This default implementation just sets the status of the port according to the driver message.

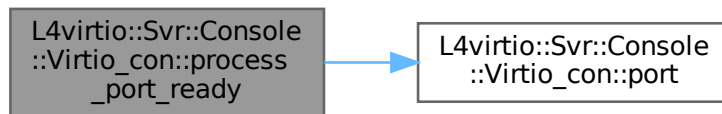
Reimplemented in [L4virtio::Svr::Console::Device](#).

Definition at line 606 of file [virtio-console](#).

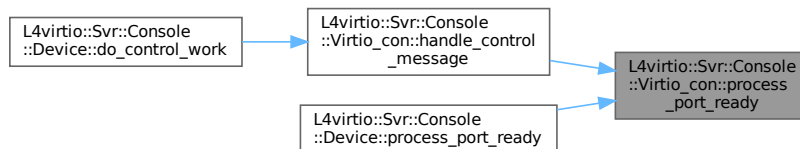
References [port\(\)](#), [L4virtio::Svr::Console::Port::Port\\_added](#), [L4virtio::Svr::Console::Port::Port\\_failed](#), [L4virtio::Svr::Console::Port::Port\\_ready](#), and [L4virtio::Svr::Console::Port::status](#).

Referenced by [handle\\_control\\_message\(\)](#), and [L4virtio::Svr::Console::Device::process\\_port\\_ready\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.395.3.10 reset\_device()

```
virtual void L4virtio::Svr::Console::Virtio_con::reset_device ( ) [inline], [virtual]
```

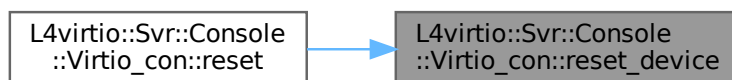
Reset the state of the actual console device.

This callback is called at the end of [reset\(\)](#), allowing the derived class to reset internal state.

Definition at line 560 of file [virtio-console](#).

Referenced by [reset\(\)](#).

Here is the caller graph for this function:



### 15.395.3.11 send\_control\_message()

```
int L4virtio::Svr::Console::Virtio_con::send_control_message (
    l4_uint32_t idx,
    l4_uint16_t event,
    l4_uint16_t value = 0,
    const char * name = 0 ) [inline]
```

Send control message to driver.

#### Parameters

<i>idx</i>	Port number.
<i>event</i>	Kind of control event.
<i>value</i>	Extra information for the event.
<i>name</i>	Name to be used for Port_name message

#### Return values

<i>L4_EOK</i>	Message has been sent.
<i>-L4_ENODEV</i>	Control queue is not ready.
<i>-L4_EBUSY</i>	Currently no descriptor available in the control queue.
<i>-L4_ENOMEM</i>	Client-issued descriptor too small. Device will be set to failed state.

#### Precondition

`port` must be smaller than the configured number of ports.

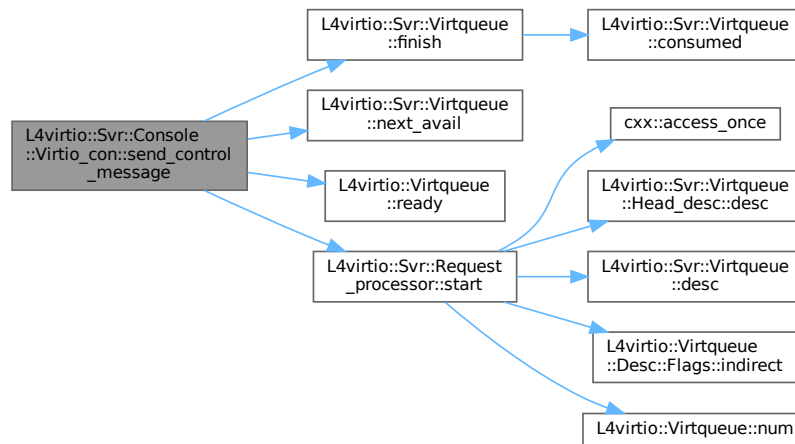
The convenience functions `port_add()`, `port_remove()` and `port_open()` should cover the most use cases and are the preferred way of communication with the driver. If you use this function directly, it is your responsibility to guarantee no invalid control messages are sent to the driver.

Definition at line 393 of file `virtio-console`.

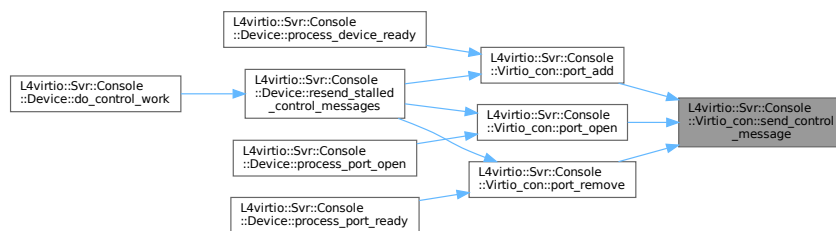
References `L4virtio::Svr::Virtqueue::finish()`, `L4_EBUSY`, `L4_ENODEV`, `L4_ENOMEM`, `L4_EOK`, `L4virtio::Svr::Console::Control_request::msg`, `L4virtio::Svr::Virtqueue::next_avail()`, `L4virtio::Svr::Console::Control_message::Port_name`, `L4virtio::Virtqueue::ready()`, `L4virtio::Svr::Console::Port::rx`, and `L4virtio::Svr::Request_processor::start()`.

Referenced by `port_add()`, `port_open()`, and `port_remove()`.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- I4/I4virtio/server/virtio-console

## 15.396 L4virtio::Svr::Data\_buffer Struct Reference

Abstract data buffer.

```
#include <virtio>
```

Inherited by L4virtio::Svr::Console::Device\_port::Buffer.

Collaboration diagram for L4virtio::Svr::Data\_buffer:

L4virtio::Svr::Data_buffer
+ pos
+ left
+ Data_buffer()
+ set()
+ copy_to()
+ skip()
+ done()

## Public Member Functions

- template<typename T >  
Data\_buffer (T \*p)  
*Create buffer for object p.*
- template<typename T >  
void set (T \*p)  
*Set buffer for object p.*
- l4\_uint32\_t copy\_to (Data\_buffer \*dst, l4\_uint32\_t max=UINT\_MAX)  
*Copy contents from this buffer to the destination buffer.*
- l4\_uint32\_t skip (l4\_uint32\_t bytes)  
*Skip given number of bytes in this buffer.*
- bool done () const  
*Check if there are no more bytes left in the buffer.*

## Data Fields

- char \* pos  
*Current buffer position.*
- l4\_uint32\_t left  
*Bytes left in buffer.*

## 15.396.1 Detailed Description

Abstract data buffer.

Definition at line 287 of file virtio.

## 15.396.2 Constructor & Destructor Documentation

### 15.396.2.1 Data\_buffer()

```
template<typename T >
L4virtio::Svr::Data_buffer::Data_buffer (
    T * p ) [inline], [explicit]
```

Create buffer for object *p*.

#### Template Parameters

<i>T</i>	Type of object (implicit)
----------	---------------------------

#### Parameters

<i>p</i>	Pointer to object.
----------	--------------------

The buffer shall point to the start of the object *p* and the size left is `sizeof(T)`.

Definition at line 304 of file [virtio](#).

## 15.396.3 Member Function Documentation

### 15.396.3.1 copy\_to()

```
l4_uint32_t L4virtio::Svr::Data_buffer::copy_to (
    Data_buffer * dst,
    l4_uint32_t max = UINT_MAX ) [inline]
```

Copy contents from this buffer to the destination buffer.

#### Parameters

<i>dst</i>	Destination buffer.
<i>max</i>	(optional) Maximum number of bytes to copy.

#### Returns

the number of bytes copied.

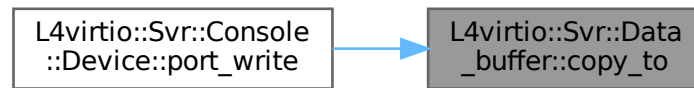
This function copies at most *max* bytes from this to *dst*. If *max* is omitted, copies the maximum number of bytes available that fit *dst*.

Definition at line 335 of file [virtio](#).

References [left](#), and [pos](#).

Referenced by [L4virtio::Svr::Console::Device::port\\_write\(\)](#).

Here is the caller graph for this function:



### 15.396.3.2 done()

```
bool L4virtio::Svr::Data_buffer::done ( ) const [inline]
```

Check if there are no more bytes left in the buffer.

#### Returns

true if there are no more bytes left in the buffer.

Definition at line 369 of file [virtio](#).

References [left](#).

### 15.396.3.3 set()

```
template<typename T >
void L4virtio::Svr::Data_buffer::set (
    T * p ) [inline]
```

Set buffer for object p.

#### Template Parameters

<i>T</i>	Type of object (implicit)
----------	---------------------------

#### Parameters

<i>p</i>	Pointer to object.
----------	--------------------

The buffer shall point to the start of the object p and the size left is sizeof(T).

Definition at line 318 of file [virtio](#).

References [left](#), and [pos](#).

### 15.396.3.4 skip()

```
l4_uint32_t L4virtio::Svr::Data_buffer::skip (
    l4_uint32_t bytes ) [inline]
```

Skip given number of bytes in this buffer.

#### Parameters

<i>bytes</i>	Number of bytes that shall be skipped.
--------------	--

#### Returns

The number of bytes skipped.

Try to skip the given number of bytes in this buffer, if there are less bytes left in the buffer that given then at most left bytes are skipped and the amount is returned.

Definition at line 356 of file [virtio](#).

References [left](#), and [pos](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/server/virtio

## 15.397 L4virtio::Svr::Dev\_config Class Reference

Abstraction for L4-Virtio device config memory.

```
#include <l4virtio>
```

Inherited by L4virtio::Svr::Dev\_config\_t< l4virtio\_block\_config\_t >, L4virtio::Svr::Dev\_config\_t< Serial\_config\_↵space >, L4virtio::Svr::Dev\_config\_t< L4virtio::Svr::No\_custom\_data >, and L4virtio::Svr::Dev\_config\_t< PRIV↵\_CONFIG >.

Collaboration diagram for L4virtio::Svr::Dev\_config:



## Public Member Functions

- [Dev\\_config](#) ([l4\\_uint32\\_t](#) vendor, [l4\\_uint32\\_t](#) device, unsigned cfg\_size, [l4\\_uint32\\_t](#) num\_queues=0)  
*Create a L4-Virtio config data space.*
- [Dev\\_config](#) (Cfg\_cap const &cfg, [l4\\_addr\\_t](#) cfg\_offset, [l4\\_uint32\\_t](#) vendor, [l4\\_uint32\\_t](#) device, unsigned cfg\_size, [l4\\_uint32\\_t](#) num\_queues=0)  
*Setup an L4-Virtio config space in an existing data space.*
- [l4\\_uint32\\_t num\\_queues](#) () const  
*Return the number of queues currently usable.*
- [l4\\_uint32\\_t guest\\_features](#) (unsigned idx) const  
*Return a specific set of guest features.*
- [l4\\_uint32\\_t negotiated\\_features](#) (unsigned idx) const  
*Compute a specific set of negotiated features.*
- [Status status](#) () const  
*Get current device status (trusted).*
- [l4\\_uint32\\_t get\\_cmd](#) () const  
*Get the value from the cmd register.*
- void [reset\\_cmd](#) ()  
*Reset the cmd register after execution of a command.*
- void [set\\_status](#) ([Status status](#))  
*Set device status register.*
- void [add\\_irq\\_status](#) ([l4\\_uint32\\_t status](#))  
*Adds irq status bit.*
- void [set\\_device\\_needs\\_reset](#) ()  
*Set DEVICE\_NEEDS\_RESET bit in device status register.*



- bool [change\\_queue\\_config](#) ([l4\\_uint32\\_t](#) num\_queues)  
*Setup new queue configuration.*
- [l4virtio\\_config\\_queue\\_t](#) volatile const \* [qconfig](#) (unsigned index) const  
*Get queue read-only config data for queue with the given index.*
- void [reset\\_hdr](#) (bool inc\_generation=false) const  
*Reset the config header to the initial contents.*
- bool [reset\\_queue](#) (unsigned index, unsigned num\_max, bool inc\_generation=false) const  
*Reset queue config for the given queue.*
- [l4virtio\\_config\\_hdr\\_t](#) const volatile \* [hdr](#) () const  
*Get a read-only pointer to the config header.*
- [L4::Cap](#)< [L4Re::Dataspace](#) > [ds](#) () const  
*Get data-space capability for the shared config data space.*
- [l4\\_addr\\_t](#) [ds\\_offset](#) () const  
*Return the offset into the config dataspace where the device configuration starts.*

## 15.397.1 Detailed Description

Abstraction for L4-Virtio device config memory.

Virtio defines a device configuration mechanism, L4-Virtio implements this mechanism based on shared memory a [set\\_status\(\)](#) and a [config\\_queue\(\)](#) call. This class provides an abstraction for L4-Virtio host implementations to establish such a shared memory data space and providing the necessary contents and access functions.

Definition at line 52 of file [l4virtio](#).

## 15.397.2 Constructor & Destructor Documentation

### 15.397.2.1 Dev\_config() [1/2]

```
L4virtio::Svr::Dev_config::Dev_config (
    l4\_uint32\_t vendor,
    l4\_uint32\_t device,
    unsigned cfg_size,
    l4\_uint32\_t num_queues = 0 ) [inline]
```

Create a L4-Virtio config data space.

#### Parameters

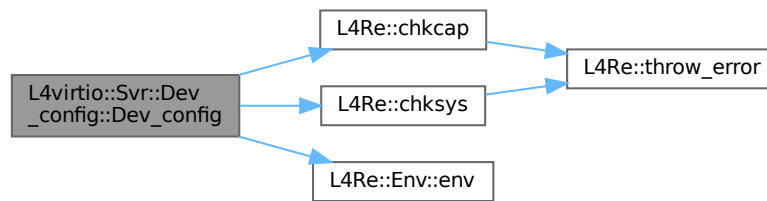
<i>vendor</i>	The vendor ID to store in config header.
<i>device</i>	The device ID to store in config header.
<i>cfg_size</i>	The size of the device-specific config data in bytes.
<i>num_queues</i>	The number of queues provided by the device.

This constructor allocates a data space used for L4-virtio config attaches the data space to the local address space and writes the initial contents to the config header.

Definition at line 112 of file [l4virtio](#).

References [L4Re::chkcap\(\)](#), [L4Re::chksys\(\)](#), [L4Re::Env::env\(\)](#), and [L4\\_PAGESIZE](#).

Here is the call graph for this function:



### 15.397.2.2 Dev\_config() [2/2]

```

L4virtio::Svr::Dev_config::Dev_config (
    Cfg_cap const & cfg,
    l4_addr_t cfg_offset,
    l4_uint32_t vendor,
    l4_uint32_t device,
    unsigned cfg_size,
    l4_uint32_t num_queues = 0 ) [inline]

```

Setup an L4-Virtio config space in an existing data space.

#### Parameters

<i>cfg</i>	Dataspace that should hold the L4-Virtio configuration.
<i>cfg_offset</i>	Offset into the dataspace where the configuration starts.
<i>vendor</i>	The vendor ID to store in config header.
<i>device</i>	The device ID to store in config header.
<i>cfg_size</i>	The size of the device-specific config data in bytes.
<i>num_queues</i>	The number of queues provided by the device.

Definition at line 146 of file [l4virtio](#).

References [L4\\_PAGESIZE](#).

## 15.397.3 Member Function Documentation

### 15.397.3.1 add\_irq\_status()

```

void L4virtio::Svr::Dev_config::add_irq_status (
    l4_uint32_t status ) [inline]

```

Adds irq status bit.

## Parameters

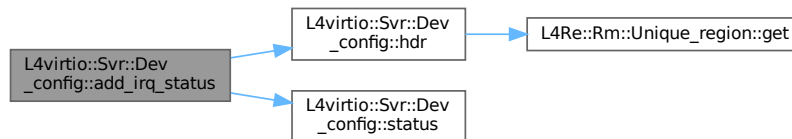
<i>status</i>	The value to add to the irq status register.
---------------	--

This function adds the status bit to the irq status register.

Definition at line 265 of file [l4virtio](#).

References [hdr\(\)](#), and [status\(\)](#).

Here is the call graph for this function:



## 15.397.3.2 change\_queue\_config()

```
bool L4virtio::Svr::Dev_config::change_queue_config (
    l4_uint32_t num_queues ) [inline]
```

Setup new queue configuration.

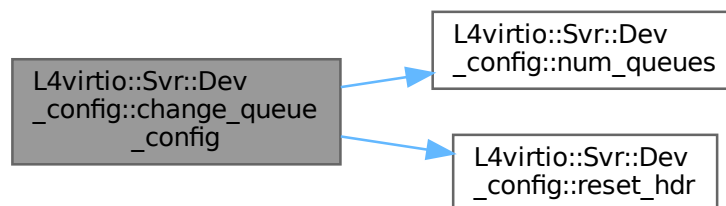
## Parameters

<i>num_queues</i>	The number of queues provided by the device.
-------------------	--

Definition at line 286 of file [l4virtio](#).

References [L4\\_PAGESIZE](#), [num\\_queues\(\)](#), and [reset\\_hdr\(\)](#).

Here is the call graph for this function:



### 15.397.3.3 ds()

```
L4::Cap< L4Re::Dataspace > L4virtio::Svr::Dev_config::ds ( ) const [inline]
```

Get data-space capability for the shared config data space.

#### Returns

Capability for the shared config data space.

Definition at line 375 of file [l4virtio](#).

### 15.397.3.4 get\_cmd()

```
l4_uint32_t L4virtio::Svr::Dev_config::get_cmd ( ) const [inline]
```

Get the value from the cmd register.

Note, the most significant eight bits are the command (0 is nothing to do). The upper eight bit are reset to zero after the command was handled.

Definition at line 230 of file [l4virtio](#).

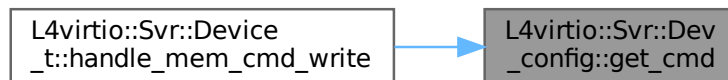
References [hdr\(\)](#).

Referenced by [L4virtio::Svr::Device\\_t< DATA >::handle\\_mem\\_cmd\\_write\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.397.3.5 guest\_features()

```
l4_uint32_t L4virtio::Svr::Dev_config::guest_features (
    unsigned idx ) const [inline]
```

Return a specific set of guest features.

## Parameters

<i>idx</i>	Index into the guest features array.
------------	--------------------------------------

## Return values

<i>The</i>	selected set of guest features.
------------	---------------------------------

This function returns a specific 32bit set of features enabled by the guest/driver. `idx` is the index in the guest features array, resp. the 32 bit set to return.

Definition at line 198 of file [l4virtio](#).

### 15.397.3.6 `hdr()`

```
l4virtio_config_hdr_t const volatile * L4virtio::Svr::Dev_config::hdr ( ) const [inline]
```

Get a read-only pointer to the config header.

## Returns

Read-only pointer to the shared config header.

Definition at line 368 of file [l4virtio](#).

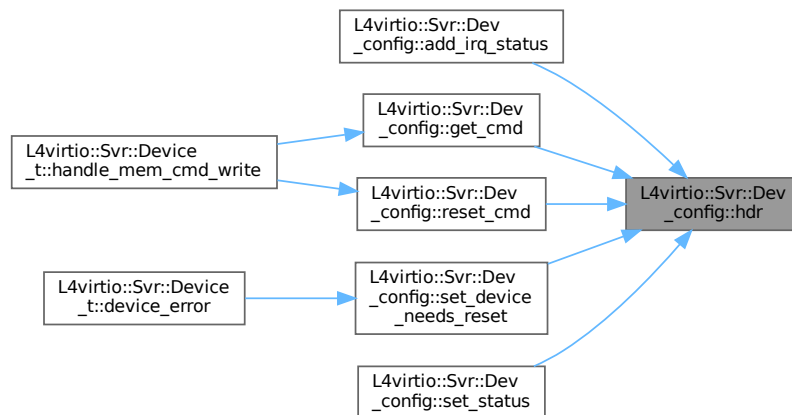
References [L4Re::Rm::Unique\\_region< T >::get\(\)](#).

Referenced by [add\\_irq\\_status\(\)](#), [get\\_cmd\(\)](#), [reset\\_cmd\(\)](#), [set\\_device\\_needs\\_reset\(\)](#), and [set\\_status\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.397.3.7 negotiated\_features()

```
l4_uint32_t L4virtio::Svr::Dev_config::negotiated_features (
    unsigned idx ) const [inline]
```

Compute a specific set of negotiated features.

#### Parameters

<i>idx</i>	Index into the guest/host features array.
------------	---

#### Return values

<i>The</i>	selected set of negotiated features.
------------	--------------------------------------

This function returns a specific 32-bit set of features negotiated by the guest/driver and host/device. *idx* is the index in the guest/host features array, resp. the 32-bit set to return.

Definition at line 212 of file [l4virtio](#).

### 15.397.3.8 qconfig()

```
l4virtio_config_queue_t volatile const * L4virtio::Svr::Dev_config::qconfig (
    unsigned index ) const [inline]
```

Get queue read-only config data for queue with the given *index*.

#### Parameters

<i>index</i>	The index of the queue.
--------------	-------------------------

**Returns**

Read-only pointer to the config of the queue with the given *index*, or NULL if *index* is out of range.

Definition at line 303 of file [l4virtio](#).

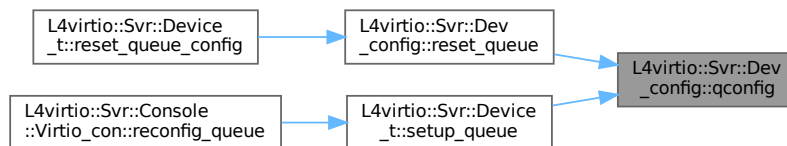
References [L4Re::Rm::Unique\\_region< T >::get\(\)](#), and [L4\\_UNLIKELY](#).

Referenced by [reset\\_queue\(\)](#), and [L4virtio::Svr::Device\\_t< DATA >::setup\\_queue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.397.3.9 reset\_cmd()**

```
void L4virtio::Svr::Dev_config::reset_cmd ( ) [inline]
```

Reset the `cmd` register after execution of a command.

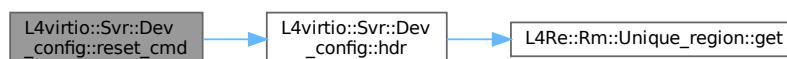
This function resets the `cmd` register in order for the client to detect that the command was executed by the device.

Definition at line 241 of file [l4virtio](#).

References [hdr\(\)](#).

Referenced by [L4virtio::Svr::Device\\_t< DATA >::handle\\_mem\\_cmd\\_write\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.397.3.10 reset\_queue()

```

bool L4virtio::Svr::Dev_config::reset_queue (
    unsigned index,
    unsigned num_max,
    bool inc_generation = false ) const [inline]
  
```

Reset queue config for the given queue.

#### Parameters

<i>index</i>	The index of the queue to reset.
<i>num_max</i>	The maximum number of descriptors supported by this queue.
<i>inc_generation</i>	The config generation will be incremented when this is true.

#### Returns

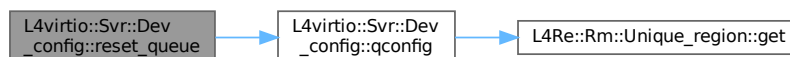
true on success, or false when *index* is out of range.

Definition at line 345 of file [l4virtio](#).

References [L4\\_UNLIKELY](#), [l4virtio\\_config\\_queue\\_t::num](#), [l4virtio\\_config\\_queue\\_t::num\\_max](#), [qconfig\(\)](#), and [l4virtio\\_config\\_queue\\_t::ready](#).

Referenced by [L4virtio::Svr::Device\\_t< DATA >::reset\\_queue\\_config\(\)](#).

Here is the call graph for this function:





Here is the caller graph for this function:



### 15.397.3.11 set\_device\_needs\_reset()

```
void L4virtio::Svr::Dev_config::set_device_needs_reset ( ) [inline]
```

Set DEVICE\_NEEDS\_RESET bit in device status register.

This function sets the internal status register and also the status register in the shared memory to DEVICE\_NEEDS\_RESET.

Definition at line 276 of file [l4virtio](#).

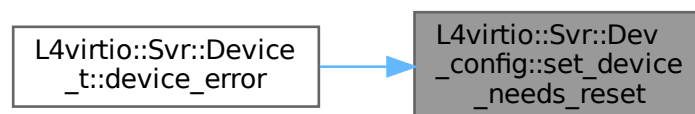
References [L4virtio::Svr::Dev\\_status::device\\_needs\\_reset\(\)](#), [hdr\(\)](#), and [L4virtio::Svr::Dev\\_status::raw](#).

Referenced by [L4virtio::Svr::Device\\_t< DATA >::device\\_error\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



**15.397.3.12 set\_status()**

```
void L4virtio::Svr::Dev_config::set_status (  
    Status status ) [inline]
```

Set device status register.

## Parameters

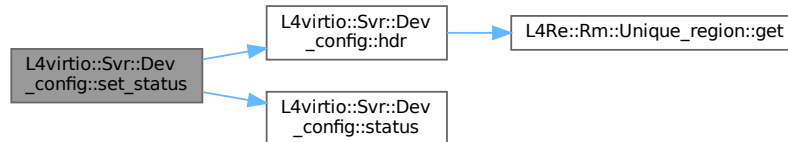
<i>status</i>	The new value for the device status register.
---------------	---

This function sets the internal status register and also the status register in the shared memory to *status*.

Definition at line 253 of file [l4virtio](#).

References [hdr\(\)](#), [L4virtio::Svr::Dev\\_status::raw](#), and [status\(\)](#).

Here is the call graph for this function:

**15.397.3.13 status()**

```
Status L4virtio::Svr::Dev_config::status ( ) const [inline]
```

Get current device status (trusted).

## Returns

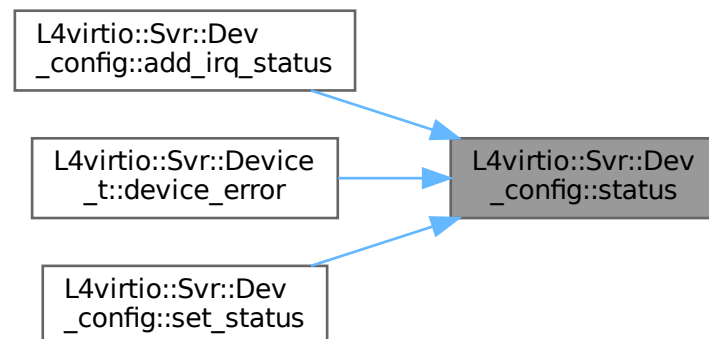
Current device status register (trusted).

The status returned by this function is value stored internally and cannot be written by the guest (i.e., the value can be taken as trusted.)

Definition at line 222 of file [l4virtio](#).

Referenced by [add\\_irq\\_status\(\)](#), [L4virtio::Svr::Device\\_t< DATA >::device\\_error\(\)](#), and [set\\_status\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

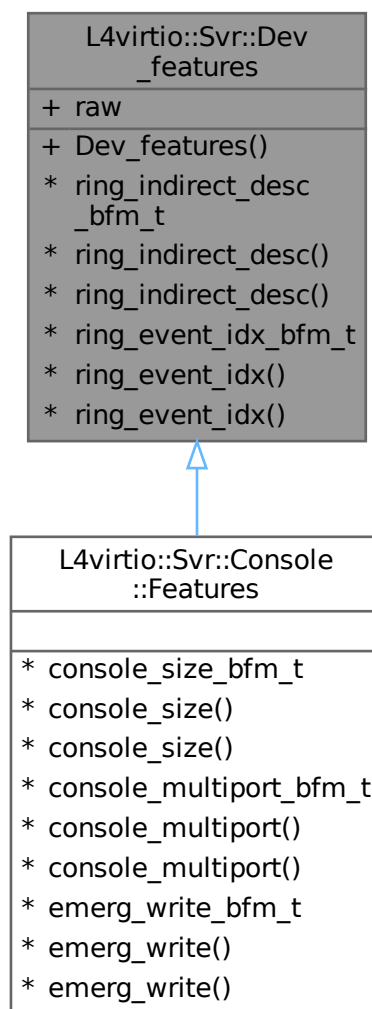
- l4/l4virtio/server/l4virtio

## 15.398 L4virtio::Svr::Dev\_features Struct Reference

Type for device feature bitmap.

```
#include <virtio>
```

Inheritance diagram for L4virtio::Svr::Dev\_features:



Collaboration diagram for L4virtio::Svr::Dev\_features:

L4virtio::Svr::Dev_features
+ raw
+ Dev_features()
* ring_indirect_desc_bfm_t
* ring_indirect_desc()
* ring_indirect_desc()
* ring_event_idx_bfm_t
* ring_event_idx()
* ring_event_idx()

## Public Member Functions

- **Dev\_features** ([l4\\_uint32\\_t](#) v)  
*Make Features from a raw bitmap.*

## Data Fields

- [l4\\_uint32\\_t](#) **raw**  
*The raw value of the features bitmap.*

### 15.398.1 Detailed Description

Type for device feature bitmap.

Definition at line 66 of file [virtio](#).

The documentation for this struct was generated from the following file:

- [l4/l4virtio/server/virtio](#)

## 15.399 L4virtio::Svr::Dev\_status Struct Reference

Type of the device status register.

```
#include <virtio>
```

Collaboration diagram for L4virtio::Svr::Dev\_status:

L4virtio::Svr::Dev_status
+ raw
+ Dev_status()
+ running()
* acked_bfm_t
* acked()
* acked()
* driver_bfm_t
* driver()
* driver()
* driver_ok_bfm_t
* driver_ok()
* driver_ok()
* features_ok_bfm_t
* features_ok()
* features_ok()
* fail_state_bfm_t
* fail_state()
* fail_state()
* device_needs_reset_bfm_t
* device_needs_reset()
* device_needs_reset()
* failed_bfm_t
* failed()
* failed()

### Public Member Functions

- **Dev\_status** ([l4\\_uint32\\_t](#) v)  
*Make Status from raw value.*
- bool [running](#) () const  
*Check if the device is in running state.*

## Data Fields

- unsigned char **raw**  
*Raw value of the VIRTIO device status register.*

### 15.399.1 Detailed Description

Type of the device status register.

Definition at line 32 of file [virtio](#).

### 15.399.2 Member Function Documentation

#### 15.399.2.1 running()

```
bool L4virtio::Svr::Dev_status::running ( ) const [inline]
```

Check if the device is in running state.

#### Returns

true if the device is in running state.

The device is in running state when [acked\(\)](#), [driver\(\)](#), [features\\_ok\(\)](#), and [driver\\_ok\(\)](#) return true, and [device\\_needs\\_reset\(\)](#) and [failed\(\)](#) return false.

Definition at line 57 of file [virtio](#).

References [raw](#).

The documentation for this struct was generated from the following file:

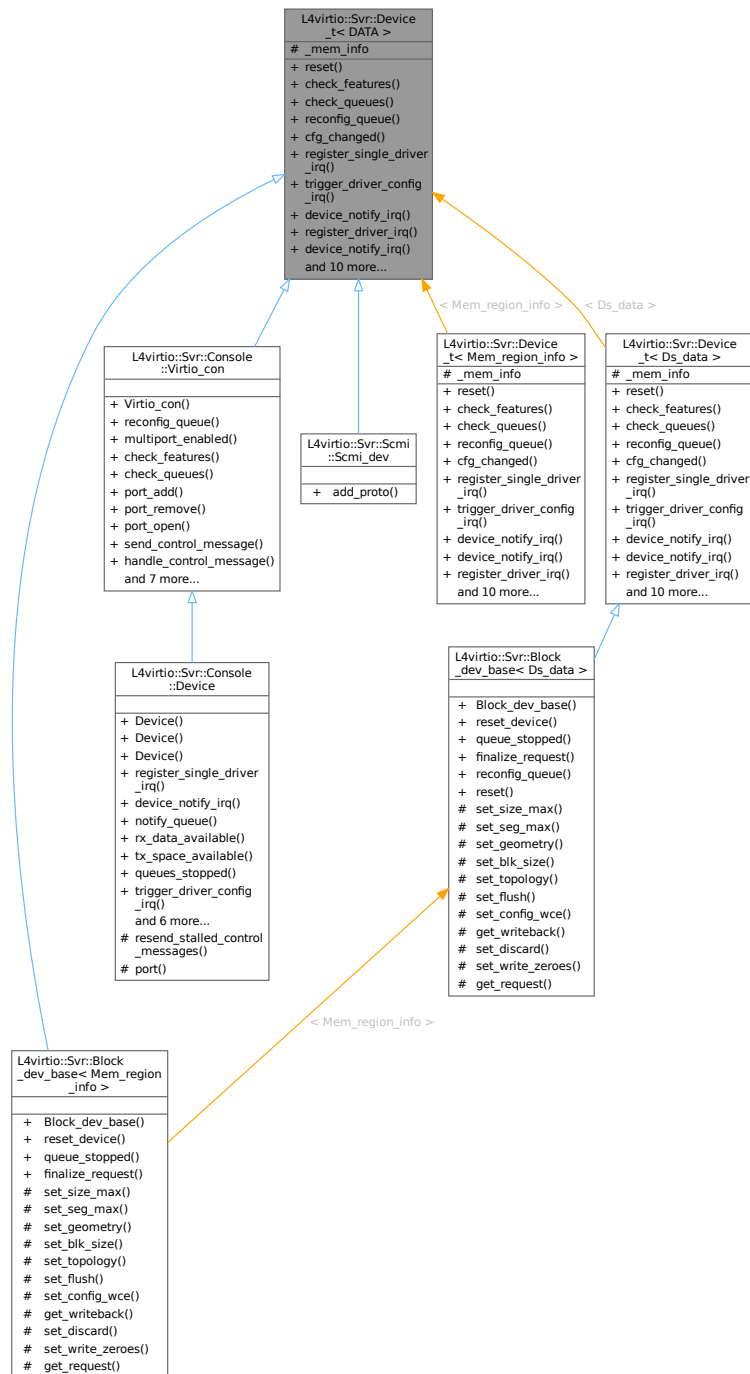
- l4/l4virtio/server/virtio

## 15.400 L4virtio::Svr::Device\_t< DATA > Class Template Reference

Server-side L4-VIRTIO device stub.

```
#include <l4virtio>
```

Inheritance diagram for L4virtio::Svr::Device\_t< DATA >:





Collaboration diagram for L4virtio::Svr::Device\_t< DATA >:

L4virtio::Svr::Device _t< DATA >
# _mem_info
+ reset() + check_features() + check_queues() + reconfig_queue() + cfg_changed() + register_single_driver_irq() + trigger_driver_config_irq() + device_notify_irq() + register_driver_irq() + device_notify_irq() and 10 more...

## Public Member Functions

- virtual void **reset** ()=0  
*reset callback, called for doing a device reset*
- virtual bool **check\_features** ()  
*callback for checking the subset of accepted features*
- virtual bool **check\_queues** ()=0  
*callback for checking if the queues at DRIVER\_OK transition*
- virtual int **reconfig\_queue** (unsigned idx)=0  
*callback for client queue-config request*
- virtual void **cfg\_changed** (unsigned)  
*callback for client device configuration changes*
- virtual void **register\_single\_driver\_irq** ()  
*callback for registering a single guest IRQ for all queues (old-style)*
- virtual void **trigger\_driver\_config\_irq** ()=0  
*callback for triggering configuration change notification IRQ*
- virtual [L4::Cap](#)< [L4::Irq](#) > **device\_notify\_irq** () const  
*callback to gather the device notification IRQ (old-style)*
- virtual void [register\\_driver\\_irq](#) (unsigned idx)  
*Callback for registering an notification IRQ (multi IRQ).*
- virtual [L4::Cap](#)< [L4::Irq](#) > **device\_notify\_irq** (unsigned idx)  
*Callback to gather the device notification IRQ (multi IRQ).*
- virtual unsigned **num\_events\_supported** () const  
*Return the highest notification index supported.*

- **Device\_t** ([Dev\\_config](#) \*dev\_config)  
*Make a device for the given config.*
- **Mem\_list** const \* **mem\_info** () const  
*Get the memory region list used for this device.*
- void [reset\\_queue\\_config](#) (unsigned idx, unsigned num\_max, bool inc\_generation=false)  
*Trigger reset for the configuration space for queue idx.*
- void [init\\_mem\\_info](#) (unsigned num)  
*Initialize the memory region list to the given maximum.*
- void [device\\_error](#) ()  
*Transition device into DEVICE\_NEEDS\_RESET state.*
- bool [setup\\_queue](#) ([Virtqueue](#) \*q, unsigned qn, unsigned num\_max)  
*Enable/disable the specified queue.*
- bool [handle\\_mem\\_cmd\\_write](#) ()  
*Check for a value in the cmd register and handle a write.*
- void **enable\_trusted\_ds\_validation** ()  
*Enable trusted dataspace validation.*
- void [add\\_trusted\\_dataspaces](#) (std::shared\_ptr< Ds\_vector const > ds)  
*Provide a list of trusted dataspaces that can be used for validation.*

## Protected Attributes

- **Mem\_list** **\_mem\_info**  
*Memory region list.*

## 15.400.1 Detailed Description

```
template<typename DATA>
class L4virtio::Svr::Device_t< DATA >
```

Server-side L4-VIRTIO device stub.

This stub supports new-style multi-event registration (using [get\\_device\\_config\(\)](#), [bind\(\)](#) and [get\\_device\\_notification\\_irq\(\)](#)).

Definition at line [795](#) of file [l4virtio](#).

## 15.400.2 Member Function Documentation

### 15.400.2.1 [add\\_trusted\\_dataspaces\(\)](#)

```
template<typename DATA >
void L4virtio::Svr::Device_t< DATA >::add_trusted_dataspaces (
    std::shared_ptr< Ds_vector const > ds ) [inline]
```

Provide a list of trusted dataspaces that can be used for validation.

#### Parameters

<i>ds</i>	list of trusted dataspaces.
-----------	-----------------------------

Definition at line 1192 of file [l4virtio](#).

### 15.400.2.2 device\_error()

```
template<typename DATA >
void L4virtio::Svr::Device_t< DATA >::device_error ( ) [inline]
```

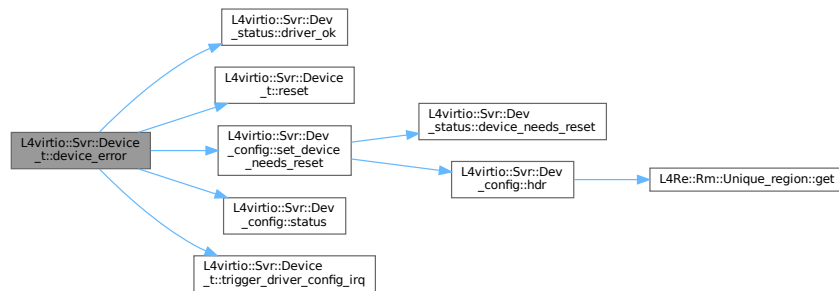
Transition device into DEVICE\_NEEDS\_RESET state.

This function does a full reset, sets the DEVICE\_NEEDS\_RESET bit in the device status register, triggering a guest config IRQ if necessary. The driver still needs to perform its own reset and initialization sequence.

Definition at line 1018 of file [l4virtio](#).

References [L4virtio::Svr::Dev\\_status::driver\\_ok\(\)](#), [L4virtio::Svr::Device\\_t< DATA >::reset\(\)](#), [L4virtio::Svr::Dev\\_config::set\\_device\\_needs\\_reset\(\)](#), [L4virtio::Svr::Dev\\_config::status\(\)](#), and [L4virtio::Svr::Device\\_t< DATA >::trigger\\_driver\\_config\\_irq\(\)](#).

Here is the call graph for this function:



### 15.400.2.3 device\_notify\_irq()

```
template<typename DATA >
virtual L4::Cap< L4::Irq > L4virtio::Svr::Device_t< DATA >::device_notify_irq (
    unsigned idx ) [inline], [virtual]
```

Callback to gather the device notification IRQ (multi IRQ).

The default implementation maps to the implementation for single IRQ notification points.

Definition at line 868 of file [l4virtio](#).

References [L4Re::chksys\(\)](#), [L4virtio::Svr::Device\\_t< DATA >::device\\_notify\\_irq\(\)](#), and [L4\\_ENOSYS](#).

Here is the call graph for this function:



#### 15.400.2.4 `handle_mem_cmd_write()`

```
template<typename DATA >
bool L4virtio::Svr::Device_t< DATA >::handle_mem_cmd_write ( ) [inline]
```

Check for a value in the `cmd` register and handle a write.

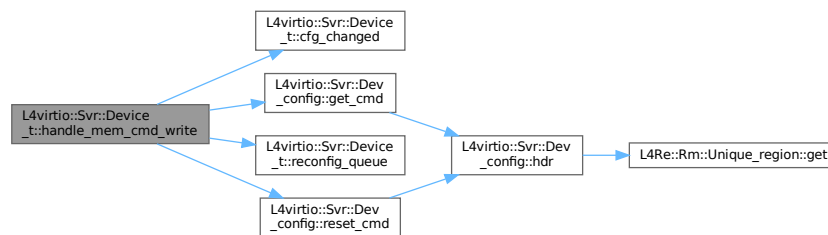
This function checks for a value in the `cmd` register and executes the command if there is any, or returns false if there was no command.

Execution of the command is signaled by a zero in the `cmd` register.

Definition at line 1148 of file `l4virtio`.

References `L4virtio::Svr::Device_t< DATA >::cfg_changed()`, `L4virtio::Svr::Dev_config::get_cmd()`, `L4_LIKELY`, `L4VIRTIO_CMD_CFG_CHANGED`, `L4VIRTIO_CMD_CFG_QUEUE`, `L4VIRTIO_CMD_MASK`, `L4VIRTIO_CMD_SET_STATUS`, `L4virtio::Svr::Device_t< DATA >::reconfig_queue()`, and `L4virtio::Svr::Dev_config::reset_cmd()`.

Here is the call graph for this function:



#### 15.400.2.5 `init_mem_info()`

```
template<typename DATA >
void L4virtio::Svr::Device_t< DATA >::init_mem_info (
    unsigned num ) [inline]
```

Initialize the memory region list to the given maximum.

##### Parameters

<i>num</i>	Maximum number of memory regions that can be managed.
------------	---

Definition at line 1006 of file `l4virtio`.

References `L4virtio::Svr::Device_t< DATA >::_mem_info`.

#### 15.400.2.6 `register_driver_irq()`

```
template<typename DATA >
virtual void L4virtio::Svr::Device_t< DATA >::register_driver_irq (
    unsigned idx ) [inline], [virtual]
```

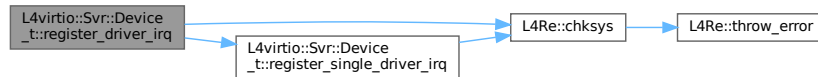
Callback for registering an notification IRQ (multi IRQ).

The default implementation maps to the implementation for single IRQ notification points.

Definition at line 854 of file [l4virtio](#).

References [L4Re::chksys\(\)](#), [L4\\_ENOSYS](#), and [L4virtio::Svr::Device\\_t< DATA >::register\\_single\\_driver\\_irq\(\)](#).

Here is the call graph for this function:



### 15.400.2.7 reset\_queue\_config()

```

template<typename DATA >
void L4virtio::Svr::Device_t< DATA >::reset_queue_config (
    unsigned idx,
    unsigned num_max,
    bool inc_generation = false ) [inline]
  
```

Trigger reset for the configuration space for queue *idx*.

#### Parameters

<i>idx</i>	The queue index to reset.
<i>num_max</i>	Maximum number of entries in this queue.
<i>inc_generation</i>	The config generation will be incremented when this is true.

This function resets the driver-readable configuration space for the queue with the given index. The queue configuration is reset to all 0, and the maximum number of entries in the queue is set to *num\_max*.

Definition at line 996 of file [l4virtio](#).

References [L4virtio::Svr::Dev\\_config::reset\\_queue\(\)](#).

Here is the call graph for this function:



### 15.400.2.8 setup\_queue()

```
template<typename DATA >
bool L4virtio::Svr::Device_t< DATA >::setup_queue (
    Virtqueue * q,
    unsigned qn,
    unsigned num_max ) [inline]
```

Enable/disable the specified queue.

#### Parameters

<i>q</i>	Pointer to the ring that represents the virtqueue internally.
<i>qn</i>	Index of the queue.
<i>num_max</i>	Maximum number of supported entries in this queue.

#### Returns

true for success.

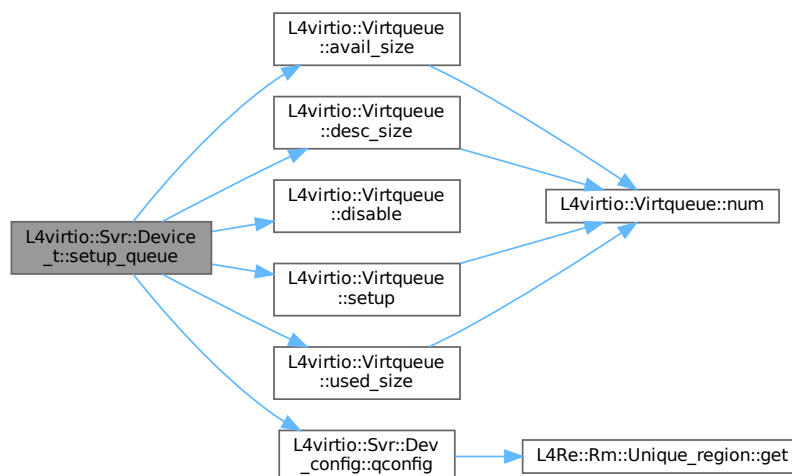
- This function calculates the parameters of the virtqueue from the clients configuration space values, checks the accessibility of the queue data structures and initializes *q* to ready state when all checks succeeded.

Definition at line 1041 of file [l4virtio](#).

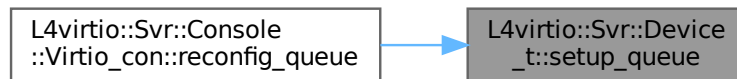
References [L4virtio::Svr::Device\\_t< DATA >::\\_mem\\_info](#), [l4virtio\\_config\\_queue\\_t::avail\\_addr](#), [L4virtio::Virtqueue::avail\\_size\(\)](#), [l4virtio\\_config\\_queue\\_t::desc\\_addr](#), [L4virtio::Virtqueue::desc\\_size\(\)](#), [L4virtio::Virtqueue::disable\(\)](#), [L4\\_UNLIKELY](#), [l4virtio\\_config\\_queue\\_t::num](#), [L4virtio::Svr::Dev\\_config::qconfig\(\)](#), [l4virtio\\_config\\_queue\\_t::ready](#), [L4virtio::Virtqueue::setup\(\)](#), [l4virtio\\_config\\_queue\\_t::used\\_addr](#), and [L4virtio::Virtqueue::used\\_size\(\)](#).

Referenced by [L4virtio::Svr::Console::Virtio\\_con::reconfig\\_queue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

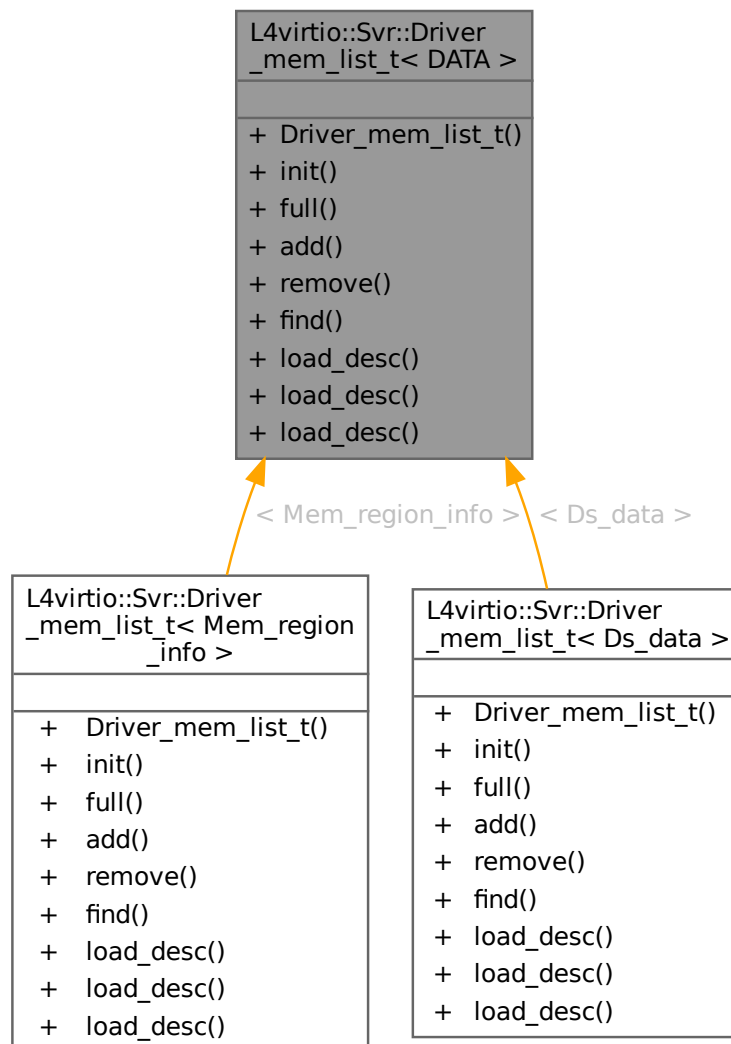
- l4/l4virtio/server/l4virtio

## 15.401 L4virtio::Svr::Driver\_mem\_list\_t< DATA > Class Template Reference

List of driver memory regions assigned to a single L4-VIRTIO transport instance.

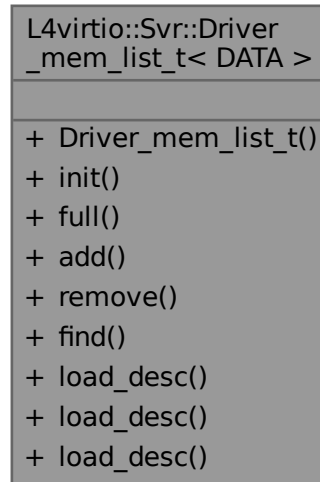
```
#include <l4virtio>
```

Inheritance diagram for L4virtio::Svr::Driver\_mem\_list\_t< DATA >:





Collaboration diagram for L4virtio::Svr::Driver\_mem\_list\_t< DATA >:



## Public Types

- typedef [L4Re::Util::Unique\\_cap< L4Re::Dataspace >](#) **Ds\_cap**  
type for storing a data-space capability internally

## Public Member Functions

- **Driver\_mem\_list\_t ()**  
*Make an empty, zero capacity list.*
- void [init](#) (unsigned max)  
*Make a fresh list with capacity max.*
- bool [full](#) () const
- Mem\_region const \* [add](#) (l4\_uint64\_t drv\_base, l4\_umword\_t size, l4\_addr\_t offset, [Ds\\_cap](#) &&ds)  
*Add a new region to the list.*
- void [remove](#) (Mem\_region const \*r)  
*Remove the given region from the list.*
- Mem\_region \* [find](#) (l4\_uint64\_t base, l4\_umword\_t size) const  
*Find memory region containing the given driver address region.*
- void [load\\_desc](#) ([Virtqueue::Desc](#) const &desc, [Request\\_processor](#) const \*p, [Virtqueue::Desc](#) const \*\*table) const  
*Default implementation for loading an indirect descriptor.*
- void [load\\_desc](#) ([Virtqueue::Desc](#) const &desc, [Request\\_processor](#) const \*p, Mem\_region const \*\*data) const  
*Default implementation returning the Driver\_mem\_region.*
- template<typename ARG >  
void [load\\_desc](#) ([Virtqueue::Desc](#) const &desc, [Request\\_processor](#) const \*p, ARG \*data) const  
*Default implementation returning generic information.*

### 15.401.1 Detailed Description

```
template<typename DATA>
class L4virtio::Svr::Driver_mem_list_t< DATA >
```

List of driver memory regions assigned to a single L4-VIRTIO transport instance.

#### Note

The regions added to this list *must* never overlap.

Definition at line 629 of file [l4virtio](#).

### 15.401.2 Member Function Documentation

#### 15.401.2.1 add()

```
template<typename DATA >
Mem_region const * L4virtio::Svr::Driver_mem_list_t< DATA >::add (
    l4_uint64_t drv_base,
    l4_umword_t size,
    l4_addr_t offset,
    Ds_cap && ds ) [inline]
```

Add a new region to the list.

#### Parameters

<i>drv_base</i>	Driver base address of the region.
<i>size</i>	Size of the region in bytes.
<i>offset</i>	Offset within the data space attached to <i>drv_base</i> .
<i>ds</i>	Data space backing the driver memory.

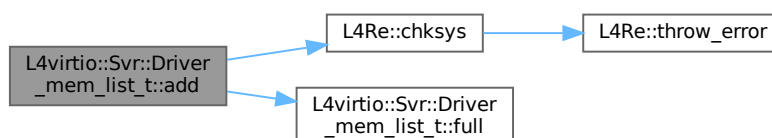
#### Returns

A pointer to the new region.

Definition at line 669 of file [l4virtio](#).

References [L4Re::chksys\(\)](#), [L4virtio::Svr::Driver\\_mem\\_list\\_t< DATA >::full\(\)](#), and [L4\\_ENOMEM](#).

Here is the call graph for this function:



## 15.401.2.2 find()

```
template<typename DATA >
Mem_region * L4virtio::Svr::Driver_mem_list_t< DATA >::find (
    l4_uint64_t base,
    l4_umword_t size ) const [inline]
```

Find memory region containing the given driver address region.

## Parameters

<i>base</i>	Driver base address.
<i>size</i>	Size of the region.

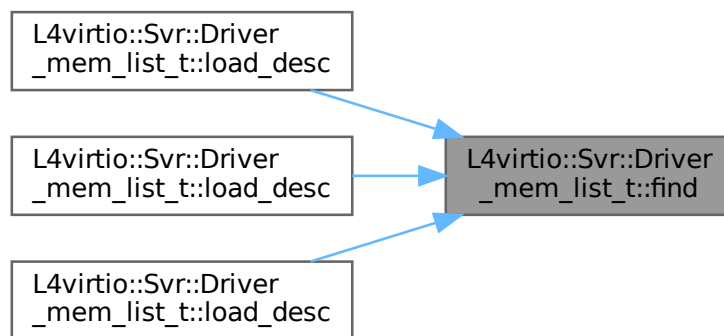
## Returns

Pointer to the region containing the given region, NULL if none is found.

Definition at line 703 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_list\\_t< DATA >::load\\_desc\(\)](#), [L4virtio::Svr::Driver\\_mem\\_list\\_t< DATA >::load\\_desc\(\)](#), and [L4virtio::Svr::Driver\\_mem\\_list\\_t< DATA >::load\\_desc\(\)](#).

Here is the caller graph for this function:



## 15.401.2.3 full()

```
template<typename DATA >
bool L4virtio::Svr::Driver_mem_list_t< DATA >::full ( ) const [inline]
```

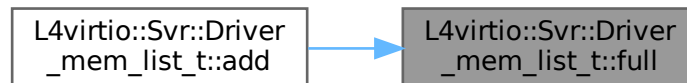
**Returns**

True if the remaining capacity is 0.

Definition at line 658 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_list\\_t< DATA >::add\(\)](#).

Here is the caller graph for this function:

**15.401.2.4 init()**

```
template<typename DATA >
void L4virtio::Svr::Driver_mem_list_t< DATA >::init (
    unsigned max ) [inline]
```

Make a fresh list with capacity *max*.

**Parameters**

<i>max</i>	The capacity of this vector.
------------	------------------------------

Definition at line 650 of file [l4virtio](#).

**15.401.2.5 load\_desc() [1/3]**

```
template<typename DATA >
template<typename ARG >
void L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc (
    Virtqueue::Desc const & desc,
    Request_processor const * p,
    ARG * data ) const [inline]
```

Default implementation returning generic information.

**Template Parameters**

<i>ARG</i>	Abstract argument type used with <a href="#">Request_processor::start()</a> and <a href="#">Request_processor::next()</a> to deliver the result of loading a descriptor. This type must provide a constructor taking three arguments: (1) pointer to a <code>Driver_mem_region</code> , (2) the <a href="#">Virtqueue::Desc</a> descriptor, and (3) a pointer to the calling <a href="#">Request_processor</a> .
------------	--

## Parameters

	<i>desc</i>	The descriptor to load
	<i>p</i>	The request processor calling us
out	<i>data</i>	Shall be assigned to ARG(mem, desc, p)

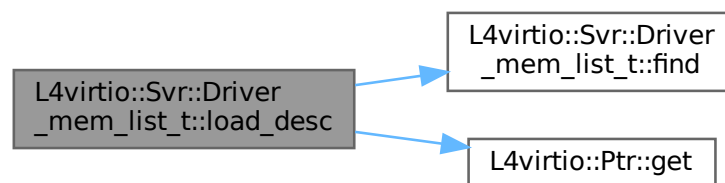
## Exceptions

<a href="#"><i>Bad_descriptor</i></a>	The descriptor address could not be translated.
---------------------------------------	---

Definition at line 764 of file [l4virtio](#).

References [L4virtio::Virtqueue::Desc::addr](#), [L4virtio::Svr::Bad\\_descriptor::Bad\\_address](#), [L4virtio::Svr::Driver\\_mem\\_list\\_t< DATA >::find](#), [L4virtio::Ptr< T >::get\(\)](#), [L4\\_UNLIKELY](#), and [L4virtio::Virtqueue::Desc::len](#).

Here is the call graph for this function:



## 15.401.2.6 load\_desc() [2/3]

```

template<typename DATA >
void L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc (
    Virtqueue::Desc const & desc,
    Request_processor const * p,
    Mem_region const ** data ) const [inline]
  
```

Default implementation returning the Driver\_mem\_region.

## Parameters

	<i>desc</i>	The descriptor to load
	<i>p</i>	The request processor calling us
out	<i>data</i>	Shall be set to a pointer to the Driver_mem_region that covers the descriptor.

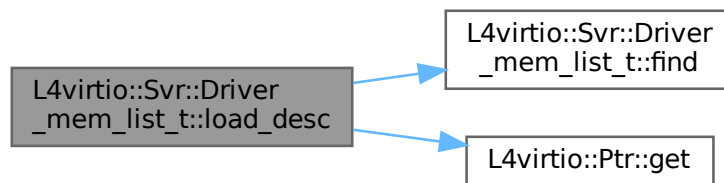
## Exceptions

<a href="#"><i>Bad_descriptor</i></a>	The descriptor address could not be translated.
---------------------------------------	---

Definition at line 737 of file [l4virtio](#).

References [L4virtio::Virtqueue::Desc::addr](#), [L4virtio::Svr::Bad\\_descriptor::Bad\\_address](#), [L4virtio::Svr::Driver\\_mem\\_list\\_t< DATA >::find](#), [L4virtio::Ptr< T >::get\(\)](#), [L4\\_UNLIKELY](#), and [L4virtio::Virtqueue::Desc::len](#).

Here is the call graph for this function:



#### 15.401.2.7 `load_desc()` [3/3]

```

template<typename DATA >
void L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc (
    Virtqueue::Desc const & desc,
    Request_processor const * p,
    Virtqueue::Desc const ** table ) const [inline]
  
```

Default implementation for loading an indirect descriptor.

##### Parameters

	<i>desc</i>	The descriptor to load
	<i>p</i>	The request processor calling us
out	<i>table</i>	Shall be set to the loaded descriptor table

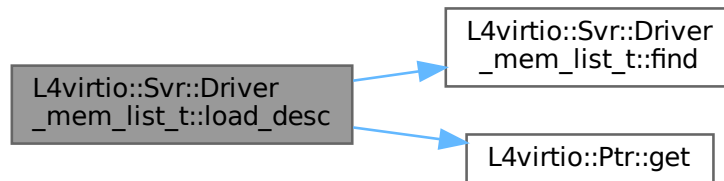
##### Exceptions

<a href="#">Bad_descriptor</a>	The descriptor address could not be translated.
--------------------------------	---

Definition at line 717 of file [l4virtio](#).

References [L4virtio::Virtqueue::Desc::addr](#), [L4virtio::Svr::Bad\\_descriptor::Bad\\_address](#), [L4virtio::Svr::Driver\\_mem\\_list\\_t< DATA >::find](#), [L4virtio::Ptr< T >::get\(\)](#), [L4\\_UNLIKELY](#), and [L4virtio::Virtqueue::Desc::len](#).

Here is the call graph for this function:



### 15.401.2.8 remove()

```
template<typename DATA >
void L4virtio::Svr::Driver_mem_list_t< DATA >::remove (
    Mem_region const * r ) [inline]
```

Remove the given region from the list.

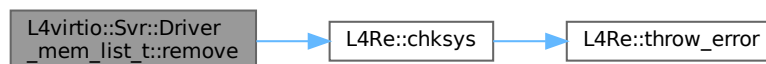
#### Parameters

<i>r</i>	The region to remove (result from <a href="#">add()</a> , or <a href="#">find()</a> ).
----------	--

Definition at line 683 of file [l4virtio](#).

References [L4Re::chksys\(\)](#), and [L4\\_ERANGE](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

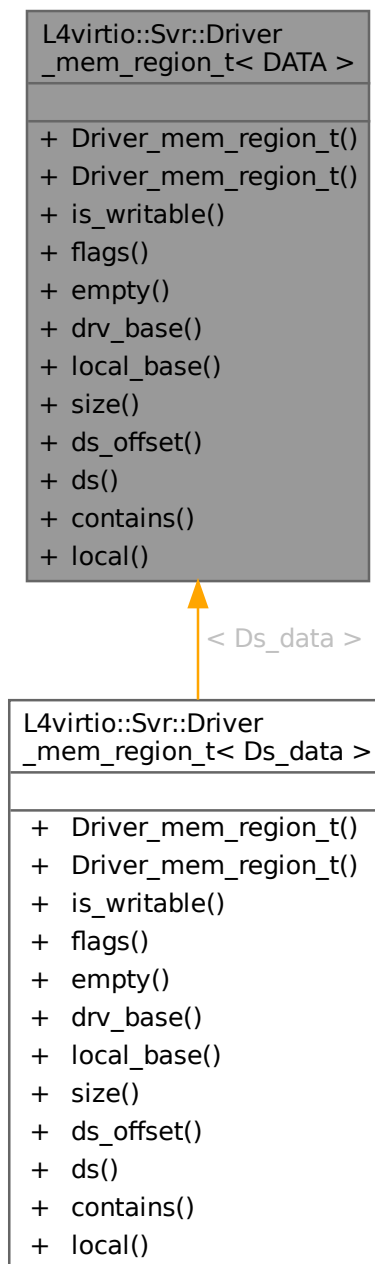
- `l4/l4virtio/server/l4virtio`

## 15.402 L4virtio::Svr::Driver\_mem\_region\_t< DATA > Class Template Reference

Region of driver memory, that shall be managed locally.

```
#include <l4virtio>
```

Inheritance diagram for L4virtio::Svr::Driver\_mem\_region\_t< DATA >:





Collaboration diagram for L4virtio::Svr::Driver\_mem\_region\_t< DATA >:

L4virtio::Svr::Driver_mem_region_t< DATA >
<ul style="list-style-type: none"> <li>+ Driver_mem_region_t()</li> <li>+ Driver_mem_region_t()</li> <li>+ is_writable()</li> <li>+ flags()</li> <li>+ empty()</li> <li>+ drv_base()</li> <li>+ local_base()</li> <li>+ size()</li> <li>+ ds_offset()</li> <li>+ ds()</li> <li>+ contains()</li> <li>+ local()</li> </ul>

### Public Member Functions

- **Driver\_mem\_region\_t ()**  
*Make default empty memory region.*
- **Driver\_mem\_region\_t (l4\_uint64\_t drv\_base, l4\_umword\_t size, l4\_addr\_t offset, Ds\_cap &&ds)**  
*Make a local memory region for the given driver values.*
- bool **is\_writable ()** const
- Flags **flags ()** const
- bool **empty ()** const
- **l4\_uint64\_t drv\_base ()** const
- **l4\_addr\_t local\_base ()** const
- **l4\_umword\_t size ()** const
- **l4\_addr\_t ds\_offset ()** const
- **L4::Cap< L4Re::Dataspace > ds ()** const
- bool **contains (l4\_uint64\_t base, l4\_umword\_t size)** const  
*Test if the given driver address range is within this region.*
- template<typename T >  
T \* **local (Ptr< T > p)** const  
*Get the local address for driver address p.*

## 15.402.1 Detailed Description

template<typename DATA>

class L4virtio::Svr::Driver\_mem\_region\_t< DATA >

Region of driver memory, that shall be managed locally.

## Template Parameters

<i>DATA</i>	Class defining additional information
-------------	---------------------------------------

Definition at line 450 of file [l4virtio](#).

## 15.402.2 Constructor & Destructor Documentation

### 15.402.2.1 Driver\_mem\_region\_t()

```
template<typename DATA >
L4virtio::Svr::Driver_mem_region_t< DATA >::Driver_mem_region_t (
    l4_uint64_t drv_base,
    l4_umword_t size,
    l4_addr_t offset,
    Ds_cap && ds ) [inline]
```

Make a local memory region for the given driver values.

## Parameters

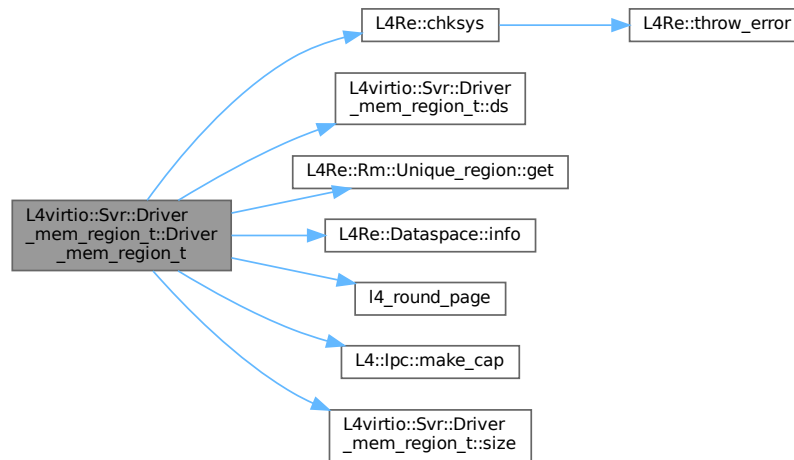
<i>drv_base</i>	Base address of the memory region used by the driver.
<i>size</i>	Size of the memory region.
<i>offset</i>	Offset within the data space that is mapped to <i>drv_base</i> within the driver.
<i>ds</i>	Data space capability backing the memory.

This constructor attaches the region of given data space to the local address space and stores the corresponding data for later reference.

Definition at line 498 of file [l4virtio](#).

References [L4Re::chksys\(\)](#), [L4virtio::Svr::Driver\\_mem\\_region\\_t< DATA >::ds\(\)](#), [L4Re::Dataspace::Stats::flags](#), [L4Re::Rm::Unique\\_region< T >::get\(\)](#), [L4Re::Dataspace::info\(\)](#), [L4\\_CAP\\_FPAGE\\_RO](#), [L4\\_CAP\\_FPAGE\\_RW](#), [L4\\_EINVAL](#), [L4\\_ENOSYS](#), [L4\\_ERANGE](#), [L4\\_PAGESIZE](#), [l4\\_round\\_page\(\)](#), [L4\\_SUPERPAGESHIFT](#), [L4::lpc::make\\_cap\(\)](#), [L4Re::Rm::F::R](#), [L4Re::Rm::F::Search\\_addr](#), [L4virtio::Svr::Driver\\_mem\\_region\\_t< DATA >::size\(\)](#), [L4Re::Dataspace::Stats::size](#), [L4Re::Dataspace::F::W](#), and [L4Re::Rm::F::W](#).

Here is the call graph for this function:



### 15.402.3 Member Function Documentation

#### 15.402.3.1 contains()

```

template<typename DATA >
bool L4virtio::Svr::Driver_mem_region_t< DATA >::contains (
    l4_uint64_t base,
    l4_umword_t size ) const [inline]

```

Test if the given driver address range is within this region.

##### Parameters

<i>base</i>	The driver base address.
<i>size</i>	The size of the region to lookup.

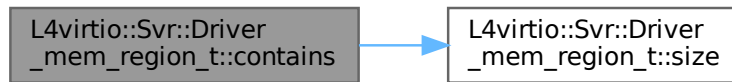
##### Returns

true if the given driver address region is contained in this region, false else.

Definition at line 592 of file [l4virtio](#).

References [L4virtio::Svr::Driver\\_mem\\_region\\_t< DATA >::size\(\)](#).

Here is the call graph for this function:



### 15.402.3.2 `drv_base()`

```
template<typename DATA >
l4_uint64_t L4virtio::Svr::Driver_mem_region_t< DATA >::drv_base ( ) const [inline]
```

#### Returns

The base address used by the driver.

Definition at line 571 of file [l4virtio](#).

### 15.402.3.3 `ds()`

```
template<typename DATA >
L4::Cap< L4Re::Dataspace > L4virtio::Svr::Driver_mem_region_t< DATA >::ds ( ) const [inline]
```

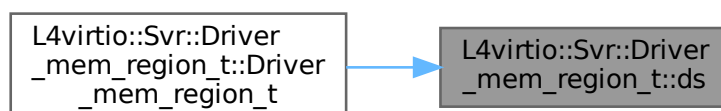
#### Returns

The data space capability for this region.

Definition at line 583 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_region\\_t< DATA >::Driver\\_mem\\_region\\_t\(\)](#).

Here is the caller graph for this function:



#### 15.402.3.4 ds\_offset()

```
template<typename DATA >
l4_addr_t L4virtio::Svr::Driver_mem_region_t< DATA >::ds_offset ( ) const [inline]
```

##### Returns

The offset within the data space.

Definition at line 580 of file [l4virtio](#).

#### 15.402.3.5 empty()

```
template<typename DATA >
bool L4virtio::Svr::Driver_mem_region_t< DATA >::empty ( ) const [inline]
```

##### Returns

True if the region is empty (size == 0), false otherwise.

Definition at line 567 of file [l4virtio](#).

#### 15.402.3.6 flags()

```
template<typename DATA >
Flags L4virtio::Svr::Driver_mem_region_t< DATA >::flags ( ) const [inline]
```

##### Returns

The flags for this region.

Definition at line 564 of file [l4virtio](#).

#### 15.402.3.7 is\_writable()

```
template<typename DATA >
bool L4virtio::Svr::Driver_mem_region_t< DATA >::is_writable ( ) const [inline]
```

##### Returns

True if the region is writable, false otherwise.

Definition at line 561 of file [l4virtio](#).

#### 15.402.3.8 local()

```
template<typename DATA >
template<typename T >
T * L4virtio::Svr::Driver_mem_region_t< DATA >::local (
    Ptr< T > p ) const [inline]
```

Get the local address for driver address *p*.

## Parameters

<i>p</i>	Driver address to translate.
----------	------------------------------

## Precondition

*p* must be contained in this region.

## Returns

Local address for the given driver address *p*.

Definition at line 616 of file [l4virtio](#).

References [L4virtio::Ptr< T >::get\(\)](#).

Here is the call graph for this function:



## 15.402.3.9 local\_base()

```
template<typename DATA >
l4_addr_t L4virtio::Svr::Driver_mem_region_t< DATA >::local_base ( ) const [inline]
```

## Returns

The local base address.

Definition at line 574 of file [l4virtio](#).

References [L4Re::Rm::Unique\\_region< T >::get\(\)](#).

Here is the call graph for this function:



**15.402.3.10 size()**

```
template<typename DATA >
l4_umword_t L4virtio::Svr::Driver_mem_region_t< DATA >::size ( ) const [inline]
```

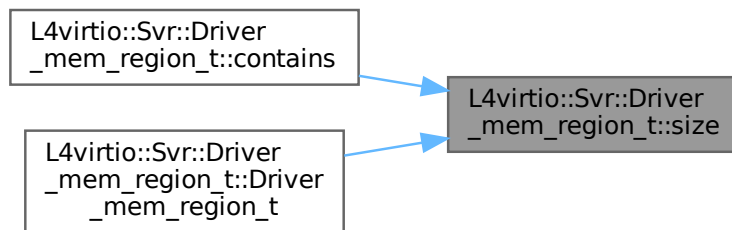
**Returns**

The size of the region in bytes.

Definition at line 577 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_region\\_t< DATA >::contains\(\)](#), and [L4virtio::Svr::Driver\\_mem\\_region\\_t< DATA >::Driver\\_m](#)

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- [l4/l4virtio/server/l4virtio](#)

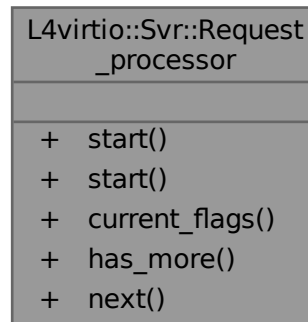
**15.403 L4virtio::Svr::Request\_processor Class Reference**

Encapsulate the state for processing a VIRTIO request.

```
#include <virtio>
```

Inherited by [L4virtio::Svr::Scmi::Queue\\_worker< L4virtio::Svr::Scmi::Scmi\\_dev >](#), and [L4virtio::Svr::Scmi::Queue\\_worker< OBSERV >](#).

Collaboration diagram for L4virtio::Svr::Request\_processor:



### Public Member Functions

- `template<typename DESC_MAN , typename ... ARGS>`  
`void start (DESC_MAN *dm, Virtqueue *ring, Virtqueue::Head_desc const &request, ARGS... args)`  
*Start processing a new request.*
- `template<typename DESC_MAN , typename ... ARGS>`  
`Virtqueue::Request const & start (DESC_MAN *dm, Virtqueue::Request const &request, ARGS... args)`  
*Start processing a new request.*
- `Virtqueue::Desc::Flags current_flags () const`  
*Get the flags of the currently processed descriptor.*
- `bool has_more () const`  
*Are there more chained descriptors?*
- `template<typename DESC_MAN , typename ... ARGS>`  
`bool next (DESC_MAN *dm, ARGS... args)`  
*Switch to the next descriptor in a descriptor chain.*

### 15.403.1 Detailed Description

Encapsulate the state for processing a VIRTIO request.

A VIRTIO request is a possibly chained list of descriptors retrieved from the available ring of a virtqueue, using [Virtqueue::next\\_avail\(\)](#).

The descriptor processing depends on helper (DESC\_MAN) for interpreting the descriptors in the context of the device implementation.

DESC\_MAN has to provide the functionality to safely dereference a descriptor from a descriptor list.

The following methods must be provided by DESC\_MAN:

- `DESC_MAN::load_desc(Virtqueue::Desc const &desc,`  
`Request_processor const *proc,`  
`Virtqueue::Desc const **table)`

This function is used to dereference `desc` as an indirect descriptor table, and must return a pointer to an indirect descriptor table.

- `DESC_MAN::load_desc(Virtqueue::Desc const &desc,`  
`Request_processor const *proc, ...)`

This function is used to dereference a descriptor as a normal data buffer, and '...' are the arguments that are passed to [start\(\)](#) and [next\(\)](#).

Definition at line 453 of file [virtio](#).



## 15.403.2 Member Function Documentation

### 15.403.2.1 current\_flags()

```
Virtqueue::Desc::Flags L4virtio::Svr::Request_processor::current_flags ( ) const [inline]
```

Get the flags of the currently processed descriptor.

#### Returns

The flags of the currently processed descriptor.

Definition at line 526 of file [virtio](#).

References [L4virtio::Virtqueue::Desc::flags](#).

### 15.403.2.2 has\_more()

```
bool L4virtio::Svr::Request_processor::has_more ( ) const [inline]
```

Are there more chained descriptors?

#### Returns

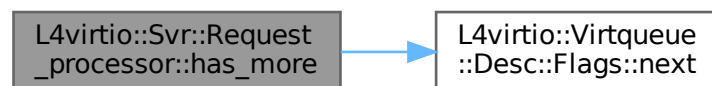
true if there are more chained descriptors in the current request.

Definition at line 534 of file [virtio](#).

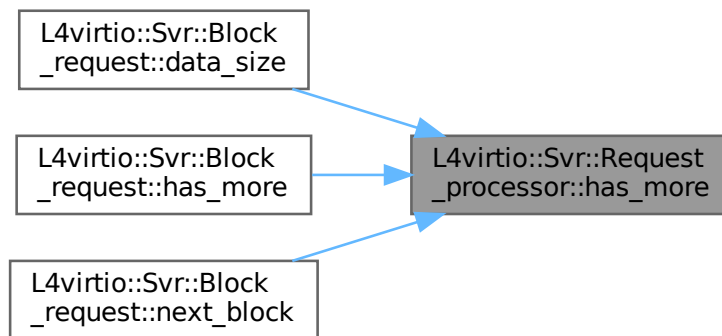
References [L4virtio::Virtqueue::Desc::flags](#), and [L4virtio::Virtqueue::Desc::Flags::next\(\)](#).

Referenced by [L4virtio::Svr::Block\\_request< Ds\\_data >::data\\_size\(\)](#), [L4virtio::Svr::Block\\_request< Ds\\_data >::has\\_more\(\)](#), and [L4virtio::Svr::Block\\_request< Ds\\_data >::next\\_block\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.403.2.3 next()

```

template<typename DESC_MAN , typename ... ARGS>
bool L4virtio::Svr::Request_processor::next (
    DESC_MAN * dm,
    ARGS... args ) [inline]
  
```

Switch to the next descriptor in a descriptor chain.

#### Template Parameters

<i>DESC_MAN</i>	Type of descriptor manager (implicit).
-----------------	--

#### Parameters

<i>dm</i>	Descriptor manager that is used to translate VIRTIO descriptor addresses.
<i>args</i>	Extra arguments passed to <code>dm-&gt;load_desc()</code>

#### Return values

<i>true</i>	A next descriptor is available.
<i>false</i>	No descriptor available.

#### Exceptions

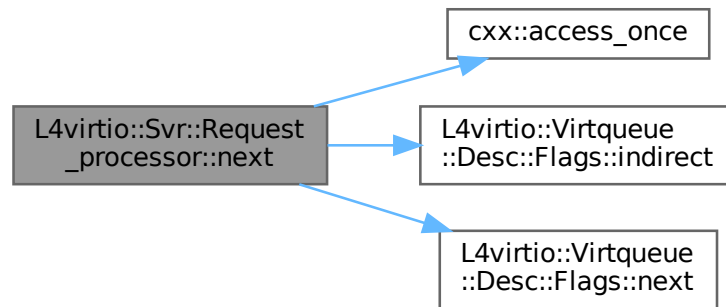
<a href="#">Bad_descriptor</a>	The <code>next</code> index of this descriptor is invalid.
--------------------------------	--

Definition at line 551 of file [virtio](#).

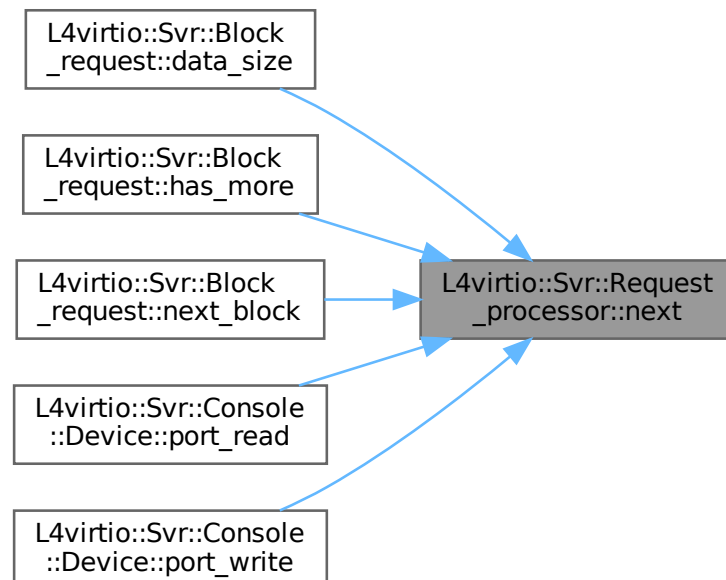
References [cxx::access\\_once\(\)](#), [L4virtio::Svr::Bad\\_descriptor::Bad\\_flags](#), [L4virtio::Svr::Bad\\_descriptor::Bad\\_next](#), [L4virtio::Virtqueue::Desc::flags](#), [L4virtio::Virtqueue::Desc::Flags::indirect\(\)](#), [L4\\_UNLIKELY](#), [L4virtio::Virtqueue::Desc::Flags::next\(\)](#), and [L4virtio::Virtqueue::Desc::next](#).

Referenced by [L4virtio::Svr::Block\\_request<Ds\\_data>::data\\_size\(\)](#), [L4virtio::Svr::Block\\_request<Ds\\_data>::has\\_more\(\)](#), [L4virtio::Svr::Block\\_request<Ds\\_data>::next\\_block\(\)](#), [L4virtio::Svr::Console::Device::port\\_read\(\)](#), and [L4virtio::Svr::Console::Device::port\\_write\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



**15.403.2.4 start() [1/2]**

```
template<typename DESC_MAN , typename ... ARGS>
void L4virtio::Svr::Request_processor::start (
    DESC_MAN * dm,
    Virtqueue * ring,
    Virtqueue::Head_desc const & request,
    ARGS... args ) [inline]
```

Start processing a new request.

**Template Parameters**

<i>DESC_MAN</i>	Type of descriptor manager (implicit).
-----------------	--

**Parameters**

<i>dm</i>	Descriptor manager that is used to translate VIRTIO descriptor addresses.
<i>ring</i>	VIRTIO ring of the request.
<i>request</i>	VIRTIO request from <a href="#">Virtqueue::next_avail()</a>
<i>args</i>	Extra arguments passed to <code>dm-&gt;load_desc()</code>

**Precondition**

The given request must be valid.

**Exceptions**

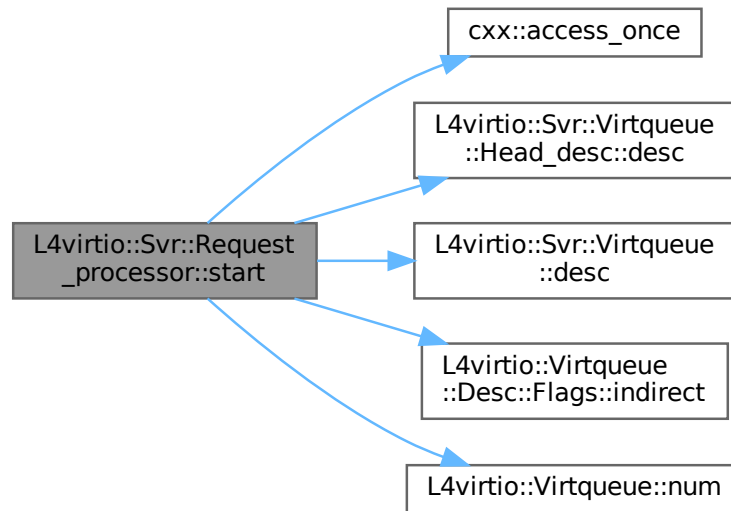
<a href="#">Bad_descriptor</a>	The descriptor has an invalid size or <code>load_desc()</code> has thrown an exception by itself.
--------------------------------	---

Definition at line 482 of file [virtio](#).

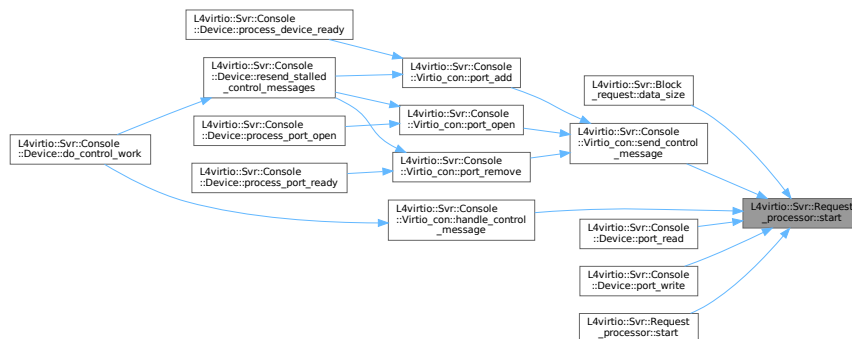
References [cxx::access\\_once\(\)](#), [L4virtio::Svr::Bad\\_descriptor::Bad\\_size](#), [L4virtio::Svr::Virtqueue::Head\\_desc::desc\(\)](#), [L4virtio::Svr::Virtqueue::desc\(\)](#), [L4virtio::Virtqueue::Desc::flags](#), [L4virtio::Virtqueue::Desc::Flags::indirect\(\)](#), [L4\\_UNLIKELY](#), [L4virtio::Virtqueue::Desc::len](#), and [L4virtio::Virtqueue::num\(\)](#).

Referenced by [L4virtio::Svr::Block\\_request<Ds\\_data>::data\\_size\(\)](#), [L4virtio::Svr::Console::Virtio\\_con::handle\\_control\\_message\(\)](#), [L4virtio::Svr::Console::Device::port\\_read\(\)](#), [L4virtio::Svr::Console::Device::port\\_write\(\)](#), [L4virtio::Svr::Console::Virtio\\_con::send\\_control\\_message\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.403.2.5 start() [2/2]

```

template<typename DESC_MAN , typename ... ARGS>
Virtqueue::Request const & L4virtio::Svr::Request_processor::start (
    DESC_MAN * dm,
    Virtqueue::Request const & request,
    ARGS... args ) [inline]
  
```

Start processing a new request.

## Template Parameters

<i>DESC_MAN</i>	Type of descriptor manager (implicit).
-----------------	--

## Parameters

<i>dm</i>	Descriptor manager that is used to translate VIRTIO descriptor addresses.
<i>request</i>	VIRTIO request from <a href="#">Virtqueue::next_avail()</a>
<i>args</i>	Extra arguments passed to <code>dm-&gt;load_desc()</code>

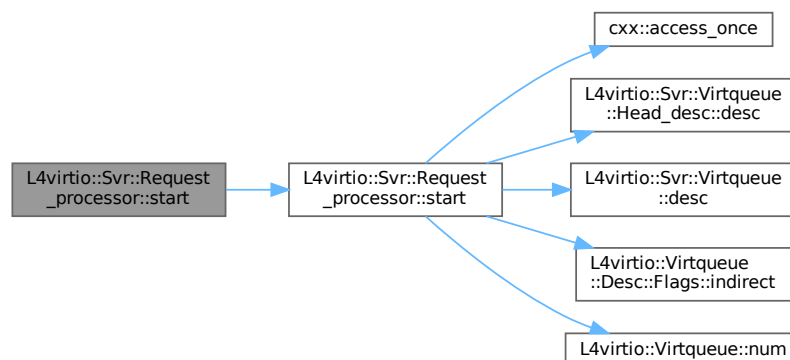
## Precondition

The given request must be valid.

Definition at line 515 of file [virtio](#).

References [start\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

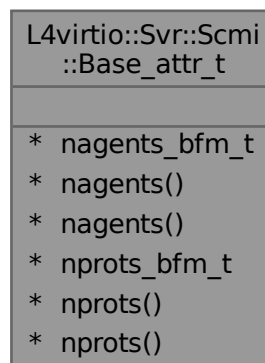
- `l4/l4virtio/server/virtio`

## 15.404 L4virtio::Svr::Scmi::Base\_attr\_t Struct Reference

SCMI base protocol attributes.

```
#include <virtio-scmi-device>
```

Collaboration diagram for L4virtio::Svr::Scmi::Base\_attr\_t:



### 15.404.1 Detailed Description

SCMI base protocol attributes.

Definition at line 90 of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

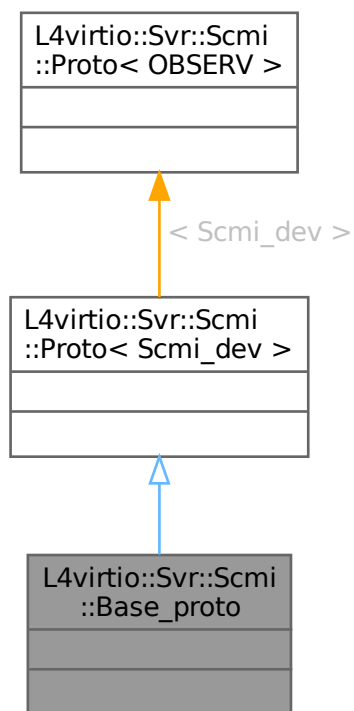
- I4/I4virtio/server/virtio-scmi-device

## 15.405 L4virtio::Svr::Scmi::Base\_proto Class Reference

Base class for the SCMI base protocol.

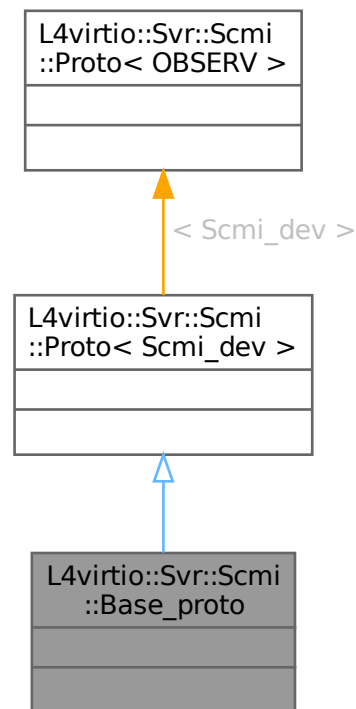
```
#include <virtio-scmi-device>
```

Inheritance diagram for L4virtio::Svr::Scmi::Base\_proto:





Collaboration diagram for L4virtio::Svr::Scmi::Base\_proto:



### 15.405.1 Detailed Description

Base class for the SCMI base protocol.

Use this class as a base to implement the base protocol.

Definition at line 458 of file [virtio-scmi-device](#).

The documentation for this class was generated from the following file:

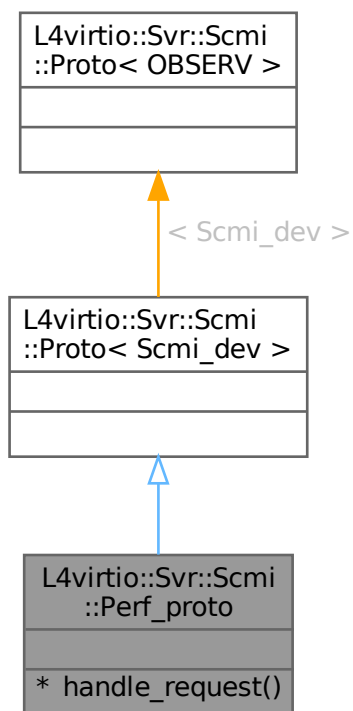
- `I4/I4virtio/server/virtio-scmi-device`

## 15.406 L4virtio::Svr::Scmi::Perf\_proto Class Reference

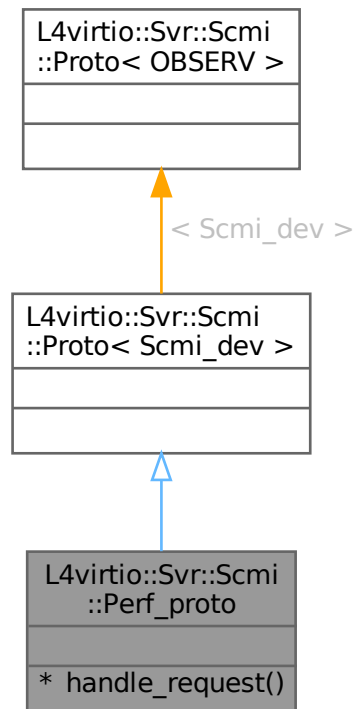
Base class for the SCMI performance protocol.

```
#include <virtio-scmi-device>
```

Inheritance diagram for L4virtio::Svr::Scmi::Perf\_proto:



Collaboration diagram for L4virtio::Svr::Scmi::Perf\_proto:



### 15.406.1 Detailed Description

Base class for the SCMI performance protocol.

Use this class as a base to implement the performance protocol.

If you want to use this from a Uvmm Linux guest, the device tree needs to look something like this:

```

firmware {
    scmi {
        compatible = "arm,scmi-virtio";

        #address-cells = <1>;
        #size-cells = <0>;

        cpufreq: protocol@13 {
            reg = <0x13>;
            #clock-cells = <1>;
        };
    };
};
....

cpu@0 {
    device_type = "cpu";
    reg = <0x0>;
    clocks = <&cpufreq 0>; // domain_id
};

```

Definition at line 662 of file [virtio-scmi-device](#).

The documentation for this class was generated from the following file:

- `I4/I4virtio/server/virtio-scmi-device`

## 15.407 L4virtio::Svr::Scmi::Performance\_attr\_t Struct Reference

SCMI performance protocol attributes.

```
#include <virtio-scmi-device>
```

Collaboration diagram for L4virtio::Svr::Scmi::Performance\_attr\_t:

L4virtio::Svr::Scmi ::Performance_attr_t
<ul style="list-style-type: none"><li>* power_bfm_t</li><li>* power()</li><li>* power()</li><li>* domains_bfm_t</li><li>* domains()</li><li>* domains()</li></ul>

### 15.407.1 Detailed Description

SCMI performance protocol attributes.

Definition at line 112 of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

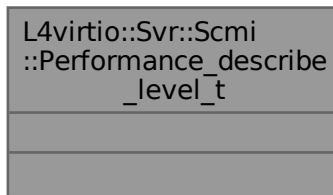
- l4/l4virtio/server/virtio-scmi-device

## 15.408 L4virtio::Svr::Scmi::Performance\_describe\_level\_t Struct Reference

SCMI performance describe level.

```
#include <virtio-scmi-device>
```

Collaboration diagram for L4virtio::Svr::Scmi::Performance\_describe\_level\_t:



### 15.408.1 Detailed Description

SCMI performance describe level.

Definition at line 150 of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

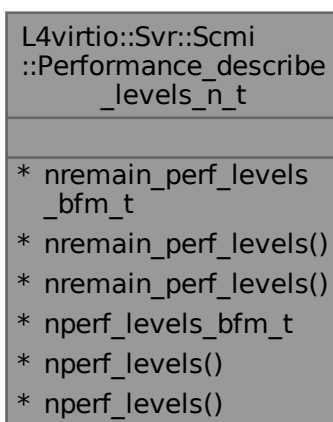
- l4/l4virtio/server/virtio-scmi-device

## 15.409 L4virtio::Svr::Scmi::Performance\_describe\_levels\_n\_t Struct Reference

SCMI performance describe levels numbers.

```
#include <virtio-scmi-device>
```

Collaboration diagram for L4virtio::Svr::Scmi::Performance\_describe\_levels\_n\_t:



### 15.409.1 Detailed Description

SCMI performance describe levels numbers.

Definition at line 142 of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/server/virtio-scmi-device

## 15.410 L4virtio::Svr::Scmi::Performance\_domain\_attr\_t Struct Reference

SCMI performance domain protocol attributes.

```
#include <virtio-scmi-device>
```

Collaboration diagram for L4virtio::Svr::Scmi::Performance\_domain\_attr\_t:

L4virtio::Svr::Scmi ::Performance_domain _attr_t
<ul style="list-style-type: none"> <li>* set_limits_bfm_t</li> <li>* set_limits()</li> <li>* set_limits()</li> <li>* set_perf_level_bfm_t</li> <li>* set_perf_level()</li> <li>* set_perf_level()</li> <li>* perf_limits_change_notify_bfm_t</li> <li>* perf_limits_change_notify()</li> <li>* perf_limits_change_notify()</li> <li>* perf_level_change_notify_bfm_t</li> <li>* perf_level_change_notify()</li> <li>* perf_level_change_notify()</li> <li>* fast_channel_bfm_t</li> <li>* fast_channel()</li> <li>* fast_channel()</li> <li>* rate_limit_bfm_t</li> <li>* rate_limit()</li> <li>* rate_limit()</li> </ul>

### 15.410.1 Detailed Description

SCMI performance domain protocol attributes.

Definition at line 124 of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

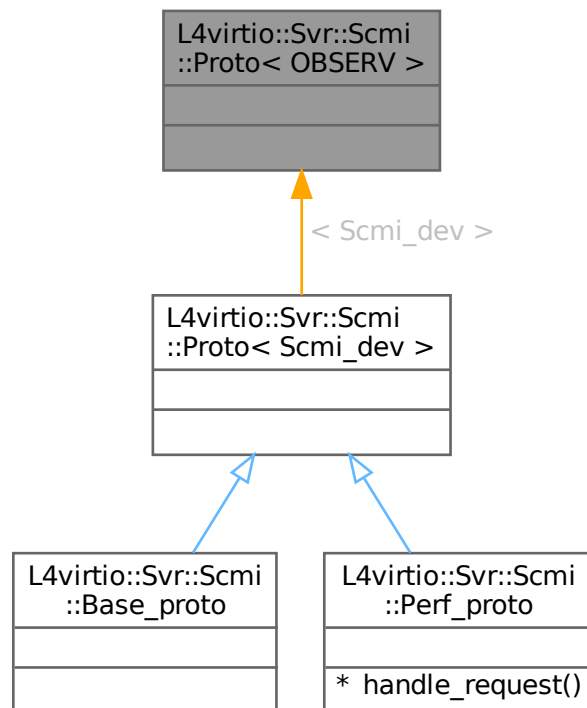
- l4/l4virtio/server/virtio-scmi-device

## 15.411 L4virtio::Svr::Scmi::Proto< OBSERV > Struct Template Reference

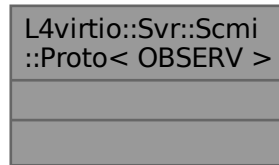
Base class for all protocols.

```
#include <virtio-scmi-device>
```

Inheritance diagram for L4virtio::Svr::Scmi::Proto< OBSERV >:



Collaboration diagram for L4virtio::Svr::Scmi::Proto< OBSERV >:



### 15.411.1 Detailed Description

```
template<typename OBSERV>
struct L4virtio::Svr::Scmi::Proto< OBSERV >
```

Base class for all protocols.

Defines an interface for processing the virtio buffers for the implemented protocol.

Definition at line 299 of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

- I4/I4virtio/server/virtio-scmi-device

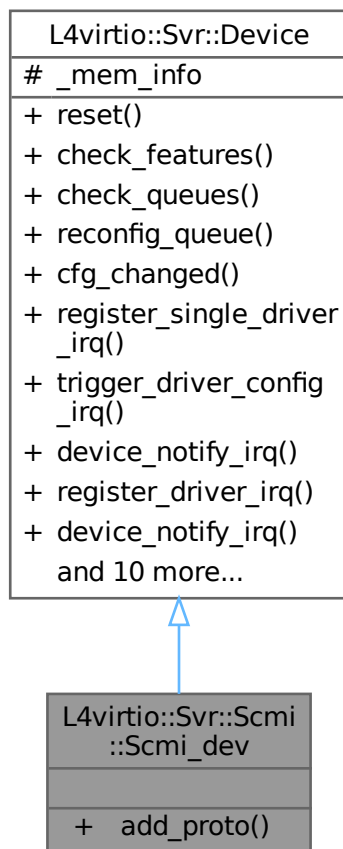
## 15.412 L4virtio::Svr::Scmi::Scmi\_dev Class Reference

A server implementation of the virtio-scmi protocol.

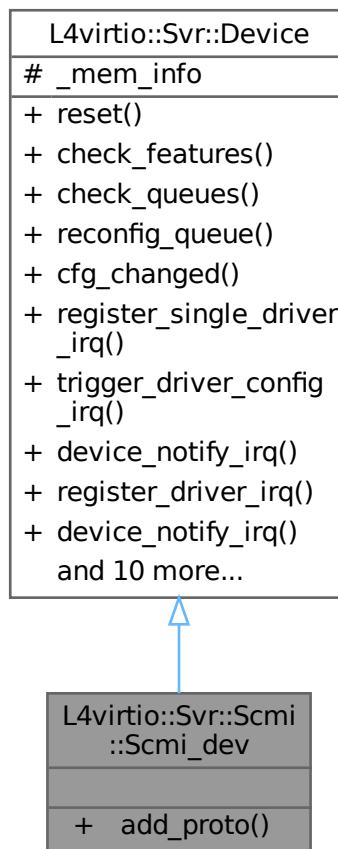
```
#include <virtio-scmi-device>
```



Inheritance diagram for L4virtio::Svr::Scmi::Scmi\_dev:



Collaboration diagram for L4virtio::Svr::Scmi::Scmi\_dev:



## Public Member Functions

- void **add\_proto** ([l4\\_uint32\\_t](#) id, [Proto](#)< [Scmi\\_dev](#) > \*proto)  
*Add an actual protocol implementation with the given id to the server.*

## Public Member Functions inherited from [L4virtio::Svr::Device\\_t](#)< [DATA](#) >

- virtual bool **check\_features** ()  
*callback for checking the subset of accepted features*
- virtual void **cfg\_changed** (unsigned)  
*callback for client device configuration changes*
- virtual void **register\_driver\_irq** (unsigned idx)  
*Callback for registering an notification IRQ (multi IRQ).*
- virtual [L4::Cap](#)< [L4::Irq](#) > **device\_notify\_irq** (unsigned idx)  
*Callback to gather the device notification IRQ (multi IRQ).*
- virtual unsigned **num\_events\_supported** () const  
*Return the highest notification index supported.*

- **Device\_t** ([Dev\\_config](#) \*dev\_config)  
*Make a device for the given config.*
- **Mem\_list** const \* **mem\_info** () const  
*Get the memory region list used for this device.*
- void **reset\_queue\_config** (unsigned idx, unsigned num\_max, bool inc\_generation=false)  
*Trigger reset for the configuration space for queue idx.*
- void **init\_mem\_info** (unsigned num)  
*Initialize the memory region list to the given maximum.*
- void **device\_error** ()  
*Transition device into DEVICE\_NEEDS\_RESET state.*
- bool **setup\_queue** ([Virtqueue](#) \*q, unsigned qn, unsigned num\_max)  
*Enable/disable the specified queue.*
- bool **handle\_mem\_cmd\_write** ()  
*Check for a value in the cmd register and handle a write.*
- void **enable\_trusted\_ds\_validation** ()  
*Enable trusted dataspace validation.*
- void **add\_trusted\_dataspaces** (std::shared\_ptr< [Ds\\_vector](#) const > ds)  
*Provide a list of trusted dataspaces that can be used for validation.*

### Additional Inherited Members

### Protected Attributes inherited from [L4virtio::Svr::Device\\_t](#)< [DATA](#) >

- **Mem\_list** **\_mem\_info**  
*Memory region list.*

## 15.412.1 Detailed Description

A server implementation of the virtio-scmi protocol.

Use this class as a base to implement your own specific SCMI device.

SCMI defines multiple protocols which can be optionally handled. This server implementation is flexible enough to handle any combination of them. The user of this server has to deviate from the provided [Proto](#) classes (for the protocols he want to handle) and needs to implement the required callbacks.

Right now, support for the base and the performance protocol is provided.

The base protocol is mandatory.

If you want to use this from a Uvmm Linux guest, the device tree needs to look something like this:

```
firmware {
    scmi {
        compatible = "arm,scmi-virtio";

        #address-cells = <1>;
        #size-cells = <0>;

        // ... supported protocols ...
    };
};
```

Definition at line 336 of file [virtio-scmi-device](#).

The documentation for this class was generated from the following file:

- l4/l4virtio/server/virtio-scmi-device

## 15.413 L4virtio::Svr::Scmi::Scmi\_hdr\_t Struct Reference

SCMI header.

```
#include <virtio-scmi-device>
```

Collaboration diagram for L4virtio::Svr::Scmi::Scmi\_hdr\_t:

L4virtio::Svr::Scmi ::Scmi_hdr_t
<ul style="list-style-type: none"> <li>* token_bfm_t</li> <li>* token()</li> <li>* token()</li> <li>* protocol_id_bfm_t</li> <li>* protocol_id()</li> <li>* protocol_id()</li> <li>* message_type_bfm_t</li> <li>* message_type()</li> <li>* message_type()</li> <li>* message_id_bfm_t</li> <li>* message_id()</li> <li>* message_id()</li> </ul>

### 15.413.1 Detailed Description

SCMI header.

Definition at line 45 of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

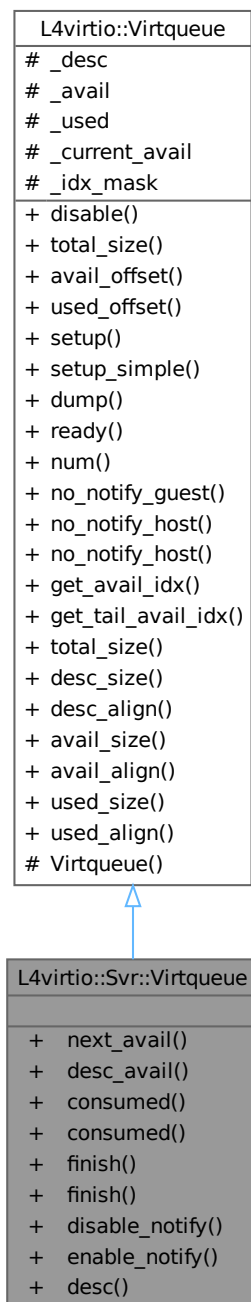
- l4/l4virtio/server/virtio-scmi-device

## 15.414 L4virtio::Svr::Virtqueue Class Reference

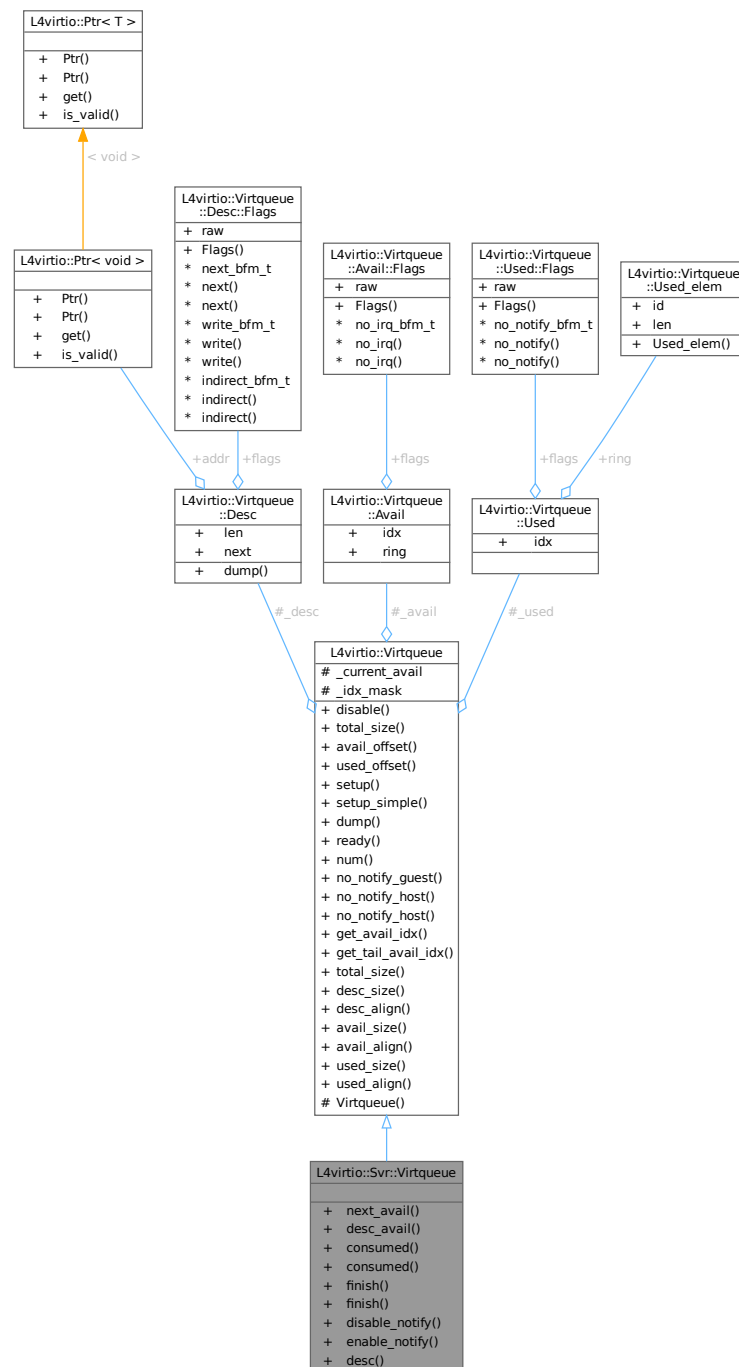
[Virtqueue](#) implementation for the device.

```
#include <virtio>
```

Inheritance diagram for L4virtio::Svr::Virtqueue:



Collaboration diagram for L4virtio::Svr::Virtqueue:



## Data Structures

- class [Head\\_desc](#)  
*VIRTIO request, essentially a descriptor from the available ring.*

## Public Member Functions

- Request [next\\_avail](#) ()

- Get the next available descriptor from the available ring.*

  - bool `desc_avail` () const
- Test for available descriptors.*

  - void `consumed` (Head\_desc const &r, l4\_uint32\_t len=0)
- Put the given descriptor into the used ring.*

  - template<typename ITER >  
void `consumed` (ITER const &begin, ITER const &end)
- Put multiple descriptors into the used ring.*

  - template<typename QUEUE\_OBSERVER >  
void `finish` (Head\_desc &d, QUEUE\_OBSERVER \*o, l4\_uint32\_t len=0)
- Add a descriptor to the used ring, and notify an observer.*

  - template<typename ITER, typename QUEUE\_OBSERVER >  
void `finish` (ITER const &begin, ITER const &end, QUEUE\_OBSERVER \*o)
- Add a range of descriptors to the used ring, and notify an observer once.*

  - void `disable_notify` ()
- Set the 'no notify' flag for this queue.*

  - void `enable_notify` ()
- Clear the 'no notify' flag for this queue.*

  - Desc const \* `desc` (unsigned idx) const
- Get a descriptor from the descriptor list.*

## Public Member Functions inherited from L4virtio::Virtqueue

- void `disable` ()
- Completely disable the queue.*
- unsigned long `total_size` () const
- Calculate the total size of this virtqueue.*
- unsigned long `avail_offset` () const
- Get the offset of the available ring from the descriptor table.*
- unsigned long `used_offset` () const
- Get the offset of the used ring from the descriptor table.*
- void `setup` (unsigned num, void \*desc, void \*avail, void \*used)
- Enable this queue.*
- void `setup_simple` (unsigned num, void \*ring)
- Enable this queue.*
- void `dump` (Desc const \*d) const
- Dump descriptors for this queue.*
- bool `ready` () const
- Test if this queue is in working state.*
- unsigned `num` () const
- bool `no_notify_guest` () const
- Get the no IRQ flag of this queue.*
- bool `no_notify_host` () const
- Get the no notify flag of this queue.*
- void `no_notify_host` (bool value)
- Set the no-notify flag for this queue.*
- l4\_uint16\_t `get_avail_idx` () const
- Get available index from available ring (for debugging).*
- l4\_uint16\_t `get_tail_avail_idx` () const
- Get tail-available index stored in local state (for debugging).*

## Additional Inherited Members

### Public Types inherited from [L4virtio::Virtqueue](#)

- `enum`  
*Fixed alignment values for different parts of a virtqueue.*

### Static Public Member Functions inherited from [L4virtio::Virtqueue](#)

- static unsigned long `total_size` (unsigned `num`)  
*Calculate the total size for a virtqueue of the given dimensions.*
- static unsigned long `desc_size` (unsigned `num`)  
*Calculate the size of the descriptor table for `num` entries.*
- static unsigned long `desc_align` ()  
*Get the alignment in zero LSBs needed for the descriptor table.*
- static unsigned long `avail_size` (unsigned `num`)  
*Calculate the size of the available ring for `num` entries.*
- static unsigned long `avail_align` ()  
*Get the alignment in zero LSBs needed for the available ring.*
- static unsigned long `used_size` (unsigned `num`)  
*Calculate the size of the used ring for `num` entries.*
- static unsigned long `used_align` ()  
*Get the alignment in zero LSBs needed for the used ring.*

### Protected Member Functions inherited from [L4virtio::Virtqueue](#)

- `Virtqueue ()`=default  
*Create a disabled virtqueue.*

### Protected Attributes inherited from [L4virtio::Virtqueue](#)

- `Desc * _desc` = nullptr  
*pointer to descriptor table, NULL if queue is off.*
- `Avail * _avail` = nullptr  
*pointer to available ring.*
- `Used * _used` = nullptr  
*pointer to used ring.*
- `l4_uint16_t _current_avail` = 0  
*The life counter for the queue.*
- `l4_uint16_t _idx_mask` = 0  
*mask used for indexing into the descriptor table and the rings.*

## 15.414.1 Detailed Description

[Virtqueue](#) implementation for the device.

This class represents a single virtqueue, with a local running available index.

#### Note

The [Virtqueue](#) implementation is not thread-safe.

Definition at line 87 of file [virtio](#).



## 15.414.2 Member Function Documentation

### 15.414.2.1 consumed() [1/2]

```
void L4virtio::Svr::Virtqueue::consumed (
    Head_desc const & r,
    14_uint32_t len = 0 ) [inline]
```

Put the given descriptor into the used ring.

#### Parameters

<i>r</i>	Request that shall be marked as finished.
<i>len</i>	The total number of bytes written.

#### Precondition

queue must be in working state.

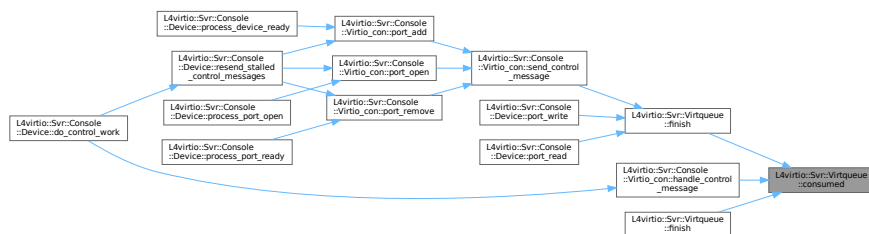
*r* must be a valid request from this queue.

Definition at line 171 of file [virtio](#).

References [L4virtio::Virtqueue::\\_desc](#), [L4virtio::Virtqueue::\\_idx\\_mask](#), [L4virtio::Virtqueue::\\_used](#), [L4virtio::Virtqueue::Used::idx](#), and [L4virtio::Virtqueue::Used::ring](#).

Referenced by [finish\(\)](#), [finish\(\)](#), and [L4virtio::Svr::Console::Virtio\\_con::handle\\_control\\_message\(\)](#).

Here is the caller graph for this function:



### 15.414.2.2 consumed() [2/2]

```
template<typename ITER >
void L4virtio::Svr::Virtqueue::consumed (
    ITER const & begin,
    ITER const & end ) [inline]
```

Put multiple descriptors into the used ring.

A range of descriptors, specified by *begin* and *end* iterators is added. Each iterator points to a struct that has a *first* member that is a [Head\\_desc](#) and a *second* member that is the corresponding number of bytes written.

## Template Parameters

<i>ITER</i>	The type of the iterator (inferred).
-------------	--------------------------------------

## Parameters

<i>begin</i>	Iterator pointing to first new descriptor.
<i>end</i>	Iterator pointing to one past last entry.

## Precondition

queue must be in working state.

Definition at line 194 of file [virtio](#).

References [L4virtio::Virtqueue::\\_desc](#), [L4virtio::Virtqueue::\\_idx\\_mask](#), [L4virtio::Virtqueue::\\_used](#), [L4virtio::Virtqueue::Used::idx](#), and [L4virtio::Virtqueue::Used::ring](#).

## 15.414.2.3 desc()

```
Desc const * L4virtio::Svr::Virtqueue::desc (
    unsigned idx ) const [inline]
```

Get a descriptor from the descriptor list.

## Parameters

<i>idx</i>	The index of the descriptor.
------------	------------------------------

## Precondition

$idx < num$

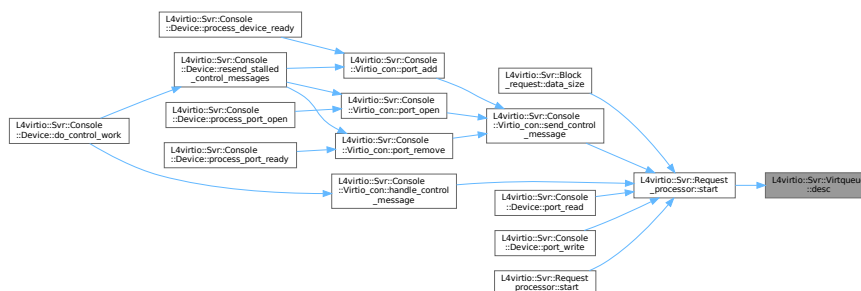
queue must be in working state

Definition at line 279 of file [virtio](#).

References [L4virtio::Virtqueue::\\_desc](#).

Referenced by [L4virtio::Svr::Request\\_processor::start\(\)](#).

Here is the caller graph for this function:



#### 15.414.2.4 desc\_avail()

```
bool L4virtio::Svr::Virtqueue::desc_avail ( ) const [inline]
```

Test for available descriptors.

##### Returns

true if there are descriptors available, false if not.

##### Precondition

The queue must be in working state.

Definition at line 156 of file [virtio](#).

References [L4virtio::Virtqueue::\\_avail](#), [L4virtio::Virtqueue::\\_current\\_avail](#), and [L4virtio::Virtqueue::Avail::idx](#).

#### 15.414.2.5 disable\_notify()

```
void L4virtio::Svr::Virtqueue::disable_notify ( ) [inline]
```

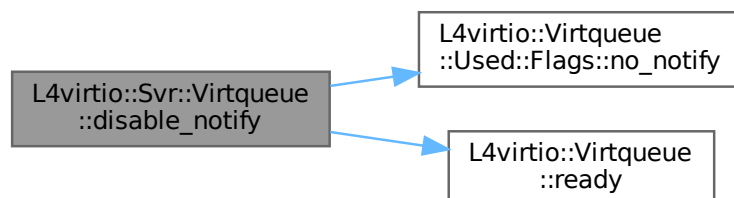
Set the 'no notify' flag for this queue.

This function may be called on a disabled queue.

Definition at line 254 of file [virtio](#).

References [L4virtio::Virtqueue::\\_used](#), [L4virtio::Virtqueue::Used::flags](#), [L4\\_LIKELY](#), [L4virtio::Virtqueue::Used::Flags::no\\_notify\(\)](#), and [L4virtio::Virtqueue::ready\(\)](#).

Here is the call graph for this function:



### 15.414.2.6 enable\_notify()

```
void L4virtio::Svr::Virtqueue::enable_notify ( ) [inline]
```

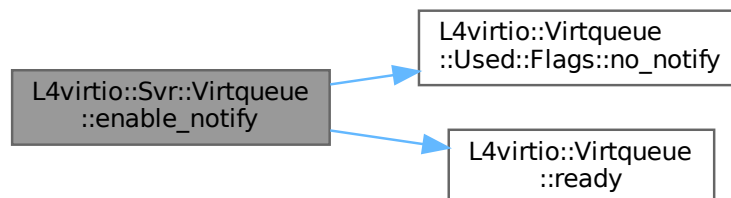
Clear the 'no notify' flag for this queue.

This function may be called on a disabled queue.

Definition at line 265 of file [virtio](#).

References [L4virtio::Virtqueue::\\_used](#), [L4virtio::Virtqueue::Used::flags](#), [L4\\_LIKELY](#), [L4virtio::Virtqueue::Used::Flags::no\\_notify\(\)](#), and [L4virtio::Virtqueue::ready\(\)](#).

Here is the call graph for this function:



### 15.414.2.7 finish() [1/2]

```
template<typename QUEUE_OBSERVER >
void L4virtio::Svr::Virtqueue::finish (
    Head_desc & d,
    QUEUE_OBSERVER * o,
    14_uint32_t len = 0 ) [inline]
```

Add a descriptor to the used ring, and notify an observer.

#### Template Parameters

<code>QUEUE_OBSERVER</code>	The type of the observer (inferred).
-----------------------------	--------------------------------------

#### Parameters

<code>d</code>	descriptor of the request that is to be marked as finished.
<code>o</code>	Pointer to the observer that is notified.
<code>len</code>	Number of bytes written for this request.

**Precondition**

- queue must be in working state.
- d must be a valid request from this queue.

Definition at line 221 of file [virtio](#).

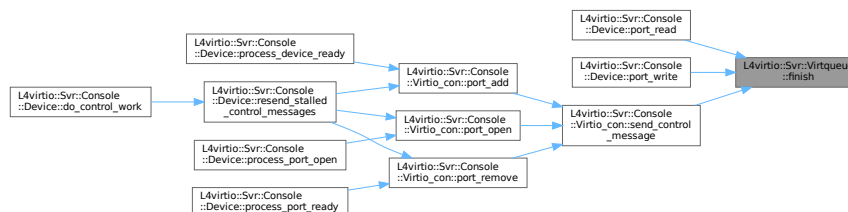
References [consumed\(\)](#).

Referenced by [L4virtio::Svr::Console::Device::port\\_read\(\)](#), [L4virtio::Svr::Console::Device::port\\_write\(\)](#), and [L4virtio::Svr::Console::Virtio\\_con::send\\_control\\_message\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.414.2.8 finish() [2/2]**

```

template<typename ITER , typename QUEUE_OBSERVER >
void L4virtio::Svr::Virtqueue::finish (
    ITER const & begin,
    ITER const & end,
    QUEUE_OBSERVER * o ) [inline]
  
```

Add a range of descriptors to the used ring, and notify an observer once.

The iterators are passed to [consumed<ITER>\(ITER const &, ITER const &\)](#), and the requirements detailed there apply.

**Template Parameters**

<i>ITER</i>	type of the iterator (inferred)
<i>QUEUE_OBSERVER</i>	the type of the observer (inferred).

**Parameters**

<i>begin</i>	iterator pointing to first element.
<i>end</i>	iterator pointing to one past last element.
<i>o</i>	pointer to the observer that is notified.

**Precondition**

queue must be in working state.

Definition at line 243 of file [virtio](#).

References [consumed\(\)](#).

Here is the call graph for this function:

**15.414.2.9 next\_avail()**

```
Request L4virtio::Svr::Virtqueue::next_avail ( ) [inline]
```

Get the next available descriptor from the available ring.

**Precondition**

The queue must be in working state.

**Returns**

A Request for the next available descriptor, the Request is invalid if there are no descriptors in the available ring.







### 15.415.2.3 valid()

```
bool L4virtio::Svr::Virtqueue::Head_desc::valid ( ) const [inline]
```

#### Returns

True if the request is valid (not NULL).

Definition at line 107 of file [virtio](#).

The documentation for this class was generated from the following file:

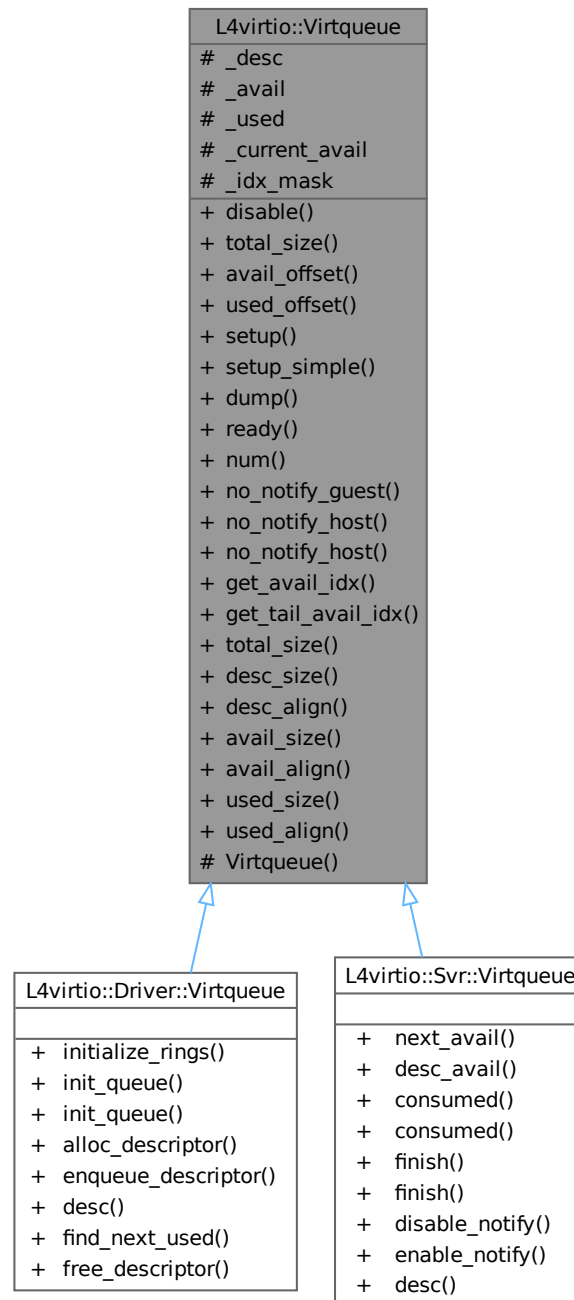
- l4/l4virtio/server/virtio

## 15.416 L4virtio::Virtqueue Class Reference

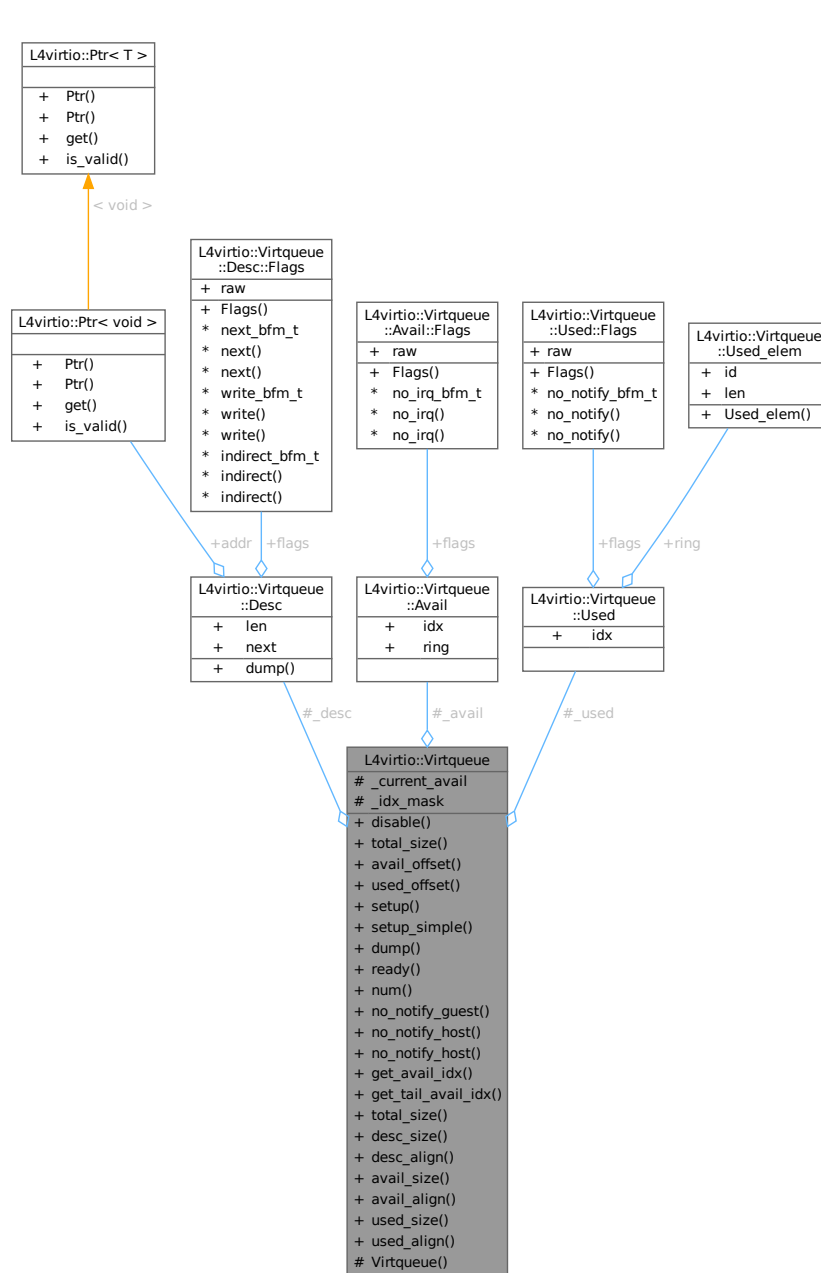
Low-level [Virtqueue](#).

```
#include <virtqueue>
```

Inheritance diagram for L4virtio::Virtqueue:



Collaboration diagram for L4virtio::Virtqueue:



## Data Structures

- class [Avail](#)  
*Type of available ring, this is read-only for the host.*
- class [Desc](#)  
*Descriptor in the descriptor table.*
- class [Used](#)  
*Used ring.*
- struct [Used\\_elem](#)  
*Type of an element of the used ring.*

## Public Types

- enum  
*Fixed alignment values for different parts of a virtqueue.*

## Public Member Functions

- void `disable` ()  
*Completely disable the queue.*
- unsigned long `total_size` () const  
*Calculate the total size of this virtqueue.*
- unsigned long `avail_offset` () const  
*Get the offset of the available ring from the descriptor table.*
- unsigned long `used_offset` () const  
*Get the offset of the used ring from the descriptor table.*
- void `setup` (unsigned `num`, void \*`desc`, void \*`avail`, void \*`used`)  
*Enable this queue.*
- void `setup_simple` (unsigned `num`, void \*`ring`)  
*Enable this queue.*
- void `dump` (`Desc` const \*`d`) const  
*Dump descriptors for this queue.*
- bool `ready` () const  
*Test if this queue is in working state.*
- unsigned `num` () const
- bool `no_notify_guest` () const  
*Get the no IRQ flag of this queue.*
- bool `no_notify_host` () const  
*Get the no notify flag of this queue.*
- void `no_notify_host` (bool `value`)  
*Set the no-notify flag for this queue.*
- `l4_uint16_t` `get_avail_idx` () const  
*Get available index from available ring (for debugging).*
- `l4_uint16_t` `get_tail_avail_idx` () const  
*Get tail-available index stored in local state (for debugging).*

## Static Public Member Functions

- static unsigned long `total_size` (unsigned `num`)  
*Calculate the total size for a virtqueue of the given dimensions.*
- static unsigned long `desc_size` (unsigned `num`)  
*Calculate the size of the descriptor table for `num` entries.*
- static unsigned long `desc_align` ()  
*Get the alignment in zero LSBs needed for the descriptor table.*
- static unsigned long `avail_size` (unsigned `num`)  
*Calculate the size of the available ring for `num` entries.*
- static unsigned long `avail_align` ()  
*Get the alignment in zero LSBs needed for the available ring.*
- static unsigned long `used_size` (unsigned `num`)  
*Calculate the size of the used ring for `num` entries.*
- static unsigned long `used_align` ()  
*Get the alignment in zero LSBs needed for the used ring.*

## Protected Member Functions

- **Virtqueue** ()=default  
*Create a disabled virtqueue.*

## Protected Attributes

- **Desc** \* **\_desc** = nullptr  
*pointer to descriptor table, NULL if queue is off.*
- **Avail** \* **\_avail** = nullptr  
*pointer to available ring.*
- **Used** \* **\_used** = nullptr  
*pointer to used ring.*
- **l4\_uint16\_t** **\_current\_avail** = 0  
*The life counter for the queue.*
- **l4\_uint16\_t** **\_idx\_mask** = 0  
*mask used for indexing into the descriptor table and the rings.*

## 15.416.1 Detailed Description

Low-level [Virtqueue](#).

This class represents a single virtqueue, with a local running available index.

### Note

The [Virtqueue](#) implementation is not thread-safe.

Definition at line 87 of file [virtqueue](#).

## 15.416.2 Member Function Documentation

### 15.416.2.1 `avail_align()`

```
static unsigned long L4virtio::Virtqueue::avail_align ( ) [inline], [static]
```

Get the alignment in zero LSBs needed for the available ring.

### Returns

The alignment in zero LSBs needed for an available ring.

Definition at line 293 of file [virtqueue](#).

### 15.416.2.2 `avail_size()`

```
static unsigned long L4virtio::Virtqueue::avail_size (
    unsigned num ) [inline], [static]
```

Calculate the size of the available ring for `num` entries.

## Parameters

<i>num</i>	The number of entries in the available ring.
------------	--

## Returns

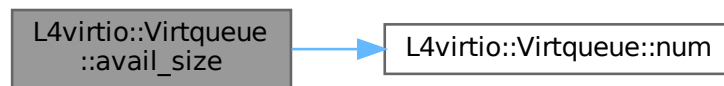
The size in bytes needed for an available ring with *num* entries.

Definition at line 285 of file [virtqueue](#).

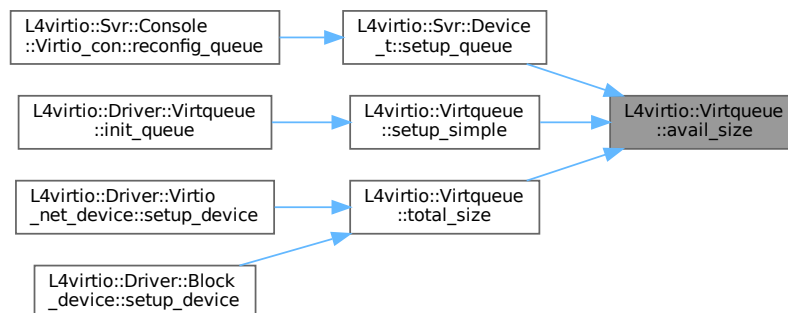
References [num\(\)](#).

Referenced by [L4virtio::Svr::Device\\_t< DATA >::setup\\_queue\(\)](#), [setup\\_simple\(\)](#), and [total\\_size\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.416.2.3 desc\_align()

```
static unsigned long L4virtio::Virtqueue::desc_align ( ) [inline], [static]
```

Get the alignment in zero LSBs needed for the descriptor table.

## Returns

The alignment in zero LSBs needed for a descriptor table.

Definition at line 275 of file [virtqueue](#).

### 15.416.2.4 desc\_size()

```
static unsigned long L4virtio::Virtqueue::desc_size (
    unsigned num ) [inline], [static]
```

Calculate the size of the descriptor table for `num` entries.

#### Parameters

<i>num</i>	The number of entries in the descriptor table.
------------	--

#### Returns

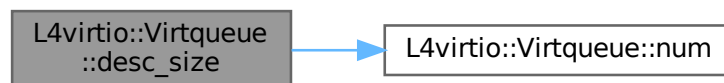
The size in bytes needed for a descriptor table with `num` entries.

Definition at line 267 of file [virtqueue](#).

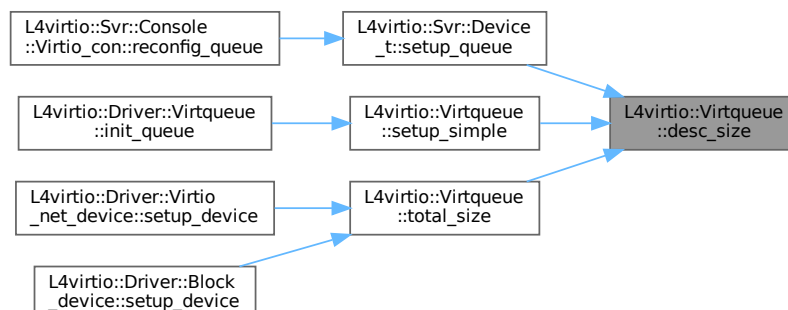
References [num\(\)](#).

Referenced by [L4virtio::Svr::Device\\_t< DATA >::setup\\_queue\(\)](#), [setup\\_simple\(\)](#), and [total\\_size\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.416.2.5 disable()

```
void L4virtio::Virtqueue::disable ( ) [inline]
```

Completely disable the queue.

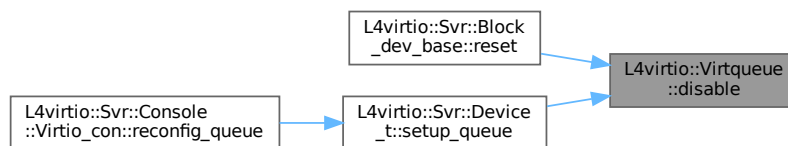
[setup\(\)](#) must be used to enable the queue again.

Definition at line 230 of file [virtqueue](#).

References [\\_desc](#).

Referenced by [L4virtio::Svr::Block\\_dev\\_base< Ds\\_data >::reset\(\)](#), and [L4virtio::Svr::Device\\_t< DATA >::setup\\_queue\(\)](#).

Here is the caller graph for this function:



### 15.416.2.6 dump()

```
void L4virtio::Virtqueue::dump (
    Desc const * d ) const [inline]
```

Dump descriptors for this queue.

#### Precondition

the queue must be in working state.

Definition at line 398 of file [virtqueue](#).

References [\\_desc](#), and [L4virtio::Virtqueue::Desc::dump\(\)](#).

Here is the call graph for this function:





### 15.416.2.7 get\_avail\_idx()

```
l4_uint16_t L4virtio::Virtqueue::get_avail_idx ( ) const [inline]
```

Get available index from available ring (for debugging).

#### Precondition

Queue must be in a working state.

#### Returns

current index in the available ring (shared between device model and device driver).

Definition at line 455 of file [virtqueue](#).

References [\\_avail](#), and [L4virtio::Virtqueue::Avail::idx](#).

### 15.416.2.8 get\_tail\_avail\_idx()

```
l4_uint16_t L4virtio::Virtqueue::get_tail_avail_idx ( ) const [inline]
```

Get tail-available index stored in local state (for debugging).

#### Returns

current tail index for the the available ring.

Definition at line 462 of file [virtqueue](#).

References [\\_current\\_avail](#).

### 15.416.2.9 no\_notify\_guest()

```
bool L4virtio::Virtqueue::no_notify_guest ( ) const [inline]
```

Get the no IRQ flag of this queue.

#### Precondition

queue must be in working state.

**Returns**

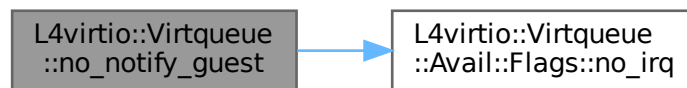
true if the guest does not want to get IRQs (currently).

Definition at line 420 of file [virtqueue](#).

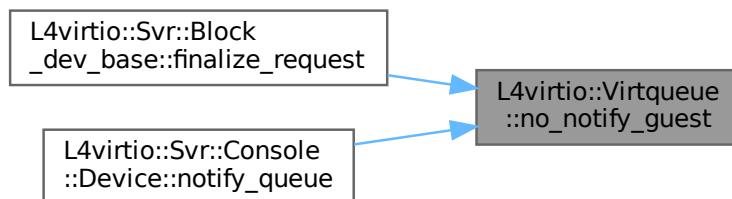
References [\\_avail](#), [L4virtio::Virtqueue::Avail::flags](#), and [L4virtio::Virtqueue::Avail::Flags::no\\_irq\(\)](#).

Referenced by [L4virtio::Svr::Block\\_dev\\_base<Ds\\_data>::finalize\\_request\(\)](#), and [L4virtio::Svr::Console::Device::notify\\_queue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.416.2.10 no\_notify\_host() [1/2]**

```
bool L4virtio::Virtqueue::no_notify_host ( ) const [inline]
```

Get the no notify flag of this queue.

**Precondition**

queue must be in working state.

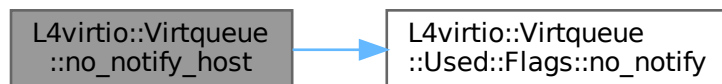
**Returns**

true if the host does not want to get IRQs (currently).

Definition at line 432 of file [virtqueue](#).

References [\\_used](#), [L4virtio::Virtqueue::Used::flags](#), and [L4virtio::Virtqueue::Used::Flags::no\\_notify\(\)](#).

Here is the call graph for this function:

**15.416.2.11 no\_notify\_host() [2/2]**

```
void L4virtio::Virtqueue::no_notify_host (
    bool value ) [inline]
```

Set the no-notify flag for this queue.

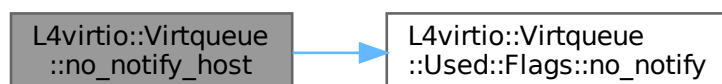
**Precondition**

Queue must be in a working state.

Definition at line 442 of file [virtqueue](#).

References [\\_used](#), [L4virtio::Virtqueue::Used::flags](#), and [L4virtio::Virtqueue::Used::Flags::no\\_notify\(\)](#).

Here is the call graph for this function:





## Returns

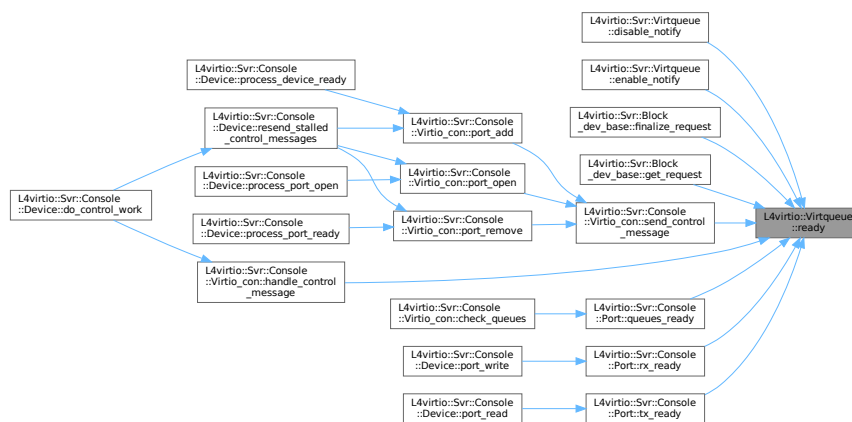
true when the queue is in working state, false else.

Definition at line 406 of file [virtqueue](#).

References [\\_desc](#), and [L4\\_LIKELY](#).

Referenced by [L4virtio::Svr::Virtqueue::disable\\_notify\(\)](#), [L4virtio::Svr::Virtqueue::enable\\_notify\(\)](#), [L4virtio::Svr::Block\\_dev\\_base<Ds\\_data>::get\\_request\(\)](#), [L4virtio::Svr::Console::Virtio\\_con::handle\\_control\\_message\(\)](#), [L4virtio::Svr::Console::Port::queues\\_ready\(\)](#), [L4virtio::Svr::Console::Port::rx\\_ready\(\)](#), [L4virtio::Svr::Console::Virtio\\_con::send\\_control\\_message\(\)](#), and [L4virtio::Svr::Console::Port::tx\\_ready\(\)](#).

Here is the caller graph for this function:



## 15.416.2.14 setup()

```

void L4virtio::Virtqueue::setup (
    unsigned num,
    void * desc,
    void * avail,
    void * used ) [inline]

```

Enable this queue.

## Parameters

<i>num</i>	The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).
<i>desc</i>	The address of the descriptor table. (Must be Desc_align aligned and at least desc_size (num) bytes in size.)
<i>avail</i>	The address of the available ring. (Must be Avail_align aligned and at least avail_size (num) bytes in size.)
<i>used</i>	The address of the used ring. (Must be Used_align aligned and at least used_size (num) bytes in size.)

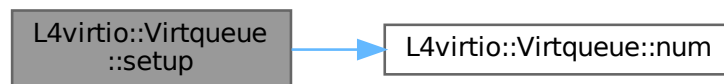
Due to the data type of the descriptors, the queue can have a maximum size of  $2^{16}$ .

Definition at line 355 of file [virtqueue](#).

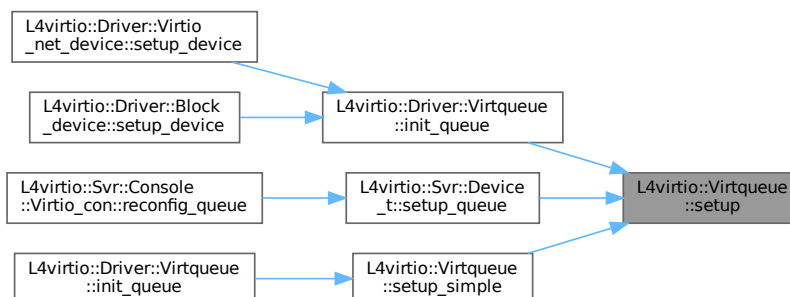
References [\\_avail](#), [\\_current\\_avail](#), [\\_desc](#), [\\_idx\\_mask](#), [\\_used](#), [L4\\_EINVAL](#), and [num\(\)](#).

Referenced by [L4virtio::Driver::Virtqueue::init\\_queue\(\)](#), [L4virtio::Svr::Device\\_t< DATA >::setup\\_queue\(\)](#), and [setup\\_simple\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.416.2.15 setup\_simple()

```
void L4virtio::Virtqueue::setup_simple (
    unsigned num,
    void * ring ) [inline]
```

Enable this queue.

#### Parameters

<i>num</i>	The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).
<i>ring</i>	The base address for the queue data structure. The memory block at <code>ring</code> must be at least <code>total_size(num)</code> bytes in size and have an alignment of <code>Desc_align(desc_align())</code> bits.

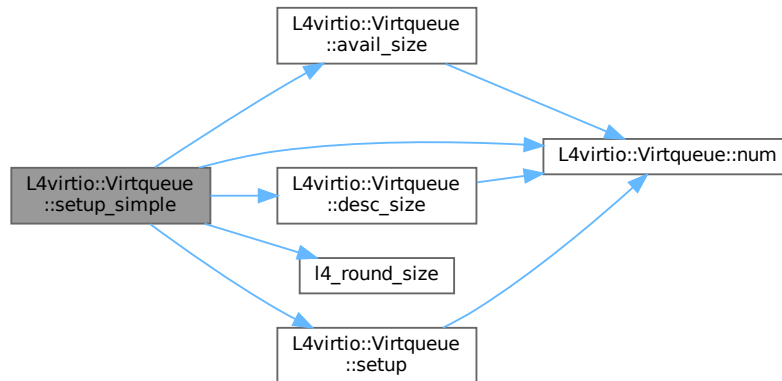
Due to the data type of the descriptors, the queue can have a maximum size of  $2^{16}$ .

Definition at line 384 of file [virtqueue](#).

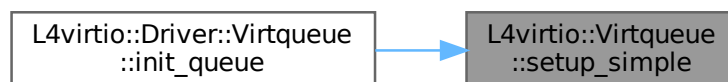
References [avail\\_size\(\)](#), [desc\\_size\(\)](#), [l4\\_round\\_size\(\)](#), [num\(\)](#), and [setup\(\)](#).

Referenced by [L4virtio::Driver::Virtqueue::init\\_queue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 15.416.2.16 total\_size() [1/2]

```
unsigned long L4virtio::Virtqueue::total_size ( ) const [inline]
```

Calculate the total size of this virtqueue.

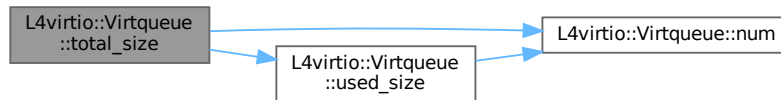
**Precondition**

The queue has been set up.

Definition at line 320 of file [virtqueue](#).

References [\\_desc](#), [\\_used](#), [num\(\)](#), and [used\\_size\(\)](#).

Here is the call graph for this function:

**15.416.2.17 total\_size() [2/2]**

```
static unsigned long L4virtio::Virtqueue::total_size (
    unsigned num ) [inline], [static]
```

Calculate the total size for a virtqueue of the given dimensions.

**Parameters**

<i>num</i>	The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).
------------	--

**Returns**

The total size in bytes of the queue data structures.

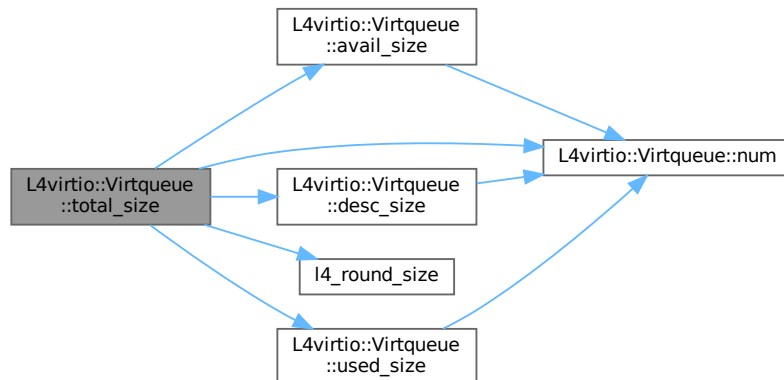
Definition at line 251 of file [virtqueue](#).

References [avail\\_size\(\)](#), [desc\\_size\(\)](#), [l4\\_round\\_size\(\)](#), [num\(\)](#), and [used\\_size\(\)](#).

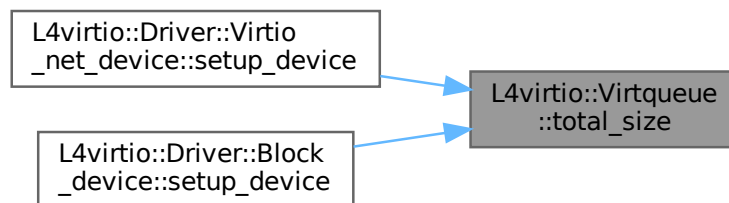
Referenced by [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#), and [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#).



Here is the call graph for this function:



Here is the caller graph for this function:



### 15.416.2.18 used\_align()

```
static unsigned long L4virtio::Virtqueue::used_align ( ) [inline], [static]
```

Get the alignment in zero LSBs needed for the used ring.

#### Returns

The alignment in zero LSBs needed for an used ring.

Definition at line 312 of file [virtqueue](#).

### 15.416.2.19 used\_size()

```
static unsigned long L4virtio::Virtqueue::used_size (
    unsigned num ) [inline], [static]
```

Calculate the size of the used ring for `num` entries.

## Parameters

<i>num</i>	The number of entries in the used ring.
------------	---

## Returns

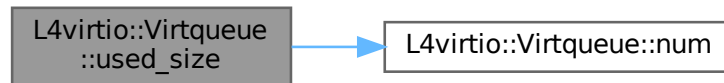
The size in bytes needed for an used ring with *num* entries.

Definition at line 304 of file [virtqueue](#).

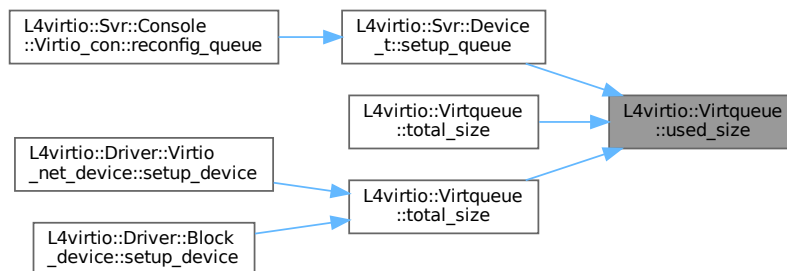
References [num\(\)](#).

Referenced by [L4virtio::Svr::Device\\_t< DATA >::setup\\_queue\(\)](#), [total\\_size\(\)](#), and [total\\_size\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- `l4/l4virtio/virtqueue`

## 15.417 L4virtio::Virtqueue::Avail Class Reference

Type of available ring, this is read-only for the host.

```
#include <virtqueue>
```

Collaboration diagram for L4virtio::Virtqueue::Avail:



### Data Structures

- struct [Flags](#)  
*Flags of the available ring.*

### Data Fields

- [Flags flags](#)  
*flags of available ring*
- [l4\\_uint16\\_t idx](#)  
*available index written by guest*
- [l4\\_uint16\\_t ring \[\]](#)  
*array of available descriptor indexes.*

### 15.417.1 Detailed Description

Type of available ring, this is read-only for the host.

Definition at line 135 of file [virtqueue](#).

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

## 15.418 L4virtio::Virtqueue::Avail::Flags Struct Reference

[Flags](#) of the available ring.

```
#include <virtqueue>
```

Collaboration diagram for L4virtio::Virtqueue::Avail::Flags:

L4virtio::Virtqueue ::Avail::Flags
+ raw
+ Flags()
* no_irq_bfm_t
* no_irq()
* no_irq()

### Public Member Functions

- **Flags** ([l4\\_uint16\\_t](#) v)  
*Make [Flags](#) from the raw value.*

### Data Fields

- [l4\\_uint16\\_t](#) raw  
*raw 16bit flags value of the available ring.*

### 15.418.1 Detailed Description

[Flags](#) of the available ring.

Definition at line 141 of file [virtqueue](#).

## 15.418.2 Member Typedef Documentation

### 15.418.2.1 no\_irq\_bfm\_t

```
typedef cxx::Bitfield<decltype( raw ), 0 , 0 > L4virtio::Virtqueue::Avail::Flags::no\_irq\_bfm\_t
```

Guest does not want to receive interrupts when requests are finished.

Type to access the `no_irq` bits ( 0 to 0 ) of `raw`.

Definition at line [150](#) of file [virtqueue](#).

The documentation for this struct was generated from the following file:

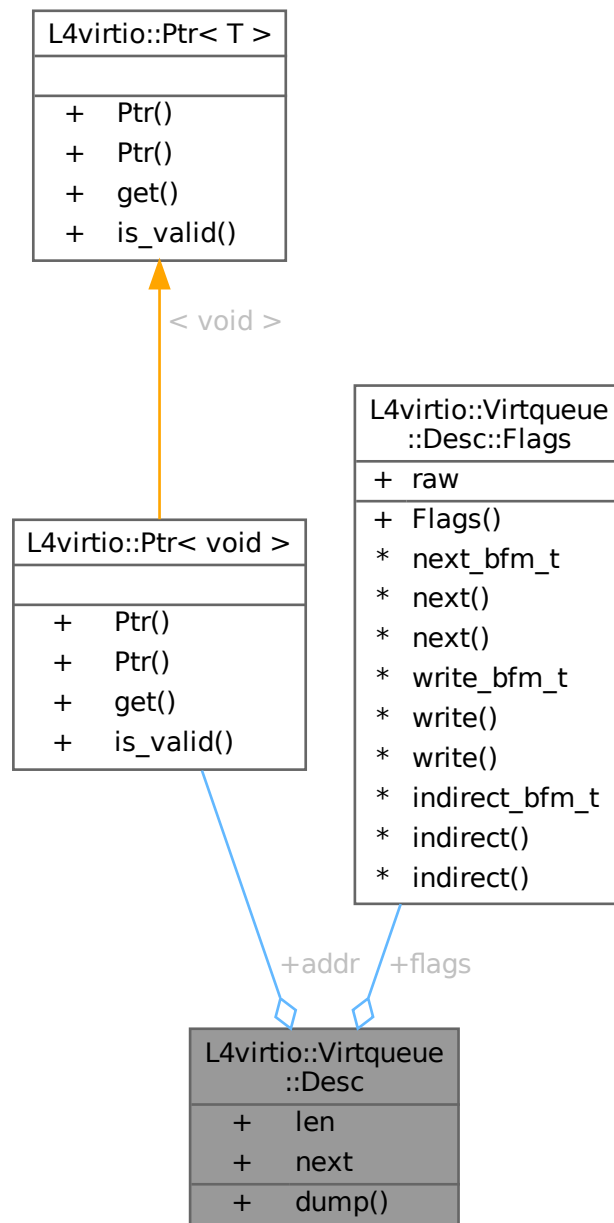
- [l4/l4virtio/virtqueue](#)

## 15.419 L4virtio::Virtqueue::Desc Class Reference

Descriptor in the descriptor table.

```
#include <virtqueue>
```

Collaboration diagram for L4virtio::Virtqueue::Desc:



## Data Structures

- struct [Flags](#)  
*Type for descriptor flags.*

## Public Member Functions

- void **dump** (unsigned idx) const  
*Dump a single descriptor.*

## Data Fields

- [Ptr](#)< void > **addr**  
*Address stored in descriptor.*
- [l4\\_uint32\\_t](#) **len**  
*Length of described buffer.*
- [Flags](#) **flags**  
*Descriptor flags.*
- [l4\\_uint16\\_t](#) **next**  
*Index of the next chained descriptor.*

### 15.419.1 Detailed Description

Descriptor in the descriptor table.

Definition at line 93 of file [virtqueue](#).

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

## 15.420 L4virtio::Virtqueue::Desc::Flags Struct Reference

Type for descriptor flags.

```
#include <virtqueue>
```

Collaboration diagram for L4virtio::Virtqueue::Desc::Flags:

L4virtio::Virtqueue ::Desc::Flags
+ raw
+ Flags()
* next_bfm_t
* next()
* next()
* write_bfm_t
* write()
* write()
* indirect_bfm_t
* indirect()
* indirect()

## Public Member Functions

- **Flags** ([l4\\_uint16\\_t](#) v)  
*Make [Flags](#) from raw 16bit value.*

## Data Fields

- [l4\\_uint16\\_t](#) raw  
*raw flags value of a virtio descriptor.*

### 15.420.1 Detailed Description

Type for descriptor flags.

Definition at line 99 of file [virtqueue](#).

### 15.420.2 Member Typedef Documentation

#### 15.420.2.1 indirect\_bfm\_t

```
typedef cxx::Bitfield<decltype( raw ), 2 , 2 > L4virtio::Virtqueue::Desc::Flags::indirect\_bfm\_t
```

Indirect descriptor, block contains a list of descriptors.

Type to access the `indirect` bits ( 2 to 2 ) of `raw`.

Definition at line 112 of file [virtqueue](#).

#### 15.420.2.2 next\_bfm\_t

```
typedef cxx::Bitfield<decltype( raw ), 0 , 0 > L4virtio::Virtqueue::Desc::Flags::next\_bfm\_t
```

Part of a descriptor chain which is continued with the next field.

Type to access the `next` bits ( 0 to 0 ) of `raw`.

Definition at line 108 of file [virtqueue](#).

#### 15.420.2.3 write\_bfm\_t

```
typedef cxx::Bitfield<decltype( raw ), 1 , 1 > L4virtio::Virtqueue::Desc::Flags::write\_bfm\_t
```

Block described by this descriptor is writeable.

Type to access the `write` bits ( 1 to 1 ) of `raw`.

Definition at line 110 of file [virtqueue](#).

The documentation for this struct was generated from the following file:

- `l4/l4virtio/virtqueue`

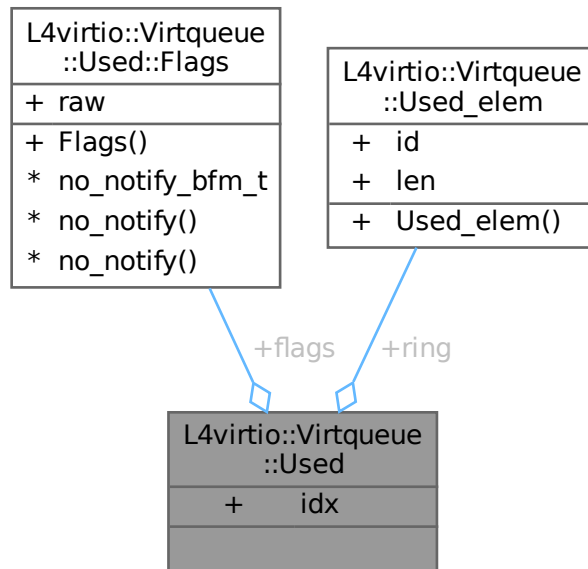


## 15.421 L4virtio::Virtqueue::Used Class Reference

Used ring.

```
#include <virtqueue>
```

Collaboration diagram for L4virtio::Virtqueue::Used:



### Data Structures

- struct [Flags](#)  
*flags for the used ring.*

### Data Fields

- [Flags](#) **flags**  
*flags of the used ring.*
- [l4\\_uint16\\_t](#) **idx**  
*index of the last entry in the ring.*
- [Used\\_elem](#) **ring** []  
*array of used descriptors.*

### 15.421.1 Detailed Description

Used ring.

Definition at line 180 of file [virtqueue](#).

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

## 15.422 L4virtio::Virtqueue::Used::Flags Struct Reference

flags for the used ring.

```
#include <virtqueue>
```

Collaboration diagram for L4virtio::Virtqueue::Used::Flags:

L4virtio::Virtqueue ::Used::Flags
+ raw
+ Flags()
* no_notify_bfm_t
* no_notify()
* no_notify()

### Public Member Functions

- **Flags** ([l4\\_uint16\\_t](#) v)  
*make [Flags](#) from raw value*

### Data Fields

- [l4\\_uint16\\_t](#) raw  
*raw flags value as specified by virtio.*

### 15.422.1 Detailed Description

flags for the used ring.

Definition at line 186 of file [virtqueue](#).

### 15.422.2 Member Typedef Documentation

#### 15.422.2.1 no\_notify\_bfm\_t

```
typedef cxx::Bitfield<decltype( raw ), 0 , 0 > L4virtio::Virtqueue::Used::Flags::no\_notify\_bfm\_t
```

host does not want to be notified when new requests have been queued.

Type to access the no\_notify bits ( 0 to 0 ) of raw.

Definition at line 195 of file [virtqueue](#).

The documentation for this struct was generated from the following file:

- [l4/l4virtio/virtqueue](#)

## 15.423 L4virtio::Virtqueue::Used\_elem Struct Reference

Type of an element of the used ring.

```
#include <virtqueue>
```

Collaboration diagram for L4virtio::Virtqueue::Used\_elem:

L4virtio::Virtqueue ::Used_elem
+ id
+ len
+ Used_elem()

### Public Member Functions

- [Used\\_elem](#) ([l4\\_uint16\\_t](#) id, [l4\\_uint32\\_t](#) len)  
*Initialize a used ring element.*

### Data Fields

- [l4\\_uint32\\_t](#) id  
*descriptor index*
- [l4\\_uint32\\_t](#) len  
*length field*

### 15.423.1 Detailed Description

Type of an element of the used ring.

Definition at line 161 of file [virtqueue](#).

### 15.423.2 Constructor & Destructor Documentation

#### 15.423.2.1 Used\_elem()

```
L4virtio::Virtqueue::Used_elem::Used_elem (
    l4\_uint16\_t id,
    l4\_uint32\_t len ) [inline]
```

Initialize a used ring element.

## Parameters

<i>id</i>	The index of the descriptor to be marked as used.
<i>len</i>	The total bytes written into the buffer of the descriptor chain.

Definition at line 172 of file [virtqueue](#).

The documentation for this struct was generated from the following file:

- I4/I4virtio/virtqueue

## 15.424 I4virtio\_block\_config\_t Struct Reference

Device configuration for block devices.

```
#include <virtio_block.h>
```

Collaboration diagram for I4virtio\_block\_config\_t:

I4virtio_block_config_t	
+	capacity
+	size_max
+	seg_max
+	blk_size

## Data Fields

- [I4\\_uint64\\_t](#) **capacity**  
*Capacity of device in 512-byte sectors.*
- [I4\\_uint32\\_t](#) **size\_max**  
*Maximum size of a single segment.*
- [I4\\_uint32\\_t](#) **seg\_max**  
*Maximum number of segments per request.*
- [I4\\_uint32\\_t](#) **blk\_size**  
*Block size of underlying disk.*

### 15.424.1 Detailed Description

Device configuration for block devices.

Definition at line 68 of file [virtio\\_block.h](#).

The documentation for this struct was generated from the following file:

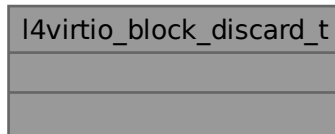
- l4/l4virtio/virtio\_block.h

## 15.425 l4virtio\_block\_discard\_t Struct Reference

Structure used for the write zeroes and discard commands.

```
#include <virtio_block.h>
```

Collaboration diagram for l4virtio\_block\_discard\_t:



### 15.425.1 Detailed Description

Structure used for the write zeroes and discard commands.

Definition at line 58 of file [virtio\\_block.h](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio\_block.h

## 15.426 l4virtio\_block\_header\_t Struct Reference

Header structure of a request for a block device.

```
#include <virtio_block.h>
```

Collaboration diagram for l4virtio\_block\_header\_t:

l4virtio_block_header_t	
+	type
+	ioprio
+	sector

### Data Fields

- [l4\\_uint32\\_t](#) **type**  
*Kind of request, see L4virtio\_block\_operations.*
- [l4\\_uint32\\_t](#) **ioprio**  
*Priority (unused)*
- [l4\\_uint64\\_t](#) **sector**  
*First sector to read/write.*

### 15.426.1 Detailed Description

Header structure of a request for a block device.

Definition at line 42 of file [virtio\\_block.h](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio\_block.h

## 15.427 l4virtio\_config\_hdr\_t Struct Reference

L4-VIRTIO config header, provided in shared data space.

```
#include <virtio.h>
```

Inherited by L4virtio::Device::Config\_hdr.

Collaboration diagram for l4virtio\_config\_hdr\_t:

l4virtio_config_hdr_t
+ magic
+ version
+ device
+ vendor
+ dev_features
+ num_queues
+ queues_offset

### Data Fields

- [l4\\_uint32\\_t](#) **magic**  
*magic value (must be 'virt').*
- [l4\\_uint32\\_t](#) **version**  
*VIRTIO version.*
- [l4\\_uint32\\_t](#) **device**  
*device ID*
- [l4\\_uint32\\_t](#) **vendor**  
*vendor ID*
- [l4\\_uint32\\_t](#) **dev\_features**  
*device features windows selected by device\_feature\_sel*
- [l4\\_uint32\\_t](#) **num\_queues**  
*number of virtqueues*
- [l4\\_uint32\\_t](#) **queues\_offset**  
*offset of virtqueue config array*

### 15.427.1 Detailed Description

L4-VIRTIO config header, provided in shared data space.

Definition at line 129 of file [virtio.h](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio.h

## 15.428 l4virtio\_config\_queue\_t Struct Reference

Queue configuration entry.

```
#include <virtio.h>
```

Collaboration diagram for l4virtio\_config\_queue\_t:

l4virtio_config_queue_t
+ num_max
+ num
+ ready
+ driver_notify_index
+ desc_addr
+ avail_addr
+ used_addr
+ device_notify_index

### Data Fields

- [l4\\_uint16\\_t](#) **num\_max**  
*R: maximum number of descriptors supported by this queue.*
- [l4\\_uint16\\_t](#) **num**  
*RW: number of descriptors configured for this queue.*
- [l4\\_uint16\\_t](#) **ready**  
*RW: queue ready flag (read-write)*
- [l4\\_uint16\\_t](#) **driver\_notify\_index**  
*W: Event index to be used for device notifications (device to driver)*
- [l4\\_uint64\\_t](#) **desc\_addr**  
*W: address of descriptor table.*
- [l4\\_uint64\\_t](#) **avail\_addr**  
*W: address of available ring.*
- [l4\\_uint64\\_t](#) **used\_addr**  
*W: address of used ring.*
- [l4\\_uint16\\_t](#) **device\_notify\_index**  
*R: Event index to be used by the driver (driver to device)*



### 15.428.1 Detailed Description

Queue configuration entry.

An array of such entries is available at the [l4virtio\\_config\\_hdr\\_t::queues\\_offset](#) in the config data space.

Consistency rules for the queue config are:

- A driver might read `num_max` at any time.
- A driver must write to `num`, `desc_addr`, `avail_addr`, and `used_addr` only when `ready` is zero (0). Values in these fields are validated and used by the device only after successfully setting `ready` to one (1), either by the IPC or by `L4VIRTIO_CMD_CFG_QUEUE`.
- The value of `device_notify_index` is valid only when `ready` is one.
- The driver might write to `device_notify_index` at any time, however the change is guaranteed to take effect after a successful `L4VIRTIO_CMD_CFG_QUEUE` or after a `config_queue` IPC. Note, the change might also have immediate effect, depending on the device implementation.

Definition at line 220 of file [virtio.h](#).

The documentation for this struct was generated from the following file:

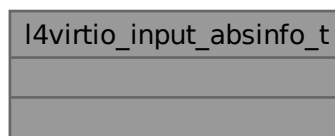
- `l4/l4virtio/virtio.h`

## 15.429 l4virtio\_input\_absinfo\_t Struct Reference

Information about the absolute axis in the underlying evdev implementation.

```
#include <virtio_input.h>
```

Collaboration diagram for `l4virtio_input_absinfo_t`:



### 15.429.1 Detailed Description

Information about the absolute axis in the underlying evdev implementation.

Definition at line 33 of file [virtio\\_input.h](#).

The documentation for this struct was generated from the following file:

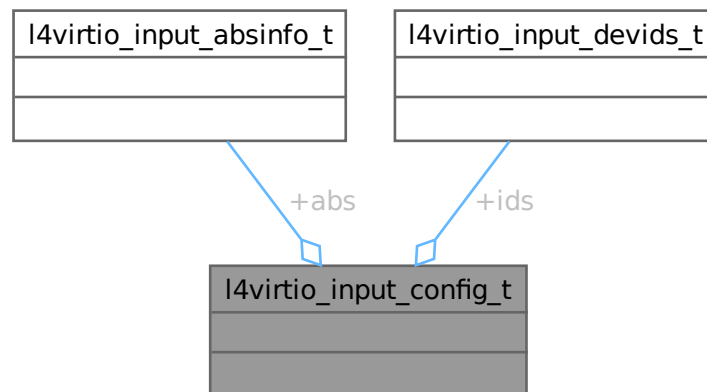
- `l4/l4virtio/virtio_input.h`

## 15.430 l4virtio\_input\_config\_t Struct Reference

Device configuration for input devices.

```
#include <virtio_input.h>
```

Collaboration diagram for l4virtio\_input\_config\_t:



### 15.430.1 Detailed Description

Device configuration for input devices.

Definition at line 56 of file [virtio\\_input.h](#).

The documentation for this struct was generated from the following file:

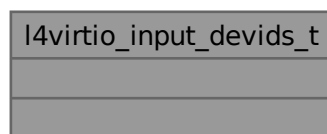
- l4/l4virtio/virtio\_input.h

## 15.431 l4virtio\_input\_devids\_t Struct Reference

Device ID information for the device.

```
#include <virtio_input.h>
```

Collaboration diagram for l4virtio\_input\_devids\_t:



### 15.431.1 Detailed Description

Device ID information for the device.

Definition at line 45 of file [virtio\\_input.h](#).

The documentation for this struct was generated from the following file:

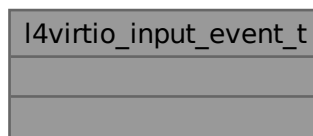
- l4/l4virtio/virtio\_input.h

## 15.432 l4virtio\_input\_event\_t Struct Reference

Single event in event or status queue.

```
#include <virtio_input.h>
```

Collaboration diagram for l4virtio\_input\_event\_t:



### 15.432.1 Detailed Description

Single event in event or status queue.

Definition at line 74 of file [virtio\\_input.h](#).

The documentation for this struct was generated from the following file:

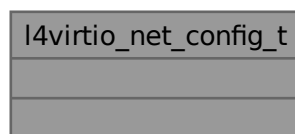
- l4/l4virtio/virtio\_input.h

## 15.433 l4virtio\_net\_config\_t Struct Reference

Device configuration for network devices.

```
#include <virtio_net.h>
```

Collaboration diagram for l4virtio\_net\_config\_t:



### 15.433.1 Detailed Description

Device configuration for network devices.

Definition at line 34 of file [virtio\\_net.h](#).

The documentation for this struct was generated from the following file:

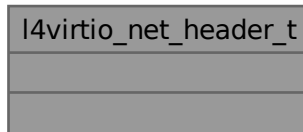
- l4/l4virtio/virtio\_net.h

## 15.434 l4virtio\_net\_header\_t Struct Reference

Header structure of a request for a network device.

```
#include <virtio_net.h>
```

Collaboration diagram for l4virtio\_net\_header\_t:



### 15.434.1 Detailed Description

Header structure of a request for a network device.

Definition at line 20 of file [virtio\\_net.h](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio\_net.h

# Chapter 16

## File Documentation

### 16.1 asm\_access.h

```
00001 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2021 Kernkonzept GmbH.
00004  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00005  */
00006
00007 #pragma once
00008
00009 #include <l4/sys/l4int.h>
00010 #include <x86/l4/drivers/asm_access.h>
00011
00012 namespace Asm_access {
00013
00014     inline
00015     l4_uint64_t
00016     read(l4_uint64_t const *mem)
00017     {
00018         l4_uint64_t val;
00019
00020         asm volatile ("movq %[mem], %[val]" : [val] "=r" (val) : [mem] "m" (*mem));
00021
00022         return val;
00023     }
00024
00025     inline
00026     void
00027     write(l4_uint64_t val, l4_uint64_t *mem)
00028     {
00029         asm volatile ("movq %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00030     }
00031 }
00032 }
```

### 16.2 asm\_access.h

```
00001 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2021 Kernkonzept GmbH.
00004  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00005  */
00006
00007 #pragma once
00008
00009 #include <l4/sys/l4int.h>
00010
00011 namespace Asm_access {
00012
00013     inline
00014     l4_uint8_t
00015     read(l4_uint8_t const *mem)
00016     {
00017         l4_uint8_t val;
00018
00019         asm volatile ("ldrb %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
```

```

00020
00021     return val;
00022 }
00023
00024 inline
00025 l4_uint16_t
00026 read(l4_uint16_t const *mem)
00027 {
00028     l4_uint16_t val;
00029
00030     asm volatile ("ldrh %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00031
00032     return val;
00033 }
00034
00035 inline
00036 l4_uint32_t
00037 read(l4_uint32_t const *mem)
00038 {
00039     l4_uint32_t val;
00040
00041     asm volatile ("ldr %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00042
00043     return val;
00044 }
00045
00046 inline
00047 void
00048 write(l4_uint8_t val, l4_uint8_t *mem)
00049 {
00050     asm volatile ("strb %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00051 }
00052
00053 inline
00054 void
00055 write(l4_uint16_t val, l4_uint16_t *mem)
00056 {
00057     asm volatile ("strh %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00058 }
00059
00060 inline
00061 void
00062 write(l4_uint32_t val, l4_uint32_t *mem)
00063 {
00064     asm volatile ("str %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00065 }
00066
00067 }

```

## 16.3 asm\_access.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2021 Kernkonzept GmbH.
00004  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00005  */
00006
00007 #pragma once
00008
00009 #include <l4/sys/l4int.h>
00010
00011 namespace Asm_access {
00012
00013     inline
00014     l4_uint8_t
00015     read(l4_uint8_t const *mem)
00016     {
00017         l4_uint8_t val;
00018
00019         asm volatile ("ldrb %w[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00020
00021         return val;
00022     }
00023
00024     inline
00025     l4_uint16_t
00026     read(l4_uint16_t const *mem)
00027     {
00028         l4_uint16_t val;
00029
00030         asm volatile ("ldrh %w[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00031
00032         return val;

```

```

00033 }
00034
00035 inline
00036 l4_uint32_t
00037 read(l4_uint32_t const *mem)
00038 {
00039     l4_uint32_t val;
00040
00041     asm volatile ("ldr %w[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00042
00043     return val;
00044 }
00045
00046 inline
00047 l4_uint64_t
00048 read(l4_uint64_t const *mem)
00049 {
00050     l4_uint64_t val;
00051
00052     asm volatile ("ldr %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00053
00054     return val;
00055 }
00056
00057 inline
00058 void
00059 write(l4_uint8_t val, l4_uint8_t *mem)
00060 {
00061     asm volatile ("strb %w[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00062 }
00063
00064 inline
00065 void
00066 write(l4_uint16_t val, l4_uint16_t *mem)
00067 {
00068     asm volatile ("strh %w[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00069 }
00070
00071 inline
00072 void
00073 write(l4_uint32_t val, l4_uint32_t *mem)
00074 {
00075     asm volatile ("str %w[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00076 }
00077
00078 inline
00079 void
00080 write(l4_uint64_t val, l4_uint64_t *mem)
00081 {
00082     asm volatile ("str %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00083 }
00084
00085 }

```

## 16.4 asm\_access.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2021 Kernkonzept GmbH.
00004  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00005  */
00006
00007 #pragma once
00008
00009 #include <l4/drivers/asm_access_gen.h>

```

## 16.5 asm\_access.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2021 Kernkonzept GmbH.
00004  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00005  */
00006
00007 #pragma once
00008
00009 #include <l4/drivers/asm_access_gen.h>

```

## 16.6 asm\_access.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2021 Kernkonzept GmbH.
00004  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00005  */
00006
00007 #pragma once
00008
00009 #include <l4/sys/l4int.h>
00010
00011 namespace Asm_access {
00012
00013 inline
00014 l4_uint8_t
00015 read(l4_uint8_t const *mem)
00016 {
00017     l4_uint8_t val;
00018
00019     asm volatile ("lb %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00020
00021     return val;
00022 }
00023
00024 inline
00025 l4_uint16_t
00026 read(l4_uint16_t const *mem)
00027 {
00028     l4_uint16_t val;
00029
00030     asm volatile ("lh %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00031
00032     return val;
00033 }
00034
00035 inline
00036 l4_uint32_t
00037 read(l4_uint32_t const *mem)
00038 {
00039     l4_uint32_t val;
00040
00041     asm volatile ("lw %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00042
00043     return val;
00044 }
00045
00046 inline
00047 void
00048 write(l4_uint8_t val, l4_uint8_t *mem)
00049 {
00050     asm volatile ("sb %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00051 }
00052
00053 inline
00054 void
00055 write(l4_uint16_t val, l4_uint16_t *mem)
00056 {
00057     asm volatile ("sh %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00058 }
00059
00060 inline
00061 void
00062 write(l4_uint32_t val, l4_uint32_t *mem)
00063 {
00064     asm volatile ("sw %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00065 }
00066
00067 #if __riscv_xlen == 64
00068
00069 inline
00070 l4_uint64_t
00071 read(l4_uint64_t const *mem)
00072 {
00073     l4_uint64_t val;
00074
00075     asm volatile ("ld %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00076
00077     return val;
00078 }
00079
00080 inline
00081 void
00082 write(l4_uint64_t val, l4_uint64_t *mem)
00083 {
00084     asm volatile ("sd %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00085 }

```



```

00086
00087 #endif
00088
00089 }

```

## 16.7 asm\_access.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2021 Kernkonzept GmbH.
00004  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00005  */
00006
00007 #pragma once
00008
00009 #include <l4/drivers/asm_access_gen.h>

```

## 16.8 asm\_access.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2021 Kernkonzept GmbH.
00004  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00005  */
00006
00007 #pragma once
00008
00009 #include <l4/sys/l4int.h>
00010
00011 namespace Asm_access {
00012
00013 inline
00014 l4_uint8_t
00015 read(l4_uint8_t const *mem)
00016 {
00017     l4_uint8_t val;
00018
00019     asm volatile ("movb %[mem], %[val]" : [val] "=q" (val) : [mem] "m" (*mem));
00020
00021     return val;
00022 }
00023
00024 inline
00025 l4_uint16_t
00026 read(l4_uint16_t const *mem)
00027 {
00028     l4_uint16_t val;
00029
00030     asm volatile ("movw %[mem], %[val]" : [val] "=r" (val) : [mem] "m" (*mem));
00031
00032     return val;
00033 }
00034
00035 inline
00036 l4_uint32_t
00037 read(l4_uint32_t const *mem)
00038 {
00039     l4_uint32_t val;
00040
00041     asm volatile ("movl %[mem], %[val]" : [val] "=r" (val) : [mem] "m" (*mem));
00042
00043     return val;
00044 }
00045
00046 inline
00047 void
00048 write(l4_uint8_t val, l4_uint8_t *mem)
00049 {
00050     asm volatile ("movb %[val], %[mem]" : [mem] "=m" (*mem) : [val] "qi" (val));
00051 }
00052
00053 inline
00054 void
00055 write(l4_uint16_t val, l4_uint16_t *mem)
00056 {
00057     asm volatile ("movw %[val], %[mem]" : [mem] "=m" (*mem) : [val] "ri" (val));
00058 }
00059
00060 inline

```

```

00061 void
00062 write(l4_uint32_t val, l4_uint32_t *mem)
00063 {
00064     asm volatile ("movl %[val], %[mem]" : [mem] "=m" (*mem) : [val] "ri" (val));
00065 }
00066
00067 }

```

## 16.9 asm\_access\_gen.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2021 Kernkonzept GmbH.
00004  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00005  */
00006
00007 #pragma once
00008
00009 #include <l4/sys/l4int.h>
00010 #include <l4/cxx/type_traits>
00011
00012 namespace Asm_access {
00013
00014 template <typename T>
00015 struct is_supported_type
00016 {
00017     static const bool value = cxx::is_same<T, l4_uint8_t>::value
00018                             || cxx::is_same<T, l4_uint16_t>::value
00019                             || cxx::is_same<T, l4_uint32_t>::value
00020                             || cxx::is_same<T, l4_uint64_t>::value;
00021 };
00022
00023 template <typename T>
00024 inline
00025 typename cxx::enable_if<is_supported_type<T>::value, T>::type
00026 read(T const *mem)
00027 {
00028     return *reinterpret_cast<volatile T const *>(mem);
00029 }
00030
00031 template <typename T>
00032 inline
00033 typename cxx::enable_if<is_supported_type<T>::value, void>::type
00034 write(T val, T *mem)
00035 {
00036     *reinterpret_cast<volatile T *>(mem) = val;
00037 }
00038
00039 }

```

## 16.10 hw\_mmio\_register\_block

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00003 /*
00004  * Copyright (C) 2014-2021 Kernkonzept GmbH.
00005  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00006  *           Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00007  */
00008 #pragma once
00009
00010 #include <l4/drivers/hw_register_block>
00011 #include <l4/drivers/asm_access.h>
00012
00013 namespace L4drivers {
00014
00015 class Mmio_register_block_base
00016 {
00017 protected:
00018     l4_addr_t _base;
00019     l4_addr_t _shift;
00020
00021 public:
00022     explicit Mmio_register_block_base(l4_addr_t base = 0, l4_addr_t shift = 0)
00023         : _base(base), _shift(shift) {}
00024
00025     template< typename T >
00026     T read(l4_addr_t reg) const
00027     { return Asm_access::read(reinterpret_cast<T const *>(_base + (reg « _shift))); }

```

```

00028
00029     template< typename T >
00030     void write(T value, l4_addr_t reg) const
00031     { Asm_access::write(value, reinterpret_cast<T *>(_base + (reg << _shift))); }
00032
00033     void set_base(l4_addr_t base) { _base = base; }
00034     void set_shift(l4_addr_t shift) { _shift = shift; }
00035 };
00036
00037 /**
00038  * An MMIO block with up to 64-bit register access (32-bit default) and little
00039  * endian byte order.
00040  */
00041     template< unsigned MAX_BITS = 32 >
00042     struct Mmio_register_block
00043     : Register_block_impl<Mmio_register_block<MAX_BITS>, MAX_BITS>,
00044       Mmio_register_block_base
00045     {
00046         explicit Mmio_register_block(l4_addr_t base = 0, l4_addr_t shift = 0)
00047         : Mmio_register_block_base(base, shift) {}
00048     };
00049
00050 }

```

## 16.11 hw\_register\_block

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00003 /*
00004  * (c) 2014-2021 Alexander Warg <alexander.warg@kernkonzept.com>
00005  */
00006 #pragma once
00007
00008 #include <l4/sys/types.h>
00009 #include <l4/cxx/type_traits>
00010
00011 namespace L4drivers {
00012
00013
00014 /**
00015  * \class Register_block
00016  * \details Example usage:
00017
00018  \code{.cpp}
00019
00020 void test()
00021 {
00022     // create a register block reference for max. 16bit accesses, using a
00023     // MMIO register block implementation (at address 0x1000).
00024     Hw::Register_block<16> regs = new Hw::Mmio_register_block<16>(0x1000);
00025
00026     // Alternatively it is allowed to use an implementation that allows
00027     // wider access than actually needed.
00028     Hw::Register_block<16> regs = new Hw::Mmio_register_block<32>(0x1000);
00029
00030     // read a 16bit register at offset 8byte
00031     unsigned short x = regs.r<16>(8);
00032     unsigned short x1 = regs[8];      // alternative
00033
00034     // read an 8bit register at offset 0byte
00035     unsigned v = regs.r<8>(0);
00036
00037     // do a 16bit write to register at offset 2byte (four variants)
00038     regs[2] = 22;
00039     regs.r<16>(2) = 22;
00040     regs[2].write(22);
00041     regs.r<16>().write(22);
00042
00043     // do an 8bit write (two variants)
00044     regs.r<8>(0) = 9;
00045     regs.r<8>(0).write(9);
00046
00047     // do 16bit read-modify-write (two variants)
00048     regs[4].modify(0xf, 3); // clear 4 lowest bits and set them to 3
00049     regs.r<16>(4).modify(0xf, 3);
00050
00051     // do 8bit read-modify-write
00052     regs.r<8>(0).modify(0xf, 3);
00053
00054     // fails to compile, because of too wide access
00055     // (32 bit access but regs is Hw::Register_block<16>)
00056     unsigned long v = regs.r<32>(4)
00057 }

```

```

00058
00059 \endcode
00060 */
00061
00062
00063 /**
00064  * \brief Abstract register block interface
00065  * \tparam MAX_BITS The maximum access width for the registers.
00066  *
00067  * This interfaces is based on virtual do_read_<xx> and do_write_<xx>
00068  * methods that have to be implemented up to the maximum access width.
00069  */
00070 template< unsigned MAX_BITS = 32 >
00071 struct Register_block_base;
00072
00073 template<>
00074 struct Register_block_base<8>
00075 {
00076     virtual l4_uint8_t do_read_8(l4_addr_t reg) const = 0;
00077     virtual void do_write_8(l4_uint8_t value, l4_addr_t reg) = 0;
00078     virtual ~Register_block_base() = 0;
00079 };
00080
00081 inline Register_block_base<8>::~Register_block_base() {}
00082
00083 template<>
00084 struct Register_block_base<16> : Register_block_base<8>
00085 {
00086     virtual l4_uint16_t do_read_16(l4_addr_t reg) const = 0;
00087     virtual void do_write_16(l4_uint16_t value, l4_addr_t reg) = 0;
00088 };
00089
00090 template<>
00091 struct Register_block_base<32> : Register_block_base<16>
00092 {
00093     virtual l4_uint32_t do_read_32(l4_addr_t reg) const = 0;
00094     virtual void do_write_32(l4_uint32_t value, l4_addr_t reg) = 0;
00095 };
00096
00097 template<>
00098 struct Register_block_base<64> : Register_block_base<32>
00099 {
00100     virtual l4_uint64_t do_read_64(l4_addr_t reg) const = 0;
00101     virtual void do_write_64(l4_uint64_t value, l4_addr_t reg) = 0;
00102 };
00103 #undef REGBLK_READ_TEMPLATE
00104 #undef REGBLK_WRITE_TEMPLATE
00105
00106 template<typename CHIL>
00107 struct Register_block_modify_mixin
00108 {
00109     template< typename T >
00110     T modify(T clear_bits, T set_bits, l4_addr_t reg) const
00111     {
00112         CHIL const *c = static_cast<CHIL const *>(this);
00113         T r = (c->template read<T>(reg) & ~clear_bits) | set_bits;
00114         c->template write<T>(r, reg);
00115         return r;
00116     }
00117
00118     template< typename T >
00119     T set(T set_bits, l4_addr_t reg) const
00120     { return this->template modify<T>(T(0), set_bits, reg); }
00121
00122     template< typename T >
00123     T clear(T clear_bits, l4_addr_t reg) const
00124     { return this->template modify<T>(clear_bits, T(0), reg); }
00125 };
00126
00127
00128 #define REGBLK_READ_TEMPLATE(sz) \
00129     template< typename T > \
00130     typename cxx::enable_if<sizeof(T) == (sz / 8), T>::type read(l4_addr_t reg) const \
00131     { \
00132         union X { T t; l4_uint##sz##_t v; } m; \
00133         m.v = _b->do_read_##sz (reg); \
00134         return m.t; \
00135     }
00136
00137 #define REGBLK_WRITE_TEMPLATE(sz) \
00138     template< typename T > \
00139     void write(T value, l4_addr_t reg, typename cxx::enable_if<sizeof(T) == (sz / 8), T>::type = T()) \
00140     const \
00141     { \
00142         union X { T t; l4_uint##sz##_t v; } m; \
00143         m.t = value; \
00144         _b->do_write_##sz(m.v, reg); \

```

```

00144     }
00145
00146 /**
00147  * \brief Helper template that translates to the Register_block_base
00148  *         interface.
00149  * \tparam BLOCK The type of the Register_block_base interface to use.
00150  *
00151  * This helper translates read<T>(), write<T>(), set<T>(), clear<T>(),
00152  * and modify<T>() calls to BLOCK::do_read<xx> and BLOCK::do_write<xx>.
00153  */
00154 template< typename BLOCK >
00155 class Register_block_tmpl
00156 : public Register_block_modify_mixin<Register_block_tmpl<BLOCK> >
00157 {
00158 private:
00159     BLOCK *_b;
00160
00161 public:
00162     Register_block_tmpl(BLOCK *blk) : _b(blk) {}
00163     Register_block_tmpl() = default;
00164
00165     operator BLOCK * () const { return _b; }
00166
00167     REGBLK_READ_TEMPLATE(8)
00168     REGBLK_WRITE_TEMPLATE(8)
00169     REGBLK_READ_TEMPLATE(16)
00170     REGBLK_WRITE_TEMPLATE(16)
00171     REGBLK_READ_TEMPLATE(32)
00172     REGBLK_WRITE_TEMPLATE(32)
00173     REGBLK_READ_TEMPLATE(64)
00174     REGBLK_WRITE_TEMPLATE(64)
00175 };
00176
00177
00178 #undef REGBLK_READ_TEMPLATE
00179 #undef REGBLK_WRITE_TEMPLATE
00180
00181 namespace __Type_helper {
00182     template<unsigned> struct Unsigned;
00183     template<> struct Unsigned<8> { typedef l4_uint8_t type; };
00184     template<> struct Unsigned<16> { typedef l4_uint16_t type; };
00185     template<> struct Unsigned<32> { typedef l4_uint32_t type; };
00186     template<> struct Unsigned<64> { typedef l4_uint64_t type; };
00187 };
00188
00189
00190 /**
00191  * \brief Single read only register inside a Register_block_base interface.
00192  * \tparam BITS The access width of the register in bits.
00193  * \tparam BLOCK The type for the Register_block_base interface.
00194  * \note Objects of this type must be used only in temporary contexts
00195  *       not in global, class, or object scope.
00196  *
00197  * Allows simple read only access to a hardware register.
00198  */
00199 template< unsigned BITS, typename BLOCK >
00200 class Ro_register_tmpl
00201 {
00202 protected:
00203     BLOCK *_b;
00204     unsigned _o;
00205
00206 public:
00207     typedef typename __Type_helper::Unsigned<BITS>::type value_type;
00208
00209     Ro_register_tmpl(BLOCK const &blk, unsigned offset) : _b(&blk), _o(offset) {}
00210     Ro_register_tmpl() = default;
00211
00212     /**
00213      * \brief read the value from the hardware register.
00214      * \return value read from the hardware register.
00215      */
00216     operator value_type () const
00217     { return _b.template read<value_type>(_o); }
00218
00219     /**
00220      * \brief read the value from the hardware register.
00221      * \return value from the hardware register.
00222      */
00223     value_type read() const
00224     { return _b.template read<value_type>(_o); }
00225 };
00226
00227
00228 /**
00229  * \brief Single hardware register inside a Register_block_base interface.
00230  * \tparam BITS The access width for the register in bits.

```

```

00231 * \tparam BLOCK the type of the Register_block_base interface.
00232 * \note Objects of this type must be used only in temporary contexts
00233 *       not in global, class, or object scope.
00234 */
00235 template< unsigned BITS, typename BLOCK >
00236 class Register_tmpl : public Ro_register_tmpl<BITS, BLOCK>
00237 {
00238 public:
00239     typedef typename Ro_register_tmpl<BITS, BLOCK>::value_type value_type;
00240
00241     Register_tmpl(BLOCK const &blk, unsigned offset)
00242     : Ro_register_tmpl<BITS, BLOCK>(blk, offset)
00243     {}
00244
00245     Register_tmpl() = default;
00246
00247     /**
00248      * \brief write \a val into the hardware register.
00249      * \param val the value to write into the hardware register.
00250      */
00251     Register_tmpl &operator = (value_type val)
00252     { this->b.template write<value_type>(val, this->o); return *this; }
00253
00254     /**
00255      * \brief write \a val into the hardware register.
00256      * \param val the value to write into the hardware register.
00257      */
00258     void write(value_type val)
00259     { this->b.template write<value_type>(val, this->o); }
00260
00261     /**
00262      * \brief set bits in \a set_bits in the hardware register.
00263      * \param set_bits bits to be set within the hardware register.
00264      *
00265      * This is a read-modify-write function that does a logical or
00266      * of the old value from the register with \a set_bits.
00267      *
00268      * \code
00269      * unsigned old_value = read();
00270      * write(old_value | set_bits);
00271      * \endcode
00272      */
00273     value_type set(value_type set_bits)
00274     { return this->b.template set<value_type>(set_bits, this->o); }
00275
00276     /**
00277      * \brief clears bits in \a clear_bits in the hardware register.
00278      * \param clear_bits bits to be cleared within the hardware register.
00279      *
00280      * This is a read-modify-write function that does a logical and
00281      * of the old value from the register with the negated value of
00282      * \a clear_bits.
00283      *
00284      * \code
00285      * unsigned old_value = read();
00286      * write(old_value & ~clear_bits);
00287      * \endcode
00288      */
00289     value_type clear(value_type clear_bits)
00290     { return this->b.template clear<value_type>(clear_bits, this->o); }
00291
00292     /**
00293      * \brief clears bits in \a clear_bits and sets bits in \a set_bits
00294      *       in the hardware register.
00295      * \param clear_bits bits to be cleared within the hardware register.
00296      * \param set_bits bits to set in the hardware register.
00297      *
00298      * This is a read-modify-write function that first does a logical and
00299      * of the old value from the register with the negated value of
00300      * \a clear_bits and then does a logical or with \a set_bits.
00301      *
00302      * \code{.c}
00303      * unsigned old_value = read();
00304      * write((old_value & ~clear_bits) | set_bits);
00305      * \endcode
00306      */
00307     value_type modify(value_type clear_bits, value_type set_bits)
00308     { return this->b.template modify<value_type>(clear_bits, set_bits, this->o); }
00309 };
00310
00311 /**
00312  * \brief Handles a reference to a register block of the given
00313  *       maximum access width.
00314  * \tparam MAX_BITS Maximum access width for the registers in this
00315  *       block.
00316  * \tparam BLOCK Type implementing the register accesses ('read<>()'),

```

```

00318 *          `write<>()`, `modify<>()`, `set<>()`, and `clear<>()`.
00319 *
00320 * Provides access to registers in this block via r<WIDTH>() and
00321 * operator[]().
00322 */
00323 template<
00324     unsigned MAX_BITS,
00325     typename BLOCK = Register_block_tmpl<
00326         Register_block_base<MAX_BITS>
00327     >
00328 >
00329 class Register_block
00330 {
00331 private:
00332     template< unsigned B, typename BLK > friend class Register_block;
00333     template< unsigned B, typename BLK > friend class Ro_register_block;
00334     typedef BLOCK Block;
00335     Block _b;
00336
00337 public:
00338     Register_block() = default;
00339     Register_block(Block const &blk) : _b(blk) {}
00340     Register_block &operator = (Block const &blk)
00341     { _b = blk; return *this; }
00342
00343     template< unsigned BITS >
00344     Register_block(Register_block<BITS> blk) : _b(blk._b) {}
00345
00346     typedef Register_tmpl<MAX_BITS, Block> Register;
00347     typedef Ro_register_tmpl<MAX_BITS, Block> Ro_register;
00348
00349     /**
00350      * \brief Read only access to register at offset \a offset.
00351      * \tparam BITS the access width in bits for the register.
00352      * \param offset The offset of the register within the register file.
00353      * \return register object allowing read only access with width \a BITS.
00354      */
00355     template< unsigned BITS >
00356     Ro_register_tmpl<BITS, Block> r(unsigned offset) const
00357     { return Ro_register_tmpl<BITS, Block>(this->_b, offset); }
00358
00359     /**
00360      * \brief Read only access to register at offset \a offset.
00361      * \param offset The offset of the register within the register file.
00362      * \return register object allowing read only access with width \a MAX_BITS.
00363      */
00364     Ro_register operator [] (unsigned offset) const
00365     { return this->r<MAX_BITS>(offset); }
00366
00367     /**
00368      * \brief Read/write access to register at offset \a offset.
00369      * \tparam BITS the access width in bits for the register.
00370      * \param offset The offset of the register within the register file.
00371      * \return register object allowing read and write access with width \a BITS.
00372      */
00373     template< unsigned BITS >
00374     Register_tmpl<BITS, Block> r(unsigned offset)
00375     { return Register_tmpl<BITS, Block>(this->_b, offset); }
00376
00377     /**
00378      * \brief Read/write access to register at offset \a offset.
00379      * \param offset The offset of the register within the register file.
00380      * \return register object allowing read and write access with
00381      *         width \a MAX_BITS.
00382      */
00383     Register operator [] (unsigned offset)
00384     { return this->r<MAX_BITS>(offset); }
00385 };
00386
00387 /**
00388 * \brief Handles a reference to a read only register block of the given
00389 *         maximum access width.
00390 * \tparam MAX_BITS Maximum access width for the registers in this block.
00391 * \tparam BLOCK Type implementing the register accesses (read<>()),
00392 *
00393 * Provides read only access to registers in this block via r<WIDTH>()
00394 * and operator[]().
00395 */
00396 template<
00397     unsigned MAX_BITS,
00398     typename BLOCK = Register_block_tmpl<
00399         Register_block_base<MAX_BITS> const
00400     >
00401 >
00402 class Ro_register_block
00403 {
00404 {

```

```

00405 private:
00406     template< unsigned B, typename BLK > friend class Ro_register_block;
00407     typedef BLOCK Block;
00408     Block _b;
00409
00410 public:
00411     Ro_register_block() = default;
00412     Ro_register_block(BLOCK const &blk) : _b(blk) {}
00413
00414     template< unsigned BITS >
00415     Ro_register_block(Register_block<BITS> const &blk) : _b(blk._b) {}
00416
00417     typedef Ro_register_tmpl<MAX_BITS, Block> Ro_register;
00418     typedef Ro_register Register;
00419
00420     /**
00421      * \brief Read only access to register at offset \a offset.
00422      * \param offset The offset of the register within the register file.
00423      * \return register object allowing read only access with width \a MAX_BITS.
00424      */
00425     Ro_register operator [] (unsigned offset) const
00426     { return Ro_register(this->_b, offset); }
00427
00428     /**
00429      * \brief Read only access to register at offset \a offset.
00430      * \tparam BITS the access width in bits for the register.
00431      * \param offset The offset of the register within the register file.
00432      * \return register object allowing read only access with width \a BITS.
00433      */
00434     template< unsigned BITS >
00435     Ro_register_tmpl<BITS, Block> r(unsigned offset) const
00436     { return Ro_register_tmpl<BITS, Block>(this->_b, offset); }
00437 };
00438
00439
00440 /**
00441  * \brief Implementation helper for register blocks.
00442  * \param BASE The class implementing read<> and write<> template functions
00443  *             for accessing the registers. This class must inherit from
00444  *             Register_block_impl.
00445  * \param MAX_BITS The maximum access width for the register file.
00446  *             Supported values are 8, 16, 32, or 64.
00447  *
00448  *
00449  * This template allows easy implementation of register files by providing
00450  * read<> and write<> template functions, see Mmio_register_block
00451  * as an example.
00452  */
00453 template< typename BASE, unsigned MAX_BITS = 32 >
00454 struct Register_block_impl;
00455
00456 #define REGBLK_IMPL_RW_TEMPLATE(sz, ...) \
00457     l4_uint##sz##_t do_read_##sz(l4_addr_t reg) const override \
00458     { return static_cast<BASE const *>(this)->template read<l4_uint##sz##_t>(reg); } \
00459     \
00460     void do_write_##sz(l4_uint##sz##_t value, l4_addr_t reg) override \
00461     { static_cast<BASE*>(this)->template write<l4_uint##sz##_t>(value, reg); }
00462
00463
00464 template< typename BASE >
00465 struct Register_block_impl<BASE, 8> : public Register_block_base<8>
00466 {
00467     REGBLK_IMPL_RW_TEMPLATE(8);
00468 };
00469
00470 template< typename BASE >
00471 struct Register_block_impl<BASE, 16> : public Register_block_base<16>
00472 {
00473     REGBLK_IMPL_RW_TEMPLATE(8);
00474     REGBLK_IMPL_RW_TEMPLATE(16);
00475 };
00476
00477 template< typename BASE >
00478 struct Register_block_impl<BASE, 32> : public Register_block_base<32>
00479 {
00480     REGBLK_IMPL_RW_TEMPLATE(8);
00481     REGBLK_IMPL_RW_TEMPLATE(16);
00482     REGBLK_IMPL_RW_TEMPLATE(32);
00483 };
00484
00485 template< typename BASE >
00486 struct Register_block_impl<BASE, 64> : public Register_block_base<64>
00487 {
00488     REGBLK_IMPL_RW_TEMPLATE(8);
00489     REGBLK_IMPL_RW_TEMPLATE(16);
00490     REGBLK_IMPL_RW_TEMPLATE(32);
00491     REGBLK_IMPL_RW_TEMPLATE(64);

```



```

00492 };
00493
00494 #undef REGBLK_IMPL_RW_TEMPLATE
00495
00496 }

```

## 16.12 io\_regblock.h

```

00001 /*
00002  * (c) 2012 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  */
00009 #pragma once
00010
00011 #ifndef __GXX_EXPERIMENTAL_CXX0X__
00012 #ifndef static_assert
00013 #define static_assert(x, y) \
00014     do { (void)sizeof(char[-(!x)]); } while (0)
00015 #endif
00016 #endif
00017
00018 namespace L4
00019 {
00020     class Io_register_block
00021     {
00022     public:
00023         virtual unsigned char  read8(unsigned long reg) const = 0;
00024
00025         virtual unsigned short read16(unsigned long reg) const = 0;
00026
00027         virtual unsigned int   read32(unsigned long reg) const = 0;
00028
00029         /*
00030          * \brief Read register with an 8 byte access.
00031          */
00032         //virtual unsigned long long read64(unsigned long reg) const = 0;
00033
00034         virtual void write8(unsigned long reg, unsigned char value) const = 0;
00035
00036         virtual void writel6(unsigned long reg, unsigned short value) const = 0;
00037
00038         virtual void write32(unsigned long reg, unsigned int value) const = 0;
00039
00040         /*
00041          * \brief Write register with an 8 byte access.
00042          */
00043         //virtual void write64(unsigned long reg, unsigned long long value) const = 0;
00044
00045         virtual unsigned long addr(unsigned long reg) const = 0;
00046
00047         virtual void delay() const = 0;
00048
00049         virtual ~Io_register_block() = 0;
00050
00051         template< typename R >
00052         R read(unsigned long reg) const
00053         {
00054             switch (sizeof(R))
00055             {
00056             case 1: return read8(reg);
00057             case 2: return read16(reg);
00058             case 4: return read32(reg);
00059             default: static_assert(sizeof(R) == 1 || sizeof(R) == 2 || sizeof(R) == 4,
00060                                     "Invalid size");
00061             };
00062         }
00063
00064         template< typename R >
00065         void write(unsigned long reg, R value) const
00066         {
00067             switch (sizeof(R))
00068             {
00069             case 1: write8(reg, value); return;
00070             case 2: writel6(reg, value); return;
00071             case 4: write32(reg, value); return;
00072             default: static_assert(sizeof(R) == 1 || sizeof(R) == 2 || sizeof(R) == 4,
00073                                     "Invalid size");
00074             };
00075         }
00076     }
00077 }

```

```

00116
00127     template< typename R >
00128     R modify(unsigned long reg, R clear_bits, R set_bits) const
00129     {
00130         R r = (read<R>(reg) & ~clear_bits) | set_bits;
00131         write(reg, r);
00132         return r;
00133     }
00134
00141     template< typename R >
00142     R set(unsigned long reg, R set_bits) const
00143     {
00144         return modify<R>(reg, 0, set_bits);
00145     }
00146
00153     template< typename R >
00154     R clear(unsigned long reg, R clear_bits) const
00155     {
00156         return modify<R>(reg, clear_bits, 0);
00157     }
00158
00159 };
00160
00161 inline Io_register_block::~Io_register_block() {}
00162
00163
00164 class Io_register_block_mmio : public Io_register_block
00165 {
00166 private:
00167     template< typename R >
00168     R _read(unsigned long reg) const
00169     { return *reinterpret_cast<volatile R *>(_base + (reg « _shift)); }
00170
00171     template< typename R >
00172     void _write(unsigned long reg, R val) const
00173     { *reinterpret_cast<volatile R *>(_base + (reg « _shift)) = val; }
00174
00175 public:
00176     Io_register_block_mmio(unsigned long base, unsigned char shift = 0)
00177     : _base(base), _shift(shift)
00178     {}
00179
00180     unsigned long addr(unsigned long reg) const override
00181     { return _base + (reg « _shift); }
00182
00183     unsigned char read8(unsigned long reg) const override
00184     { return _read<unsigned char>(reg); }
00185     unsigned short read16(unsigned long reg) const override
00186     { return _read<unsigned short>(reg); }
00187     unsigned int read32(unsigned long reg) const override
00188     { return _read<unsigned int>(reg); }
00189
00190     void write8(unsigned long reg, unsigned char val) const override
00191     { _write(reg, val); }
00192     void write16(unsigned long reg, unsigned short val) const override
00193     { _write(reg, val); }
00194     void write32(unsigned long reg, unsigned int val) const override
00195     { _write(reg, val); }
00196
00197     void delay() const override
00198     {}
00199
00200 private:
00201     unsigned long _base;
00202     unsigned char _shift;
00203 };
00204
00205 template<typename ACCESS_TYPE>
00206 class Io_register_block_mmio_fixed_width : public Io_register_block
00207 {
00208 private:
00209     template< typename R >
00210     R _read(unsigned long reg) const
00211     { return *reinterpret_cast<volatile ACCESS_TYPE *>(_base + (reg « _shift)); }
00212
00213     template< typename R >
00214     void _write(unsigned long reg, R val) const
00215     { *reinterpret_cast<volatile ACCESS_TYPE *>(_base + (reg « _shift)) = val; }
00216
00217 public:
00218     Io_register_block_mmio_fixed_width(unsigned long base, unsigned char shift = 0)
00219     : _base(base), _shift(shift)
00220     {}
00221
00222     unsigned long addr(unsigned long reg) const
00223     { return _base + (reg « _shift); }
00224

```

```

00225     unsigned char  read8(unsigned long reg) const override
00226     { return _read<unsigned char>(reg); }
00227     unsigned short read16(unsigned long reg) const override
00228     { return _read<unsigned short>(reg); }
00229     unsigned int   read32(unsigned long reg) const override
00230     { return _read<unsigned int>(reg); }
00231
00232     void write8(unsigned long reg, unsigned char val) const override
00233     { _write(reg, val); }
00234     void write16(unsigned long reg, unsigned short val) const override
00235     { _write(reg, val); }
00236     void write32(unsigned long reg, unsigned int val) const override
00237     { _write(reg, val); }
00238
00239     void delay() const override
00240     {}
00241
00242 private:
00243     unsigned long _base;
00244     unsigned char _shift;
00245 };
00246 }

```

## 16.13 io\_regblock\_port.h

```

00001 /*
00002  * (c) 2012 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  */
00009 #pragma once
00010
00011 #include "io_regblock.h"
00012
00013 namespace L4
00014 {
00015     class Io_register_block_port : public Io_register_block
00016     {
00017     public:
00018         Io_register_block_port(unsigned long base)
00019         : _base(base)
00020         {}
00021
00022         unsigned long addr(unsigned long reg) const { return _base + reg; }
00023
00024         unsigned char  read8(unsigned long reg) const
00025         {
00026             unsigned char val;
00027             asm volatile("inb %w1, %b0" : "=a" (val) : "Nd" (_base + reg));
00028             return val;
00029         }
00030
00031         unsigned short read16(unsigned long reg) const
00032         {
00033             unsigned short val;
00034             asm volatile("inw %w1, %w0" : "=a" (val) : "Nd" (_base + reg));
00035             return val;
00036         }
00037
00038         unsigned int   read32(unsigned long reg) const
00039         {
00040             unsigned int val;
00041             asm volatile("in %w1, %0" : "=a" (val) : "Nd" (_base + reg));
00042             return val;
00043         }
00044
00045         void write8(unsigned long reg, unsigned char val) const
00046         { asm volatile("outb %b0, %w1" : : "a" (val), "Nd" (_base + reg)); }
00047
00048         void write16(unsigned long reg, unsigned short val) const
00049         { asm volatile("outw %w0, %w1" : : "a" (val), "Nd" (_base + reg)); }
00050
00051         void write32(unsigned long reg, unsigned int val) const
00052         { asm volatile("out %0, %w1" : : "a" (val), "Nd" (_base + reg)); }
00053
00054         void delay() const
00055         { asm volatile ("outb %al,$0x80"); }
00056
00057     private:
00058         unsigned long _base;

```

```
00059     };
00060 }
```

## 16.14 Makefile

```
00001 PKGDIR = ..
00002 L4DIR ?= $(PKGDIR)/../..
00003
00004 PKGNAME = drivers
00005 PC_FILENAME = drivers-first
00006 EXTRA_TARGET += hw_mmio_register_block hw_register_block
00007
00008 include $(L4DIR)/mk/include.mk
```

## 16.15 Makefile

```
00001 PKGDIR = ../..
00002 L4DIR ?= $(PKGDIR)/../..
00003
00004 PKGNAME = drivers
00005
00006 include $(L4DIR)/mk/include.mk
```

## 16.16 Makefile

```
00001 PKGDIR = ../..
00002 L4DIR ?= $(PKGDIR)/../..
00003
00004 EXTRA_TARGET += cmd_control
00005
00006 include $(L4DIR)/mk/include.mk
```

## 16.17 poll\_timeout\_counter.h

```
00001 /*
00002  * (c) 2012 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  */
00009 #pragma once
00010
00011 namespace L4 {
00012
00036 class Poll_timeout_counter
00037 {
00038 public:
00044     Poll_timeout_counter(unsigned counter_val)
00045     {
00046         set(counter_val);
00047     }
00048
00055     void set(unsigned counter_val)
00056     {
00057         _c = counter_val;
00058     }
00059
00063     bool test(bool expression = true)
00064     {
00065         if (!expression)
00066             return false;
00067
00068         if (_c)
00069         {
00070             --_c;
00071             return true;
00072         }
00073
00074         return false;
00075     }
00076 }
```

```

00075     }
00076
00083     bool timed_out() const { return _c == 0; }
00084
00085 private:
00086     unsigned _c;
00087 };
00088
00089 }

```

## 16.18 uart\_16550.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2023 Kernkonzept GmbH.
00004  */
00005 /*
00006  * (c) 2008-2012 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *      Alexander Warg <alexander.warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  */
00014 #pragma once
00015
00016 #include "uart_base.h"
00017
00018 namespace L4
00019 {
00020     class Uart_16550 : public Uart
00021     {
00022     protected:
00023         enum Registers
00024         {
00025             TRB      = 0x00, // Transmit/Receive Buffer (read/write)
00026             BRD_LOW   = 0x00, // Baud Rate Divisor LSB if bit 7 of LCR is set (read/write)
00027             IER       = 0x01, // Interrupt Enable Register (read/write)
00028             BRD_HIGH  = 0x01, // Baud Rate Divisor MSB if bit 7 of LCR is set (read/write)
00029             IIR       = 0x02, // Interrupt Identification Register (read only)
00030             FCR       = 0x02, // 16550 FIFO Control Register (write only)
00031             LCR       = 0x03, // Line Control Register (read/write)
00032             MCR       = 0x04, // Modem Control Register (read/write)
00033             LSR       = 0x05, // Line Status Register (read only)
00034             MSR       = 0x06, // Modem Status Register (read only)
00035             SPR       = 0x07, // Scratch Pad Register (read/write)
00036         };
00037
00038         enum Register_value_iir
00039         {
00040             IIR_BUSY = 7,
00041         };
00042
00043         enum Register_value_lsr
00044         {
00045             LSR_DR      = 0x01, // Receiver data ready
00046             LSR_THRE    = 0x20, // Transmit hold register empty
00047             LSR_TSRE    = 0x40, // Transmitter empty
00048         };
00049
00050     public:
00051         enum
00052         {
00053             PAR_NONE = 0x00,
00054             PAR_EVEN = 0x18,
00055             PAR_ODD  = 0x08,
00056             DAT_5     = 0x00,
00057             DAT_6     = 0x01,
00058             DAT_7     = 0x02,
00059             DAT_8     = 0x03,
00060             STOP_1    = 0x00,
00061             STOP_2    = 0x04,
00062
00063             MODE_8N1 = PAR_NONE | DAT_8 | STOP_1,
00064             MODE_7E1 = PAR_EVEN | DAT_7 | STOP_1,
00065
00066             // these two values are to leave either mode
00067             // or baud rate unchanged on a call to change_mode
00068             MODE_NC  = 0x1000000,
00069             BAUD_NC   = 0x1000000,
00070
00071             Base_rate_x86 = 115200,

```

```

00072     Base_rate_pxa = 921600,
00073 };
00074
00075 explicit Uart_16550(unsigned long base_rate, unsigned char init_flags = 0,
00076                    unsigned char ier_bits = 0,
00077                    unsigned char mcr_bits = 0, unsigned char fcr_bits = 0)
00078 : _base_rate(base_rate), _init_flags(init_flags), _mcr_bits(mcr_bits),
00079   _ier_bits(ier_bits), _fcr_bits(fcr_bits)
00080 {}
00081
00082 bool startup(Io_register_block const *regs) override;
00083 void shutdown() override;
00084 bool change_mode(Transfer_mode m, Baud_rate r) override;
00085 int get_char(bool blocking = true) const override;
00086 int char_avail() const override;
00087 int tx_avail() const;
00088 void wait_tx_done() const;
00089 inline void out_char(char c) const;
00090 int write(char const *s, unsigned long count,
00091          bool blocking = true) const override;
00092 bool enable_rx_irq(bool enable = true) override;
00093
00094 private:
00095     unsigned long _base_rate;
00096     unsigned char _init_flags;
00097     unsigned char _mcr_bits;
00098     unsigned char _ier_bits;
00099     unsigned char _fcr_bits;
00100 };
00101 }

```

## 16.19 uart\_16550\_dw.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2023 Kernkonzept GmbH.
00004  */
00005 /*
00006  * (c) 2015 Adam Lackorzynski <adam@l4re.org>
00007  *
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU General Public License 2.
00010  * Please see the COPYING-GPL-2 file for details.
00011  */
00012 #pragma once
00013
00014 #include "uart_16550.h"
00015
00016 namespace L4
00017 {
00018     class Uart_16550_dw : public Uart_16550
00019     {
00020     public:
00021         explicit Uart_16550_dw(unsigned long base_rate)
00022             : Uart_16550(base_rate)
00023         {}
00024
00025         void irq_ack() override;
00026     };
00027 }

```

## 16.20 uart\_apb.h

```

00001 /*
00002  * Copyright (C) 2024 Kernkonzept GmbH.
00003  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include "uart_base.h"
00011
00012 namespace L4
00013 {
00014     class Uart_apb : public Uart
00015     {
00016     public:

```

```

00022     Uart_apb(unsigned freq) : _freq(freq) {}
00023     bool startup(Io_register_block const *) override;
00024     void shutdown() override;
00025     bool change_mode(Transfer_mode m, Baud_rate r) override;
00026     bool enable_rx_irq(bool enable) override;
00027     int get_char(bool blocking = true) const override;
00028     int char_avail() const override;
00029     int tx_avail() const;
00030     void wait_tx_done() const;
00031     inline void out_char(char c) const;
00032     int write(char const *s, unsigned long count,
00033              bool blocking = true) const override;
00034
00035 private:
00036     void set_rate(Baud_rate r);
00037     unsigned _freq;
00038 };
00039 
```

## 16.21 uart\_base.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2023 Kernkonzept GmbH.
00004  */
00005 /*
00006  * (c) 2009-2012 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  */
00013 #pragma once
00014
00015 #include <stddef.h>
00016 #include <l4/drivers/io_regblock.h>
00017
00018 #include "poll_timeout_counter.h"
00019
00020 namespace L4
00021 {
00022     class Uart
00023     {
00024     protected:
00025         unsigned _mode;
00026         unsigned _rate;
00027         Io_register_block const *_regs;
00028
00029     public:
00030         void *operator new (size_t, void* a)
00031         { return a; }
00032
00033     public:
00034         typedef unsigned Transfer_mode;
00035         typedef unsigned Baud_rate;
00036
00037         Uart()
00038         : _mode(~0U), _rate(~0U)
00039         {}
00040
00041         virtual bool startup(Io_register_block const *regs) = 0;
00042
00043         virtual ~Uart() {}
00044
00045         virtual void shutdown() = 0;
00046
00047         virtual bool change_mode(Transfer_mode m, Baud_rate r) = 0;
00048
00049         virtual int get_char(bool blocking = true) const = 0;
00050
00051         virtual int char_avail() const = 0;
00052
00053         virtual int write(char const *s, unsigned long count,
00054                          bool blocking = true) const = 0;
00055
00056         virtual void irq_ack() {}
00057
00058         virtual bool enable_rx_irq(bool = true) { return false; }
00059
00060         Transfer_mode mode() const { return _mode; }
00061
00062         Baud_rate rate() const { return _rate; }
00063     };
00064 
```

```

00131
00132 protected:
00144     template <typename Uart_driver>
00145     int generic_write(char const *s, unsigned long count,
00146                       bool blocking = true) const
00147     {
00148         auto *self = static_cast<Uart_driver const*>(this);
00149
00150         unsigned long c;
00151         for (c = 0; c < count; ++c)
00152         {
00153             if (!blocking && !self->tx_avail())
00154                 break;
00155
00156             Poll_timeout_counter i(3000000);
00157             while (i.test(!self->tx_avail()))
00158                 ;
00159
00160             self->out_char(*s++);
00161         }
00162
00163         if (blocking)
00164             self->wait_tx_done();
00165
00166         return c;
00167     }
00168 };
00169 }

```

## 16.22 uart\_cadence.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2023 Kernkonzept GmbH.
00004  */
00005 /*
00006  * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  */
00013 #pragma once
00014
00015 #include "uart_base.h"
00016
00017 namespace L4
00018 {
00019     class Uart_cadence : public Uart
00020     {
00021     public:
00022         explicit Uart_cadence(unsigned base_rate) : _base_rate(base_rate) {}
00023         bool startup(Io_register_block const *) override;
00024         void shutdown() override;
00025         bool change_mode(Transfer_mode m, Baud_rate r) override;
00026         bool enable_rx_irq(bool) override;
00027         int get_char(bool blocking = true) const override;
00028         int char_avail() const override;
00029         int tx_avail() const;
00030         void wait_tx_done() const {}
00031         inline void out_char(char c) const;
00032         int write(char const *s, unsigned long count,
00033                  bool blocking = true) const override;
00034         void irq_ack() override;
00035
00036     private:
00037         unsigned _base_rate;
00038     };
00039 };

```

## 16.23 uart\_dcc-v6.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2023 Kernkonzept GmbH.
00004  */
00005 /*
00006  * (c) 2009 Technische Universität Dresden

```



```

00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  */
00011 #pragma once
00012
00013 #include "uart_base.h"
00014
00015 namespace L4
00016 {
00017     class Uart_dcc_v6 : public Uart
00018     {
00019     public:
00020         explicit Uart_dcc_v6() {}
00021         explicit Uart_dcc_v6(unsigned /*base_rate*/) {}
00022         bool startup(Io_register_block const *) override;
00023         void shutdown() override;
00024         bool change_mode(Transfer_mode m, Baud_rate r) override;
00025         int get_char(bool blocking = true) const override;
00026         int char_avail() const override;
00027         int tx_avail() const;
00028         void wait_tx_done() const;
00029         inline void out_char(char c) const;
00030         int write(char const *s, unsigned long count,
00031                 bool blocking = true) const override;
00032     private:
00033         unsigned get_status() const;
00034     };
00035 };

```

## 16.24 uart\_dm.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2021-2022 Stephan Gerhold <stephan@gerhold.net>
00004  * Copyright (C) 2022-2023 Kernkonzept GmbH.
00005  */
00006 #pragma once
00007
00008 #include "uart_base.h"
00009
00010 namespace L4
00011 {
00012     class Uart_dm : public Uart
00013     {
00014     public:
00015         explicit Uart_dm(unsigned /*base_rate*/) {}
00016         bool startup(Io_register_block const *) override;
00017         void shutdown() override;
00018         bool change_mode(Transfer_mode m, Baud_rate r) override;
00019         bool enable_rx_irq(bool enable = true) override;
00020         int get_char(bool blocking = true) const override;
00021         int char_avail() const override;
00022         int tx_avail() const;
00023         void wait_tx_done() const;
00024         inline void out_char(char c) const;
00025         int write(char const *s, unsigned long count,
00026                 bool blocking = true) const override;
00027     };
00028 };

```

## 16.25 uart\_dummy.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2023 Kernkonzept GmbH.
00004  */
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  */
00013 #pragma once
00014
00015 #include "uart_base.h"
00016

```

```

00017 namespace L4
00018 {
00019     class Uart_dummy : public Uart
00020     {
00021     public:
00022         explicit Uart_dummy() {}
00023         explicit Uart_dummy(unsigned /*base_rate*/) {}
00024         bool startup(Io_register_block const *) override { return true; }
00025         void shutdown() override {}
00026         bool change_mode(Transfer_mode, Baud_rate) override { return true; }
00027         int get_char(bool /*blocking*/ = true) const override { return 0; }
00028         int char_avail() const override { return false; }
00029         inline void out_char(char /*ch*/) const {}
00030         int write(char const * /*str*/, unsigned long /*count*/,
00031                 bool /*blocking*/ = true) const override
00032         { return 0; }
00033     };
00034 };

```

## 16.26 uart\_geni.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2022-2023 Kernkonzept GmbH.
00004  * Author(s): Stephan Gerhold <stephan.gerhold@kernkonzept.com>
00005  */
00006 #pragma once
00007
00008 #include "uart_base.h"
00009
00010 namespace L4
00011 {
00012     class Uart_geni : public Uart
00013     {
00014     public:
00015         explicit Uart_geni(unsigned /*base_rate*/) {}
00016         bool startup(Io_register_block const *) override;
00017         void shutdown() override;
00018         bool change_mode(Transfer_mode m, Baud_rate r) override;
00019         bool enable_rx_irq(bool enable = true) override;
00020         void irq_ack() override;
00021         int get_char(bool blocking = true) const override;
00022         int char_avail() const override;
00023         int tx_avail() const;
00024         void wait_tx_done() const;
00025         void out_char(char c) const;
00026         int write(char const *s, unsigned long count,
00027                 bool blocking = true) const override;
00028     };
00029 };

```

## 16.27 uart\_imx.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2023 Kernkonzept GmbH.
00004  */
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  */
00013 #pragma once
00014
00015 #include "uart_base.h"
00016
00017 namespace L4
00018 {
00019     class Uart_imx : public Uart
00020     {
00021     public:
00022         enum platform_type
00023         {
00024             Type_imx21,
00025             Type_imx35,
00026             Type_imx51,

```

```

00027     Type_imx6,
00028     Type_imx7,
00029     Type_imx8,
00030 };
00031 explicit Uart_imx(enum platform_type type) : _type(type) {}
00032 explicit Uart_imx(enum platform_type type, unsigned /*base_rate*/)
00033     : _type(type) {}
00034 bool startup(Io_register_block const *) override;
00035 void shutdown() override;
00036 bool enable_rx_irq(bool enable = true) override;
00037 bool change_mode(Transfer_mode m, Baud_rate r) override;
00038 int get_char(bool blocking = true) const override;
00039 int char_avail() const override;
00040 int tx_avail() const;
00041 void wait_tx_done() const;
00042 inline void out_char(char c) const;
00043 int write(char const *s, unsigned long count,
00044         bool blocking = true) const override;
00045
00046 private:
00047     enum platform_type _type;
00048 };
00049
00050 class Uart_imx21 : public Uart_imx
00051 {
00052 public:
00053     Uart_imx21() : Uart_imx(Type_imx21) {}
00054     explicit Uart_imx21(unsigned base_rate) : Uart_imx(Type_imx21, base_rate) {}
00055 };
00056
00057 class Uart_imx35 : public Uart_imx
00058 {
00059 public:
00060     Uart_imx35() : Uart_imx(Type_imx35) {}
00061     explicit Uart_imx35(unsigned base_rate) : Uart_imx(Type_imx35, base_rate) {}
00062 };
00063
00064 class Uart_imx51 : public Uart_imx
00065 {
00066 public:
00067     Uart_imx51() : Uart_imx(Type_imx51) {}
00068     explicit Uart_imx51(unsigned base_rate) : Uart_imx(Type_imx51, base_rate) {}
00069 };
00070
00071 class Uart_imx6 : public Uart_imx
00072 {
00073 public:
00074     Uart_imx6() : Uart_imx(Type_imx6) {}
00075     explicit Uart_imx6(unsigned base_rate) : Uart_imx(Type_imx6, base_rate) {}
00076
00077     void irq_ack() override;
00078 };
00079
00080 class Uart_imx7 : public Uart_imx
00081 {
00082 public:
00083     Uart_imx7() : Uart_imx(Type_imx7) {}
00084     explicit Uart_imx7(unsigned base_rate) : Uart_imx(Type_imx7, base_rate) {}
00085 };
00086
00087 class Uart_imx8 : public Uart_imx
00088 {
00089 public:
00090     Uart_imx8() : Uart_imx(Type_imx8) {}
00091     explicit Uart_imx8(unsigned base_rate) : Uart_imx(Type_imx8, base_rate) {}
00092 };
00093 };

```

## 16.28 uart\_leon3.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2023 Kernkonzept GmbH.
00004  */
00005 /*
00006  * (c) 2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  */

```

```

00014 #pragma once
00015
00016 #include "uart_base.h"
00017
00018 namespace L4
00019 {
00020     class Uart_leon3 : public Uart
00021     {
00022     public:
00023         explicit Uart_leon3() {}
00024         explicit Uart_leon3(unsigned /*base_rate*/) {}
00025         bool startup(Io_register_block const *) override;
00026         void shutdown() override;
00027         bool change_mode(Transfer_mode m, Baud_rate r) override;
00028         int get_char(bool blocking = true) const override;
00029         int char_avail() const override;
00030         int tx_avail() const;
00031         void wait_tx_done() const;
00032         bool enable_rx_irq(bool = true) override;
00033         inline void out_char(char c) const;
00034         int write(char const *s, unsigned long count,
00035                 bool blocking = true) const override;
00036     };
00037 };

```

## 16.29 uart\_linflex.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2023 Kernkonzept GmbH.
00004  */
00005 /*
00006  * (c) 2018 Adam Lackorzynski <adam@l4re.org>
00007  *
00008  * This file is part of L4Re and distributed under the terms of the
00009  * GNU General Public License 2.
00010  * Please see the COPYING-GPL-2 file for details.
00011  */
00012 #pragma once
00013
00014 #include "uart_base.h"
00015
00016 namespace L4
00017 {
00018     class Uart_linflex : public Uart
00019     {
00020     public:
00021         explicit Uart_linflex(unsigned) {}
00022         bool startup(Io_register_block const *) override;
00023         void shutdown() override;
00024         bool enable_rx_irq(bool enable = true) override;
00025         bool change_mode(Transfer_mode m, Baud_rate r) override;
00026         int get_char(bool blocking = true) const override;
00027         int char_avail() const override;
00028         int tx_avail() const;
00029         void wait_tx_done() const;
00030         inline void out_char(char c) const;
00031         int write(char const *s, unsigned long count,
00032                 bool blocking = true) const override;
00033     };
00034 };

```

## 16.30 uart\_lpuart.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2023 Kernkonzept GmbH.
00004  */
00005 /*
00006  * (c) 2019 Adam Lackorzynski <adam@l4re.org>
00007  *
00008  * This file is part of L4Re and distributed under the terms of the
00009  * GNU General Public License 2.
00010  * Please see the COPYING-GPL-2 file for details.
00011  */
00012 #pragma once
00013
00014 #include "uart_base.h"
00015

```

```

00016 namespace L4
00017 {
00018     class Uart_lpuart : public Uart
00019     {
00020     public:
00021         explicit Uart_lpuart() {}
00022         explicit Uart_lpuart(unsigned /*base_rate*/) {}
00023         bool startup(Io_register_block const *) override;
00024         void shutdown() override;
00025         bool enable_rx_irq(bool enable = true) override;
00026         bool change_mode(Transfer_mode m, Baud_rate r) override;
00027         int get_char(bool blocking = true) const override;
00028         int char_avail() const override;
00029         int tx_avail() const;
00030         void wait_tx_done() const {}
00031         inline void out_char(char c) const;
00032         int write(char const *s, unsigned long count,
00033                 bool blocking = true) const override;
00034     };
00035 };

```

## 16.31 uart\_mvebu.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2023 Kernkonzept GmbH.
00004  */
00005 /*
00006  * (c) 2017 Adam Lackorzynski <adam@l4re.org>
00007  *
00008  * This file is part of L4Re and distributed under the terms of the
00009  * GNU General Public License 2.
00010  * Please see the COPYING-GPL-2 file for details.
00011  */
00012 #pragma once
00013
00014 #include "uart_base.h"
00015
00016 namespace L4
00017 {
00018     class Uart_mvebu : public Uart
00019     {
00020     public:
00021         explicit Uart_mvebu(unsigned baserate) : _baserate(baserate) {}
00022         bool startup(Io_register_block const *) override;
00023         void shutdown() override;
00024         bool enable_rx_irq(bool enable = true) override;
00025         bool change_mode(Transfer_mode m, Baud_rate r) override;
00026         int get_char(bool blocking = true) const override;
00027         int char_avail() const override;
00028         int tx_avail() const;
00029         void wait_tx_done() const {}
00030         inline void out_char(char c) const;
00031         int write(char const *s, unsigned long count,
00032                 bool blocking = true) const override;
00033     private:
00034         unsigned _baserate;
00035     };
00036 };

```

## 16.32 uart\_of.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2023 Kernkonzept GmbH.
00004  */
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  */
00013 #pragma once
00014
00015 #include "uart_base.h"
00016 #include <stdarg.h>
00017 #include <string.h>

```

```

00018 #include <l4/drivers/of.h>
00019
00020 namespace L4
00021 {
00022     class Uart_of : public Uart, public L4_drivers::Of
00023     {
00024     private:
00025         ihandle_t _serial;
00026
00027     public:
00028         Uart_of() : Of(), _serial(0) {}
00029         explicit Uart_of(unsigned /*base_rate*/) : Of(), _serial(0) {}
00030         bool startup(Io_register_block const *) override;
00031         void shutdown() override;
00032         bool change_mode(Transfer_mode m, Baud_rate r) override;
00033         int get_char(bool blocking = true) const override;
00034         int char_avail() const override;
00035         int tx_avail() const;
00036         void out_char(char c) const;
00037         int write(char const *s, unsigned long count,
00038                 bool blocking = true) const override;
00039     };
00040 };

```

## 16.33 uart\_omap35x.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2023 Kernkonzept GmbH.
00004  */
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  */
00013 #pragma once
00014
00015 #include "uart_base.h"
00016
00017 namespace L4
00018 {
00019     class Uart_omap35x : public Uart
00020     {
00021     public:
00022         explicit Uart_omap35x() {}
00023         explicit Uart_omap35x(unsigned /*base_rate*/) {}
00024         bool startup(Io_register_block const *) override;
00025         void shutdown() override;
00026         bool change_mode(Transfer_mode m, Baud_rate r) override;
00027         bool enable_rx_irq(bool) override;
00028         int get_char(bool blocking = true) const override;
00029         int char_avail() const override;
00030         int tx_avail() const;
00031         void wait_tx_done() const;
00032         inline void out_char(char c) const;
00033         int write(char const *s, unsigned long count,
00034                 bool blocking = true) const override;
00035     };
00036 };

```

## 16.34 uart\_pl011.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2023 Kernkonzept GmbH.
00004  */
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  */
00013 #pragma once
00014

```

```

00015 #include "uart_base.h"
00016
00017 namespace L4
00018 {
00019     class Uart_pl011 : public Uart
00020     {
00021     public:
00022         Uart_pl011(unsigned freq) : _freq(freq) {}
00023         bool startup(Io_register_block const *) override;
00024         void shutdown() override;
00025         bool change_mode(Transfer_mode m, Baud_rate r) override;
00026         bool enable_rx_irq(bool enable) override;
00027         int get_char(bool blocking = true) const override;
00028         int char_avail() const override;
00029         int tx_avail() const;
00030         void wait_tx_done() const;
00031         inline void out_char(char c) const;
00032         int write(char const *s, unsigned long count,
00033                 bool blocking = true) const override;
00034     private:
00035         void set_rate(Baud_rate r);
00036         unsigned _freq;
00037     };
00038 };
00039
00040 };

```

## 16.35 uart\_s3c2410.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2023 Kernkonzept GmbH.
00004  */
00005 /*
00006  * (c) 2009-2012 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  */
00013 #pragma once
00014
00015 #include "uart_base.h"
00016
00017 namespace L4
00018 {
00019     class Uart_s3c : public Uart
00020     {
00021     protected:
00022         enum Uart_type
00023         {
00024             Type_24xx, Type_64xx, Type_s5pv210,
00025         };
00026
00027         Uart_type type() const { return _type; }
00028     public:
00029         explicit Uart_s3c(Uart_type type) : _type(type) {}
00030         explicit Uart_s3c(Uart_type type, unsigned /*base_rate*/) : _type(type) {}
00031         bool startup(Io_register_block const *) override;
00032         void shutdown() override;
00033         bool change_mode(Transfer_mode m, Baud_rate r) override;
00034         int get_char(bool blocking = true) const override;
00035         int char_avail() const override;
00036         int tx_avail() const;
00037         void wait_tx_done() const;
00038         inline void out_char(char c) const;
00039         int write(char const *s, unsigned long count,
00040                 bool blocking = true) const override;
00041         void fifo_reset();
00042     protected:
00043         virtual void ack_rx_irq() const = 0;
00044         virtual void wait_for_empty_tx_fifo() const = 0;
00045         virtual unsigned is_rx_fifo_non_empty() const = 0;
00046         virtual unsigned is_tx_fifo_not_full() const = 0;
00047     private:
00048         Uart_type _type;
00049     };
00050
00051     class Uart_s3c2410 : public Uart_s3c
00052     {
00053     };
00054 };
00055

```

```

00056 public:
00057     Uart_s3c2410() : Uart_s3c(Type_24xx) {}
00058     explicit Uart_s3c2410(unsigned base_rate) : Uart_s3c(Type_24xx, base_rate) {}
00059
00060 protected:
00061     void ack_rx_irq() const override {}
00062     void wait_for_empty_tx_fifo() const override;
00063     unsigned is_rx_fifo_non_empty() const override;
00064     unsigned is_tx_fifo_not_full() const override;
00065
00066     void auto_flow_control(bool on);
00067 };
00068
00069 class Uart_s3c64xx : public Uart_s3c
00070 {
00071 public:
00072     Uart_s3c64xx() : Uart_s3c(Type_64xx) {}
00073     explicit Uart_s3c64xx(unsigned base_rate) : Uart_s3c(Type_64xx, base_rate) {}
00074
00075 protected:
00076     void ack_rx_irq() const override;
00077     void wait_for_empty_tx_fifo() const override;
00078     unsigned is_rx_fifo_non_empty() const override;
00079     unsigned is_tx_fifo_not_full() const override;
00080 };
00081
00082 class Uart_s5pv210 : public Uart_s3c
00083 {
00084 public:
00085     Uart_s5pv210() : Uart_s3c(Type_s5pv210) {}
00086     explicit Uart_s5pv210(unsigned base_rate) : Uart_s3c(Type_s5pv210, base_rate) {}
00087
00088 protected:
00089     void ack_rx_irq() const override;
00090     void wait_for_empty_tx_fifo() const override;
00091     unsigned is_rx_fifo_non_empty() const override;
00092     unsigned is_tx_fifo_not_full() const override;
00093 };
00094 };

```

## 16.36 uart\_sa1000.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2023 Kernkonzept GmbH.
00004  */
00005 /*
00006  * (c) 2008-2012 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *      Alexander Warg <alexander.warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  */
00014 #pragma once
00015
00016 #include "uart_base.h"
00017
00018 namespace L4
00019 {
00020     class Uart_sa1000 : public Uart
00021     {
00022     public:
00023         explicit Uart_sa1000() {}
00024         explicit Uart_sa1000(unsigned /*base_rate*/) {}
00025         bool startup(Io_register_block const *) override;
00026         void shutdown() override;
00027         bool change_mode(Transfer_mode m, Baud_rate r) override;
00028         int get_char(bool blocking = true) const override;
00029         int char_avail() const override;
00030         int tx_avail() const;
00031         void wait_tx_done() const;
00032         inline void out_char(char c) const;
00033         int write(char const *s, unsigned long count,
00034                 bool blocking = true) const override;
00035     };
00036 };

```



## 16.37 uart\_sbi.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2021 Kernkonzept GmbH.
00004  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00005  */
00006 #pragma once
00007
00008 #include "uart_base.h"
00009
00010 namespace L4
00011 {
00012     class Uart_sbi : public Uart
00013     {
00014     public:
00015         Uart_sbi();
00016         explicit Uart_sbi(unsigned /*base_rate*/) : Uart_sbi() {}
00017         bool startup(Io_register_block const *) override;
00018         void shutdown() override;
00019         bool change_mode(Transfer_mode m, Baud_rate r) override;
00020         int get_char(bool blocking = true) const override;
00021         int char_avail() const override;
00022         int tx_avail() const { return true; }
00023         void wait_tx_done() const {}
00024         inline void out_char(char c) const;
00025         int write(char const *s, unsigned long count,
00026                 bool blocking = true) const override;
00027     private:
00028         mutable int _bufchar;
00029     };
00030 };

```

## 16.38 uart\_sh.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2023 Kernkonzept GmbH.
00004  */
00005 /*
00006  * (c) 2016 Adam Lackorzynski <adam@l4re.org>
00007  *
00008  * This file is part of L4Re and distributed under the terms of the
00009  * GNU General Public License 2.
00010  * Please see the COPYING-GPL-2 file for details.
00011  */
00012 #pragma once
00013
00014 #include "uart_base.h"
00015
00016 namespace L4
00017 {
00018     class Uart_sh : public Uart
00019     {
00020     public:
00021         explicit Uart_sh() {}
00022         explicit Uart_sh(unsigned /*base_rate*/) {}
00023         bool startup(Io_register_block const *) override;
00024         void shutdown() override;
00025         bool enable_rx_irq(bool enable = true) override;
00026         bool change_mode(Transfer_mode m, Baud_rate r) override;
00027         void irq_ack() override;
00028         int get_char(bool blocking = true) const override;
00029         int char_avail() const override;
00030         int tx_avail() const;
00031         void wait_tx_done() const {}
00032         inline void out_char(char c) const;
00033         int write(char const *s, unsigned long count,
00034                 bool blocking = true) const override;
00035     };
00036 };

```

## 16.39 cmd\_control

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * Copyright (C) 2016 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>

```

```

00006  *
00007  * This file is distributed under the terms of the GNU General Public
00008  * License, version 2. Please see the COPYING-GPL-2 file for details.
00009  */
00010 #pragma once
00011
00012 #include <l4/sys/cxx/ipc_epiface>
00013 #include <l4/sys/cxx/ipc_string>
00014
00015 namespace L4Re { namespace Ned {
00016
00017 /**
00018  * Direct control interface for Ned.
00019  */
00020 class Cmd_control : public L4::Kobject_0t<Cmd_control>
00021 {
00022     L4_INLINE_RPC_NF(long, execute, (L4::Ipc::String<> cmd,
00023                                     L4::Ipc::Array<char> &result));
00024
00025 public:
00026     /**
00027      * Execute the given Lua code.
00028      *
00029      * \param[in] cmd      String with Lua code to execute.
00030      *
00031      * \retval L4_EOK      Code was successfully executed.
00032      * \retval -L4_EINVAL  Code could not be parsed.
00033      * \retval -L4_EIO     Error during code execution.
00034      *
00035      * The code is executed using the global Lua state of ned
00036      * which is retained between successive calls to execute.
00037      * Thus you may define data in one call to execute and use
00038      * it in a subsequent call.
00039      *
00040      * This function does not return any results from the execution
00041      * of the Lua code itself.
00042      */
00043     long execute(L4::Ipc::String<> cmd) noexcept
00044     {
00045         L4::Ipc::Array<char> res(0, NULL);
00046         return execute_t::call(c(), cmd, res);
00047     }
00048
00049     /**
00050      * Execute the given Lua code.
00051      *
00052      * \param[in] cmd      String with Lua code to execute.
00053      * \param[out] result   The first return value of the Lua code block
00054      *                      as string.
00055      *
00056      * \retval L4_EOK      Code was successfully executed.
00057      * \retval -L4_EINVAL  Code could not be parsed.
00058      * \retval -L4_EIO     Error during code execution.
00059      *
00060      * The code is executed using the global Lua state of ned
00061      * which is retained between successive calls to execute.
00062      * Thus you may define data in one call to execute and use
00063      * it in a subsequent call.
00064      */
00065     long execute(L4::Ipc::String<> cmd,
00066                 L4::Ipc::String<char> *result) noexcept
00067     {
00068         L4::Ipc::Array<char> res(result->length, result->data);
00069         long r = execute_t::call(c(), cmd, res);
00070         if (r >= 0)
00071             result->length = res.length;
00072         return r;
00073     }
00074
00075     typedef L4::Typeid::Rpc<execute_t> Rpc;
00076 };
00077
00078 } } // namespace

```

## 16.40 amd64/l4/util/idt.h File Reference

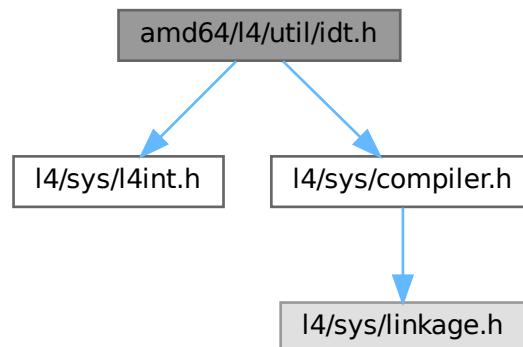
IDT related functions.

```

#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for idt.h:



## Data Structures

- struct `l4util_idt_desc_t`  
*IDT entry.*
- struct `l4util_idt_header_t`  
*Header of an IDT table.*

## 16.40.1 Detailed Description

IDT related functions.

### Date

2003

### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [idt.h](#).

## 16.41 idt.h

[Go to the documentation of this file.](#)

```

00001
00009 /*
00010  * (c) 2003-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU Lesser General Public License 2.1.
00014  * Please see the COPYING-LGPL-2.1 file for details.
00015  */

```

```

00016
00017 #ifndef __L4UTIL_IDT_H
00018 #define __L4UTIL_IDT_H
00019
00020 #include <l4/sys/l4int.h>
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00033 typedef struct
00034 {
00035     l4_uint64_t      a, b;
00036 } __attribute__((packed)) l4util_idt_desc_t;
00037
00040 typedef struct
00041 {
00042     l4_uint16_t      limit;
00043     void             *base;
00044     l4util_idt_desc_t desc[0];
00045 } __attribute__((packed)) l4util_idt_header_t;
00046
00052 static inline void
00053 l4util_idt_entry(l4util_idt_header_t *idt, int nr, void(*handler)(void))
00054 {
00055     idt->desc[nr].a = (l4_uint64_t)handler & 0x0000ffff;
00056     idt->desc[nr].b = 0x0000ef00 | ((l4_uint64_t)handler & 0xffff0000);
00057 }
00058
00063 static inline void
00064 l4util_idt_init(l4util_idt_header_t *idt, int entries)
00065 {
00066     int i;
00067     idt->limit = entries*8 - 1;
00068     idt->base = &idt->desc;
00069
00070     for (i=0; i<entries; i++)
00071         l4util_idt_entry(idt, i, 0);
00072 }
00073
00077 static inline void
00078 l4util_idt_load(l4util_idt_header_t *idt)
00079 {
00080     asm volatile ("lidt (%rax) \n\t" : : "a" (idt));
00081 }
00083 EXTERN_C_END
00084
00085 #endif
00086

```

## 16.42 x86/l4/util/idt.h File Reference

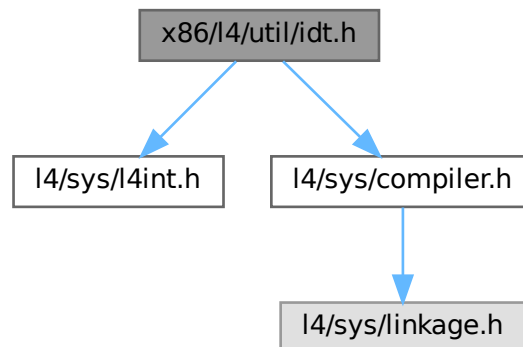
IDT related functions.

```

#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for idt.h:



## Data Structures

- struct `i4util_idt_desc_t`  
*IDT entry.*
- struct `i4util_idt_header_t`  
*Header of an IDT table.*

## 16.42.1 Detailed Description

IDT related functions.

### Date

2003

### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [idt.h](#).

## 16.43 idt.h

[Go to the documentation of this file.](#)

```

00001
00009 /*
00010  * (c) 2003-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU Lesser General Public License 2.1.
00014  * Please see the COPYING-LGPL-2.1 file for details.
00015  */

```

```

00016
00017 #ifndef __L4UTIL_IDT_H
00018 #define __L4UTIL_IDT_H
00019
00020 #include <l4/sys/l4int.h>
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00033 typedef struct
00034 {
00035     l4_uint32_t      a, b;
00036 } __attribute__((packed)) l4util_idt_desc_t;
00037
00040 typedef struct
00041 {
00042     l4_uint16_t      limit;
00043     void             *base;
00044     l4util_idt_desc_t desc[0];
00045 } __attribute__((packed)) l4util_idt_header_t;
00046
00052 static inline void
00053 l4util_idt_entry(l4util_idt_header_t *idt, int nr, void(*handler)(void))
00054 {
00055     idt->desc[nr].a = (l4_uint32_t)handler & 0x0000ffff;
00056     idt->desc[nr].b = 0x0000ef00 | ((l4_uint32_t)handler & 0xffff0000);
00057 }
00058
00063 static inline void
00064 l4util_idt_init(l4util_idt_header_t *idt, int entries)
00065 {
00066     int i;
00067     idt->limit = entries*8 - 1;
00068     idt->base = &idt->desc;
00069
00070     for (i=0; i<entries; i++)
00071         l4util_idt_entry(idt, i, 0);
00072 }
00073
00077 static inline void
00078 l4util_idt_load(l4util_idt_header_t *idt)
00079 {
00080     asm volatile ("lidt (%eax) \n\t" : : "a" (idt));
00081 }
00082
00084 EXTERN_C_END
00085
00086 #endif
00087

```

## 16.44 amd64/l4/util/perform.h File Reference

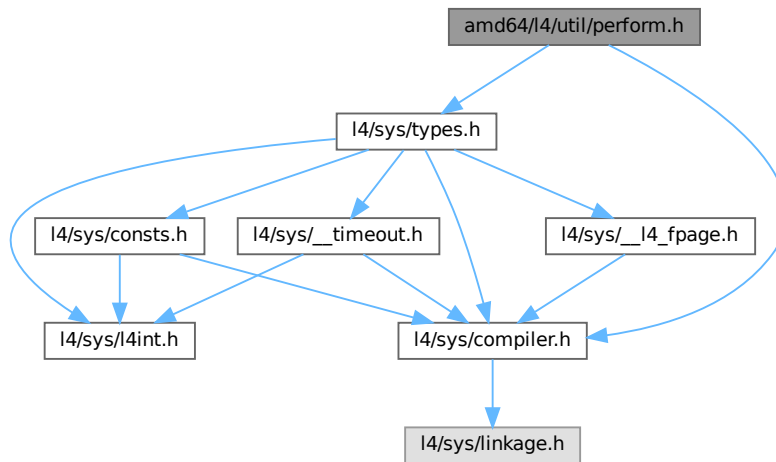
Performance Monitoring using P5/P6 Measurement Counters.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for perform.h:



### 16.44.1 Detailed Description

Performance Monitoring using P5/P6 Measurement Counters.

Define either `CPU_PENTIUM` or `CPU_P6`

Definition in file [perform.h](#).

## 16.45 perform.h

[Go to the documentation of this file.](#)

```

00001
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *          Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *          economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
00014  */
00015 #ifndef __L4UTIL_PERFORM_H
00016 #define __L4UTIL_PERFORM_H
00017
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/compiler.h>
00020
00021 EXTERN_C_BEGIN
00022
00023 extern const char*strp6pmc_event(l4_uint32_t event);
00024
00025 #ifndef CONFIG_PERFORM_ONLY_PROTOTYPES
00026
00027 #if ! (defined CPU_PENTIUM ^ defined CPU_P6 ^ defined CPU_K7)
00028
00029 #error You must define your target architecture.
00030 #error Define EITHER CPU_PENTIUM for Intel Pentium or CPU_P6 for Intel PPro/PII/PIII.
00031
00032 #else
00033
00034 /* P5/P6/K7 section */
00035

```

```

00036 /* Makros for access to model specific registers (MSR) */
00037
00038 /* Write the 64-Bit Model Specific Register. First argument is the register,
00039    second the 64-Bit value. This can only be called at privilege level 0.
00040    With L4, the kernel emulates the WRMSR when calling in PL 3.
00041    */
00042 static inline void l4_i586_wrmsr(unsigned reg,unsigned long long*val){
00043     unsigned long dummyeax, dummyecx, dummyedx;
00044
00045     asm volatile(
00046         ".byte 0xf; .byte 0x30\n" /* wrmsr */
00047         : "=a" (dummyeax), "=d" (dummyedx), "=c" (dummyecx)
00048         : "2" (reg), "0" (*(unsigned *)val), "1" (*(unsigned *)val+1))
00049     );
00050 }
00051
00052 /* Read the 64-Bit Model Specific Register. First argument is the register,
00053    second the address to a 64-Bit value. This can only be called at
00054    privilege level 0. With L4, the kernel emulates the RDMSR when calling
00055    in PL 3.
00056    */
00057 static inline void l4_i586_rdmsr(unsigned reg,unsigned long long*val){
00058     unsigned dummy;
00059
00060     asm volatile(
00061         ".byte 0xf; .byte 0x32\n" /* rdmsr */
00062         : "=a" (*(unsigned *)val), "=d" (*(unsigned *)val+1), "=c" (dummy)
00063         : "2" (reg)
00064     );
00065 }
00066
00067 #ifdef CPU_PENTIUM
00068 /* Pentium section */
00069
00070 /* functions and events defined here are only usable at Pentium
00071    Processors. P6 architecture does NOT support this kind of measuring and
00072    these events. P6 architecture has its own counters and its own events.
00073    See P6-section for details. */
00074
00075 /* from l4linux/arch/l4-i386/include/perform.h */
00076
00077 static inline void
00078 l4_i586_reset_event_counter(void){
00079     asm volatile("xor %%rax, %%rax\n"
00080         "xor %%rdx, %%rdx\n"
00081         "mov $0x12, %%rcx\n"
00082         ".byte 0x0f, 0x30\n"
00083         "movl $0x13, %%rcx\n"
00084         ".byte 0x0f, 0x30\n"
00085         : : : "cx", "ax", "dx"
00086     );
00087 };
00088
00089 static inline void
00090 l4_i586_read_event_counter_long(long long *counter0, long long *counter1)
00091 {
00092     asm volatile(
00093         /* "movl $0, %%eax\n"
00094         "movl $0x11, %%ecx\n"
00095         ".byte 0x0f, 0x30\n" */ /* stop event counting */
00096         "mov $0x12, %%rcx\n"
00097         ".byte 0x0f, 0x32\n"
00098         "mov %%rax, (%%%rbx)\n"
00099         "mov %%rdx, 4(%%rbx)\n"
00100         "mov $0x13, %%ecx\n"
00101         ".byte 0x0f, 0x32\n"
00102         "mov %%rax, (%%%rsi)\n"
00103         "mov %%rdx, 4(%%rsi)\n"
00104         : /* no output */
00105         : "b" (counter0), "S" (counter1)
00106         : "ax", "cx", "dx"
00107     );
00108 }
00109
00110 static inline void
00111 l4_i586_read_event_counter(int *counter0, int *counter1)
00112 {
00113     asm volatile("push %%rdx \n"
00114         ".byte 0x0f, 0x30 \n"
00115         "mov $0x12, %%rcx \n"
00116         ".byte 0x0f, 0x32 \n"
00117         "mov %%rax, %%rbx \n"
00118         "movl $0x13, %%rcx \n"
00119         ".byte 0x0f, 0x32\n"
00120         "popl %%edx\n"
00121         : "=b" (*counter0), "=a" (*counter1)

```



```

00123         : "l" (0), "c" (0x11)
00124     );
00125 }
00126
00127 static inline void
00128 l4_i586_select_event(int event0, int event1)
00129 {
00130     asm volatile(".byte 0x0f, 0x30\n"
00131         :
00132         :
00133         "a" (event0 + (event1 < 16)),
00134         "d" (0),
00135         "c" (0x11)
00136     );
00137 };
00138
00139 #define P5_RD_MISS          0x003 /* 000011B */
00140 #define P5_WR_MISS          0x008 /* 000100B */
00141 #define P5_RW_MISS          0x029 /* 101001B */
00142 #define P5_EX_MISS          0x00e /* 001110B */
00143
00144 #define P5_D_WBACK          0x006 /* 000110B */
00145
00146 #define P5_RW_TLB           0x002 /* 00010B */
00147 #define P5_EX_TLB           0x00d /* 01101B */
00148
00149 #define P5_A_STALL           0x01f /* 11111B */
00150 #define P5_W_STALL           0x019 /* 11001B */
00151 #define P5_R_STALL           0x01a /* 11010B */
00152 #define P5_X_STALL           0x01b /* 11011B */
00153
00154 #define P5_AGI_STALL         0x01f /* 11111B */
00155
00156 #define P5_PIPELINE_FLUSH   0x015 /* 10101B */
00157
00158 #define P5_NON_CACHE_RD      0x01e /* 11110B */
00159 #define P5_NCACHE_REFS      0x01e /* 11110B */
00160 #define P5_LOCKED_BUS        0x01c /* 11100B */
00161
00162 #define P5_MEM2PIPE          0x009 /* 01001B */
00163 #define P5_BANK_CONF         0x00a /* 01010B */
00164
00165
00166 #define P5_INSTRS_EX          0x016 /* 10110B */
00167 #define P5_INSTRS_EX_V       0x017 /* 10111B */
00168
00169
00170 #define P5_CNT_NOTHING       (0x00 < 6) /* 00B < 6 */
00171 #define P5_CNT_EVENT_PL0     (0x01 < 6) /* 01B < 6 */
00172 #define P5_CNT_EVENT_PL3     (0x02 < 6) /* 10B < 6 */
00173 #define P5_CNT_EVENT         (0x03 < 6) /* 11B < 6 */
00174 #define P5_CNT_CLOCKS_PL0    (0x05 < 6) /* 101B < 6 */
00175 #define P5_CNT_CLOCKS_PL3    (0x06 < 6) /* 110B < 6 */
00176 #define P5_CNT_CLOCKS        (0x07 < 6) /* 111B < 6 */
00177
00178
00179 #else
00180 #if defined CPU_P6
00181 /* PPro/PII/PIII section */
00182
00183 /*-
00184  * Copyright (c) 1997 The President and Fellows of Harvard College.
00185  * All rights reserved.
00186  * Copyright (c) 1997 Aaron B. Brown.
00187  *
00188  * Redistribution and use in source and binary forms, with or without
00189  * modification, are permitted provided that the following conditions
00190  * are met:
00191  * 1. Redistributions of source code must retain the above copyright
00192  *   notice, this list of conditions and the following disclaimer.
00193  * 2. Redistributions in binary form must reproduce the above copyright
00194  *   notice, this list of conditions and the following disclaimer in the
00195  *   documentation and/or other materials provided with the distribution.
00196  * 3. All advertising materials mentioning features or use of this software
00197  *   must display the following acknowledgement:
00198  *       This product includes software developed by Harvard University
00199  *       and its contributors.
00200  * 4. Neither the name of the University nor the names of its contributors
00201  *   may be used to endorse or promote products derived from this software
00202  *   without specific prior written permission.
00203  *
00204  * THIS SOFTWARE IS PROVIDED BY HARVARD AND CONTRIBUTORS ``AS IS" AND
00205  * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
00206  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
00207  * ARE DISCLAIMED. IN NO EVENT SHALL HARVARD UNIVERSITY OR CONTRIBUTORS BE
00208  * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
00209  * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF

```

```

00210 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
00211 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00212 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
00213 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
00214 * POSSIBILITY OF SUCH DAMAGE.
00215 */
00216
00217 /*****
00218 ** Symbolic names for counter numbers (used in select_p6counter()) **
00219 *****/
00220 *
00221 * These correspond in order to the Pentium Pro counters. Add new counters at
00222 * the end. These agree with the mnemonics in the Pentium Pro Family
00223 * Developer's Manual, vol 3.
00224 *
00225 * Those events marked with a $ require a MESI unit field; those marked with
00226 * a @ require a self/any unit field. Those marked with a 0 are only supported
00227 * in counter 0; those marked with 1 are only supported in counter 1.
00228 */
00229
00230 /* Data cache unit */
00231 #define P6_DATA_MEM_REFS 0x43 /* total memory refs */
00232 #define P6_DCU_LINES_IN 0x45 /* all lines allocated in cache unit */
00233 #define P6_DCU_M_LINES_IN 0x46 /* M lines allocated in cache unit */
00234 #define P6_DCU_M_LINES_OUT 0x47 /* M lines evicted from cache */
00235 #define P6_DCU_MISS_OUTSTANDING 0x48 /* #cycles a miss is outstanding */
00236
00237 /* Instruction fetch unit */
00238 #define P6_IFU_IFETCH 0x80 /* instruction fetches */
00239 #define P6_IFU_IFETCH_MISS 0x81 /* instruction fetch misses */
00240 #define P6_ITLB_MISS 0x85 /* ITLB misses */
00241 #define P6_IFU_MEM_STALL 0x86 /* number of cycles IFU is stalled */
00242 #define P6_ILD_STALL 0x87 /* #stalls in instr length decode */
00243
00244 /* L2 Cache */
00245 #define P6_L2_IFETCH 0x28 /* ($) 12 ifetches */
00246 #define P6_L2_LD 0x29 /* ($) 12 data loads */
00247 #define P6_L2_ST 0x2a /* ($) 12 data stores */
00248 #define P6_L2_LINES_IN 0x24 /* lines allocated in L2 */
00249 #define P6_L2_LINES_OUT 0x26 /* lines removed from L2 */
00250 #define P6_L2_M_LINES_INM 0x25 /* modified lines allocated in L2 */
00251 #define P6_L2_M_LINES_OUTM 0x27 /* modified lines removed from L2 */
00252 #define P6_L2_RQSTS 0x2e /* ($) number of l2 requests */
00253 #define P6_L2_ADS 0x21 /* number of l2 addr strobes */
00254 #define P6_L2_DBUS_BUSY 0x22 /* number of data bus busy cycles */
00255 #define P6_L2_DBUS_BUSY_RD 0x23 /* #bus cycles xferring l2->cpu */
00256
00257 /* External bus logic */
00258 #define P6_BUS_DRDY_CLOCKS 0x62 /* (@) #clocks DRDY is asserted */
00259 #define P6_BUS_LOCK_CLOCKS 0x63 /* (@) #clocks LOCK is asserted */
00260 #define P6_BUS_REQ_OUTSTANDING 0x60 /* #bus requests outstanding */
00261 #define P6_BUS_TRAN_BRD 0x65 /* (@) bus burst read txns */
00262 #define P6_BUS_TRAN_RFO 0x66 /* (@) bus read for ownership txns */
00263 #define P6_BUS_TRAN_WB 0x67 /* (@) bus writeback txns */
00264 #define P6_BUS_TRAN_IFETCH 0x68 /* (@) bus instr fetch txns */
00265 #define P6_BUS_TRAN_INVALID 0x69 /* (@) bus invalidate txns */
00266 #define P6_BUS_TRAN_PWR 0x6a /* (@) bus partial write txns */
00267 #define P6_BUS_TRANS_P 0x6b /* (@) bus partial txns */
00268 #define P6_BUS_TRANS_IO 0x6c /* (@) bus I/O txns */
00269 #define P6_BUS_TRAN_DEF 0x6d /* (@) bus deferred txns */
00270 #define P6_BUS_TRAN_BURST 0x6e /* (@) bus burst txns */
00271 #define P6_BUS_TRAN_ANY 0x70 /* (@) total bus txns */
00272 #define P6_BUS_TRAN_MEM 0x6f /* (@) total memory txns */
00273 #define P6_BUS_DATA_RCV 0x64 /* #busclocks CPU is receiving data */
00274 #define P6_BUS_BNR_DRV 0x61 /* #busclocks CPU is driving BNR pin */
00275 #define P6_BUS_HIT_DRV 0x7a /* #busclocks CPU is driving HIT pin */
00276 #define P6_BUS_HITM_DRV 0x7b /* #busclocks CPU is driving HITM pin */
00277 #define P6_BUS_SNOOP_STALL 0x7e /* #clkcycles bus is snoop-stalled */
00278
00279 /* FPU */
00280 #define P6_FLOPS 0xc1 /* (0) number of FP ops retired */
00281 #define P6_FP_COMP_OPS 0x10 /* (0) computational FPOPS exec'd */
00282 #define P6_FP_ASSIST 0x11 /* (1) FP excep's handled in ucode */
00283 #define P6_MUL 0x12 /* (1) number of FP multiplies */
00284 #define P6_DIV 0x13 /* (1) number of FP divides */
00285 #define P6_CYCLES_DIV_BUSY 0x14 /* (0) number of cycles divider busy */
00286
00287 /* Memory ordering */
00288 #define P6_LD_BLOCKS 0x03 /* number of store buffer blocks */
00289 #define P6_SB_DRAINS 0x04 /* # of store buffer drain cycles */
00290 #define P6_MISALING_MEM_REF 0x05 /* # misaligned data memory refs */
00291
00292 /* Instruction decoding and retirement */
00293 #define P6_INST_RETIRED 0xc0 /* number of instrs retired */
00294 #define P6_OPS_RETIRED 0xc2 /* number of micro-ops retired */
00295 #define P6_INST_DECODER 0xd0 /* number of instructions decoded */
00296

```

```

00297 /* Interrupts */
00298 #define P6_HW_INT_RX 0xc8 /* number of hardware interrupts */
00299 #define P6_CYCLES_INT_MASKED 0xc6 /* number of cycles hardints masked */
00300 #define P6_CYCLES_INT_PENDING_AND_MASKED 0xc7 /* #cycles masked but pending */
00301
00302 /* Branches */
00303 #define P6_BR_INST_RETIRED 0xc4 /* number of branch instrs retired */
00304 #define P6_BR_MISS_PRED_RETIRED 0xc5 /* number of mispred'd brs retired */
00305 #define P6_BR_TAKEN_RETIRED 0xc9 /* number of taken branches retired */
00306 #define P6_BR_MISS_PRED_TAKEN_RET 0xca /* #taken mispredictions br's retired */
00307 #define P6_BR_INST_DECODED 0xe0 /* number of branch instrs decoded */
00308 #define P6_BTBMISSES 0xe2 /* # of branches that missed in BTB */
00309 #define P6_BR_BOGUS 0xe4 /* number of bogus branches */
00310 #define P6_BACLEAR 0xe6 /* # times BACLEAR is asserted */
00311
00312 /* Stalls */
00313 #define P6_RESOURCE_STALLS 0xa2 /* # resource-related stall cycles */
00314 #define P6_PARTIAL_RAT_STALLS 0xd2 /* # cycles/events for partial stalls */
00315
00316 /* Segment register loads */
00317 #define P6_SEGMENT_REG_LOADS 0x06 /* number of segment register loads */
00318
00319 /* Clocks */
00320 #define P6_CPU_CLK_UNHALTED 0x79 /* #clocks CPU is not halted */
00321
00322 /* Unit field tags */
00323 #define P6_UNIT_M 0x0800
00324 #define P6_UNIT_E 0x0400
00325 #define P6_UNIT_S 0x0200
00326 #define P6_UNIT_I 0x0100
00327 #define P6_UNIT_MESI 0x0f00
00328
00329 #define P6_UNIT_SELF 0x0000
00330 #define P6_UNIT_ANY 0x2000
00331
00332 /*****
00333  ** Flag bit definitions (used for the 'flag' field in select_p6counter()) **
00334  *****/
00335 *
00336 * The driver accepts fully-formed counter specifications from user-level.
00337 * The following flags are mnemonics for the bits that get set in the
00338 * PerfEvtSel0 and PerfEvtSel1 MSR's
00339 *
00340 */
00341 #define P6CNT_U 0x010000 /* Monitor user-level events */
00342 #define P6CNT_K 0x020000 /* Monitor kernel-level events */
00343 #define P6CNT_E 0x040000 /* Edge detect: count state transitions */
00344 #define P6CNT_PC 0x080000 /* Pin control: ?? */
00345 #define P6CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
00346 #define P6CNT_F 0x200000 /* Freeze counter (handled in software) */
00347 #define P6CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
00348 #define P6CNT_IV 0x800000 /* Invert counter mask comparison result */
00349
00350 /*****
00351  ** Miscellaneous constants **
00352  *****/
00353 *
00354 * Number of Pentium Pro programable hardware counters.
00355 */
00356 #define NUM_P6HWC 2
00357
00358 /*****
00359  *
00360  * End of Copyright by Harvard College
00361  *
00362  *****/
00363
00364
00365 #define MSR_P6_EVTSEL0 0x186
00366 #define MSR_P6_EVTSEL1 0x187
00367 #define MSR_P6_PERFCTR0 0xc1
00368 #define MSR_P6_PERFCTR1 0xc2
00369
00370 /* P6-specific Makros to manipulate and read counters */
00371
00372 /* Read the 40 bit performance monitoring counter. This requires
00373 the PCE-flag in CR4 to be set. Otherwise GP0 is raised. Works only
00374 at P6.
00375 */
00376 #define l4_i686_rdpmc(cntnr, res_p) \
00377 __asm __volatile( \
00378 "mov %2, %%rcx # put counter number in \n\
00379 .byte 0xf; .byte 0x33 # RDPMC instruction \n\
00380 mov %%rdx, %1 # High order 32 bits \n\
00381 mov %%rax, %0 # Low order 32 bits" \
00382 : "=g" (*(int *) (res_p)), "=g" (((int *) res_p)+1)) \
00383 : "g" (cntnr) \

```

```

00384 : "ecx", "eax", "edx")
00385
00386 static inline l4_uint32_t l4_i686_rdpmc_32(int cnt){
00387     l4_uint32_t x;
00388
00389     __asm__ __volatile__(
00390         ".byte 0xf; .byte 0x33 # RDPMC instruction"
00391         : "=a" (x)
00392         : "c" (cnt)
00393         : "rcx", "rax", "rdx");
00394     return x;
00395 }
00396
00397 static inline void l4_i686_select_perfctr_event(int counter,
00398     unsigned long long val){
00399     l4_i586_wrmsr(MSR_P6_EVNTSEL0+counter, &val);
00400 }
00401
00402 static inline void l4_i686_select_perfctr0_event(long long *val){
00403     asm volatile(
00404         "mov $MSR_P6_EVNTSEL0, %%rcx\n"
00405         "mov (%%rbx), %%rax\n"
00406         "mov 4(%%rbx), %%rdx\n"
00407         /* ".byte 0xcc, 0xeb, 0x01, 0x21\n"
00408         ".byte 0x0f, 0x30\n" // wrmsr
00409         /* ".byte 0xcc, 0xeb, 0x01, 0x21\n"
00410         : /* no output */
00411         : "b" (val)
00412         : "ax", "cx", "dx", "bx"
00413         );
00414
00415 }
00416
00417 /* end of P6 section */
00418 #else
00419
00420 #define K7CNT_U 0x010000 /* Monitor user-level events */
00421 #define K7CNT_K 0x020000 /* Monitor kernel-level events */
00422 #define K7CNT_E 0x040000 /* Edge detect: count state transitions */
00423 #define K7CNT_PC 0x080000 /* Pin control: ?? */
00424 #define K7CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
00425 #define K7CNT_F 0x200000 /* Freeze counter (handled in software) */
00426 #define K7CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
00427 #define K7CNT_IV 0x800000 /* Invert counter mask comparison result */
00428
00429 #define MSR_K7_EVNTSEL0 0xC0010000
00430 #define MSR_K7_EVNTSEL1 0xC0010001
00431 #define MSR_K7_EVNTSEL2 0xC0010002
00432 #define MSR_K7_EVNTSEL3 0xC0010003
00433 #define MSR_K7_PERFCTR0 0xC0010004
00434 #define MSR_K7_PERFCTR1 0xC0010005
00435 #define MSR_K7_PERFCTR2 0xC0010006
00436 #define MSR_K7_PERFCTR3 0xC0010007
00437
00438 #endif
00439
00440 #endif
00441
00442 /* end of P5/P6/K7 section*/
00443 #endif
00444
00445 /* end of not only lib-prototypes section */
00446 #endif
00447
00448 EXTERN_C_END
00449
00450 #endif

```

## 16.46 x86/i4/util/perform.h File Reference

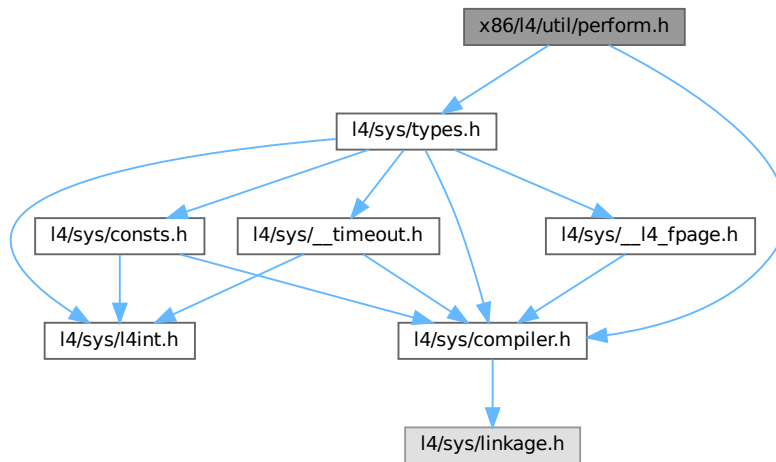
Performance Monitoring using P5/P6 Measurement Counters.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for perform.h:



### 16.46.1 Detailed Description

Performance Monitoring using P5/P6 Measurement Counters.

Define either `CPU_PENTIUM` or `CPU_P6`

Definition in file [perform.h](#).

## 16.47 perform.h

[Go to the documentation of this file.](#)

```

00001
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00010  *               Lars Reuther <reuther@os.inf.tu-dresden.de>
00011  *               economic rights: Technische Universität Dresden (Germany)
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU Lesser General Public License 2.1.
00014  * Please see the COPYING-LGPL-2.1 file for details.
00015  */
00016
00017 #ifndef __L4UTIL_PERFORM_H
00018 #define __L4UTIL_PERFORM_H
00019
00020 #include <l4/sys/types.h>
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00025 extern const char*strp6pmc_event(l4_uint32_t event);
00026
00027 #ifndef CONFIG_PERFORM_ONLY_PROTOTYPES
00028
00029 #if ! (defined CPU_PENTIUM ^ defined CPU_P6 ^ defined CPU_K7)
00030
00031 #error You must define your target architecture.
00032 #error Define EITHER CPU_PENTIUM for Intel Pentium or CPU_P6 for Intel PPro/PII/PIII.
00033
00034 #else
00035

```

```

00036 /* P5/P6/K7 section */
00037
00038 /* Makros for access to model specific registers (MSR) */
00039
00040 /* Write the 64-Bit Model Specific Register. First argument is the register,
00041    second the 64-Bit value. This can only be called at privilege level 0.
00042    With L4, the kernel emulates the WRMSR when calling in PL 3.
00043    */
00044 static inline void l4_i586_wrmsr(unsigned reg,unsigned long long*val){
00045     unsigned long dummyeax, dummyecx, dummyedx;
00046
00047     asm volatile(
00048         ".byte 0xf; .byte 0x30\n" /* wrmsr */
00049         : "=a" (dummyeax), "=d" (dummyedx), "=c" (dummyecx)
00050         : "2" (reg), "0" (*(unsigned *)val), "1" (*(unsigned *)val+1))
00051     );
00052 }
00053
00054 /* Read the 64-Bit Model Specific Register. First argument is the register,
00055    second the address to a 64-Bit value. This can only be called at
00056    privilege level 0. With L4, the kernel emulates the RDMSR when calling
00057    in PL 3.
00058    */
00059 static inline void l4_i586_rdmsr(unsigned reg,unsigned long long*val){
00060     unsigned dummy;
00061
00062     asm volatile(
00063         ".byte 0xf; .byte 0x32\n" /* rdmsr */
00064         : "=a" (*(unsigned *)val), "=d" (*(unsigned *)val+1), "=c" (dummy)
00065         : "2" (reg)
00066     );
00067 }
00068
00069
00070 #ifdef CPU_PENTIUM
00071 /* Pentium section */
00072
00073 /* functions and events defined here are only usable at Pentium
00074    Processors. P6 architecture does NOT support this kind of measuring and
00075    these events. P6 architecture has its own counters and its own events.
00076    See P6-section for details. */
00077
00078 /* from l4linux/arch/l4-i386/include/perform.h */
00079
00080 static inline void
00081 l4_i586_reset_event_counter(void){
00082     asm volatile("xor %%eax, %%eax\n"
00083         "xor %%edx, %%edx\n"
00084         "movl $0x12, %%ecx\n"
00085         ".byte 0x0f, 0x30\n"
00086         "movl $0x13, %%ecx\n"
00087         ".byte 0x0f, 0x30\n"
00088         : : : "cx", "ax", "dx"
00089     );
00090 };
00091
00092 static inline void
00093 l4_i586_read_event_counter_long(long long *counter0, long long *counter1)
00094 {
00095     asm volatile(
00096         /* "movl $0, %%eax\n"
00097         "movl $0x11, %%ecx\n"
00098         ".byte 0x0f, 0x30\n" */ /* stop event counting */
00099         "movl $0x12, %%ecx\n"
00100         ".byte 0x0f, 0x32\n"
00101         "movl %%eax, (%%ebx)\n"
00102         "movl %%edx, 4(%%ebx)\n"
00103         "movl $0x13, %%ecx\n"
00104         ".byte 0x0f, 0x32\n"
00105         "movl %%eax, (%%esi)\n"
00106         "movl %%edx, 4(%%esi)\n"
00107         : /* no output */
00108         : "b" (counter0), "S" (counter1)
00109         : "ax", "cx", "dx"
00110     );
00111 }
00112
00113 static inline void
00114 l4_i586_read_event_counter(int *counter0, int *counter1)
00115 {
00116     asm volatile("pushl %%edx\n"
00117         ".byte 0x0f, 0x30\n"
00118         "movl $0x12, %%ecx\n"
00119         ".byte 0x0f, 0x32\n"
00120         "movl %%eax, %%ebx\n"
00121         "movl $0x13, %%ecx\n"
00122         ".byte 0x0f, 0x32\n"

```

```

00123         "popl  %%edx\n"
00124         : "=b" (*counter0), "a" (*counter1)
00125         : "1" (0), "c" (0x11)
00126         );
00127     }
00128
00129     static inline void
00130     l4_i586_select_event(int event0, int event1)
00131     {
00132         asm volatile(".byte 0x0f, 0x30\n"
00133             :
00134             :
00135             "a" (event0 + (event1 < 16)),
00136             "d" (0),
00137             "c" (0x11)
00138             );
00139     };
00140
00141     #define P5_RD_MISS          0x003 /* 000011B */
00142     #define P5_WR_MISS          0x008 /* 000100B */
00143     #define P5_RW_MISS          0x029 /* 101001B */
00144     #define P5_EX_MISS          0x00e /* 001110B */
00145
00146     #define P5_D_WBACK          0x006 /* 000110B */
00147
00148     #define P5_RW_TLB           0x002 /* 00010B */
00149     #define P5_EX_TLB           0x00d /* 01101B */
00150
00151     #define P5_A_STALL           0x01f /* 11111B */
00152     #define P5_W_STALL           0x019 /* 11001B */
00153     #define P5_R_STALL           0x01a /* 11010B */
00154     #define P5_X_STALL           0x01b /* 11011B */
00155
00156     #define P5_AGI_STALL         0x01f /* 11111B */
00157
00158     #define P5_PIPELINE_FLUSH    0x015 /* 10101B */
00159
00160     #define P5_NON_CACHE_RD      0x01e /* 11110B */
00161     #define P5_NCACHE_REFS       0x01e /* 11110B */
00162     #define P5_LOCKED_BUS        0x01c /* 11100B */
00163
00164     #define P5_MEM2PIPE          0x009 /* 01001B */
00165     #define P5_BANK_CONF         0x00a /* 01010B */
00166
00167
00168     #define P5_INSTRS_EX          0x016 /* 10110B */
00169     #define P5_INSTRS_EX_V       0x017 /* 10111B */
00170
00171
00172     #define P5_CNT_NOTHING        (0x00 < 6) /* 00B < 6 */
00173     #define P5_CNT_EVENT_PL0      (0x01 < 6) /* 01B < 6 */
00174     #define P5_CNT_EVENT_PL3      (0x02 < 6) /* 10B < 6 */
00175     #define P5_CNT_EVENT          (0x03 < 6) /* 11B < 6 */
00176     #define P5_CNT_CLOCKS_PL0     (0x05 < 6) /* 101B < 6 */
00177     #define P5_CNT_CLOCKS_PL3     (0x06 < 6) /* 110B < 6 */
00178     #define P5_CNT_CLOCKS         (0x07 < 6) /* 111B < 6 */
00179
00180
00181     #else
00182     #if defined CPU_P6
00183     /* PPro/PII/PIII section */
00184
00185     /*-
00186     * Copyright (c) 1997 The President and Fellows of Harvard College.
00187     * All rights reserved.
00188     * Copyright (c) 1997 Aaron B. Brown.
00189     *
00190     * Redistribution and use in source and binary forms, with or without
00191     * modification, are permitted provided that the following conditions
00192     * are met:
00193     * 1. Redistributions of source code must retain the above copyright
00194     *   notice, this list of conditions and the following disclaimer.
00195     * 2. Redistributions in binary form must reproduce the above copyright
00196     *   notice, this list of conditions and the following disclaimer in the
00197     *   documentation and/or other materials provided with the distribution.
00198     * 3. All advertising materials mentioning features or use of this software
00199     *   must display the following acknowledgement:
00200     *       This product includes software developed by Harvard University
00201     *       and its contributors.
00202     * 4. Neither the name of the University nor the names of its contributors
00203     *   may be used to endorse or promote products derived from this software
00204     *   without specific prior written permission.
00205     *
00206     * THIS SOFTWARE IS PROVIDED BY HARVARD AND CONTRIBUTORS ``AS IS'' AND
00207     * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
00208     * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
00209     * ARE DISCLAIMED.  IN NO EVENT SHALL HARVARD UNIVERSITY OR CONTRIBUTORS BE

```

```

00210 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
00211 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
00212 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
00213 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00214 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
00215 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
00216 * POSSIBILITY OF SUCH DAMAGE.
00217 */
00218
00219 /*****
00220 ** Symbolic names for counter numbers (used in select_p6counter()) **
00221 *****/
00222 *
00223 * These correspond in order to the Pentium Pro counters. Add new counters at
00224 * the end. These agree with the mnemonics in the Pentium Pro Family
00225 * Developer's Manual, vol 3.
00226 *
00227 * Those events marked with a $ require a MESI unit field; those marked with
00228 * a @ require a self/any unit field. Those marked with a 0 are only supported
00229 * in counter 0; those marked with 1 are only supported in counter 1.
00230 */
00231
00232 /* Data cache unit */
00233 #define P6_DATA_MEM_REFS 0x43 /* total memory refs */
00234 #define P6_DCU_LINES_IN 0x45 /* all lines allocated in cache unit */
00235 #define P6_DCU_M_LINES_IN 0x46 /* M lines allocated in cache unit */
00236 #define P6_DCU_M_LINES_OUT 0x47 /* M lines evicted from cache */
00237 #define P6_DCU_MISS_OUTSTANDING 0x48 /* #cycles a miss is outstanding */
00238
00239 /* Instruction fetch unit */
00240 #define P6_IFU_IFETCH 0x80 /* instruction fetches */
00241 #define P6_IFU_IFETCH_MISS 0x81 /* instruction fetch misses */
00242 #define P6_ITLB_MISS 0x85 /* ITLB misses */
00243 #define P6_IFU_MEM_STALL 0x86 /* number of cycles IFU is stalled */
00244 #define P6_ILD_STALL 0x87 /* #stalls in instr length decode */
00245
00246 /* L2 Cache */
00247 #define P6_L2_IFETCH 0x28 /* ($) l2 ifetches */
00248 #define P6_L2_LD 0x29 /* ($) l2 data loads */
00249 #define P6_L2_ST 0x2a /* ($) l2 data stores */
00250 #define P6_L2_LINES_IN 0x24 /* lines allocated in l2 */
00251 #define P6_L2_LINES_OUT 0x26 /* lines removed from l2 */
00252 #define P6_L2_M_LINES_INM 0x25 /* modified lines allocated in l2 */
00253 #define P6_L2_M_LINES_OUTM 0x27 /* modified lines removed from l2 */
00254 #define P6_L2_RQSTS 0x2e /* ($) number of l2 requests */
00255 #define P6_L2_ADS 0x21 /* number of l2 addr strobes */
00256 #define P6_L2_DBUS_BUSY 0x22 /* number of data bus busy cycles */
00257 #define P6_L2_DBUS_BUSY_RD 0x23 /* #bus cycles xferring l2->cpu */
00258
00259 /* External bus logic */
00260 #define P6_BUS_DRDY_CLOCKS 0x62 /* (@) #clocks DRDY is asserted */
00261 #define P6_BUS_LOCK_CLOCKS 0x63 /* (@) #clocks LOCK is asserted */
00262 #define P6_BUS_REQ_OUTSTANDING 0x60 /* #bus requests outstanding */
00263 #define P6_BUS_TRAN_BRD 0x65 /* (@) bus burst read txns */
00264 #define P6_BUS_TRAN_RFO 0x66 /* (@) bus read for ownership txns */
00265 #define P6_BUS_TRAN_WB 0x67 /* (@) bus writeback txns */
00266 #define P6_BUS_TRAN_IFETCH 0x68 /* (@) bus instr fetch txns */
00267 #define P6_BUS_TRAN_INVALID 0x69 /* (@) bus invalidate txns */
00268 #define P6_BUS_TRAN_PWR 0x6a /* (@) bus partial write txns */
00269 #define P6_BUS_TRANS_P 0x6b /* (@) bus partial txns */
00270 #define P6_BUS_TRANS_IO 0x6c /* (@) bus I/O txns */
00271 #define P6_BUS_TRAN_DEF 0x6d /* (@) bus deferred txns */
00272 #define P6_BUS_TRAN_BURST 0x6e /* (@) bus burst txns */
00273 #define P6_BUS_TRAN_ANY 0x70 /* (@) total bus txns */
00274 #define P6_BUS_TRAN_MEM 0x6f /* (@) total memory txns */
00275 #define P6_BUS_DATA_RCV 0x64 /* #busclocks CPU is receiving data */
00276 #define P6_BUS_BNR_DRV 0x61 /* #busclocks CPU is driving BNR pin */
00277 #define P6_BUS_HIT_DRV 0x7a /* #busclocks CPU is driving HIT pin */
00278 #define P6_BUS_HITM_DRV 0x7b /* #busclocks CPU is driving HITM pin */
00279 #define P6_BUS_SNOOP_STALL 0x7e /* #clkcycles bus is snoop-stalled */
00280
00281 /* FPU */
00282 #define P6_FLOPS 0xc1 /* (0) number of FP ops retired */
00283 #define P6_FP_COMP_OPS 0x10 /* (0) computational FPOPS exec'd */
00284 #define P6_FP_ASSIST 0x11 /* (1) FP excep's handled in ucode */
00285 #define P6_MUL 0x12 /* (1) number of FP multiplies */
00286 #define P6_DIV 0x13 /* (1) number of FP divides */
00287 #define P6_CYCLES_DIV_BUSY 0x14 /* (0) number of cycles divider busy */
00288
00289 /* Memory ordering */
00290 #define P6_LD_BLOCKS 0x03 /* number of store buffer blocks */
00291 #define P6_SB_DRAINS 0x04 /* # of store buffer drain cycles */
00292 #define P6_MISALING_MEM_REF 0x05 /* # misaligned data memory refs */
00293
00294 /* Instruction decoding and retirement */
00295 #define P6_INST_RETIRED 0xc0 /* number of instrs retired */
00296 #define P6_UOPS_RETIRED 0xc2 /* number of micro-ops retired */

```



```

00297 #define P6_INSTDECODER 0xd0 /* number of instructions decoded */
00298
00299 /* Interrupts */
00300 #define P6_HW_INT_RX 0xc8 /* number of hardware interrupts */
00301 #define P6_CYCLES_INT_MASKED 0xc6 /* number of cycles hardints masked */
00302 #define P6_CYCLES_INT_PENDING_AND_MASKED 0xc7 /* #cycles masked but pending */
00303
00304 /* Branches */
00305 #define P6_BR_INST_RETIRED 0xc4 /* number of branch instrs retired */
00306 #define P6_BR_MISS_PRED_RETIRED 0xc5 /* number of mispred'd brs retired */
00307 #define P6_BR_TAKEN_RETIRED 0xc9 /* number of taken branches retired */
00308 #define P6_BR_MISS_PRED_TAKEN_RET 0xca /* #taken mispredictions br's retired*/
00309 #define P6_BR_INST_DECODED 0xe0 /* number of branch instrs decoded */
00310 #define P6_BT_B_MISSES 0xe2 /* # of branches that missed in BTB */
00311 #define P6_BR_BOGUS 0xe4 /* number of bogus branches */
00312 #define P6_BACLEAR 0xe6 /* # times BACLEAR is asserted */
00313
00314 /* Stalls */
00315 #define P6_RESOURCE_STALLS 0xa2 /* # resource-related stall cycles */
00316 #define P6_PARTIAL_RAT_STALLS 0xd2 /* # cycles/events for partial stalls*/
00317
00318 /* Segment register loads */
00319 #define P6_SEGMENT_REG_LOADS 0x06 /* number of segment register loads */
00320
00321 /* Clocks */
00322 #define P6_CPU_CLK_UNHALTED 0x79 /* #clocks CPU is not halted */
00323
00324 /* Unit field tags */
00325 #define P6_UNIT_M 0x0800
00326 #define P6_UNIT_E 0x0400
00327 #define P6_UNIT_S 0x0200
00328 #define P6_UNIT_I 0x0100
00329 #define P6_UNIT_MESI 0x0f00
00330
00331 #define P6_UNIT_SELF 0x0000
00332 #define P6_UNIT_ANY 0x2000
00333
00334 /*****
00335  ** Flag bit definitions (used for the 'flag' field in select_p6counter()) **
00336  *****/
00337 *
00338 * The driver accepts fully-formed counter specifications from user-level.
00339 * The following flags are mnemonics for the bits that get set in the
00340 * PerfEvtSel0 and PerfEvtSel1 MSR's
00341 *
00342 */
00343 #define P6CNT_U 0x010000 /* Monitor user-level events */
00344 #define P6CNT_K 0x020000 /* Monitor kernel-level events */
00345 #define P6CNT_E 0x040000 /* Edge detect: count state transitions */
00346 #define P6CNT_PC 0x080000 /* Pin control: ?? */
00347 #define P6CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
00348 #define P6CNT_F 0x200000 /* Freeze counter (handled in software) */
00349 #define P6CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
00350 #define P6CNT_IV 0x800000 /* Invert counter mask comparison result */
00351
00352 /*****
00353  ** Miscellaneous constants **
00354  *****/
00355 *
00356 * Number of Pentium Pro programable hardware counters.
00357 */
00358 #define NUM_P6HWC 2
00359
00360 /*****
00361  **
00362  * End of Copyright by Harvard College
00363  **
00364  *****/
00365
00366
00367 #define MSR_P6_EVTSEL0 0x186
00368 #define MSR_P6_EVTSEL1 0x187
00369 #define MSR_P6_PERFCTR0 0xc1
00370 #define MSR_P6_PERFCTR1 0xc2
00371
00372 /* P6-specific Makros to manipulate and read counters */
00373
00374 /* Read the 40 bit performance monitoring counter. This requires
00375 the PCE-flag in CR4 to be set. Otherwise GP0 is raised. Works only
00376 at P6.
00377 */
00378 #define l4_i686_rdpmmc(cntnr, res_p) \
00379 __asm __volatile( \
00380 "movl %2, %%ecx # put counter number in \n\ \
00381 .byte 0xf; .byte 0x33 # RDPMMC instruction \n\ \
00382 movl %%edx, %1 # High order 32 bits \n\ \
00383 movl %%eax, %0 # Low order 32 bits" \

```

```

00384 : "=g" (*(int *) (res_p)), "=g" (*((int *)res_p)+1)) \
00385 : "g" (cntr) \
00386 : "ecx", "eax", "edx")
00387
00388 static inline l4_uint32_t l4_i686_rdpmc_32(int cntr){
00389     l4_uint32_t x;
00390
00391     __asm__ __volatile__(
00392         ".byte 0xf; .byte 0x33 # RDPMC instruction"
00393         : "=a" (x)
00394         : "c" (cntr)
00395         : "ecx", "eax", "edx");
00396     return x;
00397 }
00398
00399 static inline void l4_i686_select_perfctr_event(int counter,
00400                                             unsigned long long val){
00401     l4_i586_wrmsr(MSR_P6_EVNTSEL0+counter, &val);
00402 }
00403
00404 static inline void l4_i686_select_perfctr0_event(long long *val){
00405     asm volatile(
00406         "movl $MSR_P6_EVNTSEL0, %%ecx\n"
00407         "movl (%%ebx), %%eax\n"
00408         "movl 4(%%ebx), %%edx\n"
00409         /* ".byte 0xcc, 0xeb, 0x01, 0x21\n"
00410          ".byte 0x0f, 0x30\n" // wrmsr
00411          /* ".byte 0xcc, 0xeb, 0x01, 0x21\n"
00412           : /* no output */
00413           : "b" (val)
00414           : "ax", "cx", "dx", "bx"
00415           );
00416
00417 }
00418
00419 /* end of P6 section */
00420 #else
00421
00422 #define K7CNT_U 0x010000 /* Monitor user-level events */
00423 #define K7CNT_K 0x020000 /* Monitor kernel-level events */
00424 #define K7CNT_E 0x040000 /* Edge detect: count state transitions */
00425 #define K7CNT_PC 0x080000 /* Pin control: ?? */
00426 #define K7CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
00427 #define K7CNT_F 0x200000 /* Freeze counter (handled in software) */
00428 #define K7CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
00429 #define K7CNT_IV 0x800000 /* Invert counter mask comparison result */
00430
00431 #define MSR_K7_EVNTSEL0 0xC0010000
00432 #define MSR_K7_EVNTSEL1 0xC0010001
00433 #define MSR_K7_EVNTSEL2 0xC0010002
00434 #define MSR_K7_EVNTSEL3 0xC0010003
00435 #define MSR_K7_PERFCTR0 0xC0010004
00436 #define MSR_K7_PERFCTR1 0xC0010005
00437 #define MSR_K7_PERFCTR2 0xC0010006
00438 #define MSR_K7_PERFCTR3 0xC0010007
00439
00440 #endif
00441
00442 #endif
00443
00444 /* end of P5/P6/K7 section */
00445 #endif
00446
00447 /* end of not only lib-prototypes section */
00448 #endif
00449
00450 EXTERN_C_END
00451
00452 #endif

```

## 16.48 amd64/l4/util/rdtsc.h File Reference

Timestamp counter related functions.

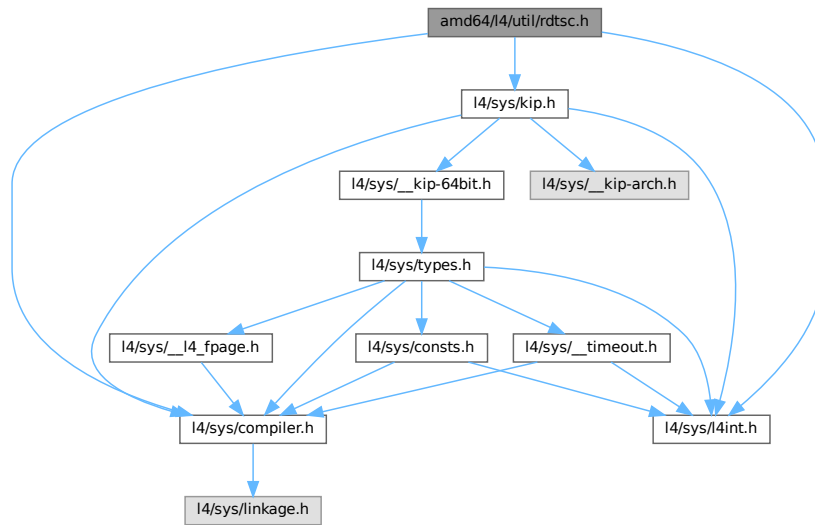
```

#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>

```

```
#include <l4/sys/kip.h>
```

Include dependency graph for rdtsc.h:



## Functions

- [l4\\_cpu\\_time\\_t l4\\_rdtsc](#) (void)  
*Read current value of CPU-internal timestamp counter.*
- [l4\\_uint32\\_t l4\\_rdtsc\\_32](#) (void)  
*Read the lest significant 32 bit of the TSC.*
- [l4\\_uint64\\_t l4\\_rdpmc](#) (int ecx)  
*Return current value of CPU-internal performance measurement counter.*
- [l4\\_uint32\\_t l4\\_rdpmc\\_32](#) (int ecx)  
*Return the least significant 32 bit of a performance counter.*
- [l4\\_uint64\\_t l4\\_tsc\\_to\\_ns](#) ([l4\\_cpu\\_time\\_t](#) tsc)  
*Convert timestamp to ns value.*
- [l4\\_uint64\\_t l4\\_tsc\\_to\\_us](#) ([l4\\_cpu\\_time\\_t](#) tsc)  
*Convert timestamp into micro seconds value.*
- void [l4\\_tsc\\_to\\_s\\_and\\_ns](#) ([l4\\_cpu\\_time\\_t](#) tsc, [l4\\_uint32\\_t](#) \*s, [l4\\_uint32\\_t](#) \*ns)  
*Convert timestamp to s.ns value.*
- [l4\\_cpu\\_time\\_t l4\\_ns\\_to\\_tsc](#) ([l4\\_uint64\\_t](#) ns)  
*Convert nano seconds into CPU ticks.*
- void [l4\\_busy\\_wait\\_ns](#) ([l4\\_uint64\\_t](#) ns)  
*Wait busy for a small amount of time.*
- void [l4\\_busy\\_wait\\_us](#) ([l4\\_uint64\\_t](#) us)  
*Wait busy for a small amount of time.*
- [l4\\_uint32\\_t l4\\_calibrate\\_tsc](#) ([l4\\_kernel\\_info\\_t](#) const \*kip)  
*Determine scalers for timestamp calculations.*
- [l4\\_uint32\\_t l4\\_tsc\\_init](#) ([l4\\_kernel\\_info\\_t](#) const \*kip)  
*Initialize scaler for TSC calibrations from the kernel.*
- [l4\\_uint32\\_t l4\\_get\\_hz](#) (void)  
*Get CPU frequency in Hz.*

## 16.48.1 Detailed Description

Timestamp counter related functions.

### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [rdtsc.h](#).

## 16.49 rdtsc.h

[Go to the documentation of this file.](#)

```

00001
00009 /*
00010  * (c) 2003-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00011  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00012  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00013  *      economic rights: Technische Universität Dresden (Germany)
00014  * This file is part of TUD:OS and distributed under the terms of the
00015  * GNU Lesser General Public License 2.1.
00016  * Please see the COPYING-LGPL-2.1 file for details.
00017  */
00018
00019 #ifndef __l4_rdtsc_h
00020 #define __l4_rdtsc_h
00021
00027 #include <l4/sys/compiler.h>
00028 #include <l4/sys/l4int.h>
00029 #include <l4/sys/kip.h>
00030
00031 EXTERN_C_BEGIN
00032
00033 /* interface */
00039 extern l4_uint32_t l4_scaler_tsc_to_ns;
00040 extern l4_uint32_t l4_scaler_tsc_to_us;
00041 extern l4_uint32_t l4_scaler_ns_to_tsc;
00042 extern l4_uint32_t l4_scaler_tsc_linux;
00043
00048 L4_INLINE l4_cpu_time_t
00049 l4_rdtsc (void);
00050
00056 L4_INLINE
00057 l4_uint32_t l4_rdtsc_32(void);
00058
00065 L4_INLINE l4_uint64_t
00066 l4_rdpmc (int ecx);
00067
00073 L4_INLINE
00074 l4_uint32_t l4_rdpmc_32(int ecx);
00075
00080 L4_INLINE l4_uint64_t
00081 l4_tsc_to_ns (l4_cpu_time_t tsc);
00082
00087 L4_INLINE l4_uint64_t
00088 l4_tsc_to_us (l4_cpu_time_t tsc);
00089
00095 L4_INLINE void
00096 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns);
00097
00103 L4_INLINE l4_cpu_time_t
00104 l4_ns_to_tsc (l4_uint64_t ns);
00105
00111 L4_INLINE void
00112 l4_busy_wait_ns (l4_uint64_t ns);
00113
00119 L4_INLINE void
00120 l4_busy_wait_us (l4_uint64_t us);
00121
00128 L4_INLINE l4_uint32_t
00129 l4_calibrate_tsc(l4_kernel_info_t const *kip);
00130
00144 L4_CV l4_uint32_t
00145 l4_tsc_init(l4_kernel_info_t const *kip);
00146

```

```

00151 L4_CV l4_uint32_t
00152 l4_get_hz (void);
00153
00156 EXTERN_C_END
00157
00158 /* implementation */
00159
00160 L4_INLINE l4_uint32_t
00161 l4_calibrate_tsc(l4_kernel_info_t const *kip)
00162 {
00163     return l4_tsc_init(kip);
00164 }
00165
00166 L4_INLINE l4_cpu_time_t
00167 l4_rdtsc (void)
00168 {
00169     l4_umword_t lo, hi;
00170
00171     __asm__ __volatile__ ("rdtsc" : "=a"(lo), "=d"(hi));
00172
00173     return ((l4_cpu_time_t)hi << 32) | lo;
00174 }
00175
00176 L4_INLINE l4_uint64_t
00177 l4_rdpmc (int ecx)
00178 {
00179     l4_umword_t lo, hi;
00180
00181     __asm__ __volatile__ ("rdpmc" : "=a"(lo), "=d"(hi) : "c"(ecx));
00182
00183     return ((l4_cpu_time_t)hi << 32) | lo;
00184 }
00185
00186 L4_INLINE
00187 l4_uint32_t l4_rdtsc_32(void)
00188 {
00189     l4_umword_t lo, hi;
00190
00191     __asm__ __volatile__ ("rdtsc" : "=a"(lo), "=d"(hi));
00192
00193     return lo;
00194 }
00195
00196 L4_INLINE
00197 l4_uint32_t l4_rdpmc_32(int ecx)
00198 {
00199     l4_umword_t lo, hi;
00200
00201     __asm__ __volatile__ ("rdpmc" : "=a"(lo), "=d"(hi) : "c"(ecx));
00202
00203     return lo;
00204 }
00205
00206 L4_INLINE l4_uint64_t
00207 l4_tsc_to_ns (l4_cpu_time_t tsc)
00208 {
00209     l4_uint64_t ns, dummy;
00210     __asm__
00211     ("
00212      \"mulq  %3      \n\t\"
00213      \"shrd  $27, %%rdx, %%rax      \n\t\"
00214      :\"=a\" (ns), \"=d\" (dummy)
00215      :\"a\" (tsc), \"r\" ((l4_uint64_t)l4_scaler_tsc_to_ns)
00216      );
00217     return ns;
00218 }
00219
00220 L4_INLINE l4_uint64_t
00221 l4_tsc_to_us (l4_cpu_time_t tsc)
00222 {
00223     l4_uint64_t ns, dummy;
00224     __asm__
00225     ("
00226      \"mulq  %3      \n\t\"
00227      \"shrd  $32, %%rdx, %%rax      \n\t\"
00228      :\"=a\" (ns), \"=d\" (dummy)
00229      :\"a\" (tsc), \"r\" ((l4_uint64_t)l4_scaler_tsc_to_us)
00230      );
00231     return ns;
00232 }
00233
00234 L4_INLINE void
00235 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns)
00236 {
00237     __asm__
00238     ("
00239      \"mulq  %3      \n\t\"

```

```

00240         "shrd $27, %%rdx, %%rax      \n\t"
00241         "xorq  %%rdx, %%rdx          \n\t"
00242         "divq  %4                    \n\t"
00243         : "=a" (*s), "=d" (*ns)
00244         : "a" (tsc), "r" ((l4_uint64_t)l4_scaler_tsc_to_ns),
00245         "rm"(1000000000ULL)
00246     );
00247 }
00248
00249 L4_INLINE l4_cpu_time_t
00250 l4_ns_to_tsc (l4_uint64_t ns)
00251 {
00252     l4_uint64_t tsc, dummy;
00253     __asm__
00254     (
00255         "mulq %3                    \n\t"
00256         "shrd $27, %%rdx, %%rax      \n\t"
00257         : "=a" (tsc), "=d" (dummy)
00258         : "a" (ns), "r" ((l4_uint64_t)l4_scaler_ns_to_tsc)
00259     );
00260     return tsc;
00261 }
00262
00263 L4_INLINE void
00264 l4_busy_wait_ns (l4_uint64_t ns)
00265 {
00266     l4_cpu_time_t stop = l4_rdtsc();
00267     stop += l4_ns_to_tsc(ns);
00268
00269     while (l4_rdtsc() < stop)
00270         ;
00271 }
00272
00273 L4_INLINE void
00274 l4_busy_wait_us (l4_uint64_t us)
00275 {
00276     l4_cpu_time_t stop = l4_rdtsc();
00277     stop += l4_ns_to_tsc(us*1000ULL);
00278
00279     while (l4_rdtsc() < stop)
00280         ;
00281 }
00282
00283 #endif /* __l4_rdtsc_h */
00284

```

## 16.50 x86/l4/util/rdtsc.h File Reference

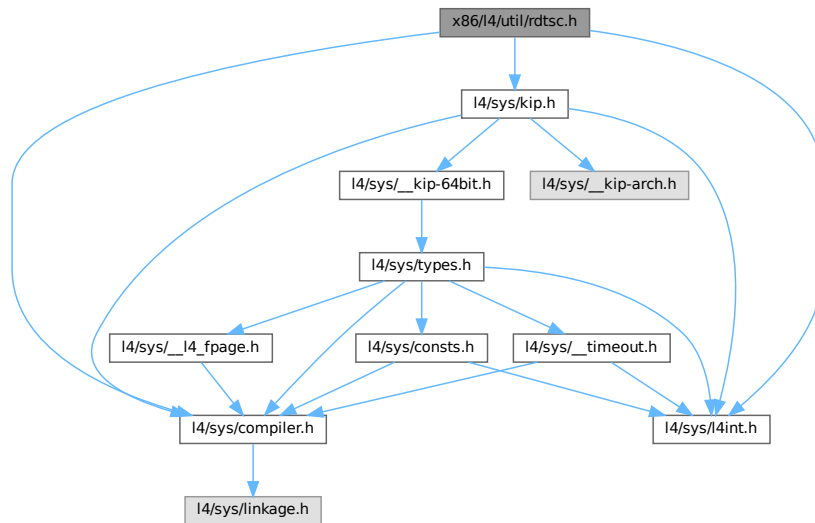
Timestamp counter related functions.

```

#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/kip.h>

```

Include dependency graph for rdtsc.h:



## Functions

- [l4\\_cpu\\_time\\_t l4\\_rdtsc](#) (void)  
*Read current value of CPU-internal timestamp counter.*
- [l4\\_uint32\\_t l4\\_rdtsc\\_32](#) (void)  
*Read the lest significant 32 bit of the TSC.*
- [l4\\_uint64\\_t l4\\_rdpmc](#) (int ecx)  
*Return current value of CPU-internal performance measurement counter.*
- [l4\\_uint32\\_t l4\\_rdpmc\\_32](#) (int ecx)  
*Return the least significant 32 bit of a performance counter.*
- [l4\\_uint64\\_t l4\\_tsc\\_to\\_ns](#) ([l4\\_cpu\\_time\\_t](#) tsc)  
*Convert timestamp to ns value.*
- [l4\\_uint64\\_t l4\\_tsc\\_to\\_us](#) ([l4\\_cpu\\_time\\_t](#) tsc)  
*Convert timestamp into micro seconds value.*
- void [l4\\_tsc\\_to\\_s\\_and\\_ns](#) ([l4\\_cpu\\_time\\_t](#) tsc, [l4\\_uint32\\_t](#) \*s, [l4\\_uint32\\_t](#) \*ns)  
*Convert timestamp to s.ns value.*
- [l4\\_cpu\\_time\\_t l4\\_ns\\_to\\_tsc](#) ([l4\\_uint64\\_t](#) ns)  
*Convert nano seconds into CPU ticks.*
- void [l4\\_busy\\_wait\\_ns](#) ([l4\\_uint64\\_t](#) ns)  
*Wait busy for a small amount of time.*
- void [l4\\_busy\\_wait\\_us](#) ([l4\\_uint64\\_t](#) us)  
*Wait busy for a small amount of time.*
- [l4\\_uint32\\_t l4\\_calibrate\\_tsc](#) ([l4\\_kernel\\_info\\_t](#) const \*kip)  
*Determine scalers for timestamp calculations.*
- [l4\\_uint32\\_t l4\\_tsc\\_init](#) ([l4\\_kernel\\_info\\_t](#) const \*kip)  
*Initialize scaler for TSC calibrations from the kernel.*
- [l4\\_uint32\\_t l4\\_get\\_hz](#) (void)  
*Get CPU frequency in Hz.*

## 16.50.1 Detailed Description

Timestamp counter related functions.

### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [rdtsc.h](#).

## 16.51 rdtsc.h

[Go to the documentation of this file.](#)

```

00001
00009 /*
00010  * (c) 2003-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00011  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00012  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00013  *      Jork Löser <jork@os.inf.tu-dresden.de>,
00014  *      Martin Pohlack <mp26@os.inf.tu-dresden.de>
00015  *      economic rights: Technische Universität Dresden (Germany)
00016  * This file is part of TUD:OS and distributed under the terms of the
00017  * GNU Lesser General Public License 2.1.
00018  * Please see the COPYING-LGPL-2.1 file for details.
00019  */
00020
00021 #ifndef __l4_rdtsc_h
00022 #define __l4_rdtsc_h
00023
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/l4int.h>
00031 #include <l4/sys/kip.h>
00032
00033 EXTERN_C_BEGIN
00034
00035 /* interface */
00041 extern l4_uint32_t l4_scaler_tsc_to_ns;
00042 extern l4_uint32_t l4_scaler_tsc_to_us;
00043 extern l4_uint32_t l4_scaler_ns_to_tsc;
00044 extern l4_uint32_t l4_scaler_tsc_linux;
00045
00050 L4_INLINE l4_cpu_time_t
00051 l4_rdtsc (void);
00052
00058 L4_INLINE
00059 l4_uint32_t l4_rdtsc_32(void);
00060
00067 L4_INLINE l4_uint64_t
00068 l4_rdpmc (int ecx);
00069
00075 L4_INLINE
00076 l4_uint32_t l4_rdpmc_32(int ecx);
00077
00082 L4_INLINE l4_uint64_t
00083 l4_tsc_to_ns (l4_cpu_time_t tsc);
00084
00089 L4_INLINE l4_uint64_t
00090 l4_tsc_to_us (l4_cpu_time_t tsc);
00091
00097 L4_INLINE void
00098 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns);
00099
00105 L4_INLINE l4_cpu_time_t
00106 l4_ns_to_tsc (l4_uint64_t ns);
00107
00113 L4_INLINE void
00114 l4_busy_wait_ns (l4_uint64_t ns);
00115
00121 L4_INLINE void
00122 l4_busy_wait_us (l4_uint64_t us);
00123
00130 L4_INLINE l4_uint32_t
00131 l4_calibrate_tsc(l4_kernel_info_t const *kip);
00132
00146 L4_CV l4_uint32_t

```



```

00147 l4_tsc_init(l4_kernel_info_t const *kip);
00148
00153 L4_CV l4_uint32_t
00154 l4_get_hz (void);
00155
00158 EXTERN_C_END
00159
00160 /* implementation */
00161
00162 L4_INLINE l4_uint32_t
00163 l4_calibrate_tsc(l4_kernel_info_t const *kip)
00164 {
00165     return l4_tsc_init(kip);
00166 }
00167
00168 L4_INLINE l4_cpu_time_t
00169 l4_rdtsc (void)
00170 {
00171     l4_cpu_time_t v;
00172
00173     __asm__ __volatile__ (
00174         ".byte 0x0f, 0x31\n\t"
00175         /*"rdtsc\n\t"*/
00176         :
00177         "=A" (v)
00178         : /* no inputs */
00179         );
00180
00181     return v;
00182 }
00183
00184
00185 /* the same, but only 32 bit. Useful for smaller differences,
00186     needs less cycles. */
00187 L4_INLINE
00188 l4_uint32_t l4_rdtsc_32(void)
00189 {
00190     l4_uint32_t x;
00191
00192     __asm__ __volatile__ (
00193         ".byte 0x0f, 0x31\n\t" // rdtsc
00194         : "=a" (x)
00195         :
00196         : "edx");
00197
00198     return x;
00199 }
00200
00201 L4_INLINE l4_uint64_t
00202 l4_rdpmc (int ecx)
00203 {
00204     l4_cpu_time_t v;
00205
00206     __asm__ __volatile__ (
00207         "rdpmc\n\t"
00208         :
00209         "=A" (v)
00210         : "c" (ecx)
00211         );
00212
00213     return v;
00214 }
00215
00216 /* the same, but only 32 bit. Useful for smaller differences,
00217     needs less cycles. */
00218 L4_INLINE
00219 l4_uint32_t l4_rdpmc_32(int ecx)
00220 {
00221     l4_uint32_t x;
00222
00223     __asm__ __volatile__ (
00224         "rdpmc\n\t"
00225         : "=a" (x)
00226         : "c" (ecx)
00227         : "edx");
00228
00229     return x;
00230 }
00231
00232 L4_INLINE l4_uint64_t
00233 l4_tsc_to_ns (l4_cpu_time_t tsc)
00234 {
00235     l4_uint32_t dummy;
00236     l4_uint64_t ns;
00237     __asm__ (
00238         ".byte 0x0f, 0x31\n\t"
00239         "movl %%edx, %%ecx\n\t"

```

```

00240     "mull    %3        \n\t"
00241     "movl    %%ecx, %%eax \n\t"
00242     "movl    %%edx, %%ecx \n\t"
00243     "mull    %3        \n\t"
00244     "addl    %%ecx, %%eax \n\t"
00245     "adcl    $0, %%edx \n\t"
00246     "shld    $5, %%eax, %%edx \n\t"
00247     "shll    $5, %%eax \n\t"
00248     : "=A" (ns),
00249     "=&c" (dummy)
00250     : "0" (tsc),
00251     "g" (l4_scaler_tsc_to_ns)
00252     );
00253     return ns;
00254 }
00255
00256 L4_INLINE l4_uint64_t
00257 l4_tsc_to_us (l4_cpu_time_t tsc)
00258 {
00259     l4_uint32_t dummy;
00260     l4_uint64_t us;
00261     __asm__
00262     (
00263     "movl    %%edx, %%ecx \n\t"
00264     "mull    %3        \n\t"
00265     "movl    %%ecx, %%eax \n\t"
00266     "movl    %%edx, %%ecx \n\t"
00267     "mull    %3        \n\t"
00268     "addl    %%ecx, %%eax \n\t"
00269     "adcl    $0, %%edx \n\t"
00270     : "=A" (us),
00271     "=&c" (dummy)
00272     : "0" (tsc),
00273     "g" (l4_scaler_tsc_to_us)
00274     );
00275     return us;
00276 }
00277
00278 L4_INLINE void
00279 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns)
00280 {
00281     l4_uint32_t dummy;
00282     __asm__
00283     (
00284     "movl    %%edx, %%ecx \n\t"
00285     "mull    %4        \n\t"
00286     "movl    %%ecx, %%eax \n\t"
00287     "movl    %%edx, %%ecx \n\t"
00288     "mull    %4        \n\t"
00289     "addl    %%ecx, %%eax \n\t"
00290     "adcl    $0, %%edx \n\t"
00291     "movl    $1000000000, %%ecx \n\t"
00292     "shld    $5, %%eax, %%edx \n\t"
00293     "shll    $5, %%eax \n\t"
00294     "divl    %%ecx \n\t"
00295     : "=a" (*s), "=d" (*ns), "=&c" (dummy)
00296     : "A" (tsc), "g" (l4_scaler_tsc_to_ns)
00297     );
00298 }
00299
00300 L4_INLINE l4_cpu_time_t
00301 l4_ns_to_tsc (l4_uint64_t ns)
00302 {
00303     l4_uint32_t dummy;
00304     l4_cpu_time_t tsc;
00305     __asm__
00306     (
00307     "movl    %%edx, %%ecx \n\t"
00308     "mull    %3        \n\t"
00309     "movl    %%ecx, %%eax \n\t"
00310     "movl    %%edx, %%ecx \n\t"
00311     "mull    %3        \n\t"
00312     "addl    %%ecx, %%eax \n\t"
00313     "adcl    $0, %%edx \n\t"
00314     "shld    $5, %%eax, %%edx \n\t"
00315     "shll    $5, %%eax \n\t"
00316     : "=A" (tsc),
00317     "=&c" (dummy)
00318     : "0" (ns),
00319     "g" (l4_scaler_ns_to_tsc)
00320     );
00321     return tsc;
00322 }
00323
00324 L4_INLINE void
00325 l4_busy_wait_ns (l4_uint64_t ns)
00326 {

```

```

00327  l4_cpu_time_t stop = l4_rdtsc();
00328  stop += l4_ns_to_tsc(ns);
00329
00330  while (l4_rdtsc() < stop)
00331      ;
00332 }
00333
00334 L4_INLINE void
00335 l4_busy_wait_us (l4_uint64_t us)
00336 {
00337     l4_cpu_time_t stop = l4_rdtsc ();
00338     stop += l4_ns_to_tsc(us*1000ULL);
00339
00340     while (l4_rdtsc() < stop)
00341         ;
00342 }
00343
00344 #endif /* __l4_rdtsc_h */
00345

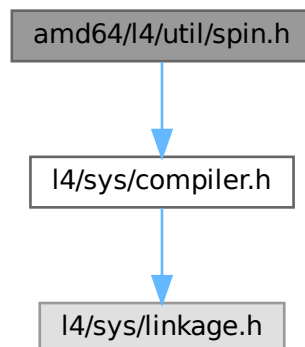
```

## 16.52 amd64/l4/util/spin.h File Reference

Spinning for amd64.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for spin.h:



### 16.52.1 Detailed Description

Spinning for amd64.

Definition in file [spin.h](#).

## 16.53 spin.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,

```

```

00007  *           Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008  *           economic rights: Technische Universität Dresden (Germany)
00009  *           This file is part of TUD:OS and distributed under the terms of the
00010  *           GNU Lesser General Public License 2.1.
00011  *           Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013 #ifndef __l4util_spin_h
00014 #define __l4util_spin_h
00015
00016 #include <l4/sys/compiler.h>
00017
00018 EXTERN_C_BEGIN
00019
00020 L4_CV void l4_spin(int x,int y);
00021 L4_CV void l4_spin_vga(int x,int y);
00022 L4_CV void l4_spin_n_text(int x, int y, int len, const char*s);
00023 L4_CV void l4_spin_n_text_vga(int x, int y, int len, const char*s);
00024
00025 /*****
00026  *
00027  * spin_text()      - spinning wheel at the hercules screen. The given text
00028  *                   must be a text constant, no variables or arrays. Its
00029  *                   size is determined with the sizeof operator, it's much
00030  *                   faster than the strlen function.
00031  * spin_text_vga() - same for vga.
00032  *
00033  *****/
00034 #define l4_spin_text(x, y, text) \
00035     l4_spin_n_text((x), (y), sizeof(text)-1, "" text)
00036 #define l4_spin_text_vga(x, y, text) \
00037     l4_spin_n_text_vga((x), (y), sizeof(text)-1, "" text)
00038
00039 EXTERN_C_END
00040
00041 #endif

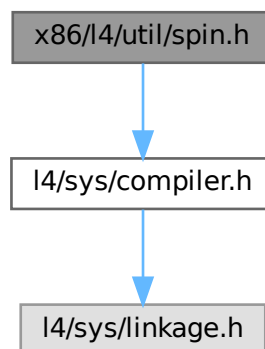
```

## 16.54 x86/l4/util/spin.h File Reference

Spinning for x86.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for spin.h:



### 16.54.1 Detailed Description

Spinning for x86.

Definition in file [spin.h](#).

## 16.55 spin.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU Lesser General Public License 2.1.
00011  * Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013 #ifndef __l4util_spin_h
00014 #define __l4util_spin_h
00015
00016 #include <l4/sys/compiler.h>
00017
00018 EXTERN_C_BEGIN
00019
00020 L4_CV void l4_spin(int x,int y);
00021 L4_CV void l4_spin_vga(int x,int y);
00022 L4_CV void l4_spin_n_text(int x, int y, int len, const char*s);
00023 L4_CV void l4_spin_n_text_vga(int x, int y, int len, const char*s);
00024
00025 /*****
00026  *
00027  * spin_text()      - spinning wheel at the hercules screen. The given text
00028  *                   must be a text constant, no variables or arrays. Its
00029  *                   size is determined with the sizeof operator, it's much
00030  *                   faster than the strlen function.
00031  * spin_text_vga() - same for vga.
00032  *
00033  *****/
00034 #define l4_spin_text(x, y, text) \
00035     l4_spin_n_text((x), (y), sizeof(text)-1, " text)
00036 #define l4_spin_text_vga(x, y, text) \
00037     l4_spin_n_text_vga((x), (y), sizeof(text)-1, " text)
00038
00039 EXTERN_C_END
00040
00041 #endif

```

## 16.56 amd64/l4/sys/segment.h File Reference

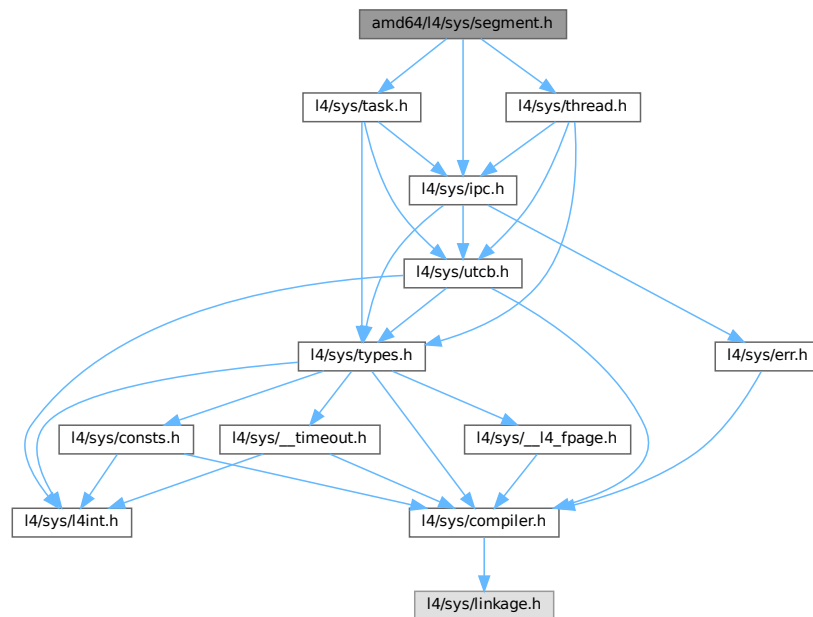
Segment handling.

```

#include <l4/sys/ipc.h>
#include <l4/sys/task.h>
#include <l4/sys/thread.h>

```

Include dependency graph for segment.h:



## Enumerations

- enum [L4\\_task\\_ldt\\_x86\\_consts](#) { [L4\\_TASK\\_LDT\\_X86\\_ENTRY\\_SIZE](#) = 8 , [L4\\_TASK\\_LDT\\_X86\\_MAX\\_ENTRIES](#) }
- Constants for LDT handling.*
- enum [L4\\_sys\\_segment](#) { [L4\\_AMD64\\_SEGMENT\\_FS](#) = 0 , [L4\\_AMD64\\_SEGMENT\\_GS](#) = 1 }
- Constants for identifying segments.*

## Functions

- long [fiasco\\_ldt\\_set](#) ([l4\\_cap\\_idx\\_t](#) task, void \*ldt, unsigned int num\_desc, unsigned int entry\_number\_start, [l4\\_utcb\\_t](#) \*utcb)  
*Set LDT segments descriptors.*
- long [fiasco\\_gdt\\_set](#) ([l4\\_cap\\_idx\\_t](#) thread, void \*desc, unsigned int size, unsigned int entry\_number\_start, [l4\\_utcb\\_t](#) \*utcb)  
*Set GDT segment descriptors.*
- unsigned [fiasco\\_gdt\\_get\\_entry\\_offset](#) ([l4\\_cap\\_idx\\_t](#) thread, [l4\\_utcb\\_t](#) \*utcb)  
*Return the offset of the entry in the GDT.*
- long [fiasco\\_amd64\\_set\\_fs](#) ([l4\\_cap\\_idx\\_t](#) thread, [l4\\_umword\\_t](#) base, [l4\\_utcb\\_t](#) \*utcb)  
*Set the base address for the FS segment.*
- long [fiasco\\_amd64\\_set\\_segment\\_base](#) ([l4\\_cap\\_idx\\_t](#) thread, enum [L4\\_sys\\_segment](#) segr, [l4\\_umword\\_t](#) base, [l4\\_utcb\\_t](#) \*utcb)  
*Set the base address for a segment.*
- long [fiasco\\_amd64\\_segment\\_info](#) ([l4\\_cap\\_idx\\_t](#) thread, unsigned \*user\_ds, unsigned \*user\_cs, unsigned \*user32\_cs, [l4\\_utcb\\_t](#) \*utcb)  
*Get segment information.*

## 16.56.1 Detailed Description

Segment handling.

Definition in file [segment.h](#).

## 16.56.2 Enumeration Type Documentation

### 16.56.2.1 L4\_sys\_segment

```
enum L4_sys_segment
```

Constants for identifying segments.

Enumerator

L4_AMD64_SEGMENT_FS	Constant identifying the FS segment.
L4_AMD64_SEGMENT_GS	Constant identifying the GS segment.

Definition at line 117 of file [segment.h](#).

### 16.56.2.2 L4\_task\_ldt\_x86\_consts

```
enum L4_task_ldt_x86_consts
```

Constants for LDT handling.

Enumerator

L4_TASK_LDT_X86_ENTRY_SIZE	Size of an LDT entry.
L4_TASK_LDT_X86_MAX_ENTRIES	Maximum number of LDT entries that can be written with one call.

Definition at line 86 of file [segment.h](#).

## 16.56.3 Function Documentation

### 16.56.3.1 fiasco\_amd64\_segment\_info()

```
long fiasco_amd64_segment_info (  
    l4_cap_idx_t thread,  
    unsigned * user_ds,  
    unsigned * user_cs,  
    unsigned * user32_cs,  
    l4_utcb_t * utcb ) [inline]
```

Get segment information.

## Parameters

in	<i>thread</i>	Thread to get info from.
out	<i>user_ds</i>	DS segment selector.
out	<i>user_cs</i>	64-bit CS segment selector.
out	<i>user32_cs</i>	32-bit CS segment selector.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

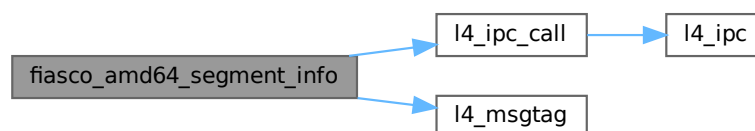
## Returns

System call error

Definition at line 186 of file [segment.h](#).

References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_THREAD](#), [L4\\_THREAD\\_AMD64\\_GET\\_SEGMENT\\_INFO\\_OP](#), and [l4\\_msg\\_regs\\_t::mr](#).

Here is the call graph for this function:



## 16.56.3.2 fiasco\_amd64\_set\_fs()

```

long fiasco_amd64_set_fs (
    l4_cap_idx_t thread,
    l4_umword_t base,
    l4_utcb_t * utcb ) [inline]

```

Set the base address for the FS segment.

## Parameters

<i>thread</i>	Thread for which the FS base address shall be modified.
<i>base</i>	Base address.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

## Return values

<i>L4_EOK</i>	Success.
<i>-L4_EINVAL</i>	Invalid base address ( <i>base</i> ).
<i>-L4_ENOSYS</i>	Operation not supported with current kernel configuration.



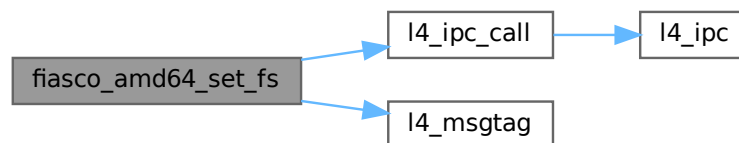
## Note

Calling this function is equivalent to calling `fiasco_amd64_set_segment_base(thread, L4_↔AMD64_SEGMENT_FS, base, utcb)`.

Definition at line 35 of file [segment.h](#).

References [L4\\_AMD64\\_SEGMENT\\_FS](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_THREAD](#), [L4\\_THREAD\\_AMD64\\_SET\\_SEGMENT\\_BASE\\_OP](#), and [l4\\_msg\\_regs\\_t::mr](#).

Here is the call graph for this function:



### 16.56.3.3 fiasco\_amd64\_set\_segment\_base()

```

long fiasco_amd64_set_segment_base (
    l4_cap_idx_t thread,
    enum L4_sys_segment segr,
    l4_umword_t base,
    l4_utcb_t * utcb ) [inline]
  
```

Set the base address for a segment.

## Parameters

<i>thread</i>	Thread for which the base address of the selected segment shall be modified.
<i>segr</i>	Segment to modify (one of <a href="#">L4_sys_segment</a> ).
<i>base</i>	Base address.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

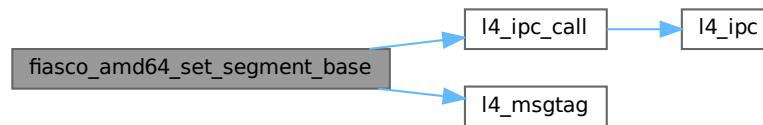
## Return values

<i>L4_EOK</i>	Success.
<i>-L4_EINVAL</i>	Invalid segment ( <i>segr</i> ) or base address ( <i>base</i> ).
<i>-L4_ENOSYS</i>	Operation not supported with current kernel configuration.

Definition at line 43 of file [segment.h](#).

References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_THREAD](#), [L4\\_THREAD\\_AMD64\\_SET\\_SEGMENT\\_BASE\\_OP](#), and [l4\\_msg\\_regs\\_t::mr](#).

Here is the call graph for this function:



## 16.57 segment.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 /*****
00024 #ifndef __L4_SYS_ARCH_X86__SEGMENT_H__
00025 #define __L4_SYS_ARCH_X86__SEGMENT_H__
00026
00027 #ifndef L4API_l4f
00028 #error This header file can only be used with a L4API version!
00029 #endif
00030
00031 #include <l4/sys/ipc.h>
00032
00050 L4_INLINE long
00051 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00052               unsigned int entry_number_start, l4_utcb_t *utcb);
00053
00070 L4_INLINE long
00071 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00072               unsigned int entry_number_start, l4_utcb_t *utcb);
00073
00080 L4_INLINE unsigned
00081 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb);
00082
00086 enum L4_task_ldt_x86_consts
00087 {
00089     L4_TASK_LDT_X86_ENTRY_SIZE = 8,
00091     L4_TASK_LDT_X86_MAX_ENTRIES
00092         = (L4_UTCB_GENERIC_DATA_SIZE - 2)
00093         / (L4_TASK_LDT_X86_ENTRY_SIZE / (L4_MWORD_BITS / 8)),
00094 };
00095
00111 L4_INLINE long
00112 fiasco_amd64_set_fs(l4_cap_idx_t thread, l4_umword_t base, l4_utcb_t *utcb);
00113
00117 enum L4_sys_segment
00118 {
00120     L4_AMD64_SEGMENT_FS = 0,
00122     L4_AMD64_SEGMENT_GS = 1
00123 };
00124
00138 L4_INLINE long
00139 fiasco_amd64_set_segment_base(l4_cap_idx_t thread, enum L4_sys_segment segr,
00140                               l4_umword_t base, l4_utcb_t *utcb);
00141

```

```

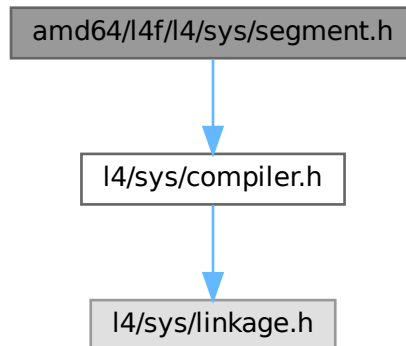
00151 L4_INLINE long
00152 fiasco_amd64_segment_info(l4_cap_idx_t thread, unsigned *user_ds,
00153                          unsigned *user_cs, unsigned *user32_cs,
00154                          l4_utcb_t *utcb);
00155
00156 /*****
00157 *** Implementation
00158 *****/
00159
00160 #include <l4/sys/task.h>
00161 #include <l4/sys/thread.h>
00162
00163 L4_INLINE long
00164 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00165              unsigned int entry_number_start, l4_utcb_t *utcb)
00166 {
00167     if (num_desc > L4_TASK_LDT_X86_MAX_ENTRIES)
00168         return -L4_EINVAL;
00169     l4_utcb_mr_u(utcb)->mr[0] = L4_TASK_LDT_SET_X86_OP;
00170     l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00171     __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], ldt,
00172                   num_desc * L4_TASK_LDT_X86_ENTRY_SIZE);
00173     return l4_error_u(l4_ipc_call(task, utcb, l4_msgtag(L4_PROTO_TASK, 2 + num_desc * 2, 0, 0),
00174                        L4_IPC_NEVER), utcb);
00175 }
00176
00177 L4_INLINE unsigned
00178 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb)
00179 {
00180     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00181     if (l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER), utcb))
00182         return -1;
00183     return l4_utcb_mr_u(utcb)->mr[0];
00184 }
00185
00186 L4_INLINE long
00187 fiasco_amd64_segment_info(l4_cap_idx_t thread, unsigned *user_ds,
00188                          unsigned *user_cs, unsigned *user32_cs,
00189                          l4_utcb_t *utcb)
00190 {
00191     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00192     int r;
00193     m->mr[0] = L4_THREAD_AMD64_GET_SEGMENT_INFO_OP;
00194     r = l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0),
00195                        L4_IPC_NEVER), utcb);
00196     if (r < 0)
00197         return r;
00198     *user_ds = m->mr[0];
00199     *user_cs = m->mr[1];
00200     *user32_cs = m->mr[2];
00201     return 0;
00202 }
00203
00204 #endif /* ! __L4_SYS_ARCH_X86_SEGMENT_H__ */

```

## 16.58 amd64/l4f/l4/sys/segment.h File Reference

l4f specific fs/gs manipulation

```
#include <l4/sys/compiler.h>
Include dependency graph for segment.h:
```



## Functions

- long [fiasco\\_amd64\\_set\\_fs](#) ([l4\\_cap\\_idx\\_t](#) thread, [l4\\_umword\\_t](#) base, [l4\\_utcb\\_t](#) \*utcb)  
*Set the base address for the FS segment.*
- long [fiasco\\_amd64\\_set\\_segment\\_base](#) ([l4\\_cap\\_idx\\_t](#) thread, enum [L4\\_sys\\_segment](#) segr, [l4\\_umword\\_t](#) base, [l4\\_utcb\\_t](#) \*utcb)  
*Set the base address for a segment.*
- long [fiasco\\_gdt\\_set](#) ([l4\\_cap\\_idx\\_t](#) thread, void \*desc, unsigned int size, unsigned int entry\_number\_start, [l4\\_utcb\\_t](#) \*utcb)  
*Set GDT segment descriptors.*

## 16.58.1 Detailed Description

l4f specific fs/gs manipulation

Definition in file [segment.h](#).

## 16.58.2 Function Documentation

### 16.58.2.1 fiasco\_amd64\_set\_fs()

```
long fiasco_amd64_set_fs (
    l4\_cap\_idx\_t thread,
    l4\_umword\_t base,
    l4\_utcb\_t * utcb ) [inline]
```

Set the base address for the FS segment.

## Parameters

<i>thread</i>	Thread for which the FS base address shall be modified.
<i>base</i>	Base address.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

## Return values

<i>L4_EOK</i>	Success.
<i>-L4_EINVAL</i>	Invalid base address ( <i>base</i> ).
<i>-L4_ENOSYS</i>	Operation not supported with current kernel configuration.

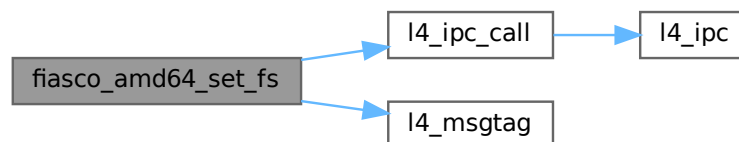
## Note

Calling this function is equivalent to calling `fiasco_amd64_set_segment_base(thread, L4_↔AMD64_SEGMENT_FS, base, utcb)`.

Definition at line 35 of file [segment.h](#).

References [L4\\_AMD64\\_SEGMENT\\_FS](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_THREAD](#), [L4\\_THREAD\\_AMD64\\_SET\\_SEGMENT\\_BASE\\_OP](#), and [l4\\_msg\\_regs\\_t::mr](#).

Here is the call graph for this function:



## 16.58.2.2 fiasco\_amd64\_set\_segment\_base()

```

long fiasco_amd64_set_segment_base (
    l4_cap_idx_t thread,
    enum L4_sys_segment sgr,
    l4_umword_t base,
    l4_utcb_t * utcb ) [inline]
  
```

Set the base address for a segment.

## Parameters

<i>thread</i>	Thread for which the base address of the selected segment shall be modified.
<i>sgr</i>	Segment to modify (one of <a href="#">L4_sys_segment</a> ).
<i>base</i>	Base address.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

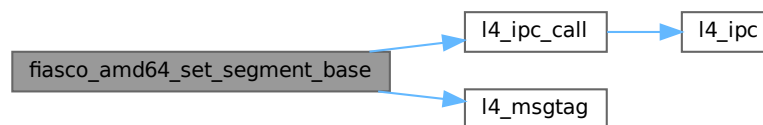
## Return values

<code>L4_EOK</code>	Success.
<code>-L4_EINVAL</code>	Invalid segment ( <code>segr</code> ) or base address ( <code>base</code> ).
<code>-L4_ENOSYS</code>	Operation not supported with current kernel configuration.

Definition at line 43 of file [segment.h](#).

References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_THREAD](#), [L4\\_THREAD\\_AMD64\\_SET\\_SEGMENT\\_BASE\\_OP](#), and [l4\\_msg\\_regs\\_t::mr](#).

Here is the call graph for this function:



## 16.59 segment.h

[Go to the documentation of this file.](#)

```

00001 #include_next <l4/sys/segment.h>
00002
00008 /*
00009  * (c) 2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #ifndef __L4_SYS__ARCH_AMD64__L4API_L4F__SEGMENT_H__
00026 #define __L4_SYS__ARCH_AMD64__L4API_L4F__SEGMENT_H__
00027
00028 #include <l4/sys/compiler.h>
00029
00030 /***** Implementation *****/
00031
00032
00033
00034 L4_INLINE long
00035 fiasco_amd64_set_fs(l4_cap_idx_t thread, l4_umword_t base, l4_utcb_t *utcb)
00036 {
00037     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_AMD64_SET_SEGMENT_BASE_OP | ((l4_umword_t)L4_AMD64_SEGMENT_FS
00038     << 16);
00039     l4_utcb_mr_u(utcb)->mr[1] = base;
00040     return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER),
00041     utcb);
00042 }
00043
00044 L4_INLINE long
00045 fiasco_amd64_set_segment_base(l4_cap_idx_t thread, enum L4_sys_segment segr,
00046     l4_umword_t base, l4_utcb_t *utcb)

```

```

00045 {
00046     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_AMD64_SET_SEGMENT_BASE_OP | ((l4_umword_t)segr << 16);
00047     l4_utcb_mr_u(utcb)->mr[1] = base;
00048     return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER),
00049         utcb);
00049 }
00050
00051 L4_INLINE long
00052 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00053     unsigned int entry_number_start, l4_utcb_t *utcb)
00054 {
00055     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00056     l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00057     __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], desc, size);
00058     return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2 + (size / 8), 0, 0),
00059         L4_IPC_NEVER), utcb);
00059 }
00060
00061 #endif /* ! __L4_SYS_ARCH_X86__L4API_L4F__SEGMENT_H__ */

```

## 16.60 x86/I4/sys/segment.h File Reference

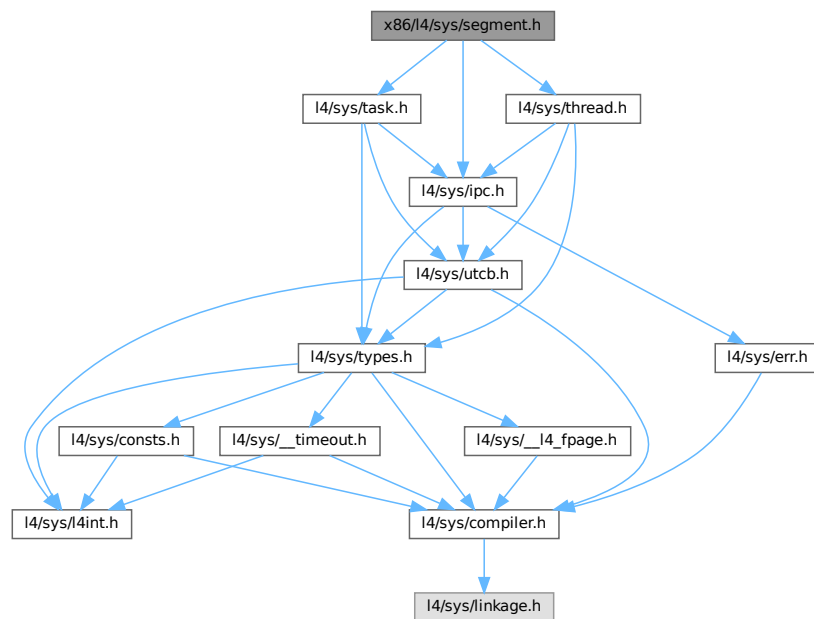
Segment handling.

```

#include <l4/sys/ipc.h>
#include <l4/sys/task.h>
#include <l4/sys/thread.h>

```

Include dependency graph for segment.h:



### Enumerations

- enum [L4\\_task\\_ldt\\_x86\\_consts](#) { [L4\\_TASK\\_LDT\\_X86\\_ENTRY\\_SIZE](#) = 8, [L4\\_TASK\\_LDT\\_X86\\_MAX\\_ENTRIES](#) }

*Contants for LDT handling.*

## Functions

- long `fiasco_ldt_set` (`l4_cap_idx_t` task, void \*ldt, unsigned int num\_desc, unsigned int entry\_number\_start, `l4_utcb_t` \*utcb)  
*Set LDT segments descriptors.*
- long `fiasco_gdt_set` (`l4_cap_idx_t` thread, void \*desc, unsigned int size, unsigned int entry\_number\_start, `l4_utcb_t` \*utcb)  
*Set GDT segment descriptors.*
- unsigned `fiasco_gdt_get_entry_offset` (`l4_cap_idx_t` thread, `l4_utcb_t` \*utcb)  
*Return the offset of the entry in the GDT.*

### 16.60.1 Detailed Description

Segment handling.

Definition in file [segment.h](#).

### 16.60.2 Enumeration Type Documentation

#### 16.60.2.1 L4\_task\_ldt\_x86\_consts

enum `L4_task_ldt_x86_consts`

Constants for LDT handling.

#### Enumerator

<code>L4_TASK_LDT_X86_ENTRY_SIZE</code>	Size of an LDT entry.
<code>L4_TASK_LDT_X86_MAX_ENTRIES</code>	Maximum number of LDT entries that can be written with one call.

Definition at line 86 of file [segment.h](#).

## 16.61 segment.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */

```



```

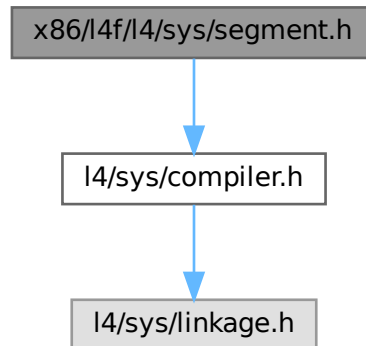
00023 /*****
00024 #ifndef __L4_SYS__ARCH_X86__SEGMENT_H__
00025 #define __L4_SYS__ARCH_X86__SEGMENT_H__
00026
00027 #ifndef L4API_I4f
00028 #error This header file can only be used with a L4API version!
00029 #endif
00030
00031 #include <l4/sys/ipc.h>
00032
00050 L4_INLINE long
00051 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00052               unsigned int entry_number_start, l4_utcb_t *utcb);
00053
00070 L4_INLINE long
00071 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00072               unsigned int entry_number_start, l4_utcb_t *utcb);
00073
00080 L4_INLINE unsigned
00081 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb);
00082
00086 enum L4_task_ldt_x86_consts
00087 {
00089     L4_TASK_LDT_X86_ENTRY_SIZE = 8,
00091     L4_TASK_LDT_X86_MAX_ENTRIES
00092         = (L4_UTCB_GENERIC_DATA_SIZE - 2)
00093           / (L4_TASK_LDT_X86_ENTRY_SIZE / (L4_MWORD_BITS / 8)),
00094 };
00095
00096 /*****
00097 *** Implementation
00098 *****/
00099
00100 #include <l4/sys/task.h>
00101 #include <l4/sys/thread.h>
00102
00103 L4_INLINE long
00104 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00105               unsigned int entry_number_start, l4_utcb_t *utcb)
00106 {
00107     if (num_desc > L4_TASK_LDT_X86_MAX_ENTRIES)
00108         return -L4_EINVAL;
00109     l4_utcb_mr_u(utcb)->mr[0] = L4_TASK_LDT_SET_X86_OP;
00110     l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00111     __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], ldt,
00112                     num_desc * L4_TASK_LDT_X86_ENTRY_SIZE);
00113     return l4_error_u(l4_ipc_call(task, utcb, l4_msgtag(L4_PROTO_TASK, 2 + num_desc * 2, 0, 0),
00114                     L4_IPC_NEVER), utcb);
00115 }
00116
00117 L4_INLINE unsigned
00118 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb)
00119 {
00120     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00121     if (l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER), utcb))
00122         return -1;
00123     return l4_utcb_mr_u(utcb)->mr[0];
00124 }
00125 #endif /* ! __L4_SYS__ARCH_X86__SEGMENT_H__ */

```

## 16.62 x86/I4f/I4/sys/segment.h File Reference

I4f specific segment manipulation

```
#include <l4/sys/compiler.h>
Include dependency graph for segment.h:
```



## Functions

- long [fiasco\\_gdt\\_set](#) (l4\_cap\_idx\_t thread, void \*desc, unsigned int size, unsigned int entry\_number\_start, l4\_utcb\_t \*utcb)  
Set GDT segment descriptors.

### 16.62.1 Detailed Description

l4f specific segment manipulation

Definition in file [segment.h](#).

## 16.63 segment.h

[Go to the documentation of this file.](#)

```
00001 #include_next <l4/sys/segment.h>
00002
00008 /*
00009  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #ifndef __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__
00026 #define __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__
00027
```

```

00028 #include <l4/sys/compiler.h>
00029
00030 /*****
00031  *** Implementation
00032  *****/
00033
00034 L4_INLINE long
00035 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00036               unsigned int entry_number_start, l4_utcb_t *utcb)
00037 {
00038     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00039     l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00040     __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], desc, size);
00041     return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2 + (size » 2), 0, 0),
00042                      L4_IPC_NEVER), utcb);
00043 }
00044 #endif /* ! __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__ */

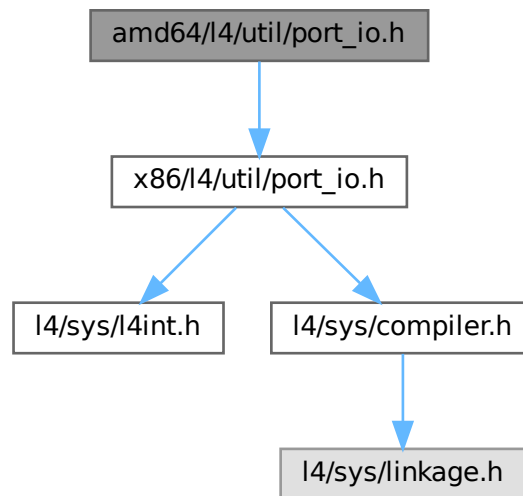
```

## 16.64 amd64/l4/util/port\_io.h File Reference

Port I/O functions.

```
#include <x86/l4/util/port_io.h>
```

Include dependency graph for port\_io.h:



### 16.64.1 Detailed Description

Port I/O functions.

Definition in file [port\\_io.h](#).

## 16.65 port\_io.h

[Go to the documentation of this file.](#)

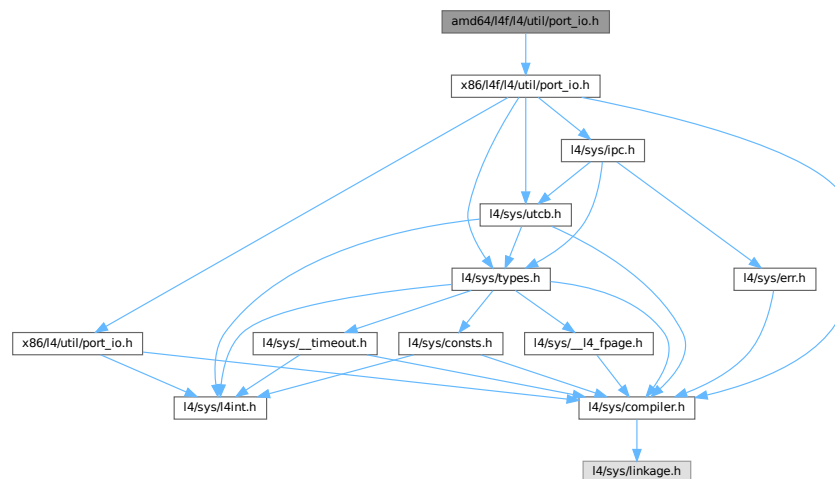
```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012 #include <x86/l4/l4/util/port_io.h>
```

## 16.66 amd64/l4f/l4/util/port\_io.h File Reference

Port I/O functions.

```
#include <x86/l4f/l4/util/port_io.h>
```

Include dependency graph for port\_io.h:



### 16.66.1 Detailed Description

Port I/O functions.

Definition in file [port\\_io.h](#).

## 16.67 port\_io.h

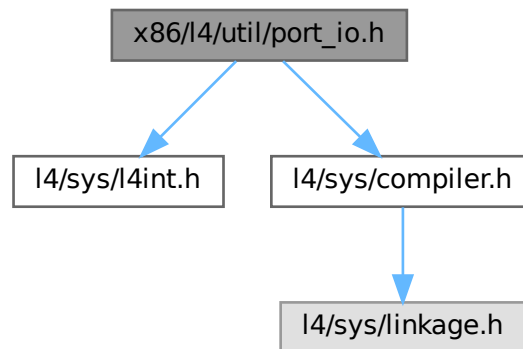
[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012 #include <x86/l4f/l4/util/port_io.h>
```

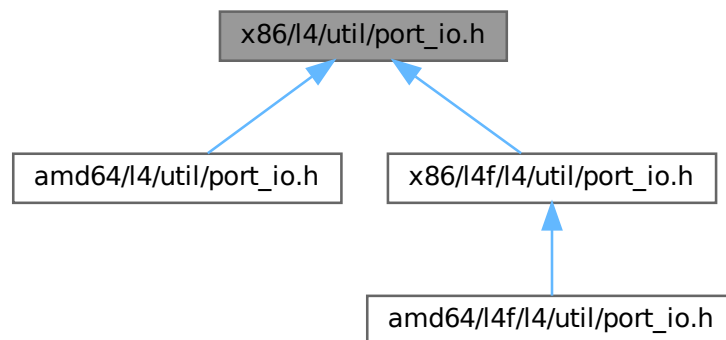
## 16.68 x86/l4/util/port\_io.h File Reference

x86 port I/O

```
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
Include dependency graph for port_io.h:
```



This graph shows which files directly or indirectly include this file:



### Functions

- `l4_uint8_t l4util_in8 (l4_uint16_t port)`  
*Read byte from I/O port.*
- `l4_uint16_t l4util_in16 (l4_uint16_t port)`  
*Read 16-bit-value from I/O port.*

- `l4_uint32_t l4util_in32 (l4_uint16_t port)`  
*Read 32-bit-value from I/O port.*
- `void l4util_ins8 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
*Read a block of 8-bit-values from I/O ports.*
- `void l4util_ins16 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
*Read a block of 16-bit-values from I/O ports.*
- `void l4util_ins32 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
*Read a block of 32-bit-values from I/O ports.*
- `void l4util_out8 (l4_uint8_t value, l4_uint16_t port)`  
*Write byte to I/O port.*
- `void l4util_out16 (l4_uint16_t value, l4_uint16_t port)`  
*Write 16-bit-value to I/O port.*
- `void l4util_out32 (l4_uint32_t value, l4_uint16_t port)`  
*Write 32-bit-value to I/O port.*
- `void l4util_outs8 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
*Write a block of bytes to I/O port.*
- `void l4util_outs16 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
*Write a block of 16-bit-values to I/O port.*
- `void l4util_outs32 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
*Write block of 32-bit-values to I/O port.*
- `void l4util_iodelay (void)`  
*delay I/O port access by writing to port 0x80*

## 16.68.1 Detailed Description

x86 port I/O

Date

06/2003

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [port\\_io.h](#).

## 16.69 port\_io.h

[Go to the documentation of this file.](#)

```

00001 /*****
00009 /*****
00010
00011 */
00012 * (c) 2003-2009 Author(s)
00013 *     economic rights: Technische Universität Dresden (Germany)
00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU Lesser General Public License 2.1.
00016 * Please see the COPYING-LGPL-2.1 file for details.
00017 */
00018
00019 #ifndef _L4UTIL_PORT_IO_H
00020 #define _L4UTIL_PORT_IO_H
00021

```

```

00027 /* L4 includes */
00028 #include <l4/sys/l4int.h>
00029 #include <l4/sys/compiler.h>
00030
00031 /*****
00032  *** Prototypes
00033  *****/
00034
00035 EXTERN_C_BEGIN
00046 L4_INLINE l4_uint8_t
00047 l4util_in8(l4_uint16_t port);
00048
00055 L4_INLINE l4_uint16_t
00056 l4util_in16(l4_uint16_t port);
00057
00064 L4_INLINE l4_uint32_t
00065 l4util_in32(l4_uint16_t port);
00066
00074 L4_INLINE void
00075 l4util_ins8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00076
00084 L4_INLINE void
00085 l4util_ins16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00086
00094 L4_INLINE void
00095 l4util_ins32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00096
00103 L4_INLINE void
00104 l4util_out8(l4_uint8_t value, l4_uint16_t port);
00105
00113 L4_INLINE void
00114 l4util_out16(l4_uint16_t value, l4_uint16_t port);
00115
00122 L4_INLINE void
00123 l4util_out32(l4_uint32_t value, l4_uint16_t port);
00124
00132 L4_INLINE void
00133 l4util_outs8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00134
00143 L4_INLINE void
00144 l4util_outs16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00145
00153 L4_INLINE void
00154 l4util_outs32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00155
00159 L4_INLINE void
00160 l4util_iodelay(void);
00161
00164 EXTERN_C_END
00165
00166
00167 /*****
00168  *** Implementation
00169  *****/
00170
00171 L4_INLINE l4_uint8_t
00172 l4util_in8(l4_uint16_t port)
00173 {
00174     l4_uint8_t value;
00175     asm volatile ("inb %w1, %b0" : "=a" (value) : "Nd" (port));
00176     return value;
00177 }
00178
00179 L4_INLINE l4_uint16_t
00180 l4util_in16(l4_uint16_t port)
00181 {
00182     l4_uint16_t value;
00183     asm volatile ("inw %w1, %w0" : "=a" (value) : "Nd" (port));
00184     return value;
00185 }
00186
00187 L4_INLINE l4_uint32_t
00188 l4util_in32(l4_uint16_t port)
00189 {
00190     l4_uint32_t value;
00191     asm volatile ("inl %w1, %0" : "=a" (value) : "Nd" (port));
00192     return value;
00193 }
00194
00195 L4_INLINE void
00196 l4util_ins8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00197 {
00198     l4_umword_t dummy1, dummy2;
00199     asm volatile ("rep insb" : "=D" (dummy1), "=c" (dummy2)
00200                  : "d" (port), "D" (addr), "c" (count)
00201                  : "memory");
00202 }

```

```

00203
00204 L4_INLINE void
00205 l4util_ins16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00206 {
00207     l4_umword_t dummy1, dummy2;
00208     asm volatile ("rep insw" : "=D"(dummy1), "=c"(dummy2)
00209 : "d" (port), "D" (addr), "c"(count)
00210 : "memory");
00211 }
00212
00213 L4_INLINE void
00214 l4util_ins32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00215 {
00216     l4_umword_t dummy1, dummy2;
00217     asm volatile ("rep insl" : "=D"(dummy1), "=c"(dummy2)
00218 : "d" (port), "D" (addr), "c"(count)
00219 : "memory");
00220 }
00221
00222 L4_INLINE void
00223 l4util_out8(l4_uint8_t value, l4_uint16_t port)
00224 {
00225     asm volatile ("outb %b0, %w1" : : "a" (value), "Nd" (port));
00226 }
00227
00228 L4_INLINE void
00229 l4util_out16(l4_uint16_t value, l4_uint16_t port)
00230 {
00231     asm volatile ("outw %w0, %w1" : : "a" (value), "Nd" (port));
00232 }
00233
00234 L4_INLINE void
00235 l4util_out32(l4_uint32_t value, l4_uint16_t port)
00236 {
00237     asm volatile ("outl %l0, %w1" : : "a" (value), "Nd" (port));
00238 }
00239
00240 L4_INLINE void
00241 l4util_outs8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00242 {
00243     l4_umword_t dummy1, dummy2;
00244     asm volatile ("rep outsb" : "=S"(dummy1), "=c"(dummy2)
00245 : "d" (port), "S" (addr), "c"(count)
00246 : "memory");
00247 }
00248
00249 L4_INLINE void
00250 l4util_outs16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00251 {
00252     l4_umword_t dummy1, dummy2;
00253     asm volatile ("rep outsw" : "=S"(dummy1), "=c"(dummy2)
00254 : "d" (port), "S" (addr), "c"(count)
00255 : "memory");
00256 }
00257
00258 L4_INLINE void
00259 l4util_outs32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00260 {
00261     l4_umword_t dummy1, dummy2;
00262     asm volatile ("rep outsl" : "=S"(dummy1), "=c"(dummy2)
00263 : "d" (port), "S" (addr), "c"(count)
00264 : "memory");
00265 }
00266
00267 L4_INLINE void
00268 l4util_iodelay(void)
00269 {
00270     asm volatile ("outb %al, $0x80");
00271 }
00272
00273 #endif

```

## 16.70 x86/I4f/I4/util/port\_io.h File Reference

port I/O functions

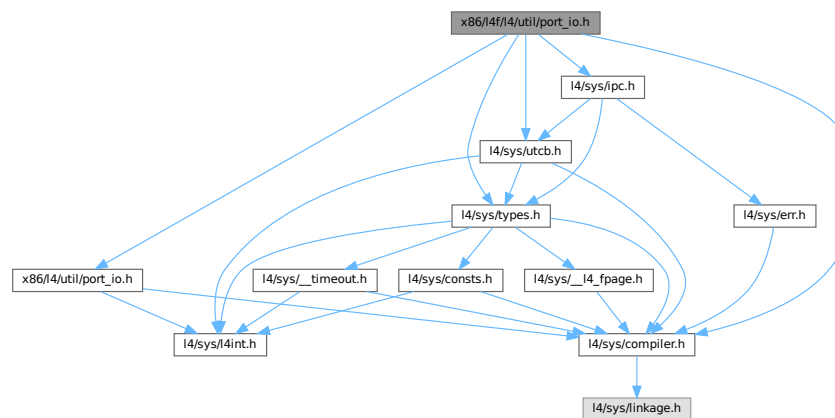
```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <x86/l4/util/port_io.h>

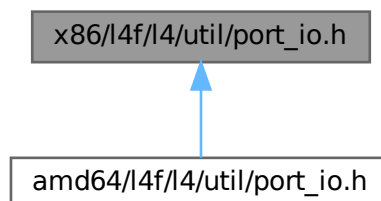
```



```
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
Include dependency graph for port_io.h:
```



This graph shows which files directly or indirectly include this file:



## Functions

- int [l4util\\_ioport\\_map](#) ([l4\\_cap\\_idx\\_t](#) sigma0id, unsigned port\_start, unsigned log2size)  
Map a range of I/O ports.

## 16.70.1 Detailed Description

port I/O functions

Date

06/2003

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [port\\_io.h](#).

## 16.70.2 Function Documentation

### 16.70.2.1 l4util\_ioport\_map()

```
int l4util_ioport_map (
    l4_cap_idx_t sigma0id,
    unsigned port_start,
    unsigned log2size ) [inline]
```

Map a range of I/O ports.

#### Parameters

<i>sigma0id</i>	I/O port service (sigma0).
<i>port_start</i>	(Start) Port to request.
<i>log2size</i>	Log2size of range to request.

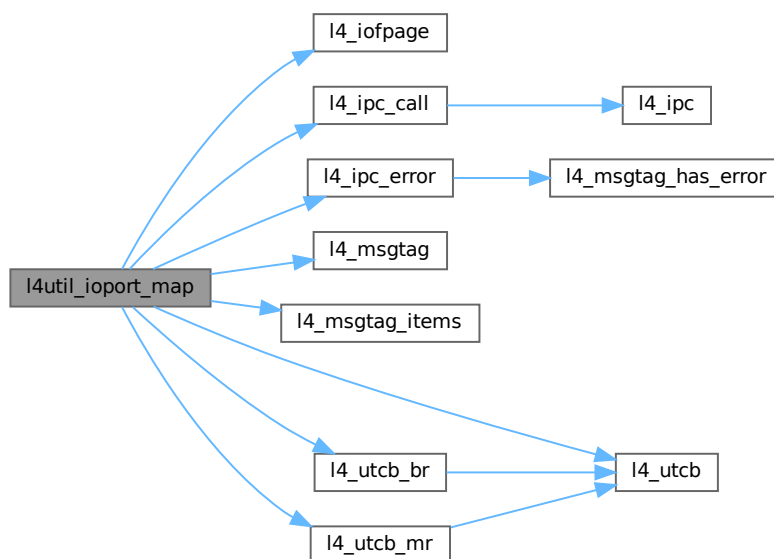
#### Returns

IPC result: 0 if the range could be successfully mapped on error: IPC failure, or -L4\_ENOENT if nothing mapped

Definition at line 56 of file [port\\_io.h](#).

References [l4\\_buf\\_regs\\_t::bdr](#), [l4\\_buf\\_regs\\_t::br](#), [L4\\_ENOENT](#), [l4\\_iofpage\(\)](#), [l4\\_ipc\\_call\(\)](#), [l4\\_ipc\\_error\(\)](#), [L4\\_IPC\\_NEVER](#), [L4\\_ITEM\\_MAP](#), [l4\\_msgtag\(\)](#), [l4\\_msgtag\\_items\(\)](#), [L4\\_PROTO\\_IO\\_PAGE\\_FAULT](#), [l4\\_utcb\(\)](#), [l4\\_utcb\\_br\(\)](#), [l4\\_utcb\\_mr\(\)](#), [l4\\_msg\\_regs\\_t::mr](#), and [l4\\_fpage\\_t::raw](#).

Here is the call graph for this function:



## 16.71 port\_io.h

[Go to the documentation of this file.](#)

```

00001 /*****
00009 /*****
00010
00011 /*
00012  * (c) 2003-2009 Author(s)
00013  *      economic rights: Technische Universität Dresden (Germany)
00014  * This file is part of TUD:OS and distributed under the terms of the
00015  * GNU Lesser General Public License 2.1.
00016  * Please see the COPYING-LGPL-2.1 file for details.
00017  */
00018
00019 #ifndef _L4UTIL_PORT_IO_API_H
00020 #define _L4UTIL_PORT_IO_API_H
00021
00022 #include <l4/sys/compiler.h>
00023 #include <l4/sys/types.h>
00024
00025 #include <x86/l4/util/port_io.h>
00026
00027 EXTERN_C_BEGIN
00028
00040 L4_INLINE int
00041 l4util_ioport_map(l4_cap_idx_t sigma0id,
00042                  unsigned port_start, unsigned log2size);
00043
00044 EXTERN_C_END
00045
00046
00047 /*****
00048 *** Implementation
00049 *****/
00050
00051 #include <l4/sys/utcb.h>
00052 #include <l4/sys/ipc.h>
00053
00054
00055 L4_INLINE int
00056 l4util_ioport_map(l4_cap_idx_t sigma0id,
00057                  unsigned port_start, unsigned log2size)
00058 {
00059     l4_fpage_t iofp;
00060     l4_msgtag_t tag;
00061     long err;
00062
00063     iofp = l4_iofpage(port_start, log2size);
00064     l4_utcb_mr()->mr[0] = iofp.raw;
00065     l4_utcb_br()->bdr    = 0;
00066     l4_utcb_br()->br[0] = L4_ITEM_MAP;
00067     l4_utcb_br()->br[1] = iofp.raw;
00068     tag = l4_ipc_call(sigma0id, l4_utcb(),
00069                      l4_msgtag(L4_PROTO_IO_PAGE_FAULT, 1, 0, 0),
00070                      L4_IPC_NEVER);
00071
00072     if ((err = l4_ipc_error(tag, l4_utcb())))
00073         return err;
00074
00075     return l4_msgtag_items(tag) > 0 ? 0 : -L4_ENOENT;
00076 }
00077
00078 #endif
00079

```

## 16.72 \_\_kip-arch.h

```

00001 /*
00002  * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however

```

```

00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019
00020 struct l4_kip_platform_info_arch
00021 {};

```

## 16.73 \_\_kip-arch.h

```

00001 /*
00002  * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019
00025 struct l4_kip_platform_info_arch
00026 {
00027     struct
00028     {
00029         l4_uint32_t MIDR, CTR, TCMTR, TLBTR, MPIDR, REVIDR;
00030         l4_uint32_t ID_PFR[2], ID_DFR0, ID_AFR0, ID_MMFR[4], ID_ISAR[6];
00031     } cpuinfo;
00032 };

```

## 16.74 \_\_kip-arch.h

```

00001 /*
00002  * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019
00020 struct l4_kip_platform_info_arch
00021 {};

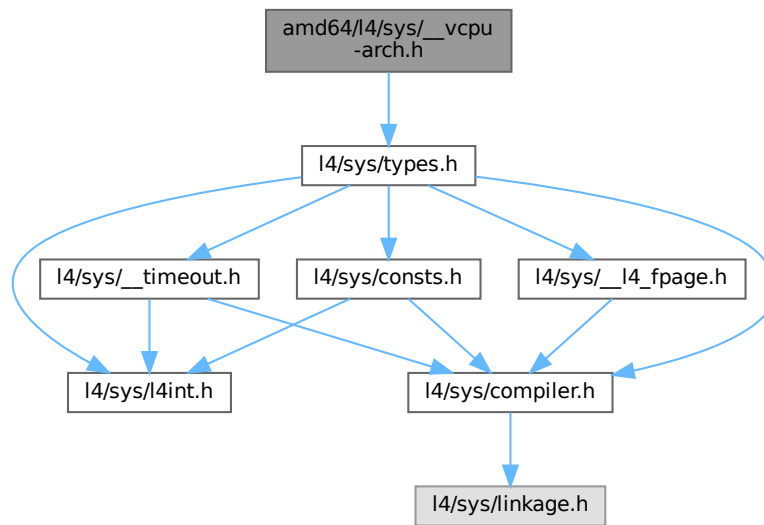
```

## 16.75 amd64/l4/sys/\_\_vcpu-arch.h File Reference

AMD64-specific vCPU interface.

```
#include <l4/sys/types.h>
```

Include dependency graph for \_\_vcpu-arch.h:



## Data Structures

- struct [l4\\_vcpu\\_arch\\_state\\_t](#)  
*Architecture-specific vCPU state.*
- struct [l4\\_vcpu\\_regs\\_t](#)  
*vCPU registers.*
- struct [l4\\_vcpu\\_ipc\\_regs\\_t](#)  
*vCPU message registers.*

## Typedefs

- typedef struct [l4\\_vcpu\\_arch\\_state\\_t](#) [l4\\_vcpu\\_arch\\_state\\_t](#)  
*Architecture-specific vCPU state.*
- typedef struct [l4\\_vcpu\\_regs\\_t](#) [l4\\_vcpu\\_regs\\_t](#)  
*vCPU registers.*
- typedef struct [l4\\_vcpu\\_ipc\\_regs\\_t](#) [l4\\_vcpu\\_ipc\\_regs\\_t](#)  
*vCPU message registers.*

## Enumerations

- enum { [L4\\_VCPU\\_STATE\\_VERSION](#) = 0x25 , [L4\\_VCPU\\_STATE\\_SIZE](#) = 0x200 , [L4\\_VCPU\\_STATE\\_EXT\\_SIZE](#) = [L4\\_PAGESIZE](#) }
  - enum [l4\\_vcpu\\_state\\_offset](#) { [L4\\_VCPU\\_OFFSET\\_EXT\\_STATE](#) = 0x400 , [L4\\_VCPU\\_OFFSET\\_EXT\\_INFOS](#) = 0x200 }
- Offsets for vCPU state layouts.*

## 16.75.1 Detailed Description

AMD64-specific vCPU interface.

Definition in file [\\_\\_vcpu-arch.h](#).

## 16.75.2 Enumeration Type Documentation

### 16.75.2.1 anonymous enum

anonymous enum

Enumerator

L4_VCPU_STATE_VERSION	Architecture-specific version ID. This ID must match the version field in the <a href="#">l4_vcpu_state_t</a> structure after enabling vCPU mode or extended vCPU mode for a thread.
-----------------------	--

Definition at line 27 of file [\\_\\_vcpu-arch.h](#).

## 16.76 \_\_vcpu-arch.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00023 #pragma once
00024
00025 #include <l4/sys/types.h>
00026
00027 enum
00028 {
00035   L4_VCPU_STATE_VERSION = 0x25,
00036
00037   L4_VCPU_STATE_SIZE = 0x200,
00038   L4_VCPU_STATE_EXT_SIZE = L4_PAGESIZE,
00039 };
00040
00045 enum L4_vcpu_state_offset
00046 {
00047   L4_VCPU_OFFSET_EXT_STATE = 0x400,
00048   L4_VCPU_OFFSET_EXT_INFOS = 0x200,
00049 };
00050
00054 typedef struct l4_vcpu_arch_state_t
00055 {
00056   l4_umword_t host_fs_base;
00057   l4_umword_t host_gs_base;
00058   l4_uint16_t host_ds, host_es, host_fs, host_gs;
00059

```

```

00060  l4_uint16_t const user_ds32;
00061  l4_uint16_t const user_cs64;
00062  l4_uint16_t const user_cs32;
00063 } l4_vcpu_arch_state_t;
00064
00065
00070 typedef struct l4_vcpu_regs_t
00071 {
00072     l4_umword_t r15;
00073     l4_umword_t r14;
00074     l4_umword_t r13;
00075     l4_umword_t r12;
00076     l4_umword_t r11;
00077     l4_umword_t r10;
00078     l4_umword_t r9;
00079     l4_umword_t r8;
00081     l4_umword_t di;
00082     l4_umword_t si;
00083     l4_umword_t bp;
00084     l4_umword_t pfa;
00085     l4_umword_t bx;
00086     l4_umword_t dx;
00087     l4_umword_t cx;
00088     l4_umword_t ax;
00090     l4_umword_t trapno;
00091     l4_umword_t err;
00093     l4_umword_t ip;
00094     l4_umword_t cs;
00095     l4_umword_t flags;
00096     l4_umword_t sp;
00097     l4_umword_t ss;
00098     l4_umword_t fs_base;
00099     l4_umword_t gs_base;
00100     l4_uint16_t ds, es, fs, gs;
00101
00102 } l4_vcpu_regs_t;
00103
00108 typedef struct l4_vcpu_ipc_regs_t
00109 {
00110     l4_umword_t _res[1];
00111     l4_umword_t label;
00112     l4_umword_t _res2[5];
00113     l4_msgtag_t tag;
00114 } l4_vcpu_ipc_regs_t;

```

## 16.77 arm/l4/sys/\_\_vcpu-arch.h File Reference

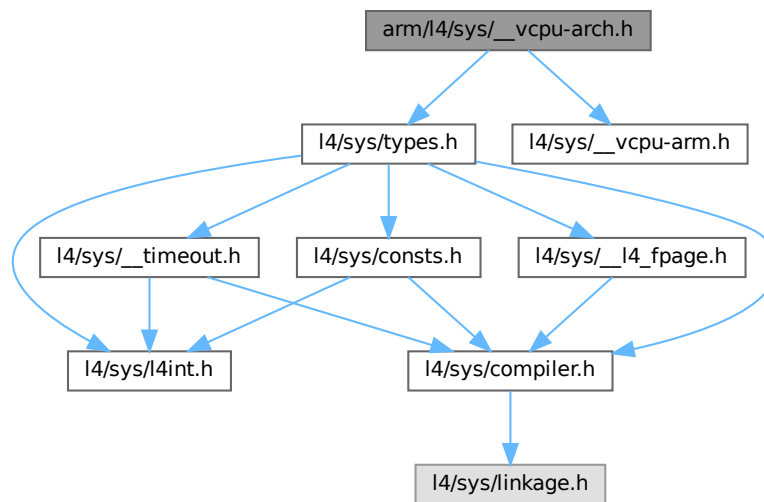
ARM-specific vCPU interface.

```

#include <l4/sys/types.h>
#include <l4/sys/__vcpu-arm.h>

```

Include dependency graph for `__vcpu-arch.h`:



## Data Structures

- struct [l4\\_vcpu\\_regs\\_t](#)  
*vCPU registers.*
- struct [l4\\_vcpu\\_arch\\_state\\_t](#)  
*Architecture-specific vCPU state.*
- struct [l4\\_vcpu\\_ipc\\_regs\\_t](#)  
*vCPU message registers.*

## Typedefs

- typedef struct [l4\\_vcpu\\_regs\\_t](#) [l4\\_vcpu\\_regs\\_t](#)  
*vCPU registers.*
- typedef struct [l4\\_vcpu\\_arch\\_state\\_t](#) [l4\\_vcpu\\_arch\\_state\\_t](#)  
*Architecture-specific vCPU state.*
- typedef struct [l4\\_vcpu\\_ipc\\_regs\\_t](#) [l4\\_vcpu\\_ipc\\_regs\\_t](#)  
*vCPU message registers.*

## Enumerations

- enum { [L4\\_VCPU\\_STATE\\_VERSION](#) = 0x37 , [L4\\_VCPU\\_STATE\\_SIZE](#) = 0x100 , [L4\\_VCPU\\_STATE\\_EXT](#)↵  
\_SIZE = 0x400 }
- enum [L4\\_vcpu\\_state\\_offset](#) { [L4\\_VCPU\\_OFFSET\\_EXT\\_STATE](#) = 0x180 , [L4\\_VCPU\\_OFFSET\\_EXT\\_INFOS](#)  
= 0x100 }
- enum [L4\\_vcpu\\_e\\_field\\_ids](#)  
*IDs for extended vCPU state fields.*



## 16.77.1 Detailed Description

ARM-specific vCPU interface.

Definition in file [\\_\\_vcpu-arch.h](#).

## 16.77.2 Enumeration Type Documentation

### 16.77.2.1 anonymous enum

anonymous enum

Enumerator

L4_VCPU_STATE_VERSION	Architecture-specific version ID. This ID must match the version field in the <a href="#">l4_vcpu_state_t</a> structure after enabling vCPU mode or extended vCPU mode for a thread.
-----------------------	--

Definition at line 28 of file [\\_\\_vcpu-arch.h](#).

### 16.77.2.2 L4\_vcpu\_e\_field\_ids

enum [L4\\_vcpu\\_e\\_field\\_ids](#)

IDs for extended vCPU state fields.

Bits 14..15: are the field size:

- 0 = 32bit field
- 1 = register width field
- 2 = 64bit field

Definition at line 110 of file [\\_\\_vcpu-arch.h](#).

## 16.78 \_\_vcpu-arch.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
```

```

00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00023 #pragma once
00024
00025 #include <l4/sys/types.h>
00026 #include <l4/sys/__vcpu-arm.h>
00027
00028 enum
00029 {
00036     L4_VCPU_STATE_VERSION = 0x37,
00037
00038     L4_VCPU_STATE_SIZE = 0x100,
00039     L4_VCPU_STATE_EXT_SIZE = 0x400,
00040 };
00041
00046 enum L4_vcpu_state_offset
00047 {
00048     L4_VCPU_OFFSET_EXT_STATE = 0x180,
00049     L4_VCPU_OFFSET_EXT_INFOS = 0x100,
00050 };
00051
00052 L4_INLINE l4_arm_vcpu_e_info_t const *
00053 l4_vcpu_e_info(void const *vcpu) L4_NOTHROW
00054 {
00055     return (l4_arm_vcpu_e_info_t const *) ((l4_addr_t)vcpu
00056                                             + L4_VCPU_OFFSET_EXT_INFOS);
00057 }
00058
00059 L4_INLINE void *l4_vcpu_e_ptr(void const *vcpu, unsigned id) L4_NOTHROW
00060 { return (void *) ((l4_addr_t)vcpu + L4_VCPU_OFFSET_EXT_STATE + (id & 0xfff)); }
00061
00066 typedef struct l4_vcpu_regs_t
00067 {
00068     l4_umword_t pfa;
00069     l4_umword_t err;
00070
00071     l4_umword_t r[13];
00072
00073     l4_umword_t sp;
00074     l4_umword_t lr;
00075     l4_umword_t _dummy;
00076     l4_umword_t ip;
00077     l4_umword_t flags;
00078     l4_umword_t tpidruro;
00079     l4_umword_t tpidrurw;
00080 } l4_vcpu_regs_t;
00081
00085 typedef struct l4_vcpu_arch_state_t
00086 {
00087     l4_umword_t host_tpidruro;
00088 } l4_vcpu_arch_state_t;
00089
00094 typedef struct l4_vcpu_ipc_regs_t
00095 {
00096     l4_msgtag_t tag;
00097     l4_umword_t _d1[3];
00098     l4_umword_t label;
00099     l4_umword_t _d2[8];
00100 } l4_vcpu_ipc_regs_t;
00101
00110 enum L4_vcpu_e_field_ids
00111 {
00112     L4_VCPU_E_HCR          = 0x8008,
00113     L4_VCPU_E_TTBRO       = 0x8010,
00114     L4_VCPU_E_TTBRI       = 0x8018,
00115     L4_VCPU_E_TTBCE       = 0x0020,
00116     L4_VCPU_E_SCTLR       = 0x0024,
00117     L4_VCPU_E_DACR        = 0x0028,
00118     L4_VCPU_E_FCSEIDR     = 0x002c,
00119
00120     L4_VCPU_E_CNTVCTL      = 0x0030,
00121     L4_VCPU_E_CNTVOFF     = 0x8038,
00122
00123     L4_VCPU_E_VMPIDR      = 0x0040,
00124     L4_VCPU_E_VPIDR       = 0x0044,
00125
00126     L4_VCPU_E_GIC_HCR     = 0x0050,
00127     L4_VCPU_E_GIC_VTR     = 0x0054,
00128     L4_VCPU_E_GIC_VMCR    = 0x0058,
00129     L4_VCPU_E_GIC_MISR    = 0x005c,
00130     L4_VCPU_E_GIC_EISR    = 0x0060,
00131     L4_VCPU_E_GIC_ELSR    = 0x0064,
00132     L4_VCPU_E_GIC_V2_LR0  = 0x0068,
00133     L4_VCPU_E_GIC_V3_LR0  = 0x8068,
00134 };

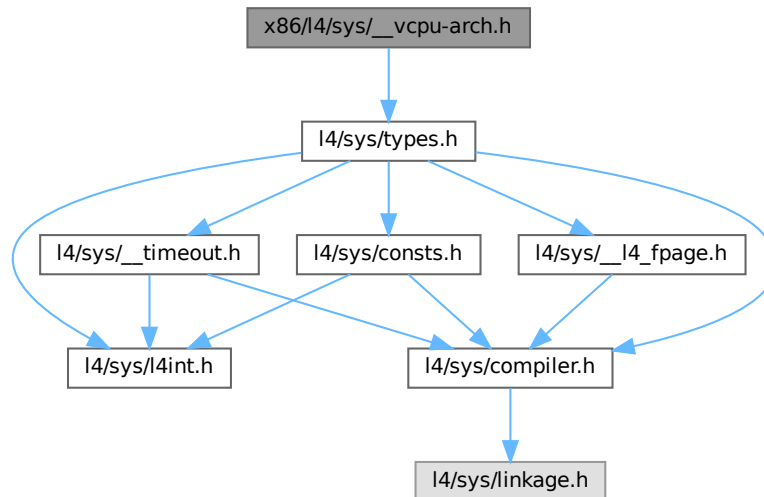
```

## 16.79 x86/l4/sys/\_\_vcpu-arch.h File Reference

x86-specific vCPU interface.

```
#include <l4/sys/types.h>
```

Include dependency graph for \_\_vcpu-arch.h:



### Data Structures

- struct `l4_vcpu_regs_t`  
*vCPU registers.*
- struct `l4_vcpu_arch_state_t`  
*Architecture-specific vCPU state.*
- struct `l4_vcpu_ipc_regs_t`  
*vCPU message registers.*

### Typedefs

- typedef struct `l4_vcpu_regs_t` `l4_vcpu_regs_t`  
*vCPU registers.*
- typedef struct `l4_vcpu_arch_state_t` `l4_vcpu_arch_state_t`  
*Architecture-specific vCPU state.*
- typedef struct `l4_vcpu_ipc_regs_t` `l4_vcpu_ipc_regs_t`  
*vCPU message registers.*

### Enumerations

- enum { `L4_VCPU_STATE_VERSION` = 0x45 , `L4_VCPU_STATE_SIZE` = 0x200 , `L4_VCPU_STATE_EXT_SIZE` = `L4_PAGESIZE` }
  - enum `l4_vcpu_state_offset` { `L4_VCPU_OFFSET_EXT_STATE` = 0x400 , `L4_VCPU_OFFSET_EXT_INFOS` = 0x200 }
- Offsets for vCPU state layouts.*

## 16.79.1 Detailed Description

x86-specific vCPU interface.

Definition in file [\\_\\_vcpu-arch.h](#).

## 16.79.2 Enumeration Type Documentation

### 16.79.2.1 anonymous enum

anonymous enum

Enumerator

L4_VCPU_STATE_VERSION	Architecture-specific version ID. This ID must match the version field in the <a href="#">l4_vcpu_state_t</a> structure after enabling vCPU mode or extended vCPU mode for a thread.
-----------------------	--

Definition at line 27 of file [\\_\\_vcpu-arch.h](#).

## 16.80 [\\_\\_vcpu-arch.h](#)

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00023 #pragma once
00024
00025 #include <l4/sys/types.h>
00026
00027 enum
00028 {
00035     L4_VCPU_STATE_VERSION = 0x45,
00036
00037     L4_VCPU_STATE_SIZE = 0x200,
00038     L4_VCPU_STATE_EXT_SIZE = L4_PAGESIZE,
00039 };
00040
00045 enum L4_vcpu_state_offset
00046 {
00047     L4_VCPU_OFFSET_EXT_STATE = 0x400,
00048     L4_VCPU_OFFSET_EXT_INFOS = 0x200,
00049 };
00050
00055 typedef struct l4_vcpu_regs_t
00056 {
00057     l4_umword_t es;
00058     l4_umword_t ds;
00059     l4_umword_t gs;
00060     l4_umword_t fs;

```

```

00062  l4_umword_t di;
00063  l4_umword_t si;
00064  l4_umword_t bp;
00065  l4_umword_t pfa;
00066  l4_umword_t bx;
00067  l4_umword_t dx;
00068  l4_umword_t cx;
00069  l4_umword_t ax;
00071  l4_umword_t trapno;
00072  l4_umword_t err;
00074  l4_umword_t ip;
00075  l4_umword_t dummy1;
00076  l4_umword_t flags;
00077  l4_umword_t sp;
00078  l4_umword_t ss;
00079 } l4_vcpu_regs_t;
00080
00084 typedef struct l4_vcpu_arch_state_t {} l4_vcpu_arch_state_t;
00085
00090 typedef struct l4_vcpu_ipc_regs_t
00091 {
00092     l4_umword_t _res[2];
00093     l4_umword_t label;
00094     l4_umword_t _res2[3];
00095     l4_msgtag_t tag;
00096 } l4_vcpu_ipc_regs_t;

```

## 16.81 ktrace\_events.h

```

00001 /* Note, automatically generated from Fiasco binary */
00002 #pragma once
00003
00004 enum L4_ktrace_tbuf_entry_fixed
00005 {
00006     l4_ktrace_tbuf_unused = 0,
00007     l4_ktrace_tbuf_pf = 1,
00008     l4_ktrace_tbuf_ipc = 2,
00009     l4_ktrace_tbuf_ipc_res = 3,
00010     l4_ktrace_tbuf_ipc_trace = 4,
00011     l4_ktrace_tbuf_ke = 5,
00012     l4_ktrace_tbuf_ke_reg = 6,
00013     l4_ktrace_tbuf_breakpoint = 7,
00014     l4_ktrace_tbuf_ke_bin = 8,
00015     l4_ktrace_tbuf_dynentries = 9,
00016     l4_ktrace_tbuf_max = 128,
00017     l4_ktrace_tbuf_hidden = 128,
00018 };
00019
00020 typedef unsigned long L4_ktrace_t__Address;
00021 typedef unsigned long L4_ktrace_t__Cap_index;
00022 typedef void L4_ktrace_t__Context;
00023 typedef void L4_ktrace_t__Context__Drq;
00024 typedef unsigned L4_ktrace_t__Context__Drq_log__Type;
00025 typedef unsigned L4_ktrace_t__Cpu_number;
00026 typedef void L4_ktrace_t__Irq_base;
00027 typedef void L4_ktrace_t__Irq_chip;
00028 typedef void L4_ktrace_t__Kobject;
00029 typedef unsigned long L4_ktrace_t__L4_error;
00030 typedef unsigned long L4_ktrace_t__L4_msg_tag;
00031 typedef unsigned long L4_ktrace_t__L4_obj_ref;
00032 typedef unsigned L4_ktrace_t__L4_timeout_pair;
00033 typedef unsigned long L4_ktrace_t__Mword;
00034 typedef void L4_ktrace_t__Rcu_item;
00035 typedef void L4_ktrace_t__Sched_context;
00036 typedef long L4_ktrace_t__Smword;
00037 typedef void L4_ktrace_t__Space;
00038 typedef unsigned short L4_ktrace_t__Unsigned16;
00039 typedef unsigned int L4_ktrace_t__Unsigned32;
00040 typedef unsigned long long L4_ktrace_t__Unsigned64;
00041 typedef unsigned char L4_ktrace_t__Unsigned8;
00042 typedef void L4_ktrace_t__cxx__Type_info;
00043
00044 typedef struct __attribute__((packed))
00045 {
00046     L4_ktrace_t__Mword _number; /* 0+8 */
00047     L4_ktrace_t__Address _ip; /* 8+8 */
00048     L4_ktrace_t__Unsigned64 _tsc; /* 16+8 */
00049     L4_ktrace_t__Context *_ctx; /* 24+8 */
00050     L4_ktrace_t__Unsigned32 _pmc1; /* 32+4 */
00051     L4_ktrace_t__Unsigned32 _pmc2; /* 36+4 */
00052     L4_ktrace_t__Unsigned32 _kclock; /* 40+4 */
00053     L4_ktrace_t__Unsigned8 _type; /* 44+1 */
00054     L4_ktrace_t__Unsigned8 _cpu; /* 45+1 */

```

```

00055 union __attribute__((__packed__))
00056 {
00057     struct __attribute__((__packed__))
00058     {
00059         char __pre_pad[2];
00060         void *func; /* 48+8 */
00061         L4_ktrace_t__Context *thread; /* 56+8 */
00062         L4_ktrace_t__Context_Drq *rq; /* 64+8 */
00063         L4_ktrace_t__Cpu_number target_cpu; /* 72+4 */
00064         L4_ktrace_t__Context_Drq_log_Type type; /* 76+4 */
00065         char wait; /* 80+1 */
00066     } drq; /* 88 */
00067     struct __attribute__((__packed__))
00068     {
00069         char __pre_pad[2];
00070         L4_ktrace_t__Mword state; /* 48+8 */
00071         L4_ktrace_t__Mword ip; /* 56+8 */
00072         L4_ktrace_t__Mword sp; /* 64+8 */
00073         L4_ktrace_t__Mword space; /* 72+8 */
00074         L4_ktrace_t__Mword err; /* 80+8 */
00075         unsigned char type; /* 88+1 */
00076         unsigned char trap; /* 89+1 */
00077     } vcpu; /* 96 */
00078     struct __attribute__((__packed__))
00079     {
00080         char __pre_pad[2];
00081         L4_ktrace_t__Smword op; /* 48+8 */
00082         L4_ktrace_t__Cap_index buffer; /* 56+8 */
00083         L4_ktrace_t__Mword id; /* 64+8 */
00084         L4_ktrace_t__Mword ram; /* 72+8 */
00085         L4_ktrace_t__Mword newo; /* 80+8 */
00086     } factory; /* 88 */
00087     struct __attribute__((__packed__))
00088     {
00089         char __pre_pad[2];
00090         L4_ktrace_t__Mword gate_dbg_id; /* 48+8 */
00091         L4_ktrace_t__Mword thread_dbg_id; /* 56+8 */
00092         L4_ktrace_t__Mword label; /* 64+8 */
00093     } gate; /* 72 */
00094     struct __attribute__((__packed__))
00095     {
00096         char __pre_pad[2];
00097         L4_ktrace_t__Irq_base *obj; /* 48+8 */
00098         L4_ktrace_t__Irq_chip *chip; /* 56+8 */
00099         L4_ktrace_t__Mword pin; /* 64+8 */
00100     } irq; /* 72 */
00101     struct __attribute__((__packed__))
00102     {
00103         char __pre_pad[2];
00104         L4_ktrace_t__Kobject *obj; /* 48+8 */
00105         L4_ktrace_t__Mword id; /* 56+8 */
00106         L4_ktrace_t__cxx_Type_info *type; /* 64+8 */
00107         L4_ktrace_t__Mword ram; /* 72+8 */
00108     } destroy; /* 80 */
00109     struct __attribute__((__packed__))
00110     {
00111         char __pre_pad[2];
00112         L4_ktrace_t__Cpu_number cpu; /* 48+4 */
00113         char __pad_1[4];
00114         L4_ktrace_t__Rcu_item *item; /* 56+8 */
00115         void *cb; /* 64+8 */
00116         unsigned char event; /* 72+1 */
00117     } rcu; /* 80 */
00118     struct __attribute__((__packed__))
00119     {
00120         char __pre_pad[2];
00121         L4_ktrace_t__Mword id; /* 48+8 */
00122         L4_ktrace_t__Mword mask; /* 56+8 */
00123         L4_ktrace_t__Mword fpage; /* 64+8 */
00124         char map; /* 72+1 */
00125     } tmap; /* 80 */
00126     struct __attribute__((__packed__))
00127     {
00128         char __pre_pad[2];
00129         L4_ktrace_t__Address _address; /* 48+8 */
00130         int _len; /* 56+4 */
00131         char __pad_1[4];
00132         L4_ktrace_t__Mword _value; /* 64+8 */
00133         int _mode; /* 72+4 */
00134     } bp; /* 80 */
00135     struct __attribute__((__packed__))
00136     {
00137         char __pre_pad[2];
00138         L4_ktrace_t__Context *dst; /* 48+8 */
00139         L4_ktrace_t__Context *dst_orig; /* 56+8 */
00140         L4_ktrace_t__Address kernel_ip; /* 64+8 */
00141         L4_ktrace_t__Mword lock_cnt; /* 72+8 */

```

```

00142     L4_ktrace_t__Space *from_space; /* 80+8 */
00143     L4_ktrace_t__Sched_context *from_sched; /* 88+8 */
00144     L4_ktrace_t__Mword from_prio; /* 96+8 */
00145 } context_switch; /* 104 */
00146 struct __attribute__((__packed__))
00147 {
00148 } empty; /* 48 */
00149 struct __attribute__((__packed__))
00150 {
00151     char __pre_pad[2];
00152     L4_ktrace_t__L4_msg_tag _tag; /* 48+8 */
00153     L4_ktrace_t__Mword _dword[2]; /* 56+16 */
00154     L4_ktrace_t__L4_obj_ref _dst; /* 72+8 */
00155     L4_ktrace_t__Mword _dbg_id; /* 80+8 */
00156     L4_ktrace_t__Mword _label; /* 88+8 */
00157     L4_ktrace_t__L4_timeout_pair _timeout; /* 96+4 */
00158 } ipc; /* 104 */
00159 struct __attribute__((__packed__))
00160 {
00161     char __pre_pad[2];
00162     L4_ktrace_t__L4_msg_tag _tag; /* 48+8 */
00163     L4_ktrace_t__Mword _dword[2]; /* 56+16 */
00164     L4_ktrace_t__L4_error _result; /* 72+8 */
00165     L4_ktrace_t__Mword _from; /* 80+8 */
00166     L4_ktrace_t__Mword _pair_event; /* 88+8 */
00167     L4_ktrace_t__Unsigned8 _have_snd; /* 96+1 */
00168     L4_ktrace_t__Unsigned8 _is_np; /* 97+1 */
00169 } ipc_res; /* 104 */
00170 struct __attribute__((__packed__))
00171 {
00172     char __pre_pad[2];
00173     L4_ktrace_t__Unsigned64 _snd_tsc; /* 48+8 */
00174     L4_ktrace_t__L4_msg_tag _result; /* 56+8 */
00175     L4_ktrace_t__L4_obj_ref _snd_dst; /* 64+8 */
00176     L4_ktrace_t__Mword _rcv_dst; /* 72+8 */
00177     L4_ktrace_t__Unsigned8 _snd_desc; /* 80+1 */
00178     L4_ktrace_t__Unsigned8 _rcv_desc; /* 81+1 */
00179 } ipc_trace; /* 88 */
00180 struct __attribute__((__packed__))
00181 {
00182     char __pre_pad[2];
00183     union __attribute__((__packed__)) {
00184         char msg[80]; /* 0+80 */
00185         struct __attribute__((__packed__)) {
00186             char tag[2]; /* 0+2 */
00187             char __pad_l[6];
00188             char *ptr; /* 8+8 */
00189         } mptr; /* 0+16 */
00190     } msg; /* 48+80 */
00191 } ke; /* 128 */
00192 struct __attribute__((__packed__))
00193 {
00194     char _msg[80]; /* 46+80 */
00195 } ke_bin; /* 128 */
00196 struct __attribute__((__packed__))
00197 {
00198     char __pre_pad[2];
00199     L4_ktrace_t__Mword v[3]; /* 48+24 */
00200     union __attribute__((__packed__)) {
00201         char msg[56]; /* 0+56 */
00202         struct __attribute__((__packed__)) {
00203             char tag[2]; /* 0+2 */
00204             char __pad_l[6];
00205             char *ptr; /* 8+8 */
00206         } mptr; /* 0+16 */
00207     } msg; /* 72+56 */
00208 } ke_reg; /* 128 */
00209 struct __attribute__((__packed__))
00210 {
00211     char __pre_pad[2];
00212     L4_ktrace_t__Address _pfa; /* 48+8 */
00213     L4_ktrace_t__Mword _error; /* 56+8 */
00214     L4_ktrace_t__Space *_space; /* 64+8 */
00215 } pf; /* 72 */
00216 struct __attribute__((__packed__))
00217 {
00218     unsigned short mode; /* 46+2 */
00219     L4_ktrace_t__Context *owner; /* 48+8 */
00220     unsigned short id; /* 56+2 */
00221     unsigned short prio; /* 58+2 */
00222     long left; /* 60+8 */
00223     unsigned long quantum; /* 68+8 */
00224 } sched; /* 80 */
00225 struct __attribute__((__packed__))
00226 {
00227     char _trapno; /* 46+1 */
00228     L4_ktrace_t__Unsigned16 _error; /* 47+2 */

```

```

00229     L4_ktrace_t_Mword _rbp; /* 49+8 */
00230     L4_ktrace_t_Mword _cr2; /* 57+8 */
00231     L4_ktrace_t_Mword _rax; /* 65+8 */
00232     L4_ktrace_t_Mword _rflags; /* 73+8 */
00233     L4_ktrace_t_Mword _rsp; /* 81+8 */
00234     L4_ktrace_t_Unsigned16 _cs; /* 89+2 */
00235     L4_ktrace_t_Unsigned16 _ds; /* 91+2 */
00236 } trap; /* 96 */
00237 struct __attribute__((__packed__))
00238 {
00239     char __padding[80]; /* 46+80 */
00240     char __post_pad[2]; /* 126+2 */
00241 } fullsize; /* 128 */
00242 struct __attribute__((__packed__))
00243 {
00244     char __pre_pad[2];
00245     L4_ktrace_t_Cap_index cap_idx; /* 48+8 */
00246 } ieh; /* 56 */
00247 struct __attribute__((__packed__))
00248 {
00249     char __pre_pad[2];
00250     L4_ktrace_t_Mword pfa; /* 48+8 */
00251     L4_ktrace_t_Cap_index cap_idx; /* 56+8 */
00252     L4_ktrace_t_Mword err; /* 64+8 */
00253 } ipfh; /* 72 */
00254 struct __attribute__((__packed__))
00255 {
00256     char __pre_pad[2];
00257     L4_ktrace_t_Mword id; /* 48+8 */
00258     L4_ktrace_t_Mword ip; /* 56+8 */
00259     L4_ktrace_t_Mword sp; /* 64+8 */
00260     L4_ktrace_t_Mword op; /* 72+8 */
00261 } exregs; /* 80 */
00262 struct __attribute__((__packed__))
00263 {
00264     char __pre_pad[2];
00265     L4_ktrace_t_Mword state; /* 48+8 */
00266     L4_ktrace_t_Address user_ip; /* 56+8 */
00267     L4_ktrace_t_Cpu_number src_cpu; /* 64+4 */
00268     L4_ktrace_t_Cpu_number target_cpu; /* 68+4 */
00269 } migration; /* 72 */
00270 struct __attribute__((__packed__))
00271 {
00272     char __pre_pad[2];
00273     L4_ktrace_t_Irq_base *obj; /* 48+8 */
00274     L4_ktrace_t_Address user_ip; /* 56+8 */
00275 } timer; /* 64 */
00276 struct __attribute__((__packed__))
00277 {
00278     char __pre_pad[2];
00279     L4_ktrace_t_Mword exitcode; /* 48+8 */
00280     L4_ktrace_t_Mword exitinfo1; /* 56+8 */
00281     L4_ktrace_t_Mword exitinfo2; /* 64+8 */
00282     L4_ktrace_t_Mword rip; /* 72+8 */
00283 } svm; /* 80 */
00284 } m;
00285 } l4_tracebuffer_entry_t;

```

## 16.82 ktrace\_events.h

```

00001 /* Note, automatically generated from Fiasco binary */
00002 #pragma once
00003
00004 enum L4_ktrace_tbuf_entry_fixed
00005 {
00006     l4_ktrace_tbuf_unused = 0,
00007     l4_ktrace_tbuf_pf = 1,
00008     l4_ktrace_tbuf_ipc = 2,
00009     l4_ktrace_tbuf_ipc_res = 3,
00010     l4_ktrace_tbuf_ipc_trace = 4,
00011     l4_ktrace_tbuf_ke = 5,
00012     l4_ktrace_tbuf_ke_reg = 6,
00013     l4_ktrace_tbuf_breakpoint = 7,
00014     l4_ktrace_tbuf_ke_bin = 8,
00015     l4_ktrace_tbuf_dynentries = 9,
00016     l4_ktrace_tbuf_max = 128,
00017     l4_ktrace_tbuf_hidden = 128,
00018 };
00019
00020 typedef unsigned long L4_ktrace_t_Address;
00021 typedef unsigned long L4_ktrace_t_Cap_index;
00022 typedef void L4_ktrace_t_Context;
00023 typedef void L4_ktrace_t_Context_Drq;

```



```

00024 typedef unsigned L4_ktrace_t__Context__Drq_log__Type;
00025 typedef unsigned L4_ktrace_t__Cpu_number;
00026 typedef void L4_ktrace_t__Irq_base;
00027 typedef void L4_ktrace_t__Irq_chip;
00028 typedef void L4_ktrace_t__Kobject;
00029 typedef unsigned long L4_ktrace_t__L4_error;
00030 typedef unsigned long L4_ktrace_t__L4_msg_tag;
00031 typedef unsigned long L4_ktrace_t__L4_obj_ref;
00032 typedef unsigned L4_ktrace_t__L4_timeout_pair;
00033 typedef unsigned long L4_ktrace_t__Mword;
00034 typedef void L4_ktrace_t__Rcu_item;
00035 typedef void L4_ktrace_t__Sched_context;
00036 typedef long L4_ktrace_t__Smword;
00037 typedef void L4_ktrace_t__Space;
00038 typedef unsigned int L4_ktrace_t__Unsigned32;
00039 typedef unsigned long long L4_ktrace_t__Unsigned64;
00040 typedef unsigned char L4_ktrace_t__Unsigned8;
00041 typedef void L4_ktrace_t__cxx__Type_info;
00042
00043 typedef struct __attribute__((packed))
00044 {
00045     L4_ktrace_t__Mword _number; /* 0+4 */
00046     L4_ktrace_t__Address_ip; /* 4+4 */
00047     L4_ktrace_t__Unsigned64 _tsc; /* 8+8 */
00048     L4_ktrace_t__Context *_ctx; /* 16+4 */
00049     L4_ktrace_t__Unsigned32 _pmc1; /* 20+4 */
00050     L4_ktrace_t__Unsigned32 _pmc2; /* 24+4 */
00051     L4_ktrace_t__Unsigned32 _kclock; /* 28+4 */
00052     L4_ktrace_t__Unsigned8 _type; /* 32+1 */
00053     L4_ktrace_t__Unsigned8 _cpu; /* 33+1 */
00054     union __attribute__((__packed__))
00055     {
00056         struct __attribute__((__packed__))
00057         {
00058             char __pre_pad[2];
00059             void *func; /* 36+4 */
00060             L4_ktrace_t__Context *thread; /* 40+4 */
00061             L4_ktrace_t__Context__Drq *rq; /* 44+4 */
00062             L4_ktrace_t__Cpu_number target_cpu; /* 48+4 */
00063             L4_ktrace_t__Context__Drq_log__Type type; /* 52+4 */
00064             char wait; /* 56+1 */
00065         } drq; /* 64 */
00066         struct __attribute__((__packed__))
00067         {
00068             char __pre_pad[2];
00069             L4_ktrace_t__Mword state; /* 36+4 */
00070             L4_ktrace_t__Mword ip; /* 40+4 */
00071             L4_ktrace_t__Mword sp; /* 44+4 */
00072             L4_ktrace_t__Mword space; /* 48+4 */
00073             L4_ktrace_t__Mword err; /* 52+4 */
00074             unsigned char type; /* 56+1 */
00075             unsigned char trap; /* 57+1 */
00076             } vcpu; /* 64 */
00077         struct __attribute__((__packed__))
00078         {
00079             char __pre_pad[2];
00080             L4_ktrace_t__Smword op; /* 36+4 */
00081             L4_ktrace_t__Cap_index buffer; /* 40+4 */
00082             L4_ktrace_t__Mword id; /* 44+4 */
00083             L4_ktrace_t__Mword ram; /* 48+4 */
00084             L4_ktrace_t__Mword newo; /* 52+4 */
00085             } factory; /* 56 */
00086         struct __attribute__((__packed__))
00087         {
00088             char __pre_pad[2];
00089             L4_ktrace_t__Mword gate_dbg_id; /* 36+4 */
00090             L4_ktrace_t__Mword thread_dbg_id; /* 40+4 */
00091             L4_ktrace_t__Mword label; /* 44+4 */
00092             } gate; /* 48 */
00093         struct __attribute__((__packed__))
00094         {
00095             char __pre_pad[2];
00096             L4_ktrace_t__Irq_base *obj; /* 36+4 */
00097             L4_ktrace_t__Irq_chip *chip; /* 40+4 */
00098             L4_ktrace_t__Mword pin; /* 44+4 */
00099             } irq; /* 48 */
00100         struct __attribute__((__packed__))
00101         {
00102             char __pre_pad[2];
00103             L4_ktrace_t__Kobject *obj; /* 36+4 */
00104             L4_ktrace_t__Mword id; /* 40+4 */
00105             L4_ktrace_t__cxx__Type_info *type; /* 44+4 */
00106             L4_ktrace_t__Mword ram; /* 48+4 */
00107             } destroy; /* 56 */
00108         struct __attribute__((__packed__))
00109         {
00110             char __pre_pad[2];

```

```

00111     L4_ktrace_t__Cpu_number cpu; /* 36+4 */
00112     L4_ktrace_t__Rcu_item *item; /* 40+4 */
00113     void *cb; /* 44+4 */
00114     unsigned char event; /* 48+1 */
00115 } rcu; /* 56 */
00116 struct __attribute__((__packed__))
00117 {
00118     char __pre_pad[2];
00119     L4_ktrace_t__Mword id; /* 36+4 */
00120     L4_ktrace_t__Mword mask; /* 40+4 */
00121     L4_ktrace_t__Mword fpage; /* 44+4 */
00122     char map; /* 48+1 */
00123 } tmap; /* 56 */
00124 struct __attribute__((__packed__))
00125 {
00126     char __pre_pad[2];
00127     L4_ktrace_t__Address _address; /* 36+4 */
00128     int _len; /* 40+4 */
00129     L4_ktrace_t__Mword _value; /* 44+4 */
00130     int _mode; /* 48+4 */
00131 } bp; /* 56 */
00132 struct __attribute__((__packed__))
00133 {
00134     char __pre_pad[2];
00135     L4_ktrace_t__Context *dst; /* 36+4 */
00136     L4_ktrace_t__Context *dst_orig; /* 40+4 */
00137     L4_ktrace_t__Address kernel_ip; /* 44+4 */
00138     L4_ktrace_t__Mword lock_cnt; /* 48+4 */
00139     L4_ktrace_t__Space *from_space; /* 52+4 */
00140     L4_ktrace_t__Sched_context *from_sched; /* 56+4 */
00141     L4_ktrace_t__Mword from_prio; /* 60+4 */
00142 } context_switch; /* 64 */
00143 struct __attribute__((__packed__))
00144 {
00145     } empty; /* 40 */
00146 struct __attribute__((__packed__))
00147 {
00148     char __pre_pad[2];
00149     L4_ktrace_t__L4_msg_tag _tag; /* 36+4 */
00150     L4_ktrace_t__Mword _dword[2]; /* 40+8 */
00151     L4_ktrace_t__L4_obj_ref _dst; /* 48+4 */
00152     L4_ktrace_t__Mword _dbg_id; /* 52+4 */
00153     L4_ktrace_t__Mword _label; /* 56+4 */
00154     L4_ktrace_t__L4_timeout_pair _timeout; /* 60+4 */
00155 } ipc; /* 64 */
00156 struct __attribute__((__packed__))
00157 {
00158     char __pre_pad[2];
00159     L4_ktrace_t__L4_msg_tag _tag; /* 36+4 */
00160     L4_ktrace_t__Mword _dword[2]; /* 40+8 */
00161     L4_ktrace_t__L4_error_result; /* 48+4 */
00162     L4_ktrace_t__Mword _from; /* 52+4 */
00163     L4_ktrace_t__Mword _pair_event; /* 56+4 */
00164     L4_ktrace_t__Unsigned8 _have_snd; /* 60+1 */
00165     L4_ktrace_t__Unsigned8 _is_np; /* 61+1 */
00166 } ipc_res; /* 64 */
00167 struct __attribute__((__packed__))
00168 {
00169     char __pre_pad[6];
00170     L4_ktrace_t__Unsigned64 _snd_tsc; /* 40+8 */
00171     L4_ktrace_t__L4_msg_tag _result; /* 48+4 */
00172     L4_ktrace_t__L4_obj_ref _snd_dst; /* 52+4 */
00173     L4_ktrace_t__Mword _rcv_dst; /* 56+4 */
00174     L4_ktrace_t__Unsigned8 _snd_desc; /* 60+1 */
00175     L4_ktrace_t__Unsigned8 _rcv_desc; /* 61+1 */
00176 } ipc_trace; /* 64 */
00177 struct __attribute__((__packed__))
00178 {
00179     char __pre_pad[2];
00180     union __attribute__((__packed__)) {
00181         char msg[24]; /* 0+24 */
00182         struct __attribute__((__packed__)) {
00183             char tag[2]; /* 0+2 */
00184             char __pad_1[2];
00185             char *ptr; /* 4+4 */
00186         } mptr; /* 0+8 */
00187     } msg; /* 36+24 */
00188 } ke; /* 64 */
00189 struct __attribute__((__packed__))
00190 {
00191     char _msg[24]; /* 34+24 */
00192 } ke_bin; /* 64 */
00193 struct __attribute__((__packed__))
00194 {
00195     char __pre_pad[2];
00196     L4_ktrace_t__Mword v[3]; /* 36+12 */
00197     union __attribute__((__packed__)) {

```

```

00198     char msg[12]; /* 0+12 */
00199     struct __attribute__((__packed__)) {
00200         char tag[2]; /* 0+2 */
00201         char __pad_1[2];
00202         char *ptr; /* 4+4 */
00203     } mptr; /* 0+8 */
00204     } msg; /* 48+12 */
00205 } ke_reg; /* 64 */
00206 struct __attribute__((__packed__))
00207 {
00208     char __pre_pad[2];
00209     L4_ktrace_t__Address _pfa; /* 36+4 */
00210     L4_ktrace_t__Mword _error; /* 40+4 */
00211     L4_ktrace_t__Space *_space; /* 44+4 */
00212 } pf; /* 48 */
00213 struct __attribute__((__packed__))
00214 {
00215     unsigned short mode; /* 34+2 */
00216     L4_ktrace_t__Context *owner; /* 36+4 */
00217     unsigned short id; /* 40+2 */
00218     unsigned short prio; /* 42+2 */
00219     long left; /* 44+4 */
00220     unsigned long quantum; /* 48+4 */
00221 } sched; /* 56 */
00222 struct __attribute__((__packed__))
00223 {
00224     char __pre_pad[2];
00225     L4_ktrace_t__Unsigned32 _error; /* 36+4 */
00226     L4_ktrace_t__Mword _cpsr; /* 40+4 */
00227     L4_ktrace_t__Mword _sp; /* 44+4 */
00228 } trap; /* 48 */
00229 struct __attribute__((__packed__))
00230 {
00231     char _padding[24]; /* 34+24 */
00232     char __post_pad[6]; /* 58+6 */
00233 } fullsize; /* 64 */
00234 struct __attribute__((__packed__))
00235 {
00236     char __pre_pad[2];
00237     L4_ktrace_t__Cap_index cap_idx; /* 36+4 */
00238 } ieh; /* 40 */
00239 struct __attribute__((__packed__))
00240 {
00241     char __pre_pad[2];
00242     L4_ktrace_t__Mword pfa; /* 36+4 */
00243     L4_ktrace_t__Cap_index cap_idx; /* 40+4 */
00244     L4_ktrace_t__Mword err; /* 44+4 */
00245 } ipfh; /* 48 */
00246 struct __attribute__((__packed__))
00247 {
00248     char __pre_pad[2];
00249     L4_ktrace_t__Mword id; /* 36+4 */
00250     L4_ktrace_t__Mword ip; /* 40+4 */
00251     L4_ktrace_t__Mword sp; /* 44+4 */
00252     L4_ktrace_t__Mword op; /* 48+4 */
00253 } exregs; /* 56 */
00254 struct __attribute__((__packed__))
00255 {
00256     char __pre_pad[2];
00257     L4_ktrace_t__Mword state; /* 36+4 */
00258     L4_ktrace_t__Address user_ip; /* 40+4 */
00259     L4_ktrace_t__Cpu_number src_cpu; /* 44+4 */
00260     L4_ktrace_t__Cpu_number target_cpu; /* 48+4 */
00261 } migration; /* 56 */
00262 struct __attribute__((__packed__))
00263 {
00264     char __pre_pad[2];
00265     L4_ktrace_t__Irq_base *obj; /* 36+4 */
00266     L4_ktrace_t__Address user_ip; /* 40+4 */
00267 } timer; /* 48 */
00268 } m;
00269 } l4_tracebuffer_entry_t;

```

## 16.83 ktrace\_events.h

```

00001 /* Note, automatically generated from Fiasco binary */
00002 #pragma once
00003
00004 enum L4_ktrace_tbuf_entry_fixed
00005 {
00006     l4_ktrace_tbuf_unused = 0,
00007     l4_ktrace_tbuf_pf = 1,
00008     l4_ktrace_tbuf_ipc = 2,

```

```

00009  l4_ktrace_tbuf_ipc_res = 3,
00010  l4_ktrace_tbuf_ipc_trace = 4,
00011  l4_ktrace_tbuf_ke = 5,
00012  l4_ktrace_tbuf_ke_reg = 6,
00013  l4_ktrace_tbuf_breakpoint = 7,
00014  l4_ktrace_tbuf_ke_bin = 8,
00015  l4_ktrace_tbuf_dynentries = 9,
00016  l4_ktrace_tbuf_max = 128,
00017  l4_ktrace_tbuf_hidden = 128,
00018 };
00019
00020 typedef unsigned long L4_ktrace_t__Address;
00021 typedef unsigned long L4_ktrace_t__Cap_index;
00022 typedef void L4_ktrace_t__Context;
00023 typedef void L4_ktrace_t__Context__Drq;
00024 typedef unsigned L4_ktrace_t__Context__Drq_log__Type;
00025 typedef unsigned L4_ktrace_t__Cpu_number;
00026 typedef void L4_ktrace_t__Irq_base;
00027 typedef void L4_ktrace_t__Irq_chip;
00028 typedef void L4_ktrace_t__Kobject;
00029 typedef unsigned long L4_ktrace_t__L4_error;
00030 typedef unsigned long L4_ktrace_t__L4_msg_tag;
00031 typedef unsigned long L4_ktrace_t__L4_obj_ref;
00032 typedef unsigned L4_ktrace_t__L4_timeout_pair;
00033 typedef unsigned long L4_ktrace_t__Mword;
00034 typedef void L4_ktrace_t__Rcu_item;
00035 typedef void L4_ktrace_t__Sched_context;
00036 typedef long L4_ktrace_t__Smword;
00037 typedef void L4_ktrace_t__Space;
00038 typedef unsigned short L4_ktrace_t__Unsigned16;
00039 typedef unsigned int L4_ktrace_t__Unsigned32;
00040 typedef unsigned long long L4_ktrace_t__Unsigned64;
00041 typedef unsigned char L4_ktrace_t__Unsigned8;
00042 typedef void L4_ktrace_t__cxx__Type_info;
00043
00044 typedef struct __attribute__((packed))
00045 {
00046     L4_ktrace_t__Mword _number; /* 0+4 */
00047     L4_ktrace_t__Address _ip; /* 4+4 */
00048     L4_ktrace_t__Unsigned64 _tsc; /* 8+8 */
00049     L4_ktrace_t__Context *_ctx; /* 16+4 */
00050     L4_ktrace_t__Unsigned32 _pmc1; /* 20+4 */
00051     L4_ktrace_t__Unsigned32 _pmc2; /* 24+4 */
00052     L4_ktrace_t__Unsigned32 _kclock; /* 28+4 */
00053     L4_ktrace_t__Unsigned8 _type; /* 32+1 */
00054     L4_ktrace_t__Unsigned8 _cpu; /* 33+1 */
00055     union __attribute__((packed))
00056     {
00057         struct __attribute__((packed))
00058         {
00059             char __pre_pad[2];
00060             void *func; /* 36+4 */
00061             L4_ktrace_t__Context *thread; /* 40+4 */
00062             L4_ktrace_t__Context__Drq *rq; /* 44+4 */
00063             L4_ktrace_t__Cpu_number target_cpu; /* 48+4 */
00064             L4_ktrace_t__Context__Drq_log__Type type; /* 52+4 */
00065             char wait; /* 56+1 */
00066         } drq; /* 64 */
00067         struct __attribute__((packed))
00068         {
00069             char __pre_pad[2];
00070             L4_ktrace_t__Mword state; /* 36+4 */
00071             L4_ktrace_t__Mword ip; /* 40+4 */
00072             L4_ktrace_t__Mword sp; /* 44+4 */
00073             L4_ktrace_t__Mword space; /* 48+4 */
00074             L4_ktrace_t__Mword err; /* 52+4 */
00075             unsigned char type; /* 56+1 */
00076             unsigned char trap; /* 57+1 */
00077         } vcpu; /* 64 */
00078     } __attribute__((packed))
00079     {
00080         char __pre_pad[2];
00081         L4_ktrace_t__Smword op; /* 36+4 */
00082         L4_ktrace_t__Cap_index buffer; /* 40+4 */
00083         L4_ktrace_t__Mword id; /* 44+4 */
00084         L4_ktrace_t__Mword ram; /* 48+4 */
00085         L4_ktrace_t__Mword newo; /* 52+4 */
00086     } factory; /* 56 */
00087     struct __attribute__((packed))
00088     {
00089         char __pre_pad[2];
00090         L4_ktrace_t__Mword gate_dbg_id; /* 36+4 */
00091         L4_ktrace_t__Mword thread_dbg_id; /* 40+4 */
00092         L4_ktrace_t__Mword label; /* 44+4 */
00093     } gate; /* 48 */
00094     struct __attribute__((packed))
00095     {

```

```

00096     char __pre_pad[2];
00097     L4_ktrace_t__Irq_base *obj; /* 36+4 */
00098     L4_ktrace_t__Irq_chip *chip; /* 40+4 */
00099     L4_ktrace_t__Mword pin; /* 44+4 */
00100 } irq; /* 48 */
00101 struct __attribute__((__packed__))
00102 {
00103     char __pre_pad[2];
00104     L4_ktrace_t__Kobject *obj; /* 36+4 */
00105     L4_ktrace_t__Mword id; /* 40+4 */
00106     L4_ktrace_t__cxx_Type_info *type; /* 44+4 */
00107     L4_ktrace_t__Mword ram; /* 48+4 */
00108 } destroy; /* 56 */
00109 struct __attribute__((__packed__))
00110 {
00111     char __pre_pad[2];
00112     L4_ktrace_t__Cpu_number cpu; /* 36+4 */
00113     L4_ktrace_t__Rcu_item *item; /* 40+4 */
00114     void *cb; /* 44+4 */
00115     unsigned char event; /* 48+1 */
00116 } rcu; /* 56 */
00117 struct __attribute__((__packed__))
00118 {
00119     char __pre_pad[2];
00120     L4_ktrace_t__Mword id; /* 36+4 */
00121     L4_ktrace_t__Mword mask; /* 40+4 */
00122     L4_ktrace_t__Mword fpage; /* 44+4 */
00123     char map; /* 48+1 */
00124 } tmap; /* 56 */
00125 struct __attribute__((__packed__))
00126 {
00127     char __pre_pad[2];
00128     L4_ktrace_t__Address _address; /* 36+4 */
00129     int _len; /* 40+4 */
00130     L4_ktrace_t__Mword _value; /* 44+4 */
00131     int _mode; /* 48+4 */
00132 } bp; /* 56 */
00133 struct __attribute__((__packed__))
00134 {
00135     char __pre_pad[2];
00136     L4_ktrace_t__Context *dst; /* 36+4 */
00137     L4_ktrace_t__Context *dst_orig; /* 40+4 */
00138     L4_ktrace_t__Address kernel_ip; /* 44+4 */
00139     L4_ktrace_t__Mword lock_cnt; /* 48+4 */
00140     L4_ktrace_t__Space *from_space; /* 52+4 */
00141     L4_ktrace_t__Sched_context *from_sched; /* 56+4 */
00142     L4_ktrace_t__Mword from_prio; /* 60+4 */
00143 } context_switch; /* 64 */
00144 struct __attribute__((__packed__))
00145 {
00146 } empty; /* 40 */
00147 struct __attribute__((__packed__))
00148 {
00149     char __pre_pad[2];
00150     L4_ktrace_t__L4_msg_tag _tag; /* 36+4 */
00151     L4_ktrace_t__Mword _dword[2]; /* 40+8 */
00152     L4_ktrace_t__L4_obj_ref _dst; /* 48+4 */
00153     L4_ktrace_t__Mword _dbg_id; /* 52+4 */
00154     L4_ktrace_t__Mword _label; /* 56+4 */
00155     L4_ktrace_t__L4_timeout_pair _timeout; /* 60+4 */
00156 } ipc; /* 64 */
00157 struct __attribute__((__packed__))
00158 {
00159     char __pre_pad[2];
00160     L4_ktrace_t__L4_msg_tag _tag; /* 36+4 */
00161     L4_ktrace_t__Mword _dword[2]; /* 40+8 */
00162     L4_ktrace_t__L4_error_result; /* 48+4 */
00163     L4_ktrace_t__Mword _from; /* 52+4 */
00164     L4_ktrace_t__Mword _pair_event; /* 56+4 */
00165     L4_ktrace_t__Unsigned8 _have_snd; /* 60+1 */
00166     L4_ktrace_t__Unsigned8 _is_np; /* 61+1 */
00167 } ipc_res; /* 64 */
00168 struct __attribute__((__packed__))
00169 {
00170     char __pre_pad[2];
00171     L4_ktrace_t__Unsigned64 _snd_tsc; /* 36+8 */
00172     L4_ktrace_t__L4_msg_tag_result; /* 44+4 */
00173     L4_ktrace_t__L4_obj_ref _snd_dst; /* 48+4 */
00174     L4_ktrace_t__Mword _rcv_dst; /* 52+4 */
00175     L4_ktrace_t__Unsigned8 _snd_desc; /* 56+1 */
00176     L4_ktrace_t__Unsigned8 _rcv_desc; /* 57+1 */
00177 } ipc_trace; /* 64 */
00178 struct __attribute__((__packed__))
00179 {
00180     char __pre_pad[2];
00181     union __attribute__((__packed__)) {
00182         char msg[24]; /* 0+24 */

```

```

00183     struct __attribute__((__packed__)) {
00184         char tag[2]; /* 0+2 */
00185         char __pad_1[2];
00186         char *ptr; /* 4+4 */
00187     } mptr; /* 0+8 */
00188     } msg; /* 36+24 */
00189 } ke; /* 64 */
00190 struct __attribute__((__packed__))
00191 {
00192     char __msg[24]; /* 34+24 */
00193 } ke_bin; /* 64 */
00194 struct __attribute__((__packed__))
00195 {
00196     char __pre_pad[2];
00197     L4_ktrace_t_Mword v[3]; /* 36+12 */
00198     union __attribute__((__packed__)) {
00199         char msg[12]; /* 0+12 */
00200         struct __attribute__((__packed__)) {
00201             char tag[2]; /* 0+2 */
00202             char __pad_1[2];
00203             char *ptr; /* 4+4 */
00204             } mptr; /* 0+8 */
00205         } msg; /* 48+12 */
00206     } ke_reg; /* 64 */
00207 struct __attribute__((__packed__))
00208 {
00209     char __pre_pad[2];
00210     L4_ktrace_t_Address _pfa; /* 36+4 */
00211     L4_ktrace_t_Mword _error; /* 40+4 */
00212     L4_ktrace_t_Space *_space; /* 44+4 */
00213 } pf; /* 48 */
00214 struct __attribute__((__packed__))
00215 {
00216     unsigned short mode; /* 34+2 */
00217     L4_ktrace_t_Context *owner; /* 36+4 */
00218     unsigned short id; /* 40+2 */
00219     unsigned short prio; /* 42+2 */
00220     long left; /* 44+4 */
00221     unsigned long quantum; /* 48+4 */
00222 } sched; /* 56 */
00223 struct __attribute__((__packed__))
00224 {
00225     L4_ktrace_t_Unsigned8 _trapno; /* 34+1 */
00226     L4_ktrace_t_Unsigned16 _error; /* 35+2 */
00227     L4_ktrace_t_Mword _ebp; /* 37+4 */
00228     L4_ktrace_t_Mword _cr2; /* 41+4 */
00229     L4_ktrace_t_Mword _eax; /* 45+4 */
00230     L4_ktrace_t_Mword _eflags; /* 49+4 */
00231     L4_ktrace_t_Mword _esp; /* 53+4 */
00232     L4_ktrace_t_Unsigned16 _cs; /* 57+2 */
00233     L4_ktrace_t_Unsigned16 _ds; /* 59+2 */
00234 } trap; /* 64 */
00235 struct __attribute__((__packed__))
00236 {
00237     char _padding[24]; /* 34+24 */
00238     char __post_pad[6]; /* 58+6 */
00239 } fullsize; /* 64 */
00240 struct __attribute__((__packed__))
00241 {
00242     char __pre_pad[2];
00243     L4_ktrace_t_Cap_index cap_idx; /* 36+4 */
00244 } ieh; /* 40 */
00245 struct __attribute__((__packed__))
00246 {
00247     char __pre_pad[2];
00248     L4_ktrace_t_Mword pfa; /* 36+4 */
00249     L4_ktrace_t_Cap_index cap_idx; /* 40+4 */
00250     L4_ktrace_t_Mword err; /* 44+4 */
00251 } ipfh; /* 48 */
00252 struct __attribute__((__packed__))
00253 {
00254     char __pre_pad[2];
00255     L4_ktrace_t_Mword id; /* 36+4 */
00256     L4_ktrace_t_Mword ip; /* 40+4 */
00257     L4_ktrace_t_Mword sp; /* 44+4 */
00258     L4_ktrace_t_Mword op; /* 48+4 */
00259 } xregs; /* 56 */
00260 struct __attribute__((__packed__))
00261 {
00262     char __pre_pad[2];
00263     L4_ktrace_t_Mword state; /* 36+4 */
00264     L4_ktrace_t_Address user_ip; /* 40+4 */
00265     L4_ktrace_t_Cpu_number src_cpu; /* 44+4 */
00266     L4_ktrace_t_Cpu_number target_cpu; /* 48+4 */
00267 } migration; /* 56 */
00268 struct __attribute__((__packed__))
00269 {

```

```

00270     char __pre_pad[2];
00271     L4_ktrace_t __Irq_base *obj; /* 36+4 */
00272     L4_ktrace_t __Address user_ip; /* 40+4 */
00273 } timer; /* 48 */
00274 struct __attribute__((__packed__))
00275 {
00276     char __pre_pad[2];
00277     L4_ktrace_t __Mword exitcode; /* 36+4 */
00278     L4_ktrace_t __Mword exitinfo1; /* 40+4 */
00279     L4_ktrace_t __Mword exitinfo2; /* 44+4 */
00280     L4_ktrace_t __Mword rip; /* 48+4 */
00281 } svm; /* 56 */
00282 } m;
00283 } l4_tracebuffer_entry_t;

```

## 16.84 amd64/l4/sys/linkage.h File Reference

Linkage.

### Macros

- **#define L4\_CV**  
*Define calling convention.*

### 16.84.1 Detailed Description

Linkage.

Definition in file [linkage.h](#).

## 16.85 linkage.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #ifndef __L4_SYS_ARCH_AMD64_LINKAGE_H__
00026 #define __L4_SYS_ARCH_AMD64_LINKAGE_H__
00027
00028 #ifdef __ASSEMBLY__
00029
00030 #ifndef ENTRY
00031 #define ENTRY(name) \
00032     .globl name; \
00033     .p2align(2); \
00034     name:
00035
00036 #endif /* __ASSEMBLY__ */
00037 #endif /* ! ENTRY */
00038
00039 #define L4_FASTCALL(x)      x
00040 #define l4_fastcall
00041
00047 #define L4_CV
00048
00049 #endif /* ! __L4_SYS_ARCH_AMD64_LINKAGE_H__ */

```

## 16.86 arm/l4/sys/linkage.h File Reference

Linkage.

### Macros

- `#define L4_CV`  
*Define calling convention.*

### 16.86.1 Detailed Description

Linkage.

Definition in file [linkage.h](#).

## 16.87 linkage.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #ifndef __L4_SYS_ARCH_ARM_LINKAGE_H_
00025 #define __L4_SYS_ARCH_ARM_LINKAGE_H_
00026
00027 #ifdef __ASSEMBLY__
00028 #ifndef ENTRY
00029 #define ENTRY(name) \
00030     .globl name; \
00031     .p2align(2); \
00032     name:
00033 #endif
00034 #endif
00035
00036 #define L4_FASTCALL(x)  x
00037 #define l4_fastcall
00038
00044 #define L4_CV
00045
00046 #ifdef __PIC__
00047 #define L4_LONG_CALL
00048 #else
00049 #define L4_LONG_CALL __attribute__((long_call))
00050 #endif
00051
00052 #endif /* ! __L4_SYS_ARCH_ARM_LINKAGE_H_ */

```

## 16.88 x86/l4/sys/linkage.h File Reference

Linkage.



**Macros**

- **#define L4\_CV**

*Define calling convention.*

**16.88.1 Detailed Description**

Linkage.

Definition in file [linkage.h](#).

**16.89 linkage.h**

[Go to the documentation of this file.](#)

```

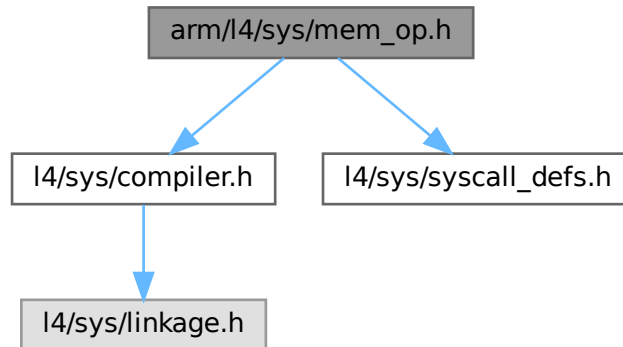
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  * Frank Mehnert <fm3@os.inf.tu-dresden.de>
00010  * economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #ifndef __L4__SYS__ARCH_X86__LINKAGE_H__
00026 #define __L4__SYS__ARCH_X86__LINKAGE_H__
00027
00028 #ifdef __ASSEMBLY__
00029
00030 #ifndef ENTRY
00031 #define ENTRY(name) \
00032     .globl name; \
00033     .p2align(2); \
00034     name:
00035
00036 #endif /* ! ENTRY */
00037 #endif /* __ASSEMBLY__ */
00038
00039 #define L4_FASTCALL(x) x __attribute__((regparm(3)))
00040 #define l4_fastcall __attribute__((regparm(3)))
00041
00047 #define L4_CV __attribute__((regparm(0)))
00048
00049 #endif /* ! __L4__SYS__ARCH_X86__LINKAGE_H__ */

```

**16.90 arm/l4/sys/mem\_op.h File Reference**

Memory access functions (ARM specific)

```
#include <l4/sys/compiler.h>
#include <l4/sys/syscall_defs.h>
Include dependency graph for mem_op.h:
```



## Enumerations

- enum `L4_mem_op_widths` { `L4_MEM_WIDTH_1BYTE` = 0 , `L4_MEM_WIDTH_2BYTE` = 1 , `L4_MEM_WIDTH_4BYTE` = 2 }

*Memory access width definitions.*

## Functions

- unsigned long `l4_mem_read` (unsigned long virtaddress, unsigned width)  
*Read user task memory from kernel privilege level.*
- void `l4_mem_write` (unsigned long virtaddress, unsigned width, unsigned long value)  
*Write user task memory from kernel privilege level.*
- unsigned long `l4_mem_arm_op_call` (unsigned long op, unsigned long va, unsigned long width, unsigned long value)  
*Implementations.*

### 16.90.1 Detailed Description

Memory access functions (ARM specific)

#### Date

2010-10

#### Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file `mem_op.h`.

## 16.91 mem\_op.h

[Go to the documentation of this file.](#)

```

00001
00009 /*
00010  * (c) 2010 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction. Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License. This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 #ifndef __L4SYS__INCLUDE__ARCH_ARM_MEM_OP_H__
00027 #define __L4SYS__INCLUDE__ARCH_ARM_MEM_OP_H__
00028
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/syscall_defs.h>
00031
00032 EXTERN_C_BEGIN
00033
00051 enum L4_mem_op_widths
00052 {
00053     L4_MEM_WIDTH_1BYTE = 0,
00054     L4_MEM_WIDTH_2BYTE = 1,
00055     L4_MEM_WIDTH_4BYTE = 2,
00056 };
00057
00070 L4_INLINE unsigned long
00071 l4_mem_read(unsigned long virtaddress, unsigned width);
00072
00085 L4_INLINE void
00086 l4_mem_write(unsigned long virtaddress, unsigned width,
00087             unsigned long value);
00088
00089 enum L4_mem_ops
00090 {
00091     L4_MEM_OP_MEM_READ = 0x10,
00092     L4_MEM_OP_MEM_WRITE = 0x11,
00093 };
00094
00098 L4_INLINE unsigned long
00099 l4_mem_arm_op_call(unsigned long op,
00100                   unsigned long va,
00101                   unsigned long width,
00102                   unsigned long value);
00103
00106 L4_INLINE unsigned long
00107 l4_mem_arm_op_call(unsigned long op,
00108                   unsigned long va,
00109                   unsigned long width,
00110                   unsigned long value)
00111 {
00112     register unsigned long _op    __asm__ ("r0") = op;
00113     register unsigned long _va    __asm__ ("r1") = va;
00114     register unsigned long _width __asm__ ("r2") = width;
00115     register unsigned long _value __asm__ ("r3") = value;
00116
00117     __asm__ __volatile__
00118     ("@ l4_cache_op_arm_call(start) \n\t"
00119      "mov     r5, %[sc] \n\t"
00120      "blx     __l4_sys_syscall \n\t"
00121      "@ l4_cache_op_arm_call(end) \n\t"
00122      :
00123      "r" (_op),
00124      "r" (_va),
00125      "r" (_width),
00126      "r" (_value)
00127      :
00128      [sc] "i" (L4_SYSCALL_MEM_OP),
00129      "0" (_op),
00130      "1" (_va),
00131      "2" (_width),
00132      "3" (_value)
00133      :
00134      "cc", "memory", "r5", "ip", "lr"
00135      );

```

```

00136
00137     return _value;
00138 }
00139
00140 L4_INLINE unsigned long
00141 l4_mem_read(unsigned long virtaddress, unsigned width)
00142 {
00143     return l4_mem_arm_op_call(L4_MEM_OP_MEM_READ, virtaddress, width, 0);
00144 }
00145
00146 L4_INLINE void
00147 l4_mem_write(unsigned long virtaddress, unsigned width,
00148             unsigned long value)
00149 {
00150     l4_mem_arm_op_call(L4_MEM_OP_MEM_WRITE, virtaddress, width, value);
00151 }
00152
00153 EXTERN_C_END
00154
00155 #endif /* ! __L4SYS__INCLUDE__ARCH_ARM__MEM_OP_H__ */

```

## 16.92 vm.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 #include <l4/sys/__vm-svm.h>
00026 #include <l4/sys/__vm-vmx.h>

```

## 16.93 arm/l4/sys/vm.h File Reference

ARM virtualization interface.

### Data Structures

- struct [l4\\_vm\\_tz\\_state](#)  
*state structure for TrustZone VMs*

### 16.93.1 Detailed Description

ARM virtualization interface.

Definition in file [vm.h](#).

## 16.94 vm.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00035 struct l4_vm_tz_state_mode
00036 {
00037     l4_umword_t sp;
00038     l4_umword_t lr;
00039     l4_umword_t spsr;
00040 };
00041
00042 struct l4_vm_tz_state_irq_inject
00043 {
00044     l4_uint32_t group;
00045     l4_uint32_t irqs[8];
00046 };
00047
00052 struct l4_vm_tz_state
00053 {
00054     l4_umword_t r[13]; // r0 - r12
00055
00056     l4_umword_t sp_usr;
00057     l4_umword_t lr_usr;
00058
00059     struct l4_vm_tz_state_mode irq;
00060
00061     l4_umword_t r_fiq[5]; // r8 - r12
00062     struct l4_vm_tz_state_mode fiq;
00063     struct l4_vm_tz_state_mode abt;
00064     struct l4_vm_tz_state_mode und;
00065     struct l4_vm_tz_state_mode svc;
00066
00067     l4_umword_t pc;
00068     l4_umword_t cpsr;
00069
00070     l4_umword_t pending_events;
00071     l4_uint32_t cpacr;
00072     l4_umword_t cp10_fpxc;
00073
00074     l4_umword_t pfs;
00075     l4_umword_t pfa;
00076     l4_umword_t exit_reason;
00077
00078     struct l4_vm_tz_state_irq_inject irq_inject;
00079 };
00080
00081 enum L4_vm_exit_reason
00082 {
00083     L4_vm_exit_reason_vmm_call    = 1,
00084     L4_vm_exit_reason_inst_abort = 2,
00085     L4_vm_exit_reason_data_abort = 3,
00086     L4_vm_exit_reason_irq        = 4,
00087     L4_vm_exit_reason_fiq        = 5,
00088     L4_vm_exit_reason_undef      = 6,
00089 };
00090
00091 L4_INLINE int
00092 l4_vm_tz_irq_inject(struct l4_vm_tz_state *state, unsigned irq);
00093
00094 L4_INLINE int
00095 l4_vm_tz_irq_inject(struct l4_vm_tz_state *state, unsigned irq)
00096 {
00097     if (irq > sizeof(state->irq_inject.irqs) * 8)
00098         return -L4_EINVAL;
00099 }

```

```

00100 unsigned g = irq / 32;
00101 state->irq_inject.group |= 1 « g;
00102 state->irq_inject.irqs[g] |= 1 « (irq & 31);
00103
00104 return 0;
00105 }

```

## 16.95 vm.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Henning Schild <hschild@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/sys/__vm-svm.h>
00027 #include <l4/sys/__vm-vmx.h>

```

## 16.96 amd64/l4/util/bitops\_arch.h File Reference

amd64 bit manipulation functions

### 16.96.1 Detailed Description

amd64 bit manipulation functions

Definition in file [bitops\\_arch.h](#).

## 16.97 bitops\_arch.h

[Go to the documentation of this file.](#)

```

00001 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2000-2009 Technische Universität Dresden (Germany)
00004  * Copyright (C) 2016, 2022 Kernkonzept GmbH. All rights reserved.
00005  * Author(s): Lars Reuther <reuther@os.inf.tu-dresden.de>
00006  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00007  *      Frank Mehnert <frank.mehnert@kernkonzept.com>
00008  */
00009
00016 #pragma once
00017
00018 /*
00019  * Note: The following Assembler statement may produce wrong code:
00020  *   asm volatile ("btsl %1, %2" : "=ccc"(r) : "Jr"(63), "m"(m) : "memory");
00021  *
00022  * The compiler might chose the first variant because the bit number is smaller
00023  * than 64. However, 'bts' is encoded as 32-bit variant ('btsl') and thus only
00024  * supports immediate bit values up to 31. Some assemblers support immediate
00025  * offsets > 31 by adapting the memory address accordingly but GAS does not.

```

```

00026  * With GAS, the instruction will encode an immediate value of 63 but the CPU
00027  * will set bit 31 instead of bit 63!
00028  *
00029  * Therefore, if we would use 'btsl' instead of 'btsq', the correct constraint
00030  * for the bit number parameter would be "Ir" instead of "Jr".
00031  */
00032
00033 EXTERN_C_BEGIN
00034
00035 /* set bit */
00036 #define __L4UTIL_BITOPS_HAVE_ARCH_SET_BIT
00037 L4_INLINE void
00038 l4util_set_bit(int b, volatile l4_umword_t * dest)
00039 {
00040     __asm__ __volatile__
00041     (
00042         "lock; btsq %1,%0    \n\t"
00043         :
00044         :
00045         "m"    (*dest), /* 0 mem, destination operand */
00046         "Jr"    ((l4_umword_t)b) /* 1, bit number */
00047         :
00048         "memory", "cc"
00049     );
00050 }
00051
00052 /* clear bit */
00053 #define __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
00054 L4_INLINE void
00055 l4util_clear_bit(int b, volatile l4_umword_t * dest)
00056 {
00057     __asm__ __volatile__
00058     (
00059         "lock; btrq %1,%0    \n\t"
00060         :
00061         :
00062         "m"    (*dest), /* 0 mem, destination operand */
00063         "Jr"    ((l4_umword_t)b) /* 1, bit number */
00064         :
00065         "memory", "cc"
00066     );
00067 }
00068
00069 /* change bit */
00070 #define __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
00071 L4_INLINE void
00072 l4util_complement_bit(int b, volatile l4_umword_t * dest)
00073 {
00074     __asm__ __volatile__
00075     (
00076         "lock; btcq %1,%0    \n\t"
00077         :
00078         :
00079         "m"    (*dest), /* 0 mem, destination operand */
00080         "Jr"    ((l4_umword_t)b) /* 1, bit number */
00081         :
00082         "memory", "cc"
00083     );
00084 }
00085
00086 /* test bit */
00087 #define __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
00088 L4_INLINE int
00089 l4util_test_bit(int b, const volatile l4_umword_t * dest)
00090 {
00091     l4_int8_t bit;
00092
00093     __asm__ __volatile__
00094     (
00095         "btq %2,%1    \n\t"
00096         :
00097         "=@ccc" (bit) /* 0, old bit value */
00098         :
00099         "m"    (*dest), /* 1 mem, destination operand */
00100         "Jr"    ((l4_umword_t)b) /* 2, bit number */
00101         :
00102         "memory"
00103     );
00104
00105     return bit;
00106 }
00107
00108 /* bit test and set */
00109 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
00110 L4_INLINE int
00111 l4util_bts(int b, volatile l4_umword_t * dest)
00112 {

```

```

00113     l4_int8_t bit;
00114
00115     __asm__ __volatile__
00116     (
00117         "lock; btsq %2,%1 \n\t"
00118         :
00119         "=@ccc" (bit) /* 0, old bit value */
00120         :
00121         "m" (*dest), /* 1 mem, destination operand */
00122         "Jr" ((l4_umword_t)b) /* 2, bit number */
00123         :
00124         "memory"
00125         );
00126
00127     return bit;
00128 }
00129
00130 /* bit test and reset */
00131 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
00132 L4_INLINE int
00133 l4util_btr(int b, volatile l4_umword_t * dest)
00134 {
00135     l4_int8_t bit;
00136
00137     __asm__ __volatile__
00138     (
00139         "lock; btrq %2,%1 \n\t"
00140         :
00141         "=@ccc" (bit) /* 0, old bit value */
00142         :
00143         "m" (*dest), /* 1 mem, destination operand */
00144         "Jr" ((l4_umword_t)b) /* 2, bit number */
00145         :
00146         "memory"
00147         );
00148
00149     return bit;
00150 }
00151
00152 /* bit test and complement */
00153 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
00154 L4_INLINE int
00155 l4util_btc(int b, volatile l4_umword_t * dest)
00156 {
00157     l4_int8_t bit;
00158
00159     __asm__ __volatile__
00160     (
00161         "lock; btcq %2,%1 \n\t"
00162         :
00163         "=@ccc" (bit) /* 0, old bit value */
00164         :
00165         "m" (*dest), /* 1 mem, destination operand */
00166         "Jr" ((l4_umword_t)b) /* 2, bit number */
00167         :
00168         "memory"
00169         );
00170
00171     return bit;
00172 }
00173
00174 /* bit scan reverse */
00175 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
00176 L4_INLINE int
00177 l4util_bsr(l4_umword_t word)
00178 {
00179     l4_umword_t tmp;
00180
00181     if (L4_UNLIKELY(word == 0))
00182         return -1;
00183
00184     __asm__ __volatile__
00185     (
00186         "bsrq %1,%0 \n\t"
00187         :
00188         "=r" (tmp) /* 0, index of most significant set bit */
00189         :
00190         "r" (word) /* 1, argument */
00191         );
00192
00193     return tmp;
00194 }
00195
00196 /* bit scan forward */
00197 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
00198 L4_INLINE int
00199 l4util_bsfl(l4_umword_t word)

```



```

00200 {
00201     l4_umword_t tmp;
00202
00203     if (L4_UNLIKELY(word == 0))
00204         return -1;
00205
00206     __asm__ __volatile__
00207     (
00208         "bsfq %1,%0 \n\t"
00209         :
00210         "=r" (tmp)          /* 0, index of least significant set bit */
00211         :
00212         "r" (word)          /* 1, argument */
00213         );
00214
00215     return tmp;
00216 }
00217
00218 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
00219 L4_INLINE int
00220 l4util_find_first_set_bit(const void * dest, l4_size_t size)
00221 {
00222     l4_mword_t dummy0, dummy1, res;
00223
00224     __asm__ __volatile__
00225     (
00226         "repe; scasq          \n\t"
00227         "jz     1f           \n\t"
00228         "lea    -8(%%rdi),%%rdi \n\t"
00229         "bsf    (%%rdi),%%rax  \n\t"
00230         "1:          \n\t"
00231         "sub    %%rbx,%%rdi    \n\t"
00232         "shl    $3,%%rdi      \n\t"
00233         "add    %%rdi,%%rax    \n\t"
00234         :
00235         "=a" (res), "=c" (dummy0), "=D" (dummy1)
00236         :
00237         "a" (0), "b" (dest), "c" ((size + 63) >> 6), "D" (dest)
00238         :
00239         "cc", "memory");
00240
00241     return res;
00242 }
00243
00244 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
00245 L4_INLINE int
00246 l4util_find_first_zero_bit(const void * dest, l4_size_t size)
00247 {
00248     l4_mword_t dummy0, dummy1, dummy2, res;
00249
00250     if (!size)
00251         return 0;
00252
00253     __asm__ __volatile__
00254     (
00255         "repe; scasq          \n\t"
00256         "je     1f           \n\t"
00257         "xor    -8(%%rdi),%%rax \n\t"
00258         "sub    $8,%%rdi      \n\t"
00259         "bsf    %%rax,%%rdx    \n\t"
00260         "1:          \n\t"
00261         "sub    %%rsi,%%rdi    \n\t"
00262         "shl    $3,%%rdi      \n\t"
00263         "add    %%rdi,%%rdx    \n\t"
00264         :
00265         "=a" (dummy0), "=c" (dummy1), "=d" (res), "=D" (dummy2)
00266         :
00267         "a" (~0UL), "c" ((size + 63) >> 6), "d" (0), "D" (dest), "S" (dest)
00268         :
00269         "cc", "memory");
00270
00271     return res;
00272 }
00273
00274 EXTERN_C_END

```

## 16.98 arm/l4/util/bitops\_arch.h File Reference

ARM specific implementation of bitops functions.

## 16.98.1 Detailed Description

ARM specific implementation of bitops functions.

Definition in file [bitops\\_arch.h](#).

## 16.99 bitops\_arch.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012 #ifndef __L4UTIL__ARCH_ARM__BITOPS_ARCH_H__
00013 #define __L4UTIL__ARCH_ARM__BITOPS_ARCH_H__
00014
00015
00016 #endif /* ! __L4UTIL__ARCH_ARM__BITOPS_ARCH_H__ */
```

## 16.100 x86/l4/util/bitops\_arch.h File Reference

x86 bit manipulation functions

### 16.100.1 Detailed Description

x86 bit manipulation functions

Definition in file [bitops\\_arch.h](#).

## 16.101 bitops\_arch.h

[Go to the documentation of this file.](#)

```
00001 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2000-2009 Technische Universität Dresden (Germany)
00004  * Copyright (C) 2016, 2022 Kernkonzept GmbH. All rights reserved.
00005  * Author(s): Lars Reuther <reuther@os.inf.tu-dresden.de>
00006  *             Frank Mehnert <frank.mehnert@kernkonzept.com>
00007  */
00008
00015 #pragma once
00016
00017 EXTERN_C_BEGIN
00018
00019 /* set bit */
00020 #define __L4UTIL_BITOPS_HAVE_ARCH_SET_BIT
00021 L4_INLINE void
00022 l4util_set_bit(int b, volatile l4_umword_t * dest)
00023 {
00024     __asm__ __volatile__
00025     (
00026         "lock; btsl  %1,%0  \n\t"
00027         :
00028         :
00029         "m"    (*dest), /* 0 mem, destination operand */
00030         "Ir"    (b)      /* 1, bit number */
00031         :
00032     );
00033 }
```

```

00032     "memory", "cc"
00033     );
00034 }
00035
00036 /* clear bit */
00037 #define __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
00038 L4_INLINE void
00039 l4util_clear_bit(int b, volatile l4_umword_t * dest)
00040 {
00041     __asm__ __volatile__
00042     (
00043         "lock; btrl %1,%0    \n\t"
00044         :
00045         :
00046         "m"    (*dest),      /* 0 mem, destination operand */
00047         "Ir"    (b)           /* 1,      bit number */
00048         :
00049         "memory", "cc"
00050         );
00051 }
00052
00053 /* change bit */
00054 #define __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
00055 L4_INLINE void
00056 l4util_complement_bit(int b, volatile l4_umword_t * dest)
00057 {
00058     __asm__ __volatile__
00059     (
00060         "lock; btc1 %1,%0    \n\t"
00061         :
00062         :
00063         "m"    (*dest),      /* 0 mem, destination operand */
00064         "Ir"    (b)           /* 1,      bit number */
00065         :
00066         "memory", "cc"
00067         );
00068 }
00069
00070 /* test bit */
00071 #define __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
00072 L4_INLINE int
00073 l4util_test_bit(int b, const volatile l4_umword_t * dest)
00074 {
00075     l4_int8_t bit;
00076
00077     __asm__ __volatile__
00078     (
00079         "btl %2,%1    \n\t"
00080         :
00081         "=@ccc" (bit)      /* 0,      old bit value */
00082         :
00083         "m"    (*dest),      /* 1 mem, destination operand */
00084         "Ir"    (b)           /* 2,      bit number */
00085         :
00086         "memory"
00087         );
00088
00089     return bit;
00090 }
00091
00092 /* bit test and set */
00093 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
00094 L4_INLINE int
00095 l4util_bts(int b, volatile l4_umword_t * dest)
00096 {
00097     l4_int8_t bit;
00098
00099     __asm__ __volatile__
00100     (
00101         "lock; btsl %2,%1    \n\t"
00102         :
00103         "=@ccc" (bit)      /* 0,      old bit value */
00104         :
00105         "m"    (*dest),      /* 1 mem, destination operand */
00106         "Ir"    (b)           /* 2,      bit number */
00107         :
00108         "memory"
00109         );
00110
00111     return bit;
00112 }
00113
00114 /* bit test and reset */
00115 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
00116 L4_INLINE int
00117 l4util_btr(int b, volatile l4_umword_t * dest)
00118 {

```

```

00119     l4_int8_t bit;
00120
00121     __asm__ __volatile__
00122     (
00123         "lock; btrl %2,%1 \n\t"
00124         :
00125         "=@ccc" (bit) /* 0, old bit value */
00126         :
00127         "m" (*dest), /* 1 mem, destination operand */
00128         "Ir" (b) /* 2, bit number */
00129         :
00130         "memory"
00131         );
00132
00133     return bit;
00134 }
00135
00136 /* bit test and complement */
00137 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
00138 L4_INLINE int
00139 l4util_btcc(int b, volatile l4_umword_t * dest)
00140 {
00141     l4_int8_t bit;
00142
00143     __asm__ __volatile__
00144     (
00145         "lock; btcl %2,%1 \n\t"
00146         :
00147         "=@ccc" (bit) /* 0, old bit value */
00148         :
00149         "m" (*dest), /* 1 mem, destination operand */
00150         "Ir" (b) /* 2, bit number */
00151         :
00152         "memory"
00153         );
00154
00155     return bit;
00156 }
00157
00158 /* bit scan reverse */
00159 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
00160 L4_INLINE int
00161 l4util_bsr(l4_umword_t word)
00162 {
00163     int tmp;
00164
00165     if (L4_UNLIKELY(word == 0))
00166         return -1;
00167
00168     __asm__ __volatile__
00169     (
00170         "bsrl %l,%0 \n\t"
00171         :
00172         "=r" (tmp) /* 0, index of most significant set bit */
00173         :
00174         "r" (word) /* 1, argument */
00175         );
00176
00177     return tmp;
00178 }
00179
00180 /* bit scan forward */
00181 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
00182 L4_INLINE int
00183 l4util_bsf(l4_umword_t word)
00184 {
00185     int tmp;
00186
00187     if (L4_UNLIKELY(word == 0))
00188         return -1;
00189
00190     __asm__ __volatile__
00191     (
00192         "bsfl %l,%0 \n\t"
00193         :
00194         "=r" (tmp) /* 0, index of least significant set bit */
00195         :
00196         "r" (word) /* 1, argument */
00197         );
00198
00199     return tmp;
00200 }
00201
00202 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
00203 L4_INLINE int
00204 l4util_find_first_set_bit(const void * dest, l4_size_t size)
00205 {

```

```

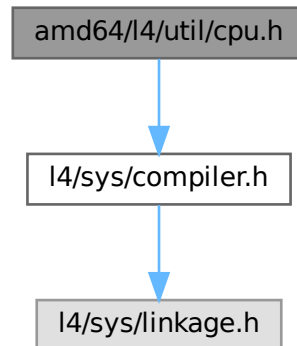
00206  l4_mword_t dummy0, dummy1, res;
00207
00208  __asm__ __volatile__
00209  (
00210      "repe; scasl                \n\t"
00211      "jz    lf                  \n\t"
00212      "lea   -4(%%edi),%%edi      \n\t"
00213      "bsf   (%%edi),%%eax        \n\t"
00214      "1:                                \n\t"
00215      "sub   %%esi,%%edi          \n\t"
00216      "shl   $3,%%edi            \n\t"
00217      "add   %%edi,%%eax          \n\t"
00218      :
00219      "=a" (res), "=c" (dummy0), "=D" (dummy1)
00220      :
00221      "a" (0), "c" ((size + 31) >> 5), "D" (dest), "S" (dest)
00222      :
00223      "cc", "memory");
00224
00225  return res;
00226 }
00227
00228 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
00229 L4_INLINE int
00230 l4util_find_first_zero_bit(const void * dest, l4_size_t size)
00231 {
00232     l4_mword_t dummy0, dummy1, dummy2, res;
00233
00234     if (!size)
00235         return 0;
00236
00237     __asm__ __volatile__
00238     (
00239         "repe; scasl                \n\t"
00240         "je    lf                  \n\t"
00241         "xor   -4(%%edi),%%eax      \n\t"
00242         "sub   $4,%%edi            \n\t"
00243         "bsf   %%eax,%%edx          \n\t"
00244         "1:                                \n\t"
00245         "sub   %%esi,%%edi          \n\t"
00246         "shl   $3,%%edi            \n\t"
00247         "add   %%edi,%%edx          \n\t"
00248         :
00249         "=a" (dummy0), "=c" (dummy1), "=d" (res), "=D" (dummy2)
00250         :
00251         "a" (~0UL), "c" ((size + 31) >> 5), "d" (0), "D" (dest), "S" (dest)
00252         :
00253         "cc", "memory");
00254
00255     return res;
00256 }
00257
00258 EXTERN_C_END

```

## 16.102 amd64/l4/util/cpu.h File Reference

CPU related functions.

```
#include <l4/sys/compiler.h>  
Include dependency graph for cpu.h:
```



## Functions

- int [l4util\\_cpu\\_has\\_cpuid](#) (void)  
*Check whether the CPU supports the "cpuid" instruction.*
- unsigned int [l4util\\_cpu\\_capabilities](#) (void)  
*Returns the CPU capabilities if the "cpuid" instruction is available.*
- unsigned int [l4util\\_cpu\\_capabilities\\_nocheck](#) (void)  
*Returns the CPU capabilities.*
- void [l4util\\_cpu\\_cpuid](#) (unsigned long mode, unsigned long \*eax, unsigned long \*ebx, unsigned long \*ecx, unsigned long \*edx)  
*Generic CPUID access function.*

### 16.102.1 Detailed Description

CPU related functions.

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [cpu.h](#).

## 16.103 cpu.h

[Go to the documentation of this file.](#)

```

00001
00007 /*
00008  * (c) 2004-2009 Author(s)
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU Lesser General Public License 2.1.
00012  * Please see the COPYING-LGPL-2.1 file for details.
00013  */
00014
00015 #ifndef __L4_UTIL_CPU_H
00016 #define __L4_UTIL_CPU_H
00017
00018 #include <l4/sys/compiler.h>
00019
00020 EXTERN_C_BEGIN
00021
00022 L4_INLINE int          l4util_cpu_has_cpuid(void);
00023
00024 L4_INLINE unsigned int l4util_cpu_capabilities(void);
00025
00026 L4_INLINE unsigned int l4util_cpu_capabilities_nocheck(void);
00027
00028 L4_INLINE void
00029 l4util_cpu_cpuid(unsigned long mode,
00030                 unsigned long *eax, unsigned long *ebx,
00031                 unsigned long *ecx, unsigned long *edx);
00032
00033 static inline void
00034 l4util_cpu_pause(void)
00035 {
00036     __asm__ __volatile__ ("rep; nop");
00037 }
00038
00039 L4_INLINE int
00040 l4util_cpu_has_cpuid(void)
00041 {
00042     return 1;
00043 }
00044
00045 L4_INLINE void
00046 l4util_cpu_cpuid(unsigned long mode,
00047                 unsigned long *eax, unsigned long *ebx,
00048                 unsigned long *ecx, unsigned long *edx)
00049 {
00050     asm volatile("cpuid"
00051                 : "=a" (*eax),
00052                   "=b" (*ebx),
00053                   "=c" (*ecx),
00054                   "=d" (*edx)
00055                 : "a" (mode)
00056                 );
00057 }
00058
00059 L4_INLINE unsigned int
00060 l4util_cpu_capabilities_nocheck(void)
00061 {
00062     unsigned long dummy, capability;
00063
00064     /* get CPU capabilities */
00065     l4util_cpu_cpuid(1, &dummy, &dummy, &dummy, &capability);
00066
00067     return capability;
00068 }
00069
00070 L4_INLINE unsigned int
00071 l4util_cpu_capabilities(void)
00072 {
00073     if (!l4util_cpu_has_cpuid())
00074         return 0; /* CPU has not cpuid instruction */
00075
00076     return l4util_cpu_capabilities_nocheck();
00077 }
00078
00079 EXTERN_C_END
00080
00081 #endif

```

## 16.104 arm/l4/util/cpu.h File Reference

CPU related functions.

### 16.104.1 Detailed Description

CPU related functions.

Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file [cpu.h](#).

## 16.105 cpu.h

[Go to the documentation of this file.](#)

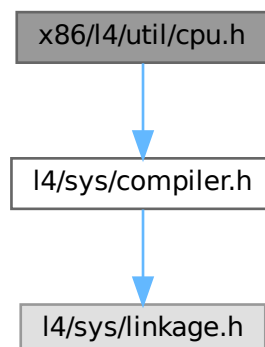
```
00001
00008 /*
00009  * (c) 2004-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
00014  */
00015
00016 #ifndef __L4_UTIL__ARCH_ARM__CPU_H__
00017 #define __L4_UTIL__ARCH_ARM__CPU_H__
00018
00019 /* Nothing yet */
00020
00021 #endif /* __L4_UTIL__ARCH_ARM__CPU_H__ */
```

## 16.106 x86/l4/util/cpu.h File Reference

CPU related functions.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for cpu.h:





## Functions

- int [l4util\\_cpu\\_has\\_cpuid](#) (void)  
*Check whether the CPU supports the "cpuid" instruction.*
- unsigned int [l4util\\_cpu\\_capabilities](#) (void)  
*Returns the CPU capabilities if the "cpuid" instruction is available.*
- unsigned int [l4util\\_cpu\\_capabilities\\_noccheck](#) (void)  
*Returns the CPU capabilities.*
- void [l4util\\_cpu\\_cpuid](#) (unsigned long mode, unsigned long \*eax, unsigned long \*ebx, unsigned long \*ecx, unsigned long \*edx)  
*Generic CPUID access function.*

### 16.106.1 Detailed Description

CPU related functions.

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [cpu.h](#).

## 16.107 cpu.h

[Go to the documentation of this file.](#)

```

00001
00007 /*
00008  * (c) 2004-2009 Author(s)
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU Lesser General Public License 2.1.
00012  * Please see the COPYING-LGPL-2.1 file for details.
00013  */
00014
00015 #ifndef __L4_UTIL_CPU_H
00016 #define __L4_UTIL_CPU_H
00017
00018 #include <l4/sys/compiler.h>
00019
00020 EXTERN_C_BEGIN
00021
00033 L4_INLINE int      l4util_cpu_has_cpuid(void);
00034
00041 L4_INLINE unsigned int l4util_cpu_capabilities(void);
00042
00048 L4_INLINE unsigned int l4util_cpu_capabilities_noccheck(void);
00049
00053 L4_INLINE void
00054 l4util_cpu_cpuid(unsigned long mode,
00055                 unsigned long *eax, unsigned long *ebx,
00056                 unsigned long *ecx, unsigned long *edx);
00057
00059 static inline void
00060 l4util_cpu_pause(void)
00061 {
00062     __asm__ __volatile__ ("rep; nop");
00063 }
00064
00065 L4_INLINE int
00066 l4util_cpu_has_cpuid(void)
00067 {
00068     unsigned long eax;
00069
00070     asm volatile("pushl %%ebx                \t\n"
00071                 "pushfl                    \t\n"
00072                 "popl %%eax                \t\n" /* get eflags */

```

```

00073         "movl %%eax, %%ebx \t\n" /* save it */
00074         "xorl $0x200000, %%eax \t\n" /* toggle ID bit */
00075         "pushl %%eax \t\n"
00076         "popfl \t\n" /* set again */
00077         "pushfl \t\n"
00078         "popl %%eax \t\n" /* get it again */
00079         "xorl %%ebx, %%eax \t\n"
00080         "pushl %%ebx \t\n"
00081         "popfl \t\n" /* restore saved flags */
00082         "popl %%ebx \t\n"
00083         : "=a" (eax)
00084         : /* no input */
00085         );
00086
00087     return eax & 0x200000;
00088 }
00089
00090 L4_INLINE void
00091 l4util_cpu_cpuid(unsigned long mode,
00092                 unsigned long *eax, unsigned long *ebx,
00093                 unsigned long *ecx, unsigned long *edx)
00094 {
00095     asm volatile("pushl %%ebx \t\n"
00096                 "cpuid \t\n"
00097                 "mov %%ebx, %%esi \t\n"
00098                 "popl %%ebx \t\n"
00099                 : "=a" (*eax),
00100                   "=S" (*ebx),
00101                   "=c" (*ecx),
00102                   "=d" (*edx)
00103                 : "a" (mode));
00104 }
00105
00106 L4_INLINE unsigned int
00107 l4util_cpu_capabilities_nocheck(void)
00108 {
00109     unsigned long dummy, capability;
00110
00111     /* get CPU capabilities */
00112     l4util_cpu_cpuid(1, &dummy, &dummy, &dummy, &capability);
00113
00114     return capability;
00115 }
00116
00117 L4_INLINE unsigned int
00118 l4util_cpu_capabilities(void)
00119 {
00120     if (!l4util_cpu_has_cpuid())
00121         return 0; /* CPU has not cpuid instruction */
00122
00123     return l4util_cpu_capabilities_nocheck();
00124 }
00125
00126 EXTERN_C_END
00127
00128 #endif
00129

```

## 16.108 amd64/l4/util/l4\_macros.h File Reference

Main function.

### 16.108.1 Detailed Description

Main function.

Date

08/29/2000

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [l4\\_macros.h](#).

## 16.109 l4\_macros.h

[Go to the documentation of this file.](#)

```
00001
00008 /*
00009  * (c) 2006-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
00014  */
00015
00016 #ifndef _L4UTIL__ARCH_AMD64__L4_MACROS_H
00017 #define _L4UTIL__ARCH_AMD64__L4_MACROS_H
00018
00019 #include_next <l4/util/l4_macros.h>
00020
00021 #ifndef l4_addr_fmt
00022 #   define l4_addr_fmt    "%016lx"
00023 #endif
00024
00025 #endif /* !_L4UTIL__ARCH_AMD64__L4_MACROS_H */
```

## 16.110 arm/l4/util/l4\_macros.h File Reference

Main function.

### 16.110.1 Detailed Description

Main function.

Date

08/29/2000

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [l4\\_macros.h](#).

## 16.111 l4\_macros.h

[Go to the documentation of this file.](#)

```
00001
00008 /*
00009  * (c) 2006-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
00014  */
00015
00016 #ifndef _L4UTIL__ARCH_ARM__L4_MACROS_H
00017 #define _L4UTIL__ARCH_ARM__L4_MACROS_H
00018
00019 #include_next <l4/util/l4_macros.h>
00020
00021 #ifndef l4_addr_fmt
00022 #   define l4_addr_fmt    "%08lx"
00023 #endif
00024
00025 #endif /* !_L4UTIL__ARCH_ARM__L4_MACROS_H */
```

## 16.112 I4/util/I4\_macros.h File Reference

Some useful generic macros, L4f version.

### 16.112.1 Detailed Description

Some useful generic macros, L4f version.

#### Date

11/12/2002

#### Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de)

Definition in file [I4\\_macros.h](#).

## 16.113 I4\_macros.h

[Go to the documentation of this file.](#)

```
00001 /*****
00008 */
00009 * (c) 2000-2009 Author(s)
00010 *     economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015
00016 /*****
00017
00018 #pragma once
00019
00020 /*****
00021 *** generic macros
00022 *****/
00023
00024 /* generate L4 thread id printf string */
00025 #ifndef l4util_idstr
00026 #   define l4util_idfmt          "%lx"
00027 #   define l4util_idfmt_adjust  "%04lx"
00028 #   define l4util_idstr(tid)    (tid >> L4_CAP_SHIFT)
00029 #endif
00030
```

## 16.114 x86/I4/util/I4\_macros.h File Reference

Main function.

### 16.114.1 Detailed Description

Main function.

Date

08/29/2000

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [l4\\_macros.h](#).

## 16.115 l4\_macros.h

[Go to the documentation of this file.](#)

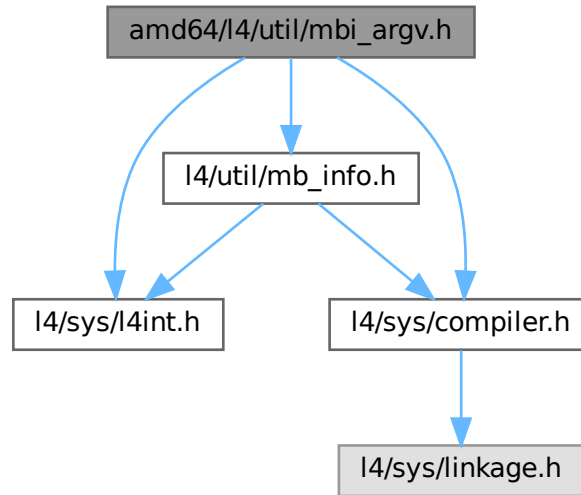
```
00001
00008 /*
00009  * (c) 2006-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
00014  */
00015
00016
00017 #ifndef _L4UTIL__ARCH_X86__L4_MACROS_H
00018 #define _L4UTIL__ARCH_X86__L4_MACROS_H
00019
00020 #include_next <l4/util/l4_macros.h>
00021
00022 #ifndef l4_addr_fmt
00023 #   define l4_addr_fmt    "%08lx"
00024 #endif
00025
00026 #endif /* !_L4UTIL__ARCH_X86__L4_MACROS_H */
```

## 16.116 amd64/l4/util/mbi\_argv.h File Reference

command line handling

```
#include <l4/sys/l4int.h>
#include <l4/util/mb_info.h>
```

```
#include <l4/sys/compiler.h>
Include dependency graph for mbi_argv.h:
```



### 16.116.1 Detailed Description

command line handling

Date

2003

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [mbi\\_argv.h](#).

### 16.117 mbi\_argv.h

[Go to the documentation of this file.](#)

```

00001
00008 /*
00009  * (c) 2003-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
00014  */
00015
00016 #ifndef L4UTIL_MBI_ARGV
00017 #define L4UTIL_MBI_ARGV
00018
00019 #include <l4/sys/l4int.h>
```

```

00020 #include <l4/util/mb_info.h>
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00025 L4_CV void l4util_mbi_to_argv(l4_mword_t flag, l4util_mb_info_t *mbi);
00026
00027 extern int l4util_argc;
00028 extern char *l4util_argv[];
00029
00030 EXTERN_C_END
00031
00032 #endif
00033

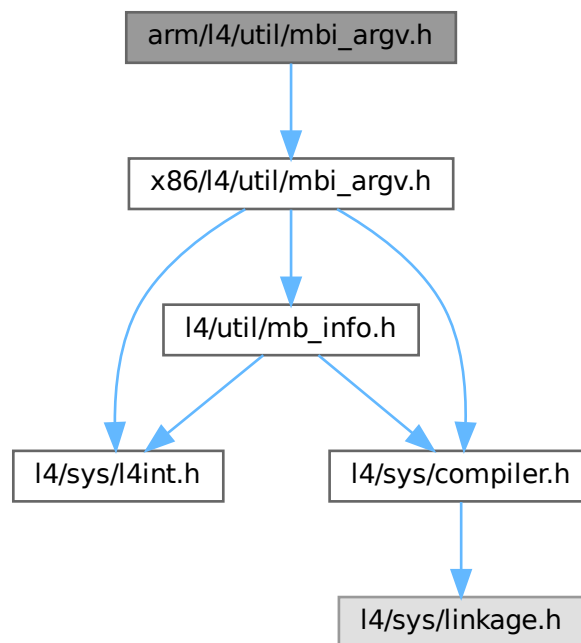
```

## 16.118 arm/l4/util/mbi\_argv.h File Reference

Multiboot.

```
#include <x86/l4/util/mbi_argv.h>
```

Include dependency graph for mbi\_argv.h:



### 16.118.1 Detailed Description

Multiboot.

Definition in file [mbi\\_argv.h](#).

## 16.119 mbi\_argv.h

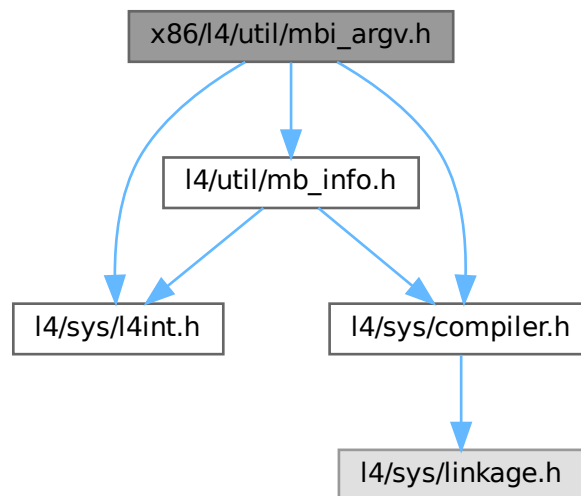
[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012 /* If this persists, move it to a generic place */
00013 #include <x86/l4/util/mbi_argv.h>
```

## 16.120 x86/l4/util/mbi\_argv.h File Reference

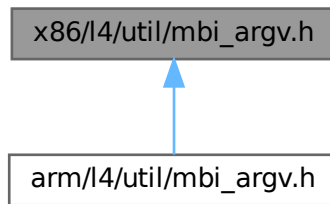
command line handling

```
#include <l4/sys/l4int.h>
#include <l4/util/mb_info.h>
#include <l4/sys/compiler.h>
Include dependency graph for mbi_argv.h:
```





This graph shows which files directly or indirectly include this file:



## 16.120.1 Detailed Description

command line handling

### Date

2003

### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [mbi\\_argv.h](#).

## 16.121 mbi\_argv.h

[Go to the documentation of this file.](#)

```

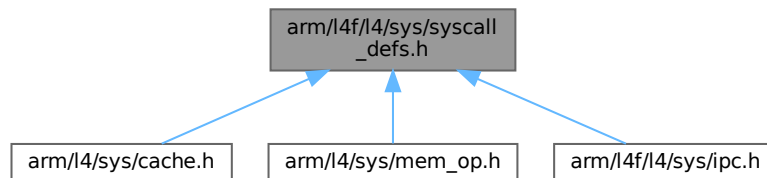
00001
00008 /*
00009  * (c) 2003-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
00014  */
00015
00016 #ifndef L4UTIL_MBI_ARGV
00017 #define L4UTIL_MBI_ARGV
00018
00019 #include <l4/sys/l4int.h>
00020 #include <l4/util/mb_info.h>
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00025 void l4util_mbi_to_argv(l4_mword_t flag, l4util_mb_info_t *mbi);
00026
00027 extern int l4util_argc;
00028 extern char *l4util_argv[];
00029
00030 EXTERN_C_END
00031
00032 #endif
00033

```

## 16.122 arm/l4f/l4/sys/syscall\_defs.h File Reference

Syscall entry definitions.

This graph shows which files directly or indirectly include this file:



### 16.122.1 Detailed Description

Syscall entry definitions.

Definition in file [syscall\\_defs.h](#).

## 16.123 syscall\_defs.h

[Go to the documentation of this file.](#)

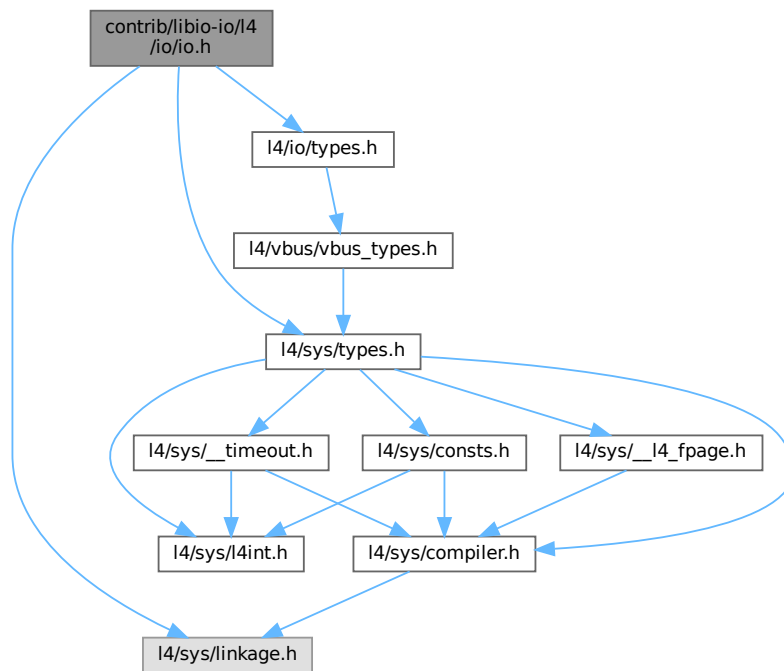
```

00001
00005 /*
00006  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #ifndef __L4SYS__ARCH_ARM__L4API_L4F__SYSCALL_DEFS_H__
00023 #define __L4SYS__ARCH_ARM__L4API_L4F__SYSCALL_DEFS_H__
00024
00025 #define L4_SYSCALL_INVOKE      (0)
00026 #define L4_SYSCALL_MEM_OP      (1)
00027
00028 #endif /* __L4SYS__ARCH_ARM__L4API_L4F__SYSCALL_DEFS_H__ */

```

## 16.124 contrib/libio-io/l4/io/io.h File Reference

```
#include <l4/sys/types.h>
#include <l4/sys/linkage.h>
#include <l4/io/types.h>
Include dependency graph for io.h:
```



### Functions

- `l4_cap_idx_t l4io_request_icu` (void)  
*Request the ICU object of the client.*
- `long l4io_request_iomem` (`l4_addr_t` phys, unsigned long size, int flags, `l4_addr_t` \*virt)  
*Request an IO memory region.*
- `long l4io_request_iomem_region` (`l4_addr_t` phys, `l4_addr_t` virt, unsigned long size, int flags)  
*Request an IO memory region and map it to a specified region.*
- `long l4io_release_iomem` (`l4_addr_t` virt, unsigned long size)  
*Release an IO memory region.*
- `long l4io_request_ioport` (unsigned portnum, unsigned len)  
*Request an IO port region.*
- `long l4io_release_ioport` (unsigned portnum, unsigned len)  
*Release an IO port region.*
- `l4io_device_handle_t l4io_get_root_device` (void)  
*Get root device handle of the device bus.*
- `int l4io_iterate_devices` (`l4io_device_handle_t` \*devhandle, `l4io_device_t` \*dev, `l4io_resource_handle_t` \*reshandle)  
*Iterate over the device bus.*

- int [l4io\\_lookup\\_device](#) (const char \*devname, l4io\_device\_handle\_t \*dev\_handle, [l4io\\_device\\_t](#) \*dev, l4io\_resource\_handle\_t \*res\_handle)  
*Find a device by name.*
- int [l4io\\_lookup\\_resource](#) (l4io\_device\_handle\_t devhandle, enum [l4io\\_resource\\_types\\_t](#) type, l4io\_resource\_handle\_t \*reshandle, [l4io\\_resource\\_t](#) \*res)  
*Request a specific resource from a device description.*
- [l4\\_addr\\_t](#) [l4io\\_request\\_resource\\_iomem](#) (l4io\_device\_handle\_t devhandle, l4io\_resource\_handle\_t \*reshandle)  
*Request IO memory.*
- void [l4io\\_request\\_all\\_ioports](#) (void(\*res\_cb)([l4vbus\\_resource\\_t](#) const \*res))  
*Request all available IO-port resources.*
- int [l4io\\_has\\_resource](#) (enum [l4io\\_resource\\_types\\_t](#) type, [l4vbus\\_paddr\\_t](#) start, [l4vbus\\_paddr\\_t](#) end)  
*Check if a resource is available.*

## 16.124.1 Function Documentation

### 16.124.1.1 l4io\_get\_root\_device()

```
l4io_device_handle_t l4io_get_root_device (
    void ) [inline]
```

Get root device handle of the device bus.

#### Returns

root device handle

Definition at line 258 of file [io.h](#).

### 16.124.1.2 l4io\_iterate\_devices()

```
int l4io_iterate_devices (
    l4io_device_handle_t * devhandle,
    l4io\_device\_t * dev,
    l4io_resource_handle_t * reshandle )
```

Iterate over the device bus.

#### Parameters

in, out	<i>devhandle</i>	Device handle to start iterating at. The next device handle is returned here.
out	<i>dev</i>	Device information, may be NULL.
out	<i>reshandle</i>	Resource handle.

#### Returns

0 on success, error code otherwise

**16.124.1.3 l4io\_request\_all\_ioports()**

```
void l4io_request_all_ioports (
    void(*) (l4vbus_resource_t const *res) res_cb )
```

Request all available IO-port resources.

**Parameters**

<i>res_cb</i>	Callback function called for every port resource found, give NULL for no callback.
---------------	--

**16.124.1.4 l4io\_request\_icu()**

```
l4_cap_idx_t l4io_request_icu (
    void )
```

Request the ICU object of the client.

**Returns**

Client ICU object, an invalid capability selector is returned if no ICU is available.

**16.125 io.h**

[Go to the documentation of this file.](#)

```
00001
00004 /*
00005  * (c) 2008-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00006  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012
00013
00014 #pragma once
00015
00016 #include <l4/sys/types.h>
00017 #include <l4/sys/linkage.h>
00018 #include <l4/io/types.h>
00019
00020 EXTERN_C_BEGIN
00021
00038 L4_CV long L4_EXPORT
00039 l4io_request_irq(int irqnum, l4_cap_idx_t irqcap);
00040
00047 L4_CV l4_cap_idx_t L4_EXPORT
00048 l4io_request_icu(void);
00049
00061 L4_CV long L4_EXPORT
00062 l4io_release_irq(int irqnum, l4_cap_idx_t irqcap);
00063
00088 L4_CV long L4_EXPORT
00089 l4io_request_iomem(l4_addr_t phys, unsigned long size, int flags,
00090                   l4_addr_t *virt);
00091
00113 L4_CV long L4_EXPORT
00114 l4io_request_iomem_region(l4_addr_t phys, l4_addr_t virt,
00115                           unsigned long size, int flags);
00116
00124 L4_CV long L4_EXPORT
00125 l4io_release_iomem(l4_addr_t virt, unsigned long size);
00126
00136 L4_CV long L4_EXPORT
```

```

00137 l4io_request_ioport(unsigned portnum, unsigned len);
00138
00148 L4_CV long L4_EXPORT
00149 l4io_release_ioport(unsigned portnum, unsigned len);
00150
00151
00152 /* ----- Device handling ----- */
00153
00159 L4_INLINE
00160 l4io_device_handle_t l4io_get_root_device(void);
00161
00172 L4_CV int L4_EXPORT
00173 l4io_iterate_devices(l4io_device_handle_t *devhandle,
00174                     l4io_device_t *dev, l4io_resource_handle_t *reshandle);
00175
00188 L4_CV int L4_EXPORT
00189 l4io_lookup_device(const char *devname,
00190                   l4io_device_handle_t *dev_handle,
00191                   l4io_device_t *dev, l4io_resource_handle_t *res_handle);
00192
00208 L4_CV int L4_EXPORT
00209 l4io_lookup_resource(l4io_device_handle_t devhandle,
00210                     enum l4io_resource_types_t type,
00211                     l4io_resource_handle_t *reshandle,
00212                     l4io_resource_t *res);
00213
00214
00215 /* ----- Convenience functions ----- */
00216
00229 L4_CV l4_addr_t L4_EXPORT
00230 l4io_request_resource_iomem(l4io_device_handle_t devhandle,
00231                             l4io_resource_handle_t *reshandle);
00232
00239 L4_CV void L4_EXPORT
00240 l4io_request_all_ioports(void (*res_cb)(l4vbus_resource_t const *res));
00241
00250 L4_CV int L4_EXPORT
00251 l4io_has_resource(enum l4io_resource_types_t type,
00252                  l4vbus_paddr_t start, l4vbus_paddr_t end);
00253
00254 /* ----- Implementations ----- */
00255 /* Implementations */
00256
00257 L4_INLINE
00258 l4io_device_handle_t l4io_get_root_device(void)
00259 { return 0; }
00260
00261 EXTERN_C_END

```

## 16.126 types.h

```

00001 /*
00002  * (c) 2008-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  * This file is part of TUD:OS and distributed under the terms of the
00005  * GNU Lesser General Public License 2.1.
00006  * Please see the COPYING-LGPL-2.1 file for details.
00007  */
00008 #pragma once
00009
00010 #include <l4/vbus/vbus_types.h>
00011
00016 enum l4io_iomem_flags_t
00017 {
00018     L4IO_MEM_NONCACHED = 0,
00019     L4IO_MEM_CACHED    = 1,
00020     L4IO_MEM_USE_MTRR   = 2,
00021     L4IO_MEM_ATTR_MASK = 0xf,
00022
00023     // combinations
00024     L4IO_MEM_WRITE_COMBINED = L4IO_MEM_USE_MTRR | L4IO_MEM_CACHED,
00025
00026
00029     L4IO_MEM_USE_RESERVED_AREA = 0x40 « 8,
00031     L4IO_MEM_EAGER_MAP         = 0x80 « 8,
00032 };
00033
00038 enum l4io_device_types_t {
00039     L4IO_DEVICE_INVALID = 0,
00040     L4IO_DEVICE_PCI,
00041     L4IO_DEVICE_USB,
00042     L4IO_DEVICE_OTHER,
00043     L4IO_DEVICE_ANY = ~0

```

```

00044 };
00045
00050 enum l4io_resource_types_t {
00051     L4IO_RESOURCE_INVALID = L4VBUS_RESOURCE_INVALID,
00052     L4IO_RESOURCE_IRQ     = L4VBUS_RESOURCE_IRQ,
00053     L4IO_RESOURCE_MEM     = L4VBUS_RESOURCE_MEM,
00054     L4IO_RESOURCE_PORT    = L4VBUS_RESOURCE_PORT,
00055     L4IO_RESOURCE_ANY     = ~0
00056 };
00057
00058
00059 typedef l4vbus_device_handle_t l4io_device_handle_t;
00060 typedef unsigned l4io_resource_handle_t;
00061
00069 typedef l4vbus_resource_t l4io_resource_t;
00070
00074 typedef l4vbus_device_t l4io_device_t;

```

## 16.127 types.h

```

00001 /*
00002  * Copyright (C) 2018-2019 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * This file is distributed under the terms of the GNU General Public
00006  * License, version 2. Please see the COPYING-GPL-2 file for details.
00007  */
00008 #pragma once
00009
00010 #include <functional>
00011
00012 #include <l4/cxx/unique_ptr>
00013 #include <l4/re/dma_space>
00014 #include <l4/l4virtio/server/l4virtio>
00015
00016 namespace Block_device {
00017
00019 enum Inout_flags
00020 {
00021     Inout_f_wb = 1,
00022     Inout_f_unmap = 2,
00023 };
00024
00025 enum Shutdown_type
00026 {
00028     Running = 0,
00030     // client had crashed.
00031     Client_gone,
00033     Client_shutdown,
00035     System_shutdown,
00037     System_suspend
00038 };
00039
00044 struct Dma_region_info
00045 {
00046     virtual ~Dma_region_info() = default;
00047 };
00048
00053 struct Mem_region_info
00054 {
00055     cxx::unique_ptr<Dma_region_info> dma_info;
00056 };
00057
00058 using Mem_region =
00059     L4virtio::Svr::Driver_mem_region_t<Mem_region_info>;
00060
00067 struct Inout_block
00068 {
00069     L4Re::Dma_space::Dma_addr dma_addr = 0;
00070     void *virt_addr = nullptr;
00072     l4_uint64_t sector = 0;
00073     l4_uint32_t num_sectors = 0;
00075     l4_uint32_t flags = 0;
00076     cxx::unique_ptr<Inout_block> next;
00077 };
00078
00079 typedef std::function<void(int, l4_size_t)> Inout_callback;
00080
00081 } // name space

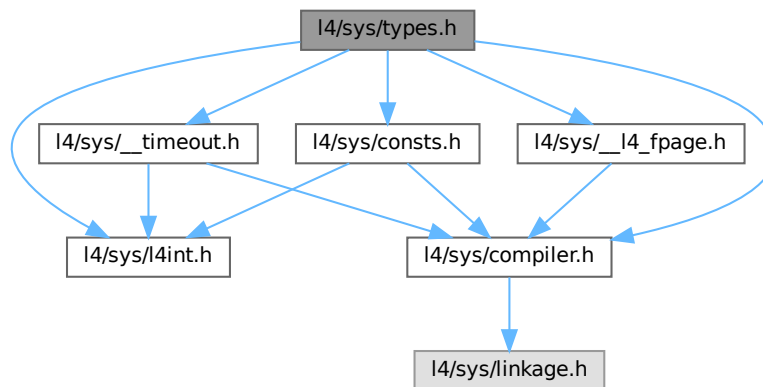
```

## 16.128 l4/sys/types.h File Reference

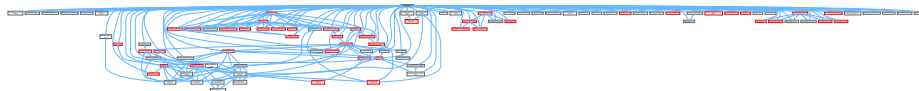
Common [L4](#) ABI Data Types.

```
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
#include <l4/sys/consts.h>
#include <l4/sys/__l4_fpage.h>
#include <l4/sys/__timeout.h>
```

Include dependency graph for types.h:



This graph shows which files directly or indirectly include this file:



### Data Structures

- struct [l4\\_msgtag\\_t](#)  
*Message tag data structure.*

### Typedefs

- typedef struct [l4\\_msgtag\\_t](#) [l4\\_msgtag\\_t](#)  
*Message tag data structure.*
- typedef unsigned long [l4\\_cap\\_idx\\_t](#)  
*Capability selector type.*



## Enumerations

- enum `L4_msgtag_protocol` {  
`L4_PROTO_NONE` = 0 , `L4_PROTO_ALLOW_SYSCALL` = 1 , `L4_PROTO_PF_EXCEPTION` = 1 ,  
`L4_PROTO_IRQ` = -1L ,  
`L4_PROTO_PAGE_FAULT` = -2L , `L4_PROTO_PREEMPTION` = -3L , `L4_PROTO_SYS_EXCEPTION` = -4L ,  
`L4_PROTO_EXCEPTION` = -5L ,  
`L4_PROTO_SIGMA0` = -6L , `L4_PROTO_IO_PAGE_FAULT` = -8L , `L4_PROTO_KOBJECT` = -10L ,  
`L4_PROTO_TASK` = -11L ,  
`L4_PROTO_THREAD` = -12L , `L4_PROTO_LOG` = -13L , `L4_PROTO_SCHEDULER` = -14L ,  
`L4_PROTO_FACTORY` = -15L ,  
`L4_PROTO_VM` = -16L , `L4_PROTO_DMA_SPACE` = -17L , `L4_PROTO_IRQ_SENDER` = -18L ,  
`L4_PROTO_IRQ_MUX` = -19L ,  
`L4_PROTO_SEMAPHORE` = -20L , `L4_PROTO_META` = -21L , `L4_PROTO_IOMMU` = -22L ,  
`L4_PROTO_DEBUGGER` = -23L ,  
`L4_PROTO_SMCCC` = -24L , `L4_PROTO_VCPU_CONTEXT` = -25L }  
*Message tag for IPC operations.*
- enum `L4_msgtag_flags` { `L4_MSGTAG_ERROR` , `L4_MSGTAG_TRANSFER_FPU` , `L4_MSGTAG_SCHEDULE`  
, `L4_MSGTAG_PROPAGATE` = 0x4000 , `L4_MSGTAG_FLAGS` }  
*Flags for message tags.*

## Functions

- `l4_msgtag_t l4_msgtag` (long label, unsigned words, unsigned items, unsigned flags) `L4_NOTHROW`  
*Create a message tag from the specified values.*
- long `l4_msgtag_label` (`l4_msgtag_t`) `L4_NOTHROW`  
*Get the protocol of tag.*
- unsigned `l4_msgtag_words` (`l4_msgtag_t`) `L4_NOTHROW`  
*Get the number of untyped words.*
- unsigned `l4_msgtag_items` (`l4_msgtag_t`) `L4_NOTHROW`  
*Get the number of typed items.*
- unsigned `l4_msgtag_flags` (`l4_msgtag_t`) `L4_NOTHROW`  
*Get the flags.*
- unsigned `l4_msgtag_has_error` (`l4_msgtag_t`) `L4_NOTHROW`  
*Test for error indicator flag.*
- unsigned `l4_msgtag_is_page_fault` (`l4_msgtag_t`) `L4_NOTHROW`  
*Test for page-fault protocol.*
- unsigned `l4_msgtag_is_preemption` (`l4_msgtag_t`) `L4_NOTHROW`  
*Test for preemption protocol.*
- unsigned `l4_msgtag_is_sys_exception` (`l4_msgtag_t`) `L4_NOTHROW`  
*Test for system-exception protocol.*
- unsigned `l4_msgtag_is_exception` (`l4_msgtag_t`) `L4_NOTHROW`  
*Test for exception protocol.*
- unsigned `l4_msgtag_is_sigma0` (`l4_msgtag_t`) `L4_NOTHROW`  
*Test for sigma0 protocol.*
- unsigned `l4_msgtag_is_io_page_fault` (`l4_msgtag_t`) `L4_NOTHROW`  
*Test for IO-page-fault protocol.*
- unsigned `l4_is_invalid_cap` (`l4_cap_idx_t` c) `L4_NOTHROW`  
*Test if a capability selector is the invalid capability.*
- unsigned `l4_is_valid_cap` (`l4_cap_idx_t` c) `L4_NOTHROW`  
*Test if a capability selector is a valid selector.*
- unsigned `l4_capability_equal` (`l4_cap_idx_t` c1, `l4_cap_idx_t` c2) `L4_NOTHROW`  
*Test if the capability indices of two capability selectors are equal.*
- `l4_cap_idx_t l4_capability_next` (`l4_cap_idx_t` c) `L4_NOTHROW`  
*Get the next capability selector after c.*

## 16.128.1 Detailed Description

Common [L4](#) ABI Data Types.

Definition in file [types.h](#).

## 16.128.2 Function Documentation

### 16.128.2.1 l4\_capability\_next()

```
l4_cap_idx_t l4_capability_next (
    l4_cap_idx_t c ) [inline]
```

Get the next capability selector after *c*.

#### Parameters

<i>c</i>	The capability selector for which the next selector shall be computed.
----------	--

#### Returns

The next capability selector after *c*.

Definition at line [480](#) of file [types.h](#).

References [L4\\_CAP\\_OFFSET](#).

## 16.129 types.h

[Go to the documentation of this file.](#)

```
00001 /*****
00002  *
00003  * (c) 2008-2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  * Alexander Warg <warg@os.inf.tu-dresden.de>,
00005  * Björn Döbel <doebel@os.inf.tu-dresden.de>,
00006  * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00007  * economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 /*****
00023  *
00024  * #pragma once
00025  *
00026  * #include <l4/sys/l4int.h>
00027  * #include <l4/sys/compiler.h>
00028  * #include <l4/sys/consts.h>
00029  *
00030  * enum l4_msgtag_protocol
```

```

00050 {
00051     L4_PROTO_NONE = 0,
00052     L4_PROTO_ALLOW_SYSCALL = 1,
00053     L4_PROTO_PF_EXCEPTION = 1,
00054
00055     L4_PROTO_IRQ = -1L,
00056     L4_PROTO_PAGE_FAULT = -2L,
00057     L4_PROTO_PREEMPTION = -3L,
00058     L4_PROTO_SYS_EXCEPTION = -4L,
00059     L4_PROTO_EXCEPTION = -5L,
00060     L4_PROTO_SIGMA0 = -6L,
00061     L4_PROTO_IO_PAGE_FAULT = -8L,
00062     L4_PROTO_KOBJECT = -10L,
00063     L4_PROTO_TASK = -11L,
00064     L4_PROTO_THREAD = -12L,
00065     L4_PROTO_LOG = -13L,
00066     L4_PROTO_SCHEDULER = -14L,
00067     L4_PROTO_FACTORY = -15L,
00068     L4_PROTO_VM = -16L,
00069     L4_PROTO_DMA_SPACE = -17L,
00070     L4_PROTO_IRQ_SENDER = -18L,
00071     L4_PROTO_IRQ_MUX = -19L,
00072     L4_PROTO_SEMAPHORE = -20L,
00073     L4_PROTO_META = -21L,
00074     L4_PROTO_IOMMU = -22L,
00075     L4_PROTO_DEBUGGER = -23L,
00076     L4_PROTO_SMCCC = -24L,
00077     L4_PROTO_VCPU_CONTEXT = -25L,
00078 };
00079
00080 enum L4_varg_type
00081 {
00082     L4_VARG_TYPE_NIL = 0x00,
00083     L4_VARG_TYPE_UMWORD = 0x01,
00084     L4_VARG_TYPE_MWORD = 0x81,
00085     L4_VARG_TYPE_STRING = 0x02,
00086     L4_VARG_TYPE_FPAGE = 0x03,
00087
00088     L4_VARG_TYPE_SIGN = 0x80,
00089 };
00090
00091
00096 enum L4_msgtag_flags
00097 {
00098     // flags for received IPC
00103     L4_MSGTAG_ERROR = 0x8000,
00104
00105     // flags for sending IPC
00115     L4_MSGTAG_TRANSFER_FPU = 0x1000,
00124     L4_MSGTAG_SCHEDULE = 0x2000,
00137     L4_MSGTAG_PROPAGATE = 0x4000,
00138
00143     L4_MSGTAG_FLAGS = 0xf000,
00144 };
00145
00146
00163 typedef struct l4_msgtag_t
00164 {
00165     l4_mword_t raw;
00166 #ifdef __cplusplus
00168     long label() const L4_NOTHROW { return raw >> 16; }
00170     void label(long v) L4_NOTHROW { raw = (raw & 0x0ffff) | ((l4_umword_t)v << 16); }
00172     unsigned words() const L4_NOTHROW { return raw & 0x3f; }
00174     unsigned items() const L4_NOTHROW { return (raw >> 6) & 0x3f; }
00181     unsigned flags() const L4_NOTHROW { return raw & 0xf000; }
00183     bool is_page_fault() const L4_NOTHROW { return label() == L4_PROTO_PAGE_FAULT; }
00185     bool is_preemption() const L4_NOTHROW { return label() == L4_PROTO_PREEMPTION; }
00187     bool is_sys_exception() const L4_NOTHROW { return label() == L4_PROTO_SYS_EXCEPTION; }
00189     bool is_exception() const L4_NOTHROW { return label() == L4_PROTO_EXCEPTION; }
00191     bool is_sigma0() const L4_NOTHROW { return label() == L4_PROTO_SIGMA0; }
00193     bool is_io_page_fault() const L4_NOTHROW { return label() == L4_PROTO_IO_PAGE_FAULT; }
00195     unsigned has_error() const L4_NOTHROW { return raw & L4_MSGTAG_ERROR; }
00196 #endif
00197 } l4_msgtag_t;
00198
00199
00200
00212 L4_INLINE l4_msgtag_t l4_msgtag(long label, unsigned words, unsigned items,
00213                                unsigned flags) L4_NOTHROW;
00214
00223 L4_INLINE long l4_msgtag_label(l4_msgtag_t t) L4_NOTHROW;
00224
00233 L4_INLINE unsigned l4_msgtag_words(l4_msgtag_t t) L4_NOTHROW;
00234
00243 L4_INLINE unsigned l4_msgtag_items(l4_msgtag_t t) L4_NOTHROW;
00244
00255 L4_INLINE unsigned l4_msgtag_flags(l4_msgtag_t t) L4_NOTHROW;

```

```

00256
00269 L4_INLINE unsigned l4_msgtag_has_error(l4_msgtag_t t) L4_NOTHROW;
00270
00279 L4_INLINE unsigned l4_msgtag_is_page_fault(l4_msgtag_t t) L4_NOTHROW;
00280
00288 L4_INLINE unsigned l4_msgtag_is_preemption(l4_msgtag_t t) L4_NOTHROW;
00289
00298 L4_INLINE unsigned l4_msgtag_is_sys_exception(l4_msgtag_t t) L4_NOTHROW;
00299
00308 L4_INLINE unsigned l4_msgtag_is_exception(l4_msgtag_t t) L4_NOTHROW;
00309
00318 L4_INLINE unsigned l4_msgtag_is_sigma0(l4_msgtag_t t) L4_NOTHROW;
00319
00328 L4_INLINE unsigned l4_msgtag_is_io_page_fault(l4_msgtag_t t) L4_NOTHROW;
00329
00359 typedef unsigned long l4_cap_idx_t;
00360
00370 L4_INLINE unsigned l4_is_invalid_cap(l4_cap_idx_t c) L4_NOTHROW;
00371
00381 L4_INLINE unsigned l4_is_valid_cap(l4_cap_idx_t c) L4_NOTHROW;
00382
00396 L4_INLINE unsigned l4_capability_equal(l4_cap_idx_t c1, l4_cap_idx_t c2) L4_NOTHROW;
00397
00406 L4_INLINE l4_cap_idx_t l4_capability_next(l4_cap_idx_t c) L4_NOTHROW;
00407
00408 /* *****
00409 /* Implementation */
00410
00411 L4_INLINE unsigned
00412 l4_is_invalid_cap(l4_cap_idx_t c) L4_NOTHROW
00413 { return c & L4_INVALID_CAP_BIT; }
00414
00415 L4_INLINE unsigned
00416 l4_is_valid_cap(l4_cap_idx_t c) L4_NOTHROW
00417 { return !(c & L4_INVALID_CAP_BIT); }
00418
00419 L4_INLINE unsigned
00420 l4_capability_equal(l4_cap_idx_t c1, l4_cap_idx_t c2) L4_NOTHROW
00421 { return (c1 >> L4_CAP_SHIFT) == (c2 >> L4_CAP_SHIFT); }
00422
00423
00427 L4_INLINE
00428 l4_msgtag_t l4_msgtag(long label, unsigned words, unsigned items,
00429                      unsigned flags) L4_NOTHROW
00430 {
00431     return (l4_msgtag_t){ (l4_mword_t)((l4_umword_t)label << 16)
00432                          | (l4_mword_t)(words & 0x3f)
00433                          | (l4_mword_t)((items & 0x3f) << 6)
00434                          | (l4_mword_t)(flags & 0xf000)};
00435 }
00436
00437
00438
00439 L4_INLINE
00440 long l4_msgtag_label(l4_msgtag_t t) L4_NOTHROW
00441 { return t.raw >> 16; }
00442
00443 L4_INLINE
00444 unsigned l4_msgtag_words(l4_msgtag_t t) L4_NOTHROW
00445 { return t.raw & 0x3f; }
00446
00447 L4_INLINE
00448 unsigned l4_msgtag_items(l4_msgtag_t t) L4_NOTHROW
00449 { return (t.raw >> 6) & 0x3f; }
00450
00451 L4_INLINE
00452 unsigned l4_msgtag_flags(l4_msgtag_t t) L4_NOTHROW
00453 { return t.raw & 0xf000; }
00454
00455
00456 L4_INLINE
00457 unsigned l4_msgtag_has_error(l4_msgtag_t t) L4_NOTHROW
00458 { return t.raw & L4_MSGTAG_ERROR; }
00459
00460
00461
00462 L4_INLINE unsigned l4_msgtag_is_page_fault(l4_msgtag_t t) L4_NOTHROW
00463 { return l4_msgtag_label(t) == L4_PROTO_PAGE_FAULT; }
00464
00465 L4_INLINE unsigned l4_msgtag_is_preemption(l4_msgtag_t t) L4_NOTHROW
00466 { return l4_msgtag_label(t) == L4_PROTO_PREEMPTION; }
00467
00468 L4_INLINE unsigned l4_msgtag_is_sys_exception(l4_msgtag_t t) L4_NOTHROW
00469 { return l4_msgtag_label(t) == L4_PROTO_SYS_EXCEPTION; }
00470
00471 L4_INLINE unsigned l4_msgtag_is_exception(l4_msgtag_t t) L4_NOTHROW
00472 { return l4_msgtag_label(t) == L4_PROTO_EXCEPTION; }

```

```

00473
00474 L4_INLINE unsigned l4_msgtag_is_sigma0(l4_msgtag_t t) L4_NOTHROW
00475 { return l4_msgtag_label(t) == L4_PROTO_SIGMA0; }
00476
00477 L4_INLINE unsigned l4_msgtag_is_io_page_fault(l4_msgtag_t t) L4_NOTHROW
00478 { return l4_msgtag_label(t) == L4_PROTO_IO_PAGE_FAULT; }
00479
00480 L4_INLINE l4_cap_idx_t l4_capability_next(l4_cap_idx_t c) L4_NOTHROW
00481 { return c + L4_CAP_OFFSET; }
00482
00483 #include <l4/sys/__l4_fpage.h>
00484 #include <l4/sys/__timeout.h>

```

## 16.130 l4/cxx/alloc.h File Reference

Alloc list.

### Data Structures

- class [L4::Alloc\\_list](#)  
*A simple list-based allocator.*

### Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

### 16.130.1 Detailed Description

Alloc list.

Definition in file [alloc.h](#).

## 16.131 alloc.h

[Go to the documentation of this file.](#)

```

00001
00002 /*
00003  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *          Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *          economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020 #pragma once
00021
00022 namespace L4 {
00023
00024     class Alloc_list

```

```

00032 {
00033     public:
00034         Alloc_list() : _free(0) {}
00035         Alloc_list(void *blk, unsigned long size) : _free(0)
00036         { free( blk, size); }
00037
00038         void free(void *blk, unsigned long size);
00039         void *alloc(unsigned long size);
00040
00041     private:
00042         struct Elem
00043         {
00044             Elem *next;
00045             unsigned long size;
00046         };
00047
00048         Elem *_free;
00049     };
00050 }

```

## 16.132 arith

```

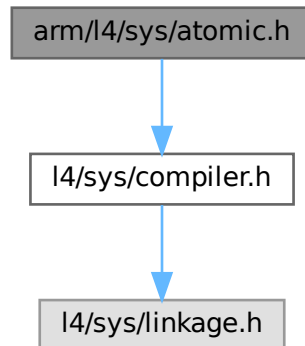
00001 // vi:set ft=c++ -- Mode: C++ --
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019
00020 #pragma once
00021
00022 namespace cxx { namespace arith {
00023
00024     template< unsigned long V >
00025     struct Ld
00026     {
00027         enum { value = Ld<V / 2>::value + 1 };
00028     };
00029
00030     template<>
00031     struct Ld<0>
00032     {
00033         enum { value = ~0UL };
00034     };
00035
00036     template<>
00037     struct Ld<1>
00038     {
00039         enum { value = 0 };
00040     };
00041
00042 }}

```

## 16.133 arm/l4/sys/atomic.h File Reference

Atomic memory modifications.

```
#include <l4/sys/compiler.h>
Include dependency graph for atomic.h:
```



### 16.133.1 Detailed Description

Atomic memory modifications.

Definition in file [atomic.h](#).

## 16.134 atomic.h

[Go to the documentation of this file.](#)

```

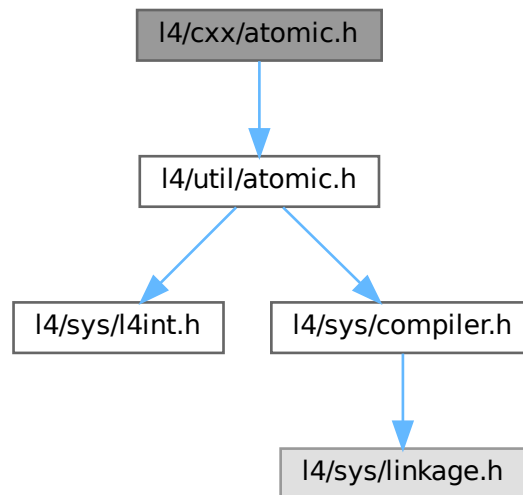
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/sys/compiler.h>
00027
00028 EXTERN_C long int
00029 l4_atomic_add(volatile long int* mem, long int offset) L4_NOTHROW L4_LONG_CALL;
00030
00031 EXTERN_C long int
00032 l4_atomic_xchg(volatile long int* mem, long int newval) L4_NOTHROW L4_LONG_CALL;
00033
00034 EXTERN_C long int
00035 l4_atomic_cmpxchg(volatile long int* mem, long int oldval, long int newval) L4_NOTHROW L4_LONG_CALL;
```

## 16.135 l4/cxx/atomic.h File Reference

Atomic template.

```
#include <l4/util/atomic.h>
```

Include dependency graph for atomic.h:



### Namespaces

- namespace [L4](#)  
*[L4](#) low-level kernel interface.*

### 16.135.1 Detailed Description

Atomic template.

Definition in file [atomic.h](#).

## 16.136 atomic.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2004-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *           Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *           economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *

```



```

00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 #include <l4/sys/l4int.h>
00026
00027 extern "C" void ____error_compare_and_swap_does_not_support_3_bytes____();
00028 extern "C" void ____error_compare_and_swap_does_not_support_more_than_4_bytes____();
00029
00030 namespace L4
00031 {
00032     template< typename X >
00033     inline int compare_and_swap(X volatile *dst, X old_val, X new_val)
00034     {
00035         switch (sizeof(X))
00036         {
00037             case 1:
00038                 return l4util_cmpxchg8((l4_uint8_t volatile*)dst, old_val, new_val);
00039             case 2:
00040                 return l4util_cmpxchg16((l4_uint16_t volatile *)dst, old_val, new_val);
00041             case 3: ____error_compare_and_swap_does_not_support_3_bytes____();
00042             case 4:
00043                 return l4util_cmpxchg32((l4_uint32_t volatile*)dst, old_val, new_val);
00044             default:
00045                 ____error_compare_and_swap_does_not_support_more_than_4_bytes____();
00046         }
00047         return 0;
00048     }
00049 }

```

## 16.137 l4/util/atomic.h File Reference

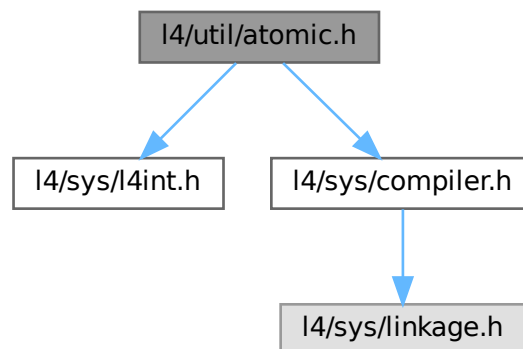
atomic operations header and generic implementations

```

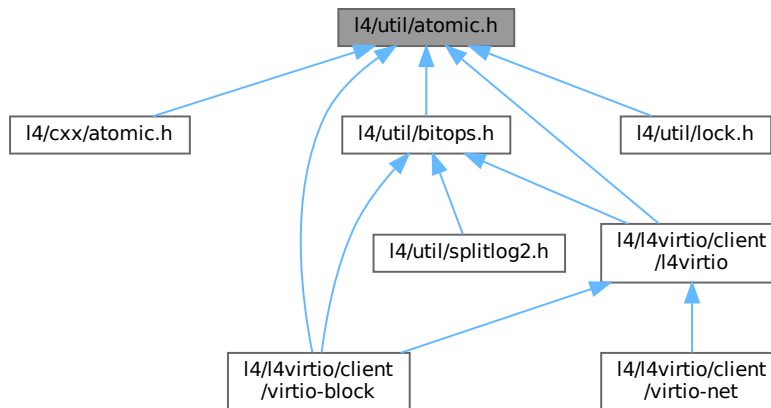
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for atomic.h:



This graph shows which files directly or indirectly include this file:



## Functions

- `int l4util_cmpxchg32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` cmp\_val, `l4_uint32_t` new\_val)  
*Atomic compare and exchange (32 bit version)*
- `int l4util_cmpxchg16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` cmp\_val, `l4_uint16_t` new\_val)  
*Atomic compare and exchange (16 bit version)*
- `int l4util_cmpxchg8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` cmp\_val, `l4_uint8_t` new\_val)  
*Atomic compare and exchange (8 bit version)*
- `int l4util_cmpxchg` (volatile `l4_umword_t` \*dest, `l4_umword_t` cmp\_val, `l4_umword_t` new\_val)  
*Atomic compare and exchange (machine wide fields)*
- `l4_uint32_t l4util_xchg32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)  
*Atomic exchange (32 bit version)*
- `l4_uint16_t l4util_xchg16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)  
*Atomic exchange (16 bit version)*
- `l4_uint8_t l4util_xchg8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)  
*Atomic exchange (8 bit version)*
- `l4_umword_t l4util_xchg` (volatile `l4_umword_t` \*dest, `l4_umword_t` val)  
*Atomic exchange (machine wide fields)*
- `void l4util_atomic_add` (volatile long \*dest, long val)  
*Atomic add.*
- `void l4util_atomic_inc` (volatile long \*dest)  
*Atomic increment.*

## Atomic add/sub/and/or (8,16,32 bit version) without result

- `void l4util_add8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `void l4util_add16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `void l4util_add32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)
- `void l4util_sub8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `void l4util_sub16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `void l4util_sub32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)
- `void l4util_and8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `void l4util_and16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)

- void [l4util\\_and32](#) (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)
- void [l4util\\_or8](#) (volatile [l4\\_uint8\\_t](#) \*dest, [l4\\_uint8\\_t](#) val)
- void [l4util\\_or16](#) (volatile [l4\\_uint16\\_t](#) \*dest, [l4\\_uint16\\_t](#) val)
- void [l4util\\_or32](#) (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)

#### Atomic add/sub/and/or operations (8,16,32 bit) with result

- [l4\\_uint8\\_t l4util\\_add8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest, [l4\\_uint8\\_t](#) val)
- [l4\\_uint16\\_t l4util\\_add16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest, [l4\\_uint16\\_t](#) val)
- [l4\\_uint32\\_t l4util\\_add32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)
- [l4\\_uint8\\_t l4util\\_sub8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest, [l4\\_uint8\\_t](#) val)
- [l4\\_uint16\\_t l4util\\_sub16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest, [l4\\_uint16\\_t](#) val)
- [l4\\_uint32\\_t l4util\\_sub32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)
- [l4\\_uint8\\_t l4util\\_and8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest, [l4\\_uint8\\_t](#) val)
- [l4\\_uint16\\_t l4util\\_and16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest, [l4\\_uint16\\_t](#) val)
- [l4\\_uint32\\_t l4util\\_and32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)
- [l4\\_uint8\\_t l4util\\_or8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest, [l4\\_uint8\\_t](#) val)
- [l4\\_uint16\\_t l4util\\_or16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest, [l4\\_uint16\\_t](#) val)
- [l4\\_uint32\\_t l4util\\_or32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)

#### Atomic inc/dec (8,16,32 bit) without result

- void [l4util\\_inc8](#) (volatile [l4\\_uint8\\_t](#) \*dest)
- void [l4util\\_inc16](#) (volatile [l4\\_uint16\\_t](#) \*dest)
- void [l4util\\_inc32](#) (volatile [l4\\_uint32\\_t](#) \*dest)
- void [l4util\\_dec8](#) (volatile [l4\\_uint8\\_t](#) \*dest)
- void [l4util\\_dec16](#) (volatile [l4\\_uint16\\_t](#) \*dest)
- void [l4util\\_dec32](#) (volatile [l4\\_uint32\\_t](#) \*dest)

#### Atomic inc/dec (8,16,32 bit) with result

- [l4\\_uint8\\_t l4util\\_inc8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest)
- [l4\\_uint16\\_t l4util\\_inc16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest)
- [l4\\_uint32\\_t l4util\\_inc32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest)
- [l4\\_uint8\\_t l4util\\_dec8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest)
- [l4\\_uint16\\_t l4util\\_dec16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest)
- [l4\\_uint32\\_t l4util\\_dec32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest)

### 16.137.1 Detailed Description

atomic operations header and generic implementations

Date

10/20/2000

Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de), Jork Loeser [jork@os.inf.tu-dresden.de](mailto:jork@os.inf.tu-dresden.de)

Definition in file [atomic.h](#).

## 16.138 atomic.h

[Go to the documentation of this file.](#)

```

00001 /*****
00010 */
00011 * (c) 2000-2009 Author(s)
00012 *     economic rights: Technische Universität Dresden (Germany)
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU Lesser General Public License 2.1.
00015 * Please see the COPYING-LGPL-2.1 file for details.
00016 */
00017
00018 /*****
00019 #ifndef __L4UTIL__INCLUDE__ATOMIC_H__
00020 #define __L4UTIL__INCLUDE__ATOMIC_H__
00021
00022 #include <l4/sys/l4int.h>
00023 #include <l4/sys/compiler.h>
00024
00025 /*****
00026 *** Prototypes
00027 *****/
00028
00029 EXTERN_C_BEGIN
00030
00036 #if __SIZEOF_LONG__ == 8
00037
00057 L4_INLINE int
00058 l4util_cmpxchg64(volatile l4_uint64_t * dest,
00059                 l4_uint64_t cmp_val, l4_uint64_t new_val);
00060
00061 #endif
00062
00076 L4_INLINE int
00077 l4util_cmpxchg32(volatile l4_uint32_t * dest,
00078                 l4_uint32_t cmp_val, l4_uint32_t new_val);
00079
00093 L4_INLINE int
00094 l4util_cmpxchg16(volatile l4_uint16_t * dest,
00095                 l4_uint16_t cmp_val, l4_uint16_t new_val);
00096
00110 L4_INLINE int
00111 l4util_cmpxchg8(volatile l4_uint8_t * dest,
00112                l4_uint8_t cmp_val, l4_uint8_t new_val);
00113
00127 L4_INLINE int
00128 l4util_cmpxchg(volatile l4_umword_t * dest,
00129               l4_umword_t cmp_val, l4_umword_t new_val);
00130
00140 L4_INLINE l4_uint32_t
00141 l4util_xchg32(volatile l4_uint32_t * dest, l4_uint32_t val);
00142
00152 L4_INLINE l4_uint16_t
00153 l4util_xchg16(volatile l4_uint16_t * dest, l4_uint16_t val);
00154
00164 L4_INLINE l4_uint8_t
00165 l4util_xchg8(volatile l4_uint8_t * dest, l4_uint8_t val);
00166
00176 L4_INLINE l4_umword_t
00177 l4util_xchg(volatile l4_umword_t * dest, l4_umword_t val);
00178
00180
00186 L4_INLINE void
00187 l4util_add8(volatile l4_uint8_t *dest, l4_uint8_t val);
00189 L4_INLINE void
00190 l4util_add16(volatile l4_uint16_t *dest, l4_uint16_t val);
00192 L4_INLINE void
00193 l4util_add32(volatile l4_uint32_t *dest, l4_uint32_t val);
00195 L4_INLINE void
00196 l4util_sub8(volatile l4_uint8_t *dest, l4_uint8_t val);
00198 L4_INLINE void
00199 l4util_sub16(volatile l4_uint16_t *dest, l4_uint16_t val);
00201 L4_INLINE void
00202 l4util_sub32(volatile l4_uint32_t *dest, l4_uint32_t val);
00204 L4_INLINE void
00205 l4util_and8(volatile l4_uint8_t *dest, l4_uint8_t val);
00207 L4_INLINE void
00208 l4util_and16(volatile l4_uint16_t *dest, l4_uint16_t val);
00210 L4_INLINE void
00211 l4util_and32(volatile l4_uint32_t *dest, l4_uint32_t val);
00213 L4_INLINE void
00214 l4util_or8(volatile l4_uint8_t *dest, l4_uint8_t val);
00216 L4_INLINE void
00217 l4util_or16(volatile l4_uint16_t *dest, l4_uint16_t val);
00219 L4_INLINE void

```

```

00220 l4util_or32(volatile l4_uint32_t *dest, l4_uint32_t val);
00222
00224
00231 L4_INLINE l4_uint8_t
00232 l4util_add8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00234 L4_INLINE l4_uint16_t
00235 l4util_add16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00237 L4_INLINE l4_uint32_t
00238 l4util_add32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00240 L4_INLINE l4_uint8_t
00241 l4util_sub8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00243 L4_INLINE l4_uint16_t
00244 l4util_sub16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00246 L4_INLINE l4_uint32_t
00247 l4util_sub32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00249 L4_INLINE l4_uint8_t
00250 l4util_and8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00252 L4_INLINE l4_uint16_t
00253 l4util_and16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00255 L4_INLINE l4_uint32_t
00256 l4util_and32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00258 L4_INLINE l4_uint8_t
00259 l4util_or8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00261 L4_INLINE l4_uint16_t
00262 l4util_or16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00264 L4_INLINE l4_uint32_t
00265 l4util_or32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00267
00269
00274 L4_INLINE void
00275 l4util_inc8(volatile l4_uint8_t *dest);
00277 L4_INLINE void
00278 l4util_inc16(volatile l4_uint16_t *dest);
00280 L4_INLINE void
00281 l4util_inc32(volatile l4_uint32_t *dest);
00283 L4_INLINE void
00284 l4util_dec8(volatile l4_uint8_t *dest);
00286 L4_INLINE void
00287 l4util_dec16(volatile l4_uint16_t *dest);
00289 L4_INLINE void
00290 l4util_dec32(volatile l4_uint32_t *dest);
00292
00294
00300 L4_INLINE l4_uint8_t
00301 l4util_inc8_res(volatile l4_uint8_t *dest);
00303 L4_INLINE l4_uint16_t
00304 l4util_inc16_res(volatile l4_uint16_t *dest);
00306 L4_INLINE l4_uint32_t
00307 l4util_inc32_res(volatile l4_uint32_t *dest);
00309 L4_INLINE l4_uint8_t
00310 l4util_dec8_res(volatile l4_uint8_t *dest);
00312 L4_INLINE l4_uint16_t
00313 l4util_dec16_res(volatile l4_uint16_t *dest);
00315 L4_INLINE l4_uint32_t
00316 l4util_dec32_res(volatile l4_uint32_t *dest);
00318
00326 L4_INLINE void
00327 l4util_atomic_add(volatile long *dest, long val);
00328
00335 L4_INLINE void
00336 l4util_atomic_inc(volatile long *dest);
00337
00338 EXTERN_C_END
00339
00340 /*****
00341  * IMPLEMENTATION
00342  *****/
00343
00344 #if __SIZEOF_LONG__ == 8
00345
00346 L4_INLINE int
00347 l4util_cmpxchg64(volatile l4_uint64_t * dest,
00348                  l4_uint64_t cmp_val, l4_uint64_t new_val)
00349 {
00350     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00351                                         __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00352 }
00353
00354 #endif
00355
00356 L4_INLINE int
00357 l4util_cmpxchg32(volatile l4_uint32_t * dest,
00358                  l4_uint32_t cmp_val, l4_uint32_t new_val)
00359 {
00360     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00361                                         __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00362 }

```

```

00363
00364 L4_INLINE int
00365 l4util_cmpxchg16(volatile l4_uint16_t * dest,
00366                  l4_uint16_t cmp_val, l4_uint16_t new_val)
00367 {
00368     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00369                                       __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00370 }
00371
00372 L4_INLINE int
00373 l4util_cmpxchg8(volatile l4_uint8_t * dest,
00374                l4_uint8_t cmp_val, l4_uint8_t new_val)
00375 {
00376     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00377                                       __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00378 }
00379
00380 L4_INLINE int
00381 l4util_cmpxchg(volatile l4_umword_t * dest,
00382               l4_umword_t cmp_val, l4_umword_t new_val)
00383 {
00384     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00385                                       __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00386 }
00387
00388 L4_INLINE l4_uint32_t
00389 l4util_xchg32(volatile l4_uint32_t * dest, l4_uint32_t val)
00390 {
00391     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00392 }
00393
00394 L4_INLINE l4_uint16_t
00395 l4util_xchg16(volatile l4_uint16_t * dest, l4_uint16_t val)
00396 {
00397     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00398 }
00399
00400 L4_INLINE l4_uint8_t
00401 l4util_xchg8(volatile l4_uint8_t * dest, l4_uint8_t val)
00402 {
00403     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00404 }
00405
00406 L4_INLINE l4_umword_t
00407 l4util_xchg(volatile l4_umword_t * dest, l4_umword_t val)
00408 {
00409     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00410 }
00411
00412 L4_INLINE void
00413 l4util_inc8(volatile l4_uint8_t *dest)
00414 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00415
00416 L4_INLINE void
00417 l4util_inc16(volatile l4_uint16_t *dest)
00418 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00419
00420 L4_INLINE void
00421 l4util_inc32(volatile l4_uint32_t *dest)
00422 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00423
00424 L4_INLINE void
00425 l4util_atomic_inc(volatile long *dest)
00426 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00427
00428 L4_INLINE void
00429 l4util_dec8(volatile l4_uint8_t *dest)
00430 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00431
00432 L4_INLINE void
00433 l4util_dec16(volatile l4_uint16_t *dest)
00434 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00435
00436 L4_INLINE void
00437 l4util_dec32(volatile l4_uint32_t *dest)
00438 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00439
00440
00441 L4_INLINE l4_uint8_t
00442 l4util_inc8_res(volatile l4_uint8_t *dest)
00443 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00444
00445 L4_INLINE l4_uint16_t
00446 l4util_inc16_res(volatile l4_uint16_t *dest)
00447 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00448
00449 L4_INLINE l4_uint32_t

```

```

00450 l4util_inc32_res(volatile l4_uint32_t *dest)
00451 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00452
00453 L4_INLINE l4_uint8_t
00454 l4util_dec8_res(volatile l4_uint8_t *dest)
00455 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00456
00457 L4_INLINE l4_uint16_t
00458 l4util_dec16_res(volatile l4_uint16_t *dest)
00459 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00460
00461 L4_INLINE l4_uint32_t
00462 l4util_dec32_res(volatile l4_uint32_t *dest)
00463 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00464
00465 L4_INLINE l4_umword_t
00466 l4util_dec_res(volatile l4_umword_t *dest)
00467 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00468
00469 L4_INLINE void
00470 l4util_add8(volatile l4_uint8_t *dest, l4_uint8_t val)
00471 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00472
00473 L4_INLINE void
00474 l4util_add16(volatile l4_uint16_t *dest, l4_uint16_t val)
00475 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00476
00477 L4_INLINE void
00478 l4util_add32(volatile l4_uint32_t *dest, l4_uint32_t val)
00479 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00480
00481 L4_INLINE void
00482 l4util_atomic_add(volatile long *dest, long val)
00483 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00484
00485 L4_INLINE void
00486 l4util_sub8(volatile l4_uint8_t *dest, l4_uint8_t val)
00487 { __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }
00488
00489 L4_INLINE void
00490 l4util_sub16(volatile l4_uint16_t *dest, l4_uint16_t val)
00491 { __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }
00492
00493 L4_INLINE void
00494 l4util_sub32(volatile l4_uint32_t *dest, l4_uint32_t val)
00495 { __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }
00496
00497 L4_INLINE void
00498 l4util_and8(volatile l4_uint8_t *dest, l4_uint8_t val)
00499 { __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }
00500
00501 L4_INLINE void
00502 l4util_and16(volatile l4_uint16_t *dest, l4_uint16_t val)
00503 { __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }
00504
00505 L4_INLINE void
00506 l4util_and32(volatile l4_uint32_t *dest, l4_uint32_t val)
00507 { __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }
00508
00509 L4_INLINE void
00510 l4util_or8(volatile l4_uint8_t *dest, l4_uint8_t val)
00511 { __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }
00512
00513 L4_INLINE void
00514 l4util_or16(volatile l4_uint16_t *dest, l4_uint16_t val)
00515 { __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }
00516
00517 L4_INLINE void
00518 l4util_or32(volatile l4_uint32_t *dest, l4_uint32_t val)
00519 { __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }
00520
00521 L4_INLINE l4_uint8_t
00522 l4util_add8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00523 { return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }
00524
00525 L4_INLINE l4_uint16_t
00526 l4util_add16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00527 { return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }
00528
00529 L4_INLINE l4_uint32_t
00530 l4util_add32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00531 { return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }
00532
00533 L4_INLINE l4_uint8_t
00534 l4util_sub8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00535 { return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }
00536

```

```

00537 L4_INLINE l4_uint16_t
00538 l4util_sub16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00539 { return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }
00540
00541 L4_INLINE l4_uint32_t
00542 l4util_sub32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00543 { return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }
00544
00545 L4_INLINE l4_uint8_t
00546 l4util_and8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00547 { return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }
00548
00549 L4_INLINE l4_uint16_t
00550 l4util_and16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00551 { return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }
00552
00553 L4_INLINE l4_uint32_t
00554 l4util_and32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00555 { return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }
00556
00557 L4_INLINE l4_uint8_t
00558 l4util_or8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00559 { return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }
00560
00561 L4_INLINE l4_uint16_t
00562 l4util_or16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00563 { return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }
00564
00565 L4_INLINE l4_uint32_t
00566 l4util_or32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00567 { return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }
00568
00569 #endif /* ! __L4UTIL__INCLUDE__ATOMIC_H__ */

```

## 16.139 l4/cxx/avl\_map File Reference

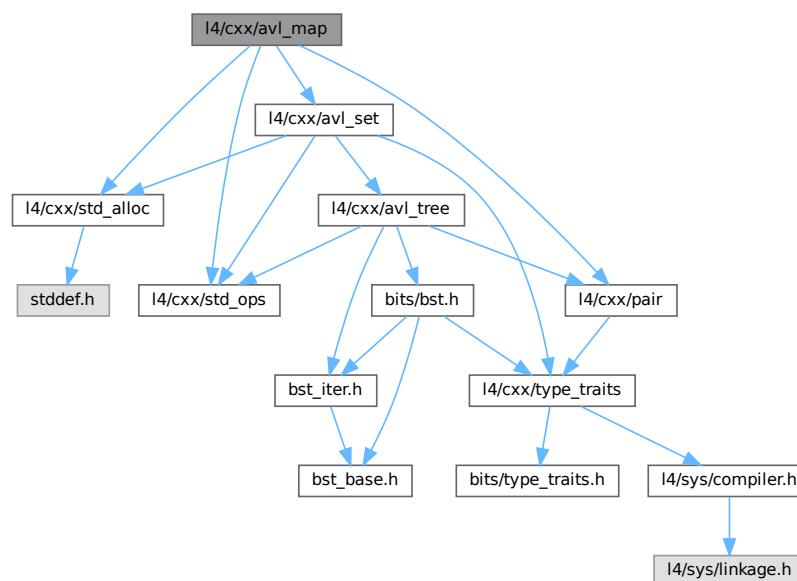
AVL map.

```

#include <l4/cxx/std_alloc>
#include <l4/cxx/std_ops>
#include <l4/cxx/pair>
#include <l4/cxx/avl_set>

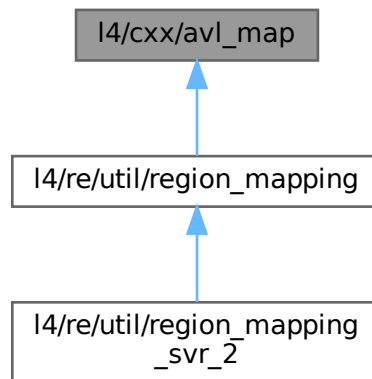
```

Include dependency graph for avl\_map:





This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [cxx::Bits::Avl\\_map\\_get\\_key< KEY\\_TYPE >](#)  
*Key-getter for [Avl\\_map](#).*
- class [cxx::Avl\\_map< KEY\\_TYPE, DATA\\_TYPE, COMPARE, ALLOC >](#)  
*AVL tree based associative container.*

## Namespaces

- namespace [cxx](#)  
*Our C++ library.*
- namespace [cxx::Bits](#)  
*Internal helpers for the cxx package.*

## 16.139.1 Detailed Description

AVL map.

Definition in file [avl\\_map](#).

## 16.140 avl\_map

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.

```

```

00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023
00024 #pragma once
00025
00026 #include <14/cxx/std_alloc>
00027 #include <14/cxx/std_ops>
00028 #include <14/cxx/pair>
00029 #include <14/cxx/avl_set>
00030
00031 namespace cxx {
00032 namespace Bits {
00033
00035 template<typename KEY_TYPE>
00036 struct Avl_map_get_key
00037 {
00038     typedef KEY_TYPE Key_type;
00039     template<typename NODE>
00040     static Key_type const &key_of(NODE const *n)
00041     { return n->item.first; }
00042 };
00043 }
00044
00053 template< typename KEY_TYPE, typename DATA_TYPE,
00054 template<typename A> class COMPARE = Lt_functor,
00055 template<typename B> class ALLOC = New_allocator >
00056 class Avl_map :
00057     public Bits::Base_avl_set<Pair<KEY_TYPE, DATA_TYPE>,
00058                             COMPARE<KEY_TYPE>, ALLOC,
00059                             Bits::Avl_map_get_key<KEY_TYPE> >
00060 {
00061 private:
00062     typedef Pair<KEY_TYPE, DATA_TYPE> Local_item_type;
00063     typedef Bits::Base_avl_set<Local_item_type, COMPARE<KEY_TYPE>, ALLOC,
00064                             Bits::Avl_map_get_key<KEY_TYPE> > Base_type;
00065
00066 public:
00068     typedef COMPARE<KEY_TYPE> Key_compare;
00070     typedef KEY_TYPE Key_type;
00072     typedef DATA_TYPE Data_type;
00074     typedef typename Base_type::Node Node;
00076     typedef typename Base_type::Node_allocator Node_allocator;
00077
00078     typedef typename Base_type::Iterator Iterator;
00079     typedef typename Base_type::Iterator iterator;
00080     typedef typename Base_type::Const_iterator Const_iterator;
00081     typedef typename Base_type::Const_iterator const_iterator;
00082     typedef typename Base_type::Rev_iterator Rev_iterator;
00083     typedef typename Base_type::Rev_iterator reverse_iterator;
00084     typedef typename Base_type::Const_rev_iterator Const_rev_iterator;
00085     typedef typename Base_type::Const_rev_iterator const_reverse_iterator;
00086
00091     Avl_map(Node_allocator const &alloc = Node_allocator())
00092         : Base_type(alloc)
00093     {}
00094
00110     cxx::Pair<Iterator, int> insert(Key_type const &key, Data_type const &data)
00111     { return Base_type::insert(Pair<Key_type, Data_type>(key, data)); }
00112
00113     template<typename... Args>
00114     cxx::Pair<Iterator, int> emplace(Args &&...args)
00115     { return Base_type::emplace(cxx::forward<Args>(args)...); }
00116
00122     Data_type const &operator [] (Key_type const &key) const
00123     { return this->find_node(key)->second; }
00124
00134     Data_type &operator [] (Key_type const &key)
00135     {
00136         Node n = this->find_node(key);
00137         if (n)
00138             return const_cast<Data_type&>(n->second);
00139         else
00140             return insert(key, Data_type()).first->second;
00141     }
00142 };
00143
00144 }
00145

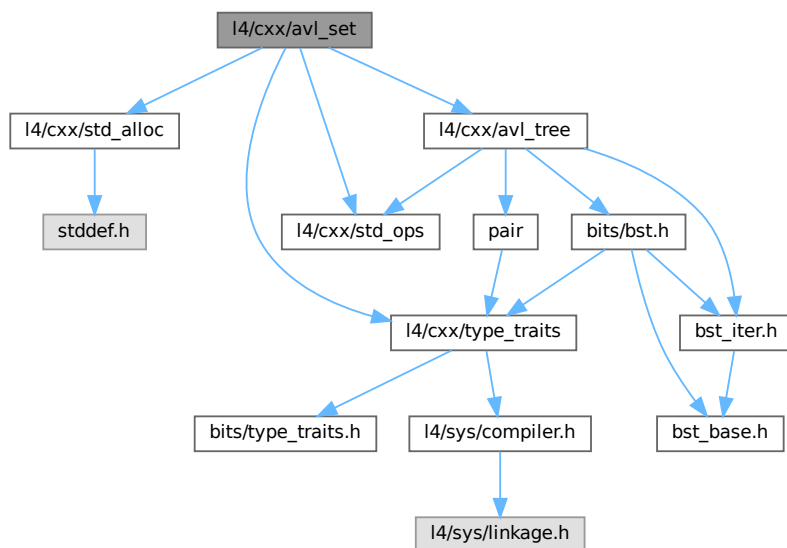
```

## 16.141 l4/cxx/avl\_set File Reference

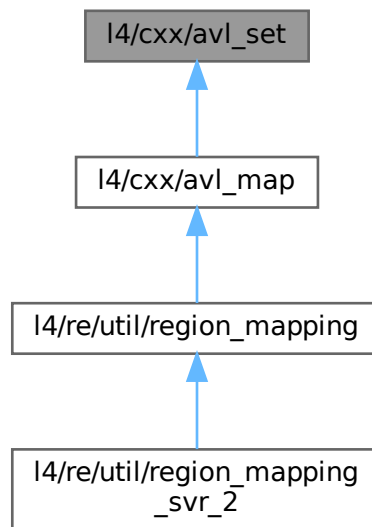
AVL set.

```
#include <l4/cxx/std_alloc>
#include <l4/cxx/std_ops>
#include <l4/cxx/type_traits>
#include <l4/cxx/avl_tree>
```

Include dependency graph for avl\_set:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [cxx::Bits::Avl\\_set\\_get\\_key< KEY\\_TYPE >](#)  
*Internal, key-getter for [Avl\\_set](#) nodes.*
- class [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >](#)  
*Internal: AVL set with internally managed nodes.*
- class [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >::Node](#)  
*A smart pointer to a tree item.*
- class [cxx::Avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC >](#)  
*AVL set for simple compareable items.*

## Namespaces

- namespace [cxx](#)  
*Our C++ library.*
- namespace [cxx::Bits](#)  
*Internal helpers for the [cxx](#) package.*

### 16.141.1 Detailed Description

AVL set.

Definition in file [avl\\_set](#).

## 16.142 avl\_set

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include <l4/cxx/std_alloc>
00028 #include <l4/cxx/std_ops>
00029 #include <l4/cxx/type_traits>
00030 #include <l4/cxx/avl_tree>
00031
00032 struct Avl_set_tester;
00033
00034 namespace cxx {
00035 namespace Bits {
00044 template< typename Node, typename Key, typename Node_op >
00045 class Avl_set_iter : public __Bst_iter_b<Node, Node_op>
00046 {
00047 private:
00049     typedef __Bst_iter_b<Node, Node_op> Base;
00050
00052     typedef typename Type_traits<Key>::Non_const_type Non_const_key;
00053
00055     typedef Avl_set_iter<Node, Non_const_key, Node_op> Non_const_iter;
00056
00057     using Base::_n;
00058     using Base::_r;
00059     using Base::_inc;
00060
00061 public:
00063     Avl_set_iter() = default;
00064
00069     Avl_set_iter(Node const *t) : Base(t) {}
00070
00075     Avl_set_iter(Base const &o) : Base(o) {}
00076
00078     Avl_set_iter(Non_const_iter const &o)
00079     : Base(o) {}
00080
00082     Avl_set_iter &operator = (Non_const_iter const &o)
00083     { Base::operator = (o); return *this; }
00084
00089     Key &operator * () const { return const_cast<Node*>(_n)->item; }
00094     Key *operator -> () const { return &const_cast<Node*>(_n)->item; }
00098     Avl_set_iter &operator ++ () { inc(); return *this; }
00102     Avl_set_iter operator ++ (int)
00103     { Avl_set_iter tmp = *this; inc(); return tmp; }
00104
00105 };
00106
00108 template<typename KEY_TYPE>
00109 struct Avl_set_get_key
00110 {
00111     typedef KEY_TYPE Key_type;
00112     template<typename NODE>
00113     static Key_type const &key_of(NODE const *n)
00114     { return n->item; }
00115 };
00116
00117
00130 template< typename ITEM_TYPE, class COMPARE,
00131           template<typename A> class ALLOC,
00132           typename GET_KEY>
00133 class Base_avl_set
00134 {
00135     friend struct ::Avl_set_tester;

```

```

00136
00137 public:
00144 enum
00145 {
00146     E_noent = 2,
00147     E_exist = 17,
00148     E_nomem = 12,
00149     E_inval = 22
00150 };
00152 typedef ITEM_TYPE Item_type;
00154 typedef GET_KEY Get_key;
00156 typedef typename GET_KEY::Key_type Key_type;
00158 typedef typename Type_traits<Item_type>::Const_type Const_item_type;
00160 typedef COMPARE Item_compare;
00161
00162 private:
00164 class _Node : public Avl_tree_node
00165 {
00166 public:
00168     Item_type item;
00169
00170     _Node() = default;
00171
00172     _Node(Item_type const &item) : Avl_tree_node(), item(item) {}
00173
00174     template<typename ...ARGS>
00175     _Node(ARGS &&...args) : Avl_tree_node(), item(cxx::forward<ARGS>(args)...)
00176     {}
00177 };
00178
00179 public:
00183 class Node
00184 {
00185 private:
00186     struct No_type;
00187     friend class Base_avl_set<ITEM_TYPE, COMPARE, ALLOC, GET_KEY>;
00188     _Node const *_n;
00189     explicit Node(_Node const *n) : _n(n) {}
00190
00191 public:
00193     Node() : _n(0) {}
00194
00200     Item_type const &operator * () { return _n->item; }
00206     Item_type const *operator -> () { return _n->item; }
00207
00212     bool valid() const { return _n; }
00213
00215     operator Item_type const * () { return _n ? _n->item : 0; }
00216 };
00217
00219 typedef ALLOC<_Node> Node_allocator;
00220
00221 private:
00222 typedef Avl_tree<_Node, GET_KEY, COMPARE> Tree;
00223 Tree _tree;
00225 Node_allocator _alloc;
00226
00227 Base_avl_set &operator = (Base_avl_set const &) = delete;
00228
00229 typedef typename Tree::Fwd_iter_ops Fwd;
00230 typedef typename Tree::Rev_iter_ops Rev;
00231
00232 public:
00233 typedef typename Type_traits<Item_type>::Param_type Item_param_type;
00234
00236 typedef Avl_set_iter<_Node, Item_type, Fwd> Iterator;
00237 typedef Iterator iterator;
00239 typedef Avl_set_iter<_Node, Const_item_type, Fwd> Const_iterator;
00240 typedef Const_iterator const_iterator;
00242 typedef Avl_set_iter<_Node, Item_type, Rev> Rev_iterator;
00243 typedef Rev_iterator reverse_iterator;
00245 typedef Avl_set_iter<_Node, Const_item_type, Rev> Const_rev_iterator;
00246 typedef Const_rev_iterator const_reverse_iterator;
00247
00254 explicit Base_avl_set(Node_allocator const &alloc = Node_allocator())
00255 : _tree(), _alloc(alloc)
00256 {}
00257
00258 ~Base_avl_set()
00259 {
00260     _tree.remove_all([this](_Node *n)
00261     {
00262         n->~_Node();
00263         _alloc.free(n);
00264     });
00265 }
00266

```

```

00274 inline Base_avl_set(Base_avl_set const &o);
00275
00293 cxx::Pair<Iterator, int> insert(Item_type const &item);
00294
00295 template<typename... Args>
00296 cxx::Pair<Iterator, int> emplace(Args&&... args);
00297
00306 int remove(Key_type const &item)
00307 {
00308     _Node *n = _tree.remove(item);
00309
00310     if (n)
00311     {
00312         n->~_Node();
00313         _alloc.free(n);
00314     }
00315     return 0;
00316
00317     return -E_noent;
00318 }
00319
00324 int erase(Key_type const &item)
00325 { return remove(item); }
00326
00335 Node find_node(Key_type const &item) const
00336 { return Node(_tree.find_node(item)); }
00337
00346 Node lower_bound_node(Key_type const &key) const
00347 { return Node(_tree.lower_bound_node(key)); }
00348
00349 Node lower_bound_node(Key_type &&key) const
00350 { return Node(_tree.lower_bound_node(key)); }
00351
00356 Const_iterator begin() const { return _tree.begin(); }
00361 Const_iterator end() const { return _tree.end(); }
00362
00367 Iterator begin() { return _tree.begin(); }
00372 Iterator end() { return _tree.end(); }
00373
00378 Const_rev_iterator rbegin() const { return _tree.rbegin(); }
00383 Const_rev_iterator rend() const { return _tree.rend(); }
00384
00389 Rev_iterator rbegin() { return _tree.rbegin(); }
00394 Rev_iterator rend() { return _tree.rend(); }
00395
00396 Const_iterator find(Key_type const &item) const
00397 { return _tree.find(item); }
00398
00399 #ifdef __DEBUG_L4_AVL
00400 bool rec_dump(bool print)
00401 {
00402     return _tree.rec_dump(print);
00403 }
00404 #endif
00405 };
00406
00407
00408 //-----
00409 /* Implementation of AVL Tree */
00410
00411 /* Create a copy */
00412 template< typename Item, class Compare, template<typename A> class Alloc, typename KEY_TYPE>
00413 Base_avl_set<Item, Compare, Alloc, KEY_TYPE>::Base_avl_set(Base_avl_set const &o)
00414 : _tree(), _alloc(o._alloc)
00415 {
00416     for (Const_iterator i = o.begin(); i != o.end(); ++i)
00417         insert(*i);
00418 }
00419
00420 /* Insert new _Node. */
00421 template< typename Item, class Compare, template< typename A > class Alloc, typename KEY_TYPE>
00422 Pair<typename Base_avl_set<Item, Compare, Alloc, KEY_TYPE>::Iterator, int>
00423 Base_avl_set<Item, Compare, Alloc, KEY_TYPE>::insert(Item const &item)
00424 {
00425     _Node *n = _alloc.alloc();
00426     if (!n)
00427         return cxx::pair(end(), -E_nomem);
00428
00429     new (n, Nothrow()) _Node(item);
00430     Pair<_Node *, bool> err = _tree.insert(n);
00431     if (!err.second)
00432     {
00433         n->~_Node();
00434         _alloc.free(n);
00435     }
00436
00437     return cxx::pair(Iterator(typename Tree::Iterator(err.first, err.first)), err.second ? 0 :

```

```

        -E_exist);
00438 }
00439
00440 /* In-place insert new _Node. */
00441 template< typename Item, class Compare, template< typename A > class Alloc, typename KEY_TYPE>
00442 template<typename... Args>
00443 Pair<typename Base_avl_set<Item,Compare,Alloc,KEY_TYPE>::Iterator, int>
00444 Base_avl_set<Item,Compare,Alloc,KEY_TYPE>::emplace(Args&&... args)
00445 {
00446     _Node *n = _alloc.alloc();
00447     if (!n)
00448         return cxx::pair(end(), -E_nomem);
00449
00450     new (n, Nothrow()) _Node(cxx::forward<Args>(args)...);
00451     Pair<_Node *, bool> err = _tree.insert(n);
00452     if (!err.second)
00453     {
00454         n->~_Node();
00455         _alloc.free(n);
00456     }
00457
00458     return cxx::pair(Iterator(typename Tree::Iterator(err.first, err.first)), err.second ? 0 :
        -E_exist);
00459 }
00460
00461 } // namespace Bits
00462
00474 template< typename ITEM_TYPE, class COMPARE = Lt_functor<ITEM_TYPE>,
00475           template<typename A> class ALLOC = New_allocator>
00476 class Avl_set :
00477     public Bits::Base_avl_set<ITEM_TYPE, COMPARE, ALLOC,
00478                             Bits::Avl_set_get_key<ITEM_TYPE> >
00479 {
00480 private:
00481     typedef Bits::Base_avl_set<ITEM_TYPE, COMPARE, ALLOC,
00482                             Bits::Avl_set_get_key<ITEM_TYPE> > Base;
00483
00484 public:
00485     typedef typename Base::Node_allocator Node_allocator;
00486     Avl_set() = default;
00487     Avl_set(Node_allocator const &alloc)
00488         : Base(alloc)
00489     {}
00490 };
00491
00492 } // namespace cxx

```

## 16.143 I4/cxx/avl\_tree File Reference

AVL tree.

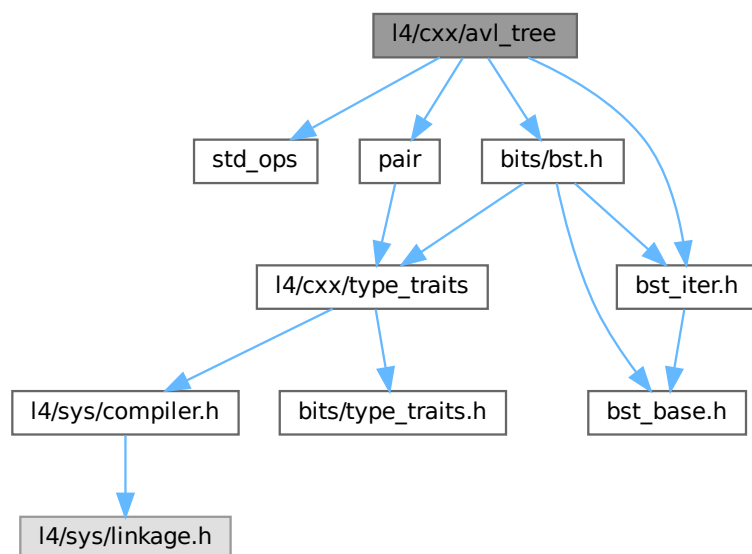
```

#include "std_ops"
#include "pair"
#include "bits/bst.h"
#include "bits/bst_iter.h"

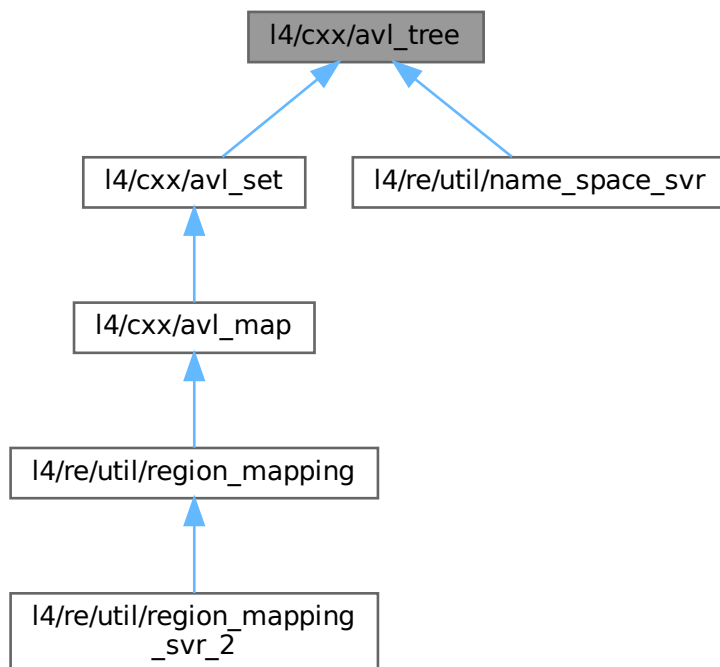
```



Include dependency graph for avl\_tree:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `cxx::Avl_tree_node`  
*Node of an AVL tree.*
- class `cxx::Avl_tree< Node, Get_key, Compare >`  
*A generic AVL tree.*

## Namespaces

- namespace `cxx`  
*Our C++ library.*

### 16.143.1 Detailed Description

AVL tree.

Definition in file [avl\\_tree](#).

## 16.144 avl\_tree

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *          Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009  *          economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include "std_ops"
00028 #include "pair"
00029
00030 #include "bits/bst.h"
00031 #include "bits/bst_iter.h"
00032
00033 struct Avl_set_tester;
00034
00035 namespace cxx {
00036
00040 class Avl_tree_node : public Bits::Bst_node
00041 {
00042     friend struct ::Avl_set_tester;
00043
00044 private:
00045     template< typename Node, typename Get_key, typename Compare >
00046     friend class Avl_tree;
00047
00049     typedef Bits::Direction Bal;
00051     typedef Bits::Direction Dir;
00052
00053     // We are a final BST node, hide interior.
00055     using Bits::Bst_node::next;
00056     using Bits::Bst_node::next_p;
```

```

00057     using Bits::Bst_node::rotate;
00061     Bal _balance;
00062
00063 protected:
00065     Avl_tree_node() = default;
00066
00067 private:
00068     Avl_tree_node(Avl_tree_node const &o) = delete;
00069     Avl_tree_node(Avl_tree_node &&o) = delete;
00070
00072     Avl_tree_node &operator = (Avl_tree_node const &o) = default;
00074     Avl_tree_node &operator = (Avl_tree_node &&o) = default;
00075
00077     explicit Avl_tree_node(bool) : Bits::Bst_node(true), _balance(Dir::N) {}
00078
00080     static Bits::Bst_node *rotate2(Bst_node **t, Bal idir, Bal pre);
00081
00083     bool balanced() const { return _balance == Bal::N; }
00084
00086     Bal balance() const { return _balance; }
00087
00089     void balance(Bal b) { _balance = b; }
00090 };
00091
00092
00109 template< typename Node, typename Get_key,
00110           typename Compare = Lt_functor<typename Get_key::Key_type> >
00111 class Avl_tree : public Bits::Bst<Node, Get_key, Compare>
00112 {
00113 private:
00114     typedef Bits::Bst<Node, Get_key, Compare> Bst;
00115
00117     using Bst::_head;
00118
00120     using Bst::k;
00121
00123     typedef typename Avl_tree_node::Bal Bal;
00125     typedef typename Avl_tree_node::Bal Dir;
00126
00127     Avl_tree(Avl_tree const &o) = delete;
00128     Avl_tree &operator = (Avl_tree const &o) = delete;
00129     Avl_tree(Avl_tree &&o) = delete;
00130     Avl_tree &operator = (Avl_tree &&o) = delete;
00131
00132 public:
00134     typedef typename Bst::Key_type Key_type;
00135     typedef typename Bst::Key_param_type Key_param_type;
00137
00138     // Grab iterator types from Bst
00141     typedef typename Bst::Iterator Iterator;
00143     typedef typename Bst::Const_iterator Const_iterator;
00145     typedef typename Bst::Rev_iterator Rev_iterator;
00147     typedef typename Bst::Const_rev_iterator Const_rev_iterator;
00149
00157     Pair<Node *, bool> insert(Node *new_node);
00158
00165     Node *remove(Key_param_type key);
00169     Node *erase(Key_param_type key) { return remove(key); }
00170
00172     Avl_tree() = default;
00173
00175     ~Avl_tree() noexcept
00176     {
00177         this->remove_all([](Node *){});
00178     }
00179
00180 #ifdef __DEBUG_L4_AVL
00181     bool rec_dump(Avl_tree_node *n, int depth, int *dp, bool print, char pfx);
00182     bool rec_dump(bool print)
00183     {
00184         int dp=0;
00185         return rec_dump(static_cast<Avl_tree_node *>(_head), 0, &dp, print, '+');
00186     }
00187 #endif
00188 };
00189
00190
00191 //-----
00192 /* IMPLEMENTATION: Bits::__Bst_iter_b */
00193
00194
00195 inline
00196 Bits::Bst_node *
00197 Avl_tree_node::rotate2(Bst_node **t, Bal idir, Bal pre)
00198 {
00199     typedef Bits::Bst_node N;
00200     typedef Avl_tree_node A;

```

```

00201 N *tmp[2] = { *t, N::next(*t, idir) };
00202 *t = N::next(tmp[1], !idir);
00203 A *n = static_cast<A*>(*t);
00204
00205 N::next(tmp[0], idir, N::next(n, !idir));
00206 N::next(tmp[1], !idir, N::next(n, idir));
00207 N::next(n, !idir, tmp[0]);
00208 N::next(n, idir, tmp[1]);
00209
00210 n->balance(Bal::N);
00211
00212 if (pre == Bal::N)
00213 {
00214     static_cast<A*>(tmp[0])->balance(Bal::N);
00215     static_cast<A*>(tmp[1])->balance(Bal::N);
00216     return 0;
00217 }
00218
00219 static_cast<A*>(tmp[pre != idir])->balance(!pre);
00220 static_cast<A*>(tmp[pre == idir])->balance(Bal::N);
00221
00222 return N::next(tmp[pre == idir], !pre);
00223 }
00224
00225 //-----
00226 /* Implementation of AVL Tree */
00227
00228 /* Insert new _Node. */
00229 template< typename Node, typename Get_key, class Compare>
00230 Pair<Node *, bool>
00231 Avl_tree<Node, Get_key, Compare>::insert(Node *new_node)
00232 {
00233     typedef Avl_tree_node A;
00234     typedef Bits::Bst_node N;
00235     N **t = &_head; /* search variable */
00236     N **s = &_head; /* node where rebalancing may occur */
00237     Key_param_type new_key = Get_key::key_of(new_node);
00238
00239     // search insertion point
00240     for (N *p; (p = *t);)
00241     {
00242         Dir b = this->dir(new_key, p);
00243         if (b == Dir::N)
00244             return pair(static_cast<Node*>(p), false);
00245
00246         if (!static_cast<A const *>(p)->balanced())
00247             s = t;
00248
00249         t = A::next_p(p, b);
00250     }
00251
00252     *static_cast<A*>(new_node) = A(true);
00253     *t = new_node;
00254
00255     N *n = *s;
00256     A *a = static_cast<A*>(n);
00257     if (!a->balanced())
00258     {
00259         A::Bal b(this->greater(new_key, n));
00260         if (a->balance() != b)
00261         {
00262             // ok we got in balance the shorter subtree go higher
00263             a->balance(Bal::N);
00264             // propagate the new balance down to the new node
00265             n = A::next(n, b);
00266         }
00267         else if (b == Bal(this->greater(new_key, A::next(n, b))))
00268         {
00269             // left-left or right-right case -> single rotation
00270             A::rotate(s, b);
00271             a->balance(Bal::N);
00272             static_cast<A*>(*s)->balance(Bal::N);
00273             n = A::next(*s, b);
00274         }
00275         else
00276         {
00277             // need a double rotation
00278             n = A::next(A::next(n, b), !b);
00279             n = A::rotate2(s, b, n == new_node ? Bal::N : Bal(this->greater(new_key, n)));
00280         }
00281     }
00282
00283     for (A::Bal b; n && n != new_node; static_cast<A*>(n)->balance(b), n = A::next(n, b))
00284         b = Bal(this->greater(new_key, n));
00285
00286     return pair(new_node, true);
00287 }

```

```

00288
00289
00290 /* remove an element */
00291 template< typename Node, typename Get_key, class Compare>
00292 inline
00293 Node *Avl_tree<Node, Get_key, Compare>::remove(Key_param_type key)
00294 {
00295     typedef Avl_tree_node A;
00296     typedef Bits::Bst_node N;
00297     N **q = &_head; /* search variable */
00298     N **s = &_head; /* last ('deepest') node on the search path to q
00299                      * with balance 0, at this place the rebalancing
00300                      * stops in any case */
00301     N **t = 0;
00302     Dir dir;
00303
00304     // find target node and rebalancing entry
00305     for (N *n; (n = *q); q = A::next_p(n, dir))
00306     {
00307         dir = Dir(this->greater(key, n));
00308         if (dir == Dir::L && !this->greater(k(n), key))
00309             /* found node */
00310             t = q;
00311
00312         if (!A::next(n, dir))
00313             break;
00314
00315         A const *a = static_cast<A const *>(n);
00316         if (a->balanced() || (a->balance() == !dir && A::next<A>(n, !dir)->balanced()))
00317             s = q;
00318     }
00319
00320     // nothing found
00321     if (!t)
00322         return 0;
00323
00324     A *i = static_cast<A*>(*t);
00325
00326     for (N *n; (n = *s); s = A::next_p(n, dir))
00327     {
00328         dir = Dir(this->greater(key, n));
00329
00330         if (!A::next(n, dir))
00331             break;
00332
00333         A *a = static_cast<A*>(n);
00334         // got one out of balance
00335         if (a->balanced())
00336             a->balance(!dir);
00337         else if (a->balance() == dir)
00338             a->balance(Bal::N);
00339         else
00340         {
00341             // we need rotations to get in balance
00342             Bal b = A::next<A>(n, !dir)->balance();
00343             if (b == dir)
00344                 A::rotate2(s, !dir, A::next<A>(A::next(n, !dir), dir)->balance());
00345             else
00346             {
00347                 A::rotate(s, !dir);
00348                 if (b != Bal::N)
00349                 {
00350                     a->balance(Bal::N);
00351                     static_cast<A*>(*s)->balance(Bal::N);
00352                 }
00353                 else
00354                 {
00355                     a->balance(!dir);
00356                     static_cast<A*>(*s)->balance(dir);
00357                 }
00358             }
00359             if (n == i)
00360                 t = A::next_p(*s, dir);
00361         }
00362     }
00363
00364     A *n = static_cast<A*>(*q);
00365     *t = n;
00366     *q = A::next(n, !dir);
00367     *n = *i;
00368
00369     return static_cast<Node*>(i);
00370 }
00371
00372 #ifdef __DEBUG_L4_AVL
00373 template< typename Node, typename Get_key, class Compare>
00374 bool Avl_tree<Node, Get_key, Compare>::rec_dump(Avl_tree_node *n, int depth, int *dp, bool print, char

```



## Namespaces

- namespace `L4`  
*L4 low-level kernel interface.*

## Variables

- `IOModifier` const `L4::hex`  
*Modifies the stream to print numbers as hexadecimal values.*
- `IOModifier` const `L4::dec`  
*Modifies the stream to print numbers as decimal values.*

### 16.145.1 Detailed Description

Basic IO stream.

Definition in file [basic\\_ostream](#).

## 16.146 basic\_ostream

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 namespace L4 {
00026
00033     class IOModifier
00034     {
00035     public:
00036         IOModifier(int x) : mod(x) {}
00037         bool operator == (IOModifier o) { return mod == o.mod; }
00038         bool operator != (IOModifier o) { return mod != o.mod; }
00039         int mod;
00040     };
00041
00046     class IOBackend
00047     {
00048     public:
00049         typedef int Mode;
00050
00051     protected:
00052         friend class BasicOStream;
00053
00054         IOBackend()
00055         : int_mode(10)
00056         {}
00057
00058         virtual ~IOBackend() {}
00059
00060         virtual void write(char const *str, unsigned len) = 0;
```

```

00061
00062 private:
00063     void write(IOModifier m);
00064     void write(long long int c, int len);
00065     void write(long long unsigned c, int len);
00066     void write(long long unsigned c, unsigned char base = 10,
00067               unsigned char len = 0, char pad = ' ');
00068
00069     Mode mode() const
00070     { return int_mode; }
00071
00072     void mode(Mode m)
00073     { int_mode = m; }
00074
00075     int int_mode;
00076 };
00077
00082 class BasicOStream
00083 {
00084 public:
00085     BasicOStream(IOBackend *b)
00086     : iob(b)
00087     {}
00088
00089     void write(char const *str, unsigned len)
00090     {
00091         if (iob)
00092             iob->write(str, len);
00093     }
00094
00095     void write(long long int c, int len)
00096     {
00097         if (iob)
00098             iob->write(c, len);
00099     }
00100
00101     void write(long long unsigned c, unsigned char base = 10,
00102               unsigned char len = 0, char pad = ' ')
00103     {
00104         if (iob)
00105             iob->write(c, base, len, pad);
00106     }
00107
00108     void write(long long unsigned c, int len)
00109     {
00110         if (iob)
00111             iob->write(c, len);
00112     }
00113
00114     void write(IOModifier m)
00115     {
00116         if (iob)
00117             iob->write(m);
00118     }
00119
00120     IOBackend::Mode be_mode() const
00121     {
00122         if (iob)
00123             return iob->mode();
00124         return 0;
00125     }
00126
00127     void be_mode(IOBackend::Mode m)
00128     {
00129         if (iob)
00130             iob->mode(m);
00131     }
00132
00133 private:
00134     IOBackend *iob;
00135 };
00136
00141 class IONumFmt
00142 {
00143 public:
00144     IONumFmt(unsigned long long n, unsigned char base = 10,
00145             unsigned char len = 0, char pad = ' ')
00146     : n(n), base(base), len(len), pad(pad)
00147     {}
00148
00149     BasicOStream &print(BasicOStream &o) const;
00150
00151 private:
00152     unsigned long long n;
00153     unsigned char base, len;
00154     char pad;
00155 };

```



```

00156
00157 inline IONumFmt n_hex(unsigned long long n) { return IONumFmt(n, 16); }
00158
00162 extern IOModifier const hex;
00163
00167 extern IOModifier const dec;
00168
00169 inline
00170 BasicOSTream &IONumFmt::print(BasicOSTream &o) const
00171 {
00172     o.write(n, base, len, pad);
00173     return o;
00174 }
00175 }
00176
00177
00178 // Implementation
00179
00180 inline
00181 L4::BasicOSTream &
00182 operator « (L4::BasicOSTream &s, char const * const str)
00183 {
00184     if (!str)
00185     {
00186         s.write("NULL", 6);
00187         return s;
00188     }
00189
00190     unsigned l = 0;
00191     for (; str[l] != 0; l++)
00192         ;
00193     s.write(str, l);
00194     return s;
00195 }
00196
00197 inline
00198 L4::BasicOSTream &
00199 operator « (L4::BasicOSTream &s, signed short u)
00200 {
00201     s.write(static_cast<long long signed>(u), -1);
00202     return s;
00203 }
00204
00205 inline
00206 L4::BasicOSTream &
00207 operator « (L4::BasicOSTream &s, signed u)
00208 {
00209     s.write(static_cast<long long signed>(u), -1);
00210     return s;
00211 }
00212
00213 inline
00214 L4::BasicOSTream &
00215 operator « (L4::BasicOSTream &s, signed long u)
00216 {
00217     s.write(static_cast<long long signed>(u), -1);
00218     return s;
00219 }
00220
00221 inline
00222 L4::BasicOSTream &
00223 operator « (L4::BasicOSTream &s, signed long long u)
00224 {
00225     s.write(u, -1);
00226     return s;
00227 }
00228
00229 inline
00230 L4::BasicOSTream &
00231 operator « (L4::BasicOSTream &s, unsigned short u)
00232 {
00233     s.write(static_cast<long long unsigned>(u), -1);
00234     return s;
00235 }
00236
00237 inline
00238 L4::BasicOSTream &
00239 operator « (L4::BasicOSTream &s, unsigned u)
00240 {
00241     s.write(static_cast<long long unsigned>(u), -1);
00242     return s;
00243 }
00244
00245 inline
00246 L4::BasicOSTream &
00247 operator « (L4::BasicOSTream &s, unsigned long u)
00248 {

```

```

00249     s.write(static_cast<long long unsigned>(u), -1);
00250     return s;
00251 }
00252
00253 inline
00254 L4::BasicOStream &
00255 operator « (L4::BasicOStream &s, unsigned long long u)
00256 {
00257     s.write(u, -1);
00258     return s;
00259 }
00260
00261 inline
00262 L4::BasicOStream &
00263 operator « (L4::BasicOStream &s, void const *u)
00264 {
00265     long unsigned x = reinterpret_cast<long unsigned>(u);
00266     L4::IOBackend::Mode mode = s.be_mode();
00267     s.write(L4::hex);
00268     s.write(static_cast<long long unsigned>(x), -1);
00269     s.be_mode(mode);
00270     return s;
00271 }
00272
00273 inline
00274 L4::BasicOStream &
00275 operator « (L4::BasicOStream &s, L4::IOModifier m)
00276 {
00277     s.write(m);
00278     return s;
00279 }
00280
00281 inline
00282 L4::BasicOStream &
00283 operator « (L4::BasicOStream &s, char c)
00284 {
00285     s.write(&c, 1);
00286     return s;
00287 }
00288
00289 inline
00290 L4::BasicOStream &
00291 operator « (L4::BasicOStream &o, L4::IONumFmt const &n)
00292 { return n.print(o); }

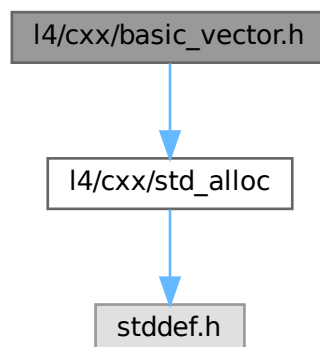
```

## 16.147 l4/cxx/basic\_vector.h File Reference

Basic vector.

```
#include <l4/cxx/std_alloc>
```

Include dependency graph for basic\_vector.h:



## Namespaces

- namespace `cxx`  
*Our C++ library.*

### 16.147.1 Detailed Description

Basic vector.

Definition in file [basic\\_vector.h](#).

## 16.148 basic\_vector.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #pragma once
00023
00024 #include <14/cxx/std_alloc>
00025
00026 namespace cxx {
00027
00028 template< typename T >
00029 class Basic_vector
00030 {
00031 public:
00032     Basic_vector(T *array, unsigned long capacity)
00033         : _array(array), _capacity(capacity)
00034     {
00035         for (unsigned long i = 0; i < capacity; ++i)
00036             new (&_array[i]) T();
00037     }
00038
00039 private:
00040     T *_array;
00041     unsigned long _capacity;
00042 };
00043
00044 };

```

## 16.149 bitfield

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2012 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate

```

```

00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019
00020 #pragma once
00021
00022 #include "type_list"
00023
00025 namespace cxx {
00026
00034 template<typename T, unsigned LSB, unsigned MSB>
00035 class Bitfield
00036 {
00037 private:
00038     typedef remove_reference_t<T> Base_type;
00039
00040     static_assert(MSB >= LSB, "boundary mismatch in bit-field definition");
00041     static_assert(MSB < sizeof(Base_type) * 8, "MSB outside of bit-field type");
00042     static_assert(LSB < sizeof(Base_type) * 8, "LSB outside of bit-field type");
00043
00049     template<unsigned BITS> struct Best_type
00050     {
00051         template< typename TY > struct Cmp { enum { value = (BITS <= sizeof(TY)*8) }; };
00052         typedef cxx::type_list<
00053             unsigned char,
00054             unsigned short,
00055             unsigned int,
00056             unsigned long,
00057             unsigned long long
00058         > Unsigned_types;
00059         typedef cxx::find_type_t<Unsigned_types, Cmp> Type;
00060     };
00061
00062 public:
00063     enum
00064     {
00065         Bits = MSB + 1 - LSB,
00066         Lsb = LSB,
00067         Msb = MSB,
00068     };
00069
00071     enum Masks : Base_type
00072     {
00073         Low_mask = static_cast<Base_type>(~0ULL) >> (sizeof(Base_type)*8 - Bits),
00074         Mask = Low_mask << Lsb,
00075     };
00076
00085     typedef typename Best_type<Bits>::Type Bits_type;
00086
00093     typedef typename Best_type<Bits + Lsb>::Type Shift_type;
00094
00095 private:
00096     static_assert(sizeof(Bits_type)*8 >= Bits, "error finding the type to store the bits");
00097     static_assert(sizeof(Shift_type)*8 >= Bits + Lsb, "error finding the type to keep the shifted
00098 bits");
00098     static_assert(sizeof(Bits_type) <= sizeof(Base_type), "size mismatch for Bits_type");
00099     static_assert(sizeof(Shift_type) <= sizeof(Base_type), "size mismatch for Shift_type");
00100     static_assert(sizeof(Bits_type) <= sizeof(Shift_type), "size mismatch for Shift_type and
00101 Bits_type");
00102
00102 public:
00110     static constexpr Bits_type get(Shift_type val)
00111     { return (val >> Lsb) & Low_mask; }
00112
00123     static constexpr Base_type get_unshifted(Shift_type val)
00124     { return val & Mask; }
00125
00138     static constexpr Base_type set_dirty(Base_type dest, Shift_type val)
00139     {
00140         //assert (! (val & ~Low_mask));
00141         return (dest & ~Mask) | (val << Lsb);
00142     }
00143
00158     static constexpr Base_type set_unshifted_dirty(Base_type dest, Shift_type val)
00159     {
00160         //assert (! (val & ~Mask));
00161         return (dest & ~Mask) | val;
00162     }
00163
00172     static Base_type set(Base_type dest, Bits_type val)
00173     { return set_dirty(dest, val & Low_mask); }
00174
00184     static Base_type set_unshifted(Base_type dest, Shift_type val)

```

```

00185 { return set_unshifted_dirty(dest, val & Mask); }
00186
00198 static constexpr Base_type val_dirty(Shift_type val) { return val << Lsb; }
00199
00207 static constexpr Base_type val(Bits_type val) { return val_dirty(val & Low_mask); }
00208
00217 static constexpr Base_type val_unshifted(Shift_type val) { return val & Mask; }
00218
00220 template< typename TT >
00221 class Value_base
00222 {
00223 private:
00224     TT v;
00225
00226 public:
00227     constexpr Value_base(TT t) : v(t) {}
00228     constexpr Bits_type get() const { return Bitfield::get(v); }
00229     constexpr Base_type get_unshifted() const { return Bitfield::get_unshifted(v); }
00230
00231     void set(Bits_type val) { v = Bitfield::set(v, val); }
00232     void set_dirty(Bits_type val) { v = Bitfield::set_dirty(v, val); }
00233     void set_unshifted(Shift_type val) { v = Bitfield::set_unshifted(v, val); }
00234     void set_unshifted_dirty(Shift_type val) { v = Bitfield::set_unshifted_dirty(v, val); }
00235 };
00236
00238 template< typename TT >
00239 class Value : public Value_base<TT>
00240 {
00241 public:
00242     constexpr Value(TT t) : Value_base<TT>(t) {}
00243     constexpr operator Bits_type () const { return this->get(); }
00244     constexpr Value &operator = (Bits_type val) { this->set(val); return *this; }
00245     constexpr Value &operator = (Value const &val)
00246     { this->set(val.get()); return *this; }
00247     Value(Value const &) = default;
00248 };
00249
00251 template< typename TT >
00252 class Value_unshifted : public Value_base<TT>
00253 {
00254 public:
00255     constexpr Value_unshifted(TT t) : Value_base<TT>(t) {}
00256     constexpr operator Shift_type () const { return this->get_unshifted(); }
00257     constexpr Value_unshifted &operator = (Shift_type val) { this->set_unshifted(val); return *this; }
00258     constexpr Value_unshifted &operator = (Value_unshifted const &val)
00259     { this->set_unshifted(val.get_unshifted()); return *this; }
00260     Value_unshifted(Value_unshifted const &) = default;
00261 };
00262
00264 typedef Value<Base_type &> Ref;
00266 typedef Value<Base_type volatile &> Ref_volatile;
00268 typedef Value<Base_type const> Val;
00269
00271 typedef Value_unshifted<Base_type &> Ref_unshifted;
00273 typedef Value_unshifted<Base_type volatile &> Ref_unshifted_volatile;
00275 typedef Value_unshifted<Base_type const> Val_unshifted;
00276 };
00277
00278 #define CXX_BITFIELD_MEMBER(LSB, MSB, name, data_member) \
00279 \
00280 \
00281 typedef cxx::Bitfield<decltype(data_member), LSB, MSB> name ## _bfm_t; \
00282 \
00283 constexpr typename name ## _bfm_t::Val name() const { return data_member; } \
00284 typename name ## _bfm_t::Val name() const volatile { return data_member; } \
00285 \
00286 constexpr typename name ## _bfm_t::Ref name() { return data_member; } \
00287 typename name ## _bfm_t::Ref_volatile name() volatile { return data_member; } \
00288
00290 #define CXX_BITFIELD_MEMBER_RO(LSB, MSB, name, data_member) \
00291 \
00292 \
00293 typedef cxx::Bitfield<decltype(data_member), LSB, MSB> name ## _bfm_t; \
00294 \
00295 constexpr typename name ## _bfm_t::Val name() const { return data_member; } \
00296 typename name ## _bfm_t::Val name() const volatile { return data_member; } \
00297
00299 #define CXX_BITFIELD_MEMBER_UNSHIFTED(LSB, MSB, name, data_member) \
00300 \
00301 \
00302 typedef cxx::Bitfield<decltype(data_member), LSB, MSB> name ## _bfm_t; \
00303 \
00304 constexpr typename name ## _bfm_t::Val_unshifted name() const { return data_member; } \
00305 typename name ## _bfm_t::Val_unshifted name() const volatile { return data_member; } \
00306 \
00307 constexpr typename name ## _bfm_t::Ref_unshifted name() { return data_member; } \
00308 typename name ## _bfm_t::Ref_unshifted_volatile name() volatile { return data_member; } \

```

```

00309
00311 #define CXX_BITFIELD_MEMBER_UNSHIFTED_RO(LSB, MSB, name, data_member) \
00312 \
00313 \
00314 typedef cxx::Bitfield<decltype(data_member), LSB, MSB> name ## _bfm_t; \
00315 \
00316 constexpr typename name ## _bfm_t::Val_unshifted name() const { return data_member; } \
00317 typename name ## _bfm_t::Val_unshifted name() const volatile { return data_member; } \
00318
00319 }

```

## 16.150 bitmap

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2014 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019
00020 #pragma once
00021
00022 namespace cxx {
00023
00024 class Bitmap_base
00025 {
00026 protected:
00027     typedef unsigned long word_type;
00028
00029     enum
00030     {
00031         W_bits = sizeof(word_type) * 8,
00032         C_bits = 8,
00033     };
00034
00035     word_type *_bits;
00036
00037     static unsigned word_index(unsigned bit) { return bit / W_bits; }
00038     static unsigned bit_index(unsigned bit) { return bit % W_bits; }
00039
00040     class Bit
00041     {
00042     public:
00043         Bit(Bitmap_base *bm, long bit) : _bm(bm), _bit(bit) {}
00044         Bit &operator = (bool val) { _bm->bit(_bit, val); return *this; }
00045         operator bool () const { return _bm->bit(_bit); }
00046     };
00047
00048 public:
00049     explicit Bitmap_base(void *bits) noexcept : _bits(reinterpret_cast<word_type *>(bits)) {}
00050
00051     static long words(long bits) noexcept { return (bits + W_bits - 1) / W_bits; }
00052     static long bit_buffer_bytes(long bits) noexcept
00053     { return words(bits) * W_bits / 8; }
00054
00055     template< long BITS >
00056     class Word
00057     {
00058     public:
00059         typedef unsigned long Type;
00060         enum
00061         {
00062             Size = (BITS + W_bits - 1) / W_bits
00063         };
00064     };
00065
00066     static long chars(long bits) noexcept

```

```

00102 { return (bits + C_bits - 1) / C_bits; }
00103
00105 template< long BITS >
00106 class Char
00107 {
00108 public:
00109     typedef unsigned char Type;
00110     enum
00111     {
00112         Size = (BITS + C_bits - 1) / C_bits
00113     };
00114 };
00115
00122 void bit(long bit, bool on) noexcept;
00123
00129 void clear_bit(long bit) noexcept;
00130
00138 void atomic_clear_bit(long bit) noexcept;
00139
00147 word_type atomic_get_and_clear(long bit) noexcept;
00148
00154 void set_bit(long bit) noexcept;
00155
00163 void atomic_set_bit(long bit) noexcept;
00164
00172 word_type atomic_get_and_set(long bit) noexcept;
00173
00182 word_type bit(long bit) const noexcept;
00183
00192 word_type operator [] (long bit) const noexcept
00193 { return this->bit(bit); }
00194
00202 Bit operator [] (long bit) noexcept
00203 { return Bit(this, bit); }
00204
00216 long scan_zero(long max_bit, long start_bit = 0) const noexcept;
00217
00218 void *bit_buffer() const noexcept { return _bits; }
00219
00220 protected:
00221     static int _bz1(unsigned long w) noexcept;
00222 };
00223
00224
00230 template<int BITS>
00231 class Bitmap : public Bitmap_base
00232 {
00233 private:
00234     char _bits[Bitmap_base::Char<BITS>::Size];
00235
00236 public:
00238     Bitmap() noexcept : Bitmap_base(_bits) {}
00239     Bitmap(Bitmap<BITS> const &o) noexcept : Bitmap_base(_bits)
00240     { __builtin_memcpy(_bits, o._bits, sizeof(_bits)); }
00253     long scan_zero(long start_bit = 0) const noexcept;
00254
00255     void clear_all()
00256     { __builtin_memset(_bits, 0, sizeof(_bits)); }
00257 };
00258
00259
00260 inline
00261 void
00262 Bitmap_base::bit(long bit, bool on) noexcept
00263 {
00264     long idx = word_index(bit);
00265     long b = bit_index(bit);
00266     _bits[idx] = (_bits[idx] & ~(1UL << b)) | (static_cast<unsigned long>(on) << b);
00267 }
00268
00269 inline
00270 void
00271 Bitmap_base::clear_bit(long bit) noexcept
00272 {
00273     long idx = word_index(bit);
00274     long b = bit_index(bit);
00275     _bits[idx] &= ~(1UL << b);
00276 }
00277
00278 inline
00279 void
00280 Bitmap_base::atomic_clear_bit(long bit) noexcept
00281 {
00282     long idx = word_index(bit);
00283     long b = bit_index(bit);
00284     word_type mask = 1UL << b;
00285     __atomic_and_fetch(&_bits[idx], ~mask, __ATOMIC_RELAXED);

```

```

00286 }
00287
00288 inline
00289 Bitmap_base::word_type
00290 Bitmap_base::atomic_get_and_clear(long bit) noexcept
00291 {
00292     long idx = word_index(bit);
00293     long b   = bit_index(bit);
00294     word_type mask = 1UL << b;
00295     return __atomic_fetch_and(&_bits[idx], ~mask, __ATOMIC_RELAXED) & mask;
00296 }
00297
00298 inline
00299 void
00300 Bitmap_base::set_bit(long bit) noexcept
00301 {
00302     long idx = word_index(bit);
00303     long b   = bit_index(bit);
00304     _bits[idx] |= (1UL << b);
00305 }
00306
00307 inline
00308 void
00309 Bitmap_base::atomic_set_bit(long bit) noexcept
00310 {
00311     long idx = word_index(bit);
00312     long b   = bit_index(bit);
00313     word_type mask = 1UL << b;
00314     __atomic_or_fetch(&_bits[idx], mask, __ATOMIC_RELAXED);
00315 }
00316
00317 inline
00318 Bitmap_base::word_type
00319 Bitmap_base::atomic_get_and_set(long bit) noexcept
00320 {
00321     long idx = word_index(bit);
00322     long b   = bit_index(bit);
00323     word_type mask = 1UL << b;
00324     return __atomic_fetch_or(&_bits[idx], mask, __ATOMIC_RELAXED) & mask;
00325 }
00326
00327 inline
00328 Bitmap_base::word_type
00329 Bitmap_base::bit(long bit) const noexcept
00330 {
00331     long idx = word_index(bit);
00332     long b   = bit_index(bit);
00333     return _bits[idx] & (1UL << b);
00334 }
00335
00336 inline
00337 int
00338 Bitmap_base::bzl(unsigned long w) noexcept
00339 {
00340     for (int i = 0; i < W_bits; ++i, w >= 1)
00341     {
00342         if ((w & 1) == 0)
00343             return i;
00344     }
00345     return -1;
00346 }
00347
00348 inline
00349 long
00350 Bitmap_base::scan_zero(long max_bit, long start_bit) const noexcept
00351 {
00352     if (!operator [] (start_bit))
00353         return start_bit;
00354     long idx = word_index(start_bit);
00355     max_bit -= start_bit & ~(W_bits - 1);
00356     for (; max_bit > 0; max_bit -= W_bits, ++idx)
00357     {
00358         if (_bits[idx] == 0)
00359             return idx * W_bits;
00360         if (_bits[idx] != ~0UL)
00361         {
00362             long zbit = _bzl(_bits[idx]);
00363             return zbit < max_bit ? idx * W_bits + zbit : -1;
00364         }
00365     }
00366     return -1;
00367 }
00368
00369
00370
00371
00372 }

```

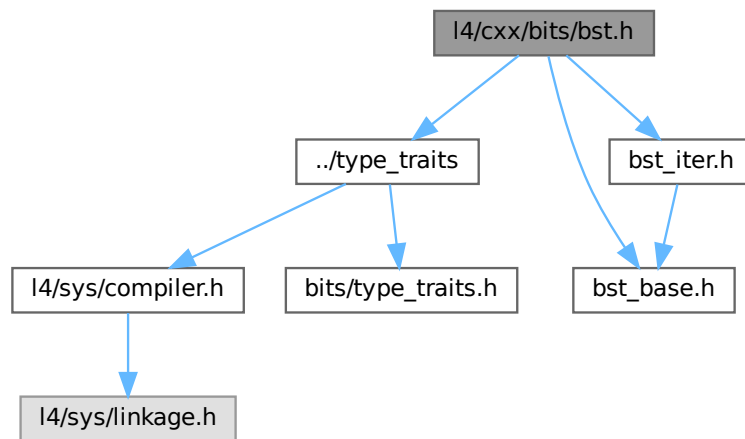


```
00373
00374 template<int BITS> inline
00375 long
00376 Bitmap<BITS>::scan_zero(long start_bit) const noexcept
00377 {
00378     return Bitmap_base::scan_zero(BITS, start_bit);
00379 }
00380
00381 };
```

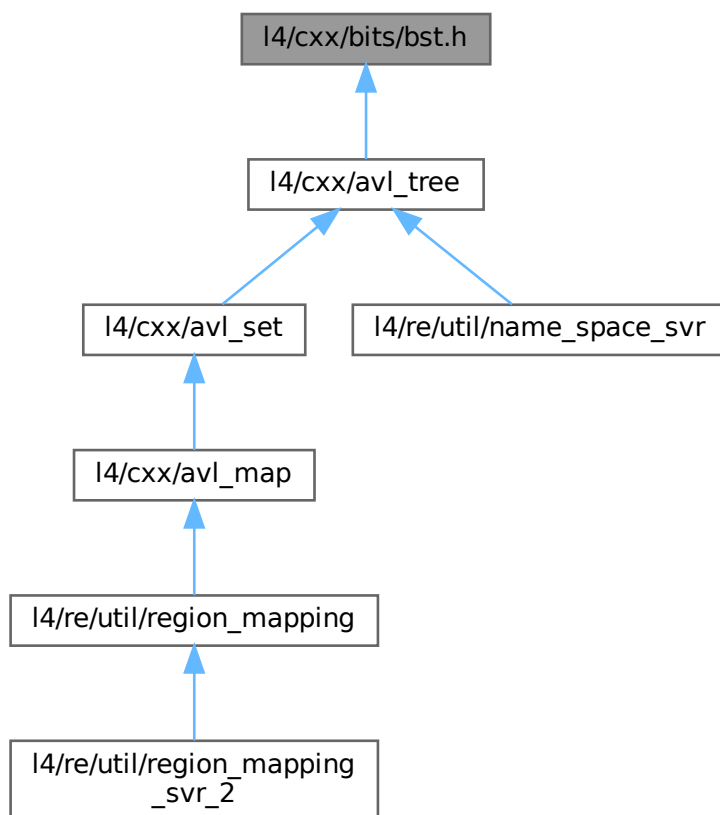
## 16.151 I4/cxx/bits/bst.h File Reference

AVL tree.

```
#include "../type_traits"
#include "bst_base.h"
#include "bst_iter.h"
Include dependency graph for bst.h:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `cxx::Bits::Bst< Node, Get_key, Compare >`  
*Basic binary search tree (BST).*

## Namespaces

- namespace `cxx`  
*Our C++ library.*
- namespace `cxx::Bits`  
*Internal helpers for the cxx package.*

## 16.151.1 Detailed Description

AVL tree.

Definition in file [bst.h](#).

## 16.152 bst.h

[Go to the documentation of this file.](#)

```

00001 // vi:ft=c++
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include "../type_traits"
00027 #include "bst_base.h"
00028 #include "bst_iter.h"
00029
00030 struct Avl_set_tester;
00031
00032 namespace cxx { namespace Bits {
00033
00041 template< typename Node, typename Get_key, typename Compare >
00042 class Bst
00043 {
00044     friend struct ::Avl_set_tester;
00045
00046 private:
00047     typedef Direction Dir;
00048     struct Fwd
00049     {
00051         static Node *child(Node const *n, Direction d)
00052         { return Bst_node::next<Node>(n, d); }
00053
00054         static bool cmp(Node const *l, Node const *r)
00055         { return Compare()(Get_key::key_of(l), Get_key::key_of(r)); }
00056     };
00057
00058     struct Rev
00059     {
00061         static Node *child(Node const *n, Direction d)
00062         { return Bst_node::next<Node>(n, !d); }
00063
00064         static bool cmp(Node const *l, Node const *r)
00065         { return Compare()(Get_key::key_of(r), Get_key::key_of(l)); }
00066     };
00067
00068 public:
00070     typedef typename Get_key::Key_type Key_type;
00072     typedef typename Type_traits<Key_type>::Param_type Key_param_type;
00073
00075     typedef Fwd Fwd_iter_ops;
00077     typedef Rev Rev_iter_ops;
00078
00080
00082     typedef __Bst_iter<Node, Node, Fwd> Iterator;
00084     typedef __Bst_iter<Node, Node const, Fwd> Const_iterator;
00086     typedef __Bst_iter<Node, Node, Rev> Rev_iterator;
00088     typedef __Bst_iter<Node, Node const, Rev> Const_rev_iterator;
00091 protected:
00102     Bst_node *_head;
00103
00105     Bst() : _head(0) {}
00106
00108     Node *head() const { return static_cast<Node*>(_head); }
00109
00111     static Key_type k(Bst_node const *n)
00112     { return Get_key::key_of(static_cast<Node const *>(n)); }
00113
00122     static Dir dir(Key_param_type l, Key_param_type r)
00123     {
00124         Compare cmp;
00125         Dir d(cmp(r, l));
00126         if (d == Direction::L && !cmp(l, r))
00127             return Direction::N;

```

```

00128     return d;
00129 }
00130
00139 static Dir dir(Key_param_type l, Bst_node const *r)
00140 { return dir(l, k(r)); }
00141
00143 static bool greater(Key_param_type l, Key_param_type r)
00144 { return Compare()(r, l); }
00145
00147 static bool greater(Key_param_type l, Bst_node const *r)
00148 { return greater(l, k(r)); }
00149
00151 static bool greater(Bst_node const *l, Bst_node const *r)
00152 { return greater(k(l), k(r)); }
00161 template<typename FUNC>
00162 static void remove_tree(Bst_node *head, FUNC &&callback)
00163 {
00164     if (Bst_node *n = Bst_node::next(head, Dir::L))
00165         remove_tree(n, callback);
00166
00167     if (Bst_node *n = Bst_node::next(head, Dir::R))
00168         remove_tree(n, callback);
00169
00170     callback(static_cast<Node *>(head));
00171 }
00172
00173 public:
00174
00183 Const_iterator begin() const { return Const_iterator(head()); }
00188 Const_iterator end() const { return Const_iterator(); }
00189
00194 Iterator begin() { return Iterator(head()); }
00199 Iterator end() { return Iterator(); }
00200
00205 Const_rev_iterator rbegin() const { return Const_rev_iterator(head()); }
00210 Const_rev_iterator rend() const { return Const_rev_iterator(); }
00211
00216 Rev_iterator rbegin() { return Rev_iterator(head()); }
00221 Rev_iterator rend() { return Rev_iterator(); }
00229
00235 Node *find_node(Key_param_type key) const;
00236
00243 Node *lower_bound_node(Key_param_type key) const;
00244
00251 Const_iterator find(Key_param_type key) const;
00252
00261 template<typename FUNC>
00262 void remove_all(FUNC &&callback)
00263 {
00264     if (!_head)
00265         return;
00266
00267     Bst_node *head = _head;
00268     _head = 0;
00269     remove_tree(head, cxx::forward<FUNC>(callback));
00270 }
00271
00272
00274 };
00275
00276 /* find an element */
00277 template< typename Node, typename Get_key, class Compare>
00278 inline
00279 Node *
00280 Bst<Node, Get_key, Compare>::find_node(Key_param_type key) const
00281 {
00282     Dir d;
00283
00284     for (Bst_node *q = _head; q; q = Bst_node::next(q, d))
00285     {
00286         d = dir(key, q);
00287         if (d == Dir::N)
00288             return static_cast<Node*>(q);
00289     }
00290     return 0;
00291 }
00292
00293 template< typename Node, typename Get_key, class Compare>
00294 inline
00295 Node *
00296 Bst<Node, Get_key, Compare>::lower_bound_node(Key_param_type key) const
00297 {
00298     Dir d;
00299     Bst_node *r = 0;
00300
00301     for (Bst_node *q = _head; q; q = Bst_node::next(q, d))
00302     {

```

```

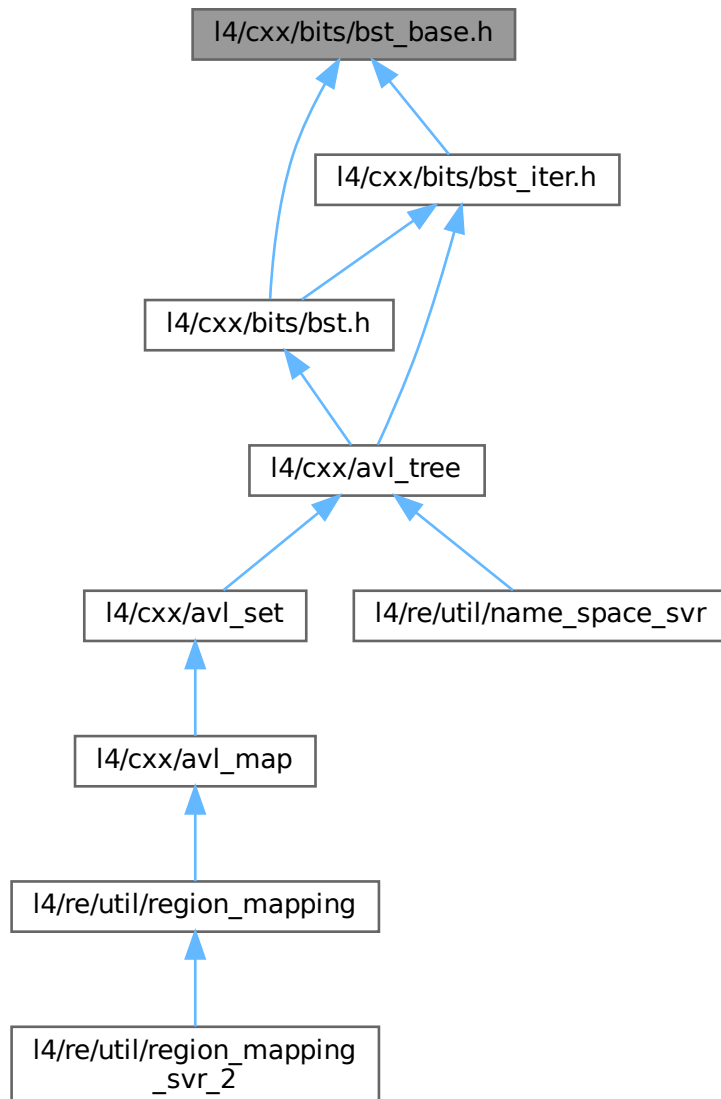
00303         d = dir(key, q);
00304         if (d == Dir::L)
00305             r = q; // found a node greater than key
00306         else if (d == Dir::N)
00307             return static_cast<Node*>(q);
00308         }
00309         return static_cast<Node*>(r);
00310     }
00311
00312     /* find an element */
00313     template< typename Node, typename Get_key, class Compare>
00314     inline
00315     typename Bst<Node, Get_key, Compare>::Const_iterator
00316     Bst<Node, Get_key, Compare>::find(Key_param_type key) const
00317     {
00318         Bst_node *q = _head;
00319         Bst_node *r = q;
00320
00321         for (Dir d; q; q = Bst_node::next(q, d))
00322         {
00323             d = dir(key, q);
00324             if (d == Dir::N)
00325                 return Iterator(static_cast<Node*>(q), static_cast<Node *>(r));
00326
00327             if (d != Dir::L && q == r)
00328                 r = Bst_node::next(q, d);
00329         }
00330         return Iterator();
00331     }
00332
00333 }

```

## 16.153 l4/cxx/bits/bst\_base.h File Reference

AVL tree.

This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [cxx::Bits::Direction](#)  
*The direction to go in a binary search tree.*
- class [cxx::Bits::Bst\\_node](#)  
*Basic type of a node in a binary search tree (BST).*

## Namespaces

- namespace [cxx](#)  
*Our C++ library.*
- namespace [cxx::Bits](#)  
*Internal helpers for the cxx package.*

## 16.153.1 Detailed Description

AVL tree.

Definition in file [bst\\_base.h](#).

## 16.154 bst\_base.h

[Go to the documentation of this file.](#)

```
00001 // vi:ft=cpp
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *                      Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 /*
00028  * This file contains very basic bits for implementing binary serach trees
00029  */
00030 namespace cxx {
00031 namespace Bits {
00032
00033 struct Direction
00034 {
00035     enum Direction_e
00036     {
00037         L = 0,
00038         R = 1,
00039         N = 2
00040     };
00041     unsigned char d;
00042
00043     Direction() = default;
00044
00045     Direction(Direction_e d) : d(d) {}
00046
00047     explicit Direction(bool b) : d(Direction_e(b)) /*d(b ? R : L)*/ {}
00048
00049     Direction operator ! () const { return Direction(!d); }
00050
00051     bool operator == (Direction_e o) const { return d == o; }
00052     bool operator != (Direction_e o) const { return d != o; }
00053     bool operator == (Direction o) const { return d == o.d; }
00054     bool operator != (Direction o) const { return d != o.d; }
00055 };
00056
00057 class Bst_node
00058 {
00059     // all BSTs are friends
00060     template< typename Node, typename Get_key, typename Compare >
00061     friend class Bst;
00062
00063 protected:
00064     static Bst_node *next(Bst_node const *p, Direction d)
00065     { return p->_c[d.d]; }
00066
00067     static void next(Bst_node *p, Direction d, Bst_node *n)
00068     { p->_c[d.d] = n; }
00069
00070     static Bst_node **next_p(Bst_node *p, Direction d)
00071     { return &p->_c[d.d]; }
00072
00073 }
```

```

00110     template< typename Node > static
00111     Node *next(Bst_node const *p, Direction d)
00112     { return static_cast<Node *>(p->_c[d.d]); }
00113
00115     static void rotate(Bst_node **t, Direction idir);
00118 private:
00119     Bst_node *_c[2];
00120
00121 protected:
00123     Bst_node() {}
00124
00126     explicit Bst_node(bool) { _c[0] = _c[1] = 0; }
00127 };
00128
00129 inline
00130 void
00131 Bst_node::rotate(Bst_node **t, Direction idir)
00132 {
00133     Bst_node *tmp = *t;
00134     *t = next(tmp, idir);
00135     next(tmp, idir, next(*t, !idir));
00136     next(*t, !idir, tmp);
00137 }
00138
00139 }}

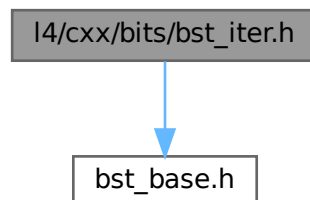
```

## 16.155 I4/cxx/bits/bst\_iter.h File Reference

AVL tree.

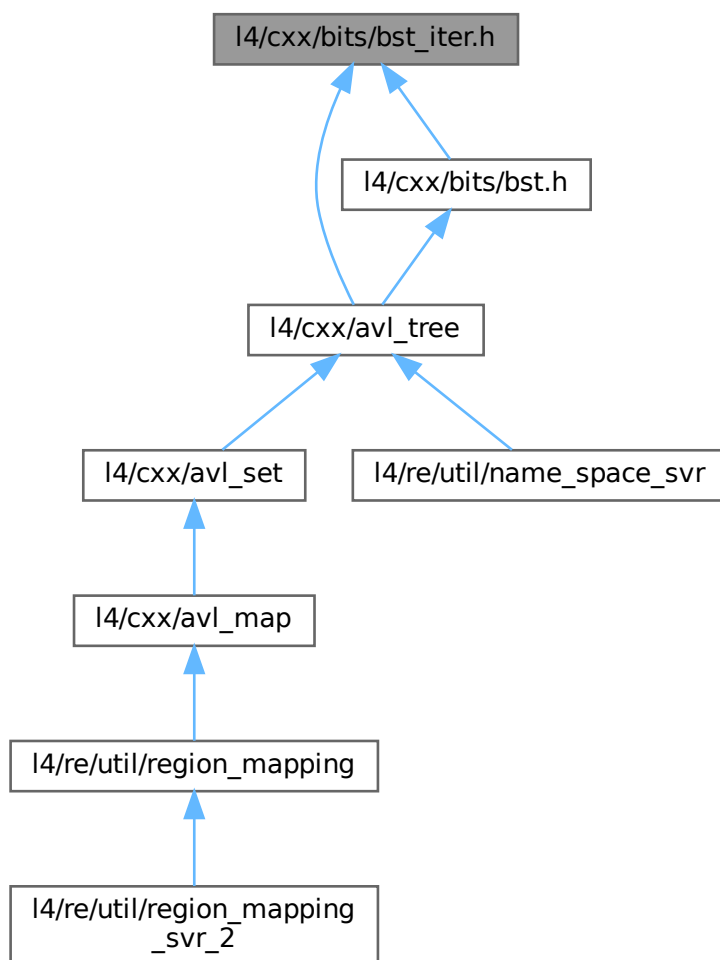
```
#include "bst_base.h"
```

Include dependency graph for bst\_iter.h:





This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace `cxx`  
*Our C++ library.*
- namespace `cxx::Bits`  
*Internal helpers for the cxx package.*

### 16.155.1 Detailed Description

AVL tree.

Definition in file `bst_iter.h`.

## 16.156 bst\_iter.h

[Go to the documentation of this file.](#)

```

00001 // vi:ft=cpp
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include "bst_base.h"
00028
00029 namespace cxx { namespace Bits {
00030
00031 template< typename Node, typename Node_op >
00032 class __Bst_iter_b
00033 {
00034 protected:
00035     typedef Direction Dir;
00036     Node const *_n;
00037     Node const *_r;
00038
00039     __Bst_iter_b() : _n(0), _r(0) {}
00040
00041     __Bst_iter_b(Node const *t)
00042         : _n(t), _r(_n)
00043     { _downmost(); }
00044
00045     __Bst_iter_b(Node const *t, Node const *r)
00046         : _n(t), _r(r)
00047     {}
00048
00049     inline void _downmost();
00050
00051     inline void inc();
00052
00053 public:
00054     bool operator == (__Bst_iter_b const &o) const { return _n == o._n; }
00055     bool operator != (__Bst_iter_b const &o) const { return _n != o._n; }
00056 };
00057
00058 template< typename Node, typename Node_type, typename Node_op >
00059 class __Bst_iter : public __Bst_iter_b<Node, Node_op>
00060 {
00061 private:
00062     typedef __Bst_iter_b<Node, Node_op> Base;
00063
00064     using Base::_n;
00065     using Base::_r;
00066     using Base::inc;
00067
00068 public:
00069     __Bst_iter() {}
00070
00071     __Bst_iter(Node const *t) : Base(t) {}
00072     __Bst_iter(Node const *t, Node const *r) : Base(t, r) {}
00073
00074     // template<typename Key2>
00075     __Bst_iter(Base const &o) : Base(o) {}
00076
00077     Node_type &operator * () const { return *const_cast<Node *>(_n); }
00078     Node_type *operator -> () const { return const_cast<Node *>(_n); }
00079     __Bst_iter &operator ++ () { inc(); return *this; }
00080     __Bst_iter &operator ++ (int)
00081     { __Bst_iter tmp = *this; inc(); return tmp; }
00082 };
00083
00084 //-----
00085 /* IMPLEMENTATION: __Bst_iter_b */

```

```

00134
00135 template< typename Node, typename Node_op>
00136 inline
00137 void __Bst_iter_b<Node, Node_op>::_downmost()
00138 {
00139     while (_n)
00140     {
00141         Node *n = Node_op::child(_n, Dir::L);
00142         if (n)
00143             _n = n;
00144         else
00145             return;
00146     }
00147 }
00148
00149 template< typename Node, typename Node_op>
00150 void __Bst_iter_b<Node, Node_op>::_inc()
00151 {
00152     if (!_n)
00153         return;
00154
00155     if (_n == _r)
00156     {
00157         _r = _n = Node_op::child(_r, Dir::R);
00158         _downmost();
00159         return;
00160     }
00161
00162     if (Node_op::child(_n, Dir::R))
00163     {
00164         _n = Node_op::child(_n, Dir::R);
00165         _downmost();
00166         return;
00167     }
00168
00169     Node const *q = _r;
00170     Node const *p = _r;
00171     while (1)
00172     {
00173         if (Node_op::cmp(_n, q))
00174         {
00175             p = q;
00176             q = Node_op::child(q, Dir::L);
00177         }
00178         else if (_n == q || Node_op::child(q, Dir::R) == _n)
00179         {
00180             _n = p;
00181             return;
00182         }
00183         else
00184             q = Node_op::child(q, Dir::R);
00185     }
00186 }
00187
00188 }

```

## 16.157 list\_basics.h

```

00001 /*
00002  * (c) 2011 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018
00019 #pragma once
00020
00021 namespace cxx { namespace Bits {
00022
00023     template< typename T >
00024     class List_iterator_end_ptr
00025     {

```

```

00026 private:
00027     template< typename U > friend class Basic_list;
00028     static void *_end;
00029 };
00030
00031 template< typename T >
00032 void *List_iterator_end_ptr<T>::_end;
00033
00034 template< typename VALUE_T, typename TYPE >
00035 struct Basic_list_policy
00036 {
00037     typedef VALUE_T *Value_type;
00038     typedef VALUE_T const *Const_value_type;
00039     typedef TYPE **Type;
00040     typedef TYPE *Const_type;
00041     typedef TYPE *Head_type;
00042     typedef TYPE Item_type;
00043
00044     static Type next(Type c) { return &(*c)->_n; }
00045     static Const_type next(Const_type c) { return c->_n; }
00046 };
00047
00048 template< typename POLICY >
00049 class Basic_list
00050 {
00051 {
00052     Basic_list(Basic_list const &) = delete;
00053     void operator = (Basic_list const &) = delete;
00054
00055 public:
00056     typedef typename POLICY::Value_type Value_type;
00057     typedef typename POLICY::Const_value_type Const_value_type;
00058
00059     class Iterator
00060     {
00061     private:
00062         typedef typename POLICY::Type Internal_type;
00063
00064     public:
00065         typedef typename POLICY::Value_type value_type;
00066         typedef typename POLICY::Value_type Value_type;
00067
00068         Value_type operator * () const { return static_cast<Value_type>(*_c); }
00069         Value_type operator -> () const { return static_cast<Value_type>(*_c); }
00070         Iterator operator ++ () { _c = POLICY::next(_c); return *this; }
00071
00072         bool operator == (Iterator const &o) const { return *_c == *o._c; }
00073         bool operator != (Iterator const &o) const { return !operator == (o); }
00074
00075         Iterator() : _c(__end()) {}
00076
00077     private:
00078         friend class Basic_list;
00079         static Internal_type __end()
00080         {
00081             union X { Internal_type l; void **v; } z;
00082             z.v = &Bits::List_iterator_end_ptr<void>::_end;
00083             return z.l;
00084         }
00085
00086         explicit Iterator(Internal_type i) : _c(i) {}
00087
00088         Internal_type _c;
00089     };
00090
00091     class Const_iterator
00092     {
00093     private:
00094         typedef typename POLICY::Const_type Internal_type;
00095
00096     public:
00097         typedef typename POLICY::Value_type value_type;
00098         typedef typename POLICY::Value_type Value_type;
00099
00100         Value_type operator * () const { return static_cast<Value_type>(_c); }
00101         Value_type operator -> () const { return static_cast<Value_type>(_c); }
00102         Const_iterator operator ++ () { _c = POLICY::next(_c); return *this; }
00103
00104         friend bool operator == (Const_iterator const &lhs, Const_iterator const &rhs)
00105         { return lhs._c == rhs._c; }
00106         friend bool operator != (Const_iterator const &lhs, Const_iterator const &rhs)
00107         { return lhs._c != rhs._c; }
00108
00109         Const_iterator() {}
00110         Const_iterator(Iterator const &o) : _c(*o) {}
00111
00112     private:
00113         friend class Basic_list;

```

```

00114
00115     explicit Const_iterator(Internal_type i) : _c(i) {}
00116
00117     Internal_type _c;
00118 };
00119
00120 // BSS allocation
00121 explicit Basic_list(bool) {}
00122 Basic_list() : _f(0) {}
00123
00124 Basic_list(Basic_list &&o) : _f(o._f)
00125 {
00126     o.clear();
00127 }
00128
00129 Basic_list &operator = (Basic_list &&o)
00130 {
00131     _f = o._f;
00132     o.clear();
00133     return *this;
00134 }
00135
00137 bool empty() const { return !_f; }
00139 Value_type front() const { return static_cast<Value_type>(_f); }
00140
00146 void clear() { _f = 0; }
00147
00149 Iterator begin() { return Iterator(&_f); }
00151 Const_iterator begin() const { return Const_iterator(_f); }
00159 static Const_iterator iter(Const_value_type c) { return Const_iterator(c); }
00161 Const_iterator end() const { return Const_iterator(nullptr); }
00163 Iterator end() { return Iterator(); }
00164
00165 protected:
00166     static typename POLICY::Type __get_internal(Iterator const &i) { return i._c; }
00167     static Iterator __iter(typename POLICY::Type c) { return Iterator(c); }
00168
00170     typename POLICY::Head_type _f;
00171 };
00172
00173 }}
00174

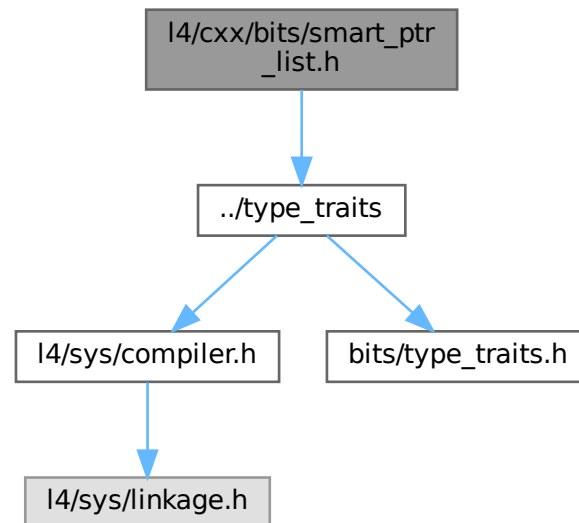
```

## 16.158 l4/cxx/bits/smart\_ptr\_list.h File Reference

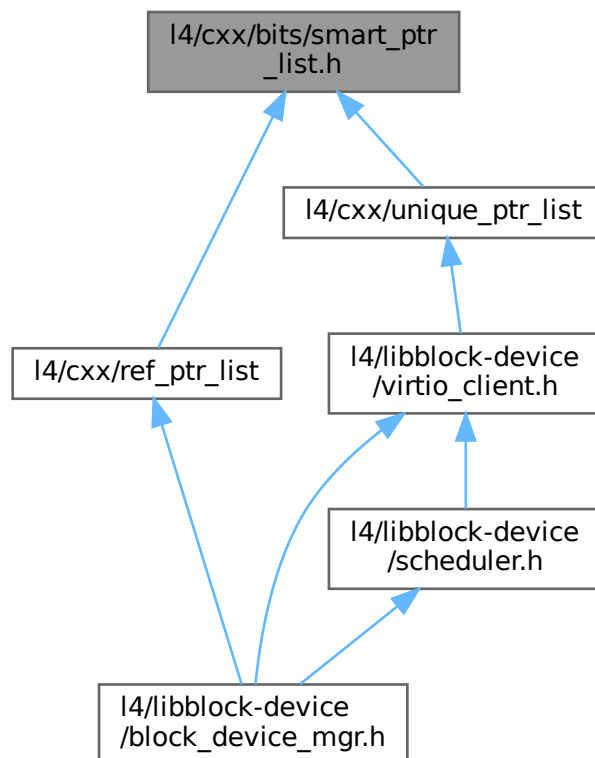
Implementation of a list of smart-pointer-managed objects.

```
#include "../type_traits"
```

Include dependency graph for smart\_ptr\_list.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [cxx::Bits::Smart\\_ptr\\_list\\_item< T, STORE\\_T >](#)  
*List item for an arbitrary item in a [Smart\\_ptr\\_list](#).*
- class [cxx::Bits::Smart\\_ptr\\_list< ITEM >](#)  
*List of smart-pointer-managed objects.*

## Namespaces

- namespace [cxx](#)  
*Our C++ library.*
- namespace [cxx::Bits](#)  
*Internal helpers for the cxx package.*

### 16.158.1 Detailed Description

Implementation of a list of smart-pointer-managed objects.

Definition in file [smart\\_ptr\\_list.h](#).

## 16.159 smart\_ptr\_list.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * Copyright (C) 2018, 2022 Kernkonzept GmbH.
00007  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00008  *
00009  * This file is distributed under the terms of the GNU General Public
00010  * License, version 2. Please see the COPYING-GPL-2 file for details.
00011  */
00012 #pragma once
00013
00014 #include "../type_traits"
00015
00016 namespace cxx { namespace Bits {
00017
00027 template <typename T, typename STORE_T>
00028 class Smart_ptr_list_item
00029 {
00030     using Value_type = T;
00031     using Storage_type = STORE_T;
00032
00033     template<typename U> friend class Smart_ptr_list;
00034     Storage_type _n;
00035 };
00036
00046 template <typename ITEM>
00047 class Smart_ptr_list
00048 {
00049     using Value_type = typename ITEM::Value_type;
00050     using Next_type = typename ITEM::Storage_type;
00051
00052 public:
00053     class Iterator
00054     {
00055     public:
00056         Iterator() : _c(nullptr) {}
00057
00058         Value_type *operator * () const { return _c; }
00059         Value_type *operator -> () const { return _c; }
00060
00061         Iterator operator ++ ()
00062         {
00063             _c = _c->_n.get();
00064             return *this;
00065         }
00066
00067         bool operator == (Iterator const &o) const { return _c == o._c; }
00068         bool operator != (Iterator const &o) const { return !operator == (o); }
00069
00070     private:
00071         friend class Smart_ptr_list;
00072
00073         explicit Iterator(Value_type *i) : _c(i) {}
00074
00075         Value_type *_c;
00076     };
00077
00078     class Const_iterator
00079     {
00080     public:
00081         Const_iterator() : _c(nullptr) {}
00082
00083         Value_type const *operator * () const { return _c; }
00084         Value_type const *operator -> () const { return _c; }
00085
00086         Const_iterator operator ++ ()
00087         {
00088             _c = _c->_n.get();
00089             return *this;
00090         }
00091
00092         bool operator == (Const_iterator const &o) const { return _c == o._c; }
00093         bool operator != (Const_iterator const &o) const { return !operator == (o); }
00094
00095     private:
00096         friend class Smart_ptr_list;
00097
00098         explicit Const_iterator(Value_type const *i) : _c(i) {}
00099
00100         Value_type const *_c;
00101     };
00102
00103     Smart_ptr_list() : _b(&_f) {}

```



```

00104
00106 void push_front(Next_type &&e)
00107 {
00108     e->_n = cxx::move(this->_f);
00109     this->_f = cxx::move(e);
00110
00111     if (_b == &_f)
00112         _b = &(_f->_n);
00113 }
00114
00116 void push_front(Next_type const &e)
00117 {
00118     e->_n = cxx::move(this->_f);
00119     this->_f = e;
00120
00121     if (_b == &_f)
00122         _b = &(_f->_n);
00123 }
00124
00126 void push_back(Next_type &&e)
00127 {
00128     *_b = cxx::move(e);
00129     _b = &((*_b)->_n);
00130 }
00131
00133 void push_back(Next_type const &e)
00134 {
00135     *_b = e;
00136     _b = &((*_b)->_n);
00137 }
00138
00140 Value_type *front() const
00141 { return _f.get(); }
00142
00150 Next_type pop_front()
00151 {
00152     Next_type ret = cxx::move(_f);
00153
00154     if (ret)
00155         _f = cxx::move(ret->_n);
00156
00157     if (!_f)
00158         _b = &_f;
00159
00160     return ret;
00161 }
00162
00164 bool empty() const
00165 { return !_f; }
00166
00167 Iterator begin() { return Iterator(_f.get()); }
00168 Iterator end() { return Iterator(); }
00169
00170 Const_iterator begin() const { return Const_iterator(_f.get()); }
00171 Const_iterator end() const { return Const_iterator(); }
00172
00173 Const_iterator cbegin() const { return const_iterator(_f.get()); }
00174 Const_iterator cend() const { return Const_iterator(); }
00175
00176 private:
00177     Next_type _f;
00178     Next_type *_b;
00179 };
00180
00181 } }

```

## 16.160 type\_traits.h

```

00001 // vi:ft=c++
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by

```

```

00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020
00021 #pragma once
00022
00023 namespace cxx {
00024
00025 class Null_type;
00026
00027 template< bool flag, typename T, typename F >
00028 class Select
00029 {
00030 public:
00031     typedef T Type;
00032 };
00033
00034 template< typename T, typename F >
00035 class Select< false, T, F >
00036 {
00037 public:
00038     typedef F Type;
00039 };
00040
00041
00042
00043 template< typename T, typename U >
00044 class Conversion
00045 {
00046     typedef char S;
00047     class B { char dummy[2]; };
00048     static S test(U);
00049     static B test(...);
00050     static T make_T();
00051 public:
00052     enum
00053     {
00054         exists = sizeof(test(make_T())) == sizeof(S),
00055         two_way = exists && Conversion<U,T>::exists,
00056         exists_2_way = two_way,
00057         same_type = false
00058     };
00059 };
00060
00061 template< >
00062 class Conversion<void, void>
00063 {
00064 public:
00065     enum { exists = 1, two_way = 1, exists_2_way = two_way, same_type = 1 };
00066 };
00067
00068 template< typename T >
00069 class Conversion<T, T>
00070 {
00071 public:
00072     enum { exists = 1, two_way = 1, exists_2_way = two_way, same_type = 1 };
00073 };
00074
00075 template< typename T >
00076 class Conversion<void, T>
00077 {
00078 public:
00079     enum { exists = 0, two_way = 0, exists_2_way = two_way, same_type = 0 };
00080 };
00081
00082 template< typename T >
00083 class Conversion<T, void>
00084 {
00085 public:
00086     enum { exists = 0, two_way = 0, exists_2_way = two_way, same_type = 0 };
00087 };
00088
00089 template< int I >
00090 class Int_to_type
00091 {
00092 public:
00093     enum { i = I };
00094 };
00095
00096 namespace TT
00097 {
00098     template< typename U > class Pointer_traits
00099     {
00100     public:
00101         typedef Null_type Pointee;
00102         enum { value = false };
00103     };
00104 }

```

```

00103     };
00104
00105     template< typename U > class Pointer_traits< U* >
00106     {
00107     public:
00108         typedef U Pointee;
00109         enum { value = true };
00110     };
00111
00112     template< typename U > struct Ref_traits
00113     {
00114         enum { value = false };
00115         typedef U Referee;
00116     };
00117
00118     template< typename U > struct Ref_traits<U&>
00119     {
00120         enum { value = true };
00121         typedef U Referee;
00122     };
00123
00124
00125     template< typename U > struct Add_ref { typedef U &Type; };
00126     template< typename U > struct Add_ref<U&> { typedef U Type; };
00127
00128     template< typename U > struct PMF_traits { enum { value = false }; };
00129     template< typename U, typename F > struct PMF_traits<U F::*>
00130     { enum { value = true }; };
00131
00132
00133     template< typename U > class Is_unsigned { public: enum { value = false }; };
00134     template<> class Is_unsigned<unsigned> { public: enum { value = true }; };
00135     template<> class Is_unsigned<unsigned char> {
00136     public: enum { value = true };
00137     };
00138     template<> class Is_unsigned<unsigned short> {
00139     public: enum { value = true };
00140     };
00141     template<> class Is_unsigned<unsigned long> {
00142     public: enum { value = true };
00143     };
00144     template<> class Is_unsigned<unsigned long long> {
00145     public: enum { value = true };
00146     };
00147
00148     template< typename U > class Is_signed { public: enum { value = false }; };
00149     template<> class Is_signed<signed char> { public: enum { value = true }; };
00150     template<> class Is_signed<signed short> { public: enum { value = true }; };
00151     template<> class Is_signed<signed> { public: enum { value = true }; };
00152     template<> class Is_signed<signed long> { public: enum { value = true }; };
00153     template<> class Is_signed<signed long long> {
00154     public: enum { value = true };
00155     };
00156
00157     template< typename U > class Is_int { public: enum { value = false }; };
00158     template<> class Is_int< char > { public: enum { value = true }; };
00159     template<> class Is_int< bool > { public: enum { value = true }; };
00160     template<> class Is_int< wchar_t > { public: enum { value = true }; };
00161
00162     template< typename U > class Is_float { public: enum { value = false }; };
00163     template<> class Is_float< float > { public: enum { value = true }; };
00164     template<> class Is_float< double > { public: enum { value = true }; };
00165     template<> class Is_float< long double > { public: enum { value = true }; };
00166
00167     template<typename T> class Const_traits
00168     {
00169     public:
00170         enum { value = false };
00171         typedef T Type;
00172         typedef const T Const_type;
00173     };
00174
00175     template<typename T> class Const_traits<const T>
00176     {
00177     public:
00178         enum { value = true };
00179         typedef T Type;
00180         typedef const T Const_type;
00181     };
00182 };
00183
00184 template< typename T >
00185 class Type_traits
00186 {
00187 public:
00188
00189     enum

```

```

00190 {
00191     is_unsigned = TT::Is_unsigned<T>::value,
00192     is_signed   = TT::Is_signed<T>::value,
00193     is_int      = TT::Is_int<T>::value,
00194     is_float    = TT::Is_float<T>::value,
00195     is_pointer  = TT::Pointer_traits<T>::value,
00196     is_pointer_to_member = TT::PMF_traits<T>::value,
00197     is_reference = TT::Ref_traits<T>::value,
00198     is_scalar = is_unsigned || is_signed || is_int || is_pointer
00199     || is_pointer_to_member || is_reference,
00200     is_fundamental = is_unsigned || is_signed || is_float
00201     || Conversion<T, void>::same_type,
00202     is_const      = TT::Const_traits<T>::value,
00203
00204     alignment =
00205     (sizeof(T) >= sizeof(unsigned long)
00206     ? sizeof(unsigned long)
00207     : (sizeof(T) >= sizeof(unsigned)
00208     ? sizeof(unsigned)
00209     : (sizeof(T) >= sizeof(short)
00210     ? sizeof(short)
00211     : 1)))
00212 };
00213
00214 typedef typename Select<is_scalar, T, typename TT::Add_ref<typename
TT::Const_traits<T>::Const_type>::Type>::Type Param_type;
00215 typedef typename TT::Pointer_traits<T>::Pointee Pointee_type;
00216 typedef typename TT::Ref_traits<T>::Referee Referee_type;
00217 typedef typename TT::Const_traits<T>::Type Non_const_type;
00218 typedef typename TT::Const_traits<T>::Const_type Const_type;
00219
00220 static unsigned long align(unsigned long a)
00221 {
00222     return (a + static_cast<unsigned long>(alignment) - 1UL)
00223     & ~(static_cast<unsigned long>(alignment) - 1UL);
00224 }
00225 };
00226
00227
00228 };
00229
00230
00231

```

## 16.161 dlist

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2011 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019
00020 #pragma once
00021
00022 namespace cxx {
00023
00024 class D_list_item
00025 {
00026 public:
00027     D_list_item() : _dli_next(0) {}
00028 private:
00029     friend class D_list_item_policy;
00030
00031     D_list_item(D_list_item const &);
00032     void operator = (D_list_item const &);
00033
00034     D_list_item *_dli_next, *_dli_prev;
00035 };
00036
00037 class D_list_item_policy

```

```

00038 {
00039 public:
00040     typedef D_list_item Item;
00041     static D_list_item *amprev(D_list_item *e) { return e->_dli_prev; }
00042     static D_list_item *ampnext(D_list_item *e) { return e->_dli_next; }
00043     static D_list_item *prev(D_list_item const *e) { return e->_dli_prev; }
00044     static D_list_item *next(D_list_item const *e) { return e->_dli_next; }
00045 };
00046
00047 template< typename T >
00048 struct Sd_list_head_policy
00049 {
00050     typedef T *Head_type;
00051     static T *head(Head_type h) { return h; }
00052     static void set_head(Head_type &h, T *v) { h = v; }
00053 };
00054
00055 template<
00056     typename T,
00057     typename C = D_list_item_policy
00058 >
00059 class D_list_cyclic
00060 {
00061 protected:
00062     template< typename VALUE, typename ITEM >
00063     class __Iterator
00064     {
00065     public:
00066         typedef VALUE *Value_type;
00067         typedef VALUE *value_type;
00068
00069         __Iterator() {}
00070
00071         bool operator == (__Iterator const &o) const
00072         { return _c == o._c; }
00073
00074         bool operator != (__Iterator const &o) const
00075         { return _c != o._c; }
00076
00077         __Iterator &operator ++ ()
00078         {
00079             _c = C::next(_c);
00080             return *this;
00081         }
00082
00083         __Iterator &operator -- ()
00084         {
00085             _c = C::prev(_c);
00086             return *this;
00087         }
00088
00089         Value_type operator * () const { return static_cast<Value_type>(_c); }
00090         Value_type operator -> () const { return static_cast<Value_type>(_c); }
00091
00092     private:
00093         friend class D_list_cyclic;
00094
00095         explicit __Iterator(ITEM *s) : _c(s) {}
00096
00097         ITEM *_c;
00098     };
00099
00100 public:
00101     typedef T *Value_type;
00102     typedef T *value_type;
00103     typedef __Iterator<T, typename C::Item> Iterator;
00104     typedef Iterator Const_iterator;
00105
00106     static void remove(T *e)
00107     {
00108         C::next(C::prev(e)) = C::next(e);
00109         C::prev(C::next(e)) = C::prev(e);
00110         C::next(e) = 0;
00111     }
00112
00113     static Iterator erase(Iterator const &e)
00114     {
00115         typename C::Item *n = C::next(*e);
00116         remove(*e);
00117         return __iter(n);
00118     }
00119
00120     static Iterator iter(T const *e) { return Iterator(const_cast<T*>(e)); }
00121
00122     static bool in_list(T const *e) { return C::next(const_cast<T*>(e)); }
00123     static bool has_sibling(T const *e) { return C::next(const_cast<T*>(e)) != e; }
00124

```

```

00125 static Iterator insert_after(T *e, Iterator const &pos)
00126 {
00127     C::prev(e) = *pos;
00128     C::next(e) = C::next(*pos);
00129     C::prev(C::next(*pos)) = e;
00130     C::next(*pos) = e;
00131     return pos;
00132 }
00133
00134 static Iterator insert_before(T *e, Iterator const &pos)
00135 {
00136     C::next(e) = *pos;
00137     C::prev(e) = C::prev(*pos);
00138     C::next(C::prev(*pos)) = e;
00139     C::prev(*pos) = e;
00140     return pos;
00141 }
00142
00143 static T *self_insert(T *e)
00144 { C::next(e) = C::prev(e) = e; return e; }
00145
00146 static void remove_last(T *e)
00147 { C::next(e) = 0; }
00148
00149 protected:
00150 static Iterator __iter(typename C::Item *e) { return Iterator(e); }
00151 };
00152
00153 template<
00154     typename T,
00155     typename C = D_list_item_policy,
00156     typename H = Sd_list_head_policy<T>,
00157     bool BSS = false
00158 >
00159 class Sd_list : public D_list_cyclic<T, C>
00160 {
00161 private:
00162     typedef D_list_cyclic<T, C> Base;
00163
00164 public:
00165     typedef typename Base::Iterator Iterator;
00166     enum Pos
00167     { Back, Front };
00168
00169     Sd_list() { if (!BSS) H::set_head(_f, 0); }
00170
00171     bool empty() const { return !H::head(_f); }
00172     T *front() const { return H::head(_f); }
00173
00174     void remove(T *e)
00175     {
00176         T *h = H::head(_f);
00177         if (e == C::next(e)) // must be the last
00178         {
00179             Base::remove_last(e);
00180             H::set_head(_f, 0);
00181             return;
00182         }
00183
00184         if (e == H::head(_f))
00185             H::set_head(_f, static_cast<T*>(C::next(h)));
00186
00187         Base::remove(e);
00188     }
00189
00190     Iterator erase(Iterator const &e)
00191     {
00192         typename C::Item *n = C::next(*e);
00193         remove(*e);
00194         return __iter(n);
00195     }
00196
00197     void push(T *e, Pos pos)
00198     {
00199         T *h = H::head(_f);
00200         if (!h)
00201             H::set_head(_f, Base::self_insert(e));
00202         else
00203         {
00204             Base::insert_before(e, this->iter(h));
00205             if (pos == Front)
00206                 H::set_head(_f, e);
00207         }
00208     }
00209
00210     void push_back(T *e) { push(e, Back); }
00211     void push_front(T *e) { push(e, Front); }

```

```

00212 void rotate_to(T *h) { H::set_head(_f, h); }
00213
00214 typename H::Head_type const &head() const { return _f; }
00215 typename H::Head_type &head() { return _f; }
00216
00217 private:
00218     Sd_list(Sd_list const &);
00219     void operator = (Sd_list const &);
00220
00221     typename H::Head_type _f;
00222 };
00223
00224 template<
00225     typename T,
00226     typename C = D_list_item_policy,
00227     bool BSS = false
00228 >
00229 class D_list : public D_list_cyclic<T, C>
00230 {
00231 private:
00232     typedef D_list_cyclic<T, C> Base;
00233     typedef typename C::Item Internal_type;
00234
00235 public:
00236     enum Pos
00237     { Back, Front };
00238
00239     typedef typename Base::Iterator Iterator;
00240     typedef typename Base::Const_iterator Const_iterator;
00241     typedef T* value_type;
00242     typedef T* Value_type;
00243
00244     D_list() { this->self_insert(static_cast<T*>(&_h)); }
00245
00246     bool empty() const { return C::next(static_cast<T const *>(&_h)) == &_h; }
00247
00248     static void remove(T *e) { Base::remove(e); }
00249     Iterator erase(Iterator const &e) { return Base::erase(e); }
00250
00251     void push(T *e, Pos pos)
00252     {
00253         if (pos == Front)
00254             Base::insert_after(e, end());
00255         else
00256             Base::insert_before(e, end());
00257     }
00258
00259     void push_back(T *e) { push(e, Back); }
00260     void push_front(T *e) { push(e, Front); }
00261
00262     Iterator begin() const { return this->__iter(C::next(const_cast<Internal_type *>(&_h))); }
00263     Iterator end() const { return this->__iter(const_cast<Internal_type *>(&_h)); }
00264
00265 private:
00266     D_list(D_list const &);
00267     void operator = (D_list const &);
00268
00269     Internal_type _h;
00270 };
00271
00272 }
00273

```

## 16.162 l4/cxx/exceptions File Reference

Base exceptions.

```

#include <l4/cxx/l4types.h>
#include <l4/cxx/basic_ostream>
#include <l4/sys/err.h>
#include <l4/sys/capability>
#include <l4/util/backtrace.h>

```





*Exception for an unknown condition.*

- class `L4::Element_not_found`

*Exception for a failed lookup (element not found).*

- class `L4::Invalid_capability`

*Indicates that an invalid object was invoked.*

- class `L4::Com_error`

*Error conditions during IPC.*

- class `L4::Bounds_error`

*Access out of bounds.*

## Namespaces

- namespace `L4`

*L4 low-level kernel interface.*

## Macros

- `#define L4_CXX_EXCEPTION_BACKTRACE 20`

*Number of instruction pointers in backtrace.*

### 16.162.1 Detailed Description

Base exceptions.

Definition in file [exceptions](#).

## 16.163 exceptions

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/cxx/l4types.h>
00028 #include <l4/cxx/basic_ostream>
00029 #include <l4/sys/err.h>
00030 #include <l4/sys/capability>
00031
00032
00039 #ifndef L4_CXX_NO_EXCEPTION_BACKTRACE
00040 # define L4_CXX_EXCEPTION_BACKTRACE 20
00041 #endif
00042
```

```

00043 #if defined(L4_CXX_EXCEPTION_BACKTRACE)
00044 #include <l4/util/backtrace.h>
00045 #endif
00046
00047 namespace L4
00048 {
00049     class Exception_tracer
00050     {
00051     private:
00052         void *_pc_array[L4_CXX_EXCEPTION_BACKTRACE];
00053         int _frame_cnt;
00054     protected:
00055     #if defined(__PIC__)
00056         Exception_tracer() noexcept : _frame_cnt(0) {}
00057     #else
00058         Exception_tracer() noexcept
00059         : _frame_cnt(l4util_backtrace(_pc_array, L4_CXX_EXCEPTION_BACKTRACE)) {}
00060     #endif
00061     public:
00062         void const *const *_pc_array() const noexcept { return _pc_array; }
00063         int frame_count() const noexcept { return _frame_cnt; }
00064     #else
00065     protected:
00066         Exception_tracer() noexcept {}
00067     public:
00068         void const *const *_pc_array() const noexcept { return 0; }
00069         int frame_count() const noexcept { return 0; }
00070     #endif
00071     };
00072
00073     class Base_exception : public Exception_tracer
00074     {
00075     protected:
00076         Base_exception() noexcept {}
00077     public:
00078         virtual char const *str() const noexcept = 0;
00079         virtual ~Base_exception() noexcept {}
00080     };
00081
00082     class Runtime_error : public Base_exception
00083     {
00084     private:
00085         long _errno;
00086         char _extra[80];
00087     public:
00088         explicit Runtime_error(long err_no, char const *extra = 0) noexcept
00089         : _errno(err_no)
00090         {
00091             if (!extra)
00092                 _extra[0] = 0;
00093             else
00094             {
00095                 unsigned i = 0;
00096                 for (; i < sizeof(_extra) && extra[i]; ++i)
00097                     _extra[i] = extra[i];
00098                 _extra[i < sizeof(_extra) ? i : sizeof(_extra) - 1] = 0;
00099             }
00100         }
00101         char const *str() const noexcept override
00102         { return l4sys_errtostr(_errno); }
00103         char const *extra_str() const { return _extra; }
00104         ~Runtime_error() noexcept {}
00105         long err_no() const noexcept { return _errno; }
00106     };
00107
00108     class Out_of_memory : public Runtime_error
00109     {
00110     public:
00111         explicit Out_of_memory(char const *extra = "") noexcept
00112         : Runtime_error(-L4_ENOMEM, extra) {}
00113         ~Out_of_memory() noexcept {}
00114     };
00115
00116     class Element_already_exists : public Runtime_error
00117     {
00118     public:
00119         explicit Element_already_exists(char const *e = "") noexcept

```

```

00207     : Runtime_error(-L4_EEXIST, e) {}
00208     ~Element_already_exists() noexcept {}
00209 };
00210
00219 class Unknown_error : public Base_exception
00220 {
00221 public:
00222     Unknown_error() noexcept {}
00223     char const *str() const noexcept override { return "unknown error"; }
00224     ~Unknown_error() noexcept {}
00225 };
00226
00231 class Element_not_found : public Runtime_error
00232 {
00233 public:
00234     explicit Element_not_found(char const *e = "") noexcept
00235         : Runtime_error(-L4_ENOENT, e) {}
00236 };
00237
00245 class Invalid_capability : public Base_exception
00246 {
00247 private:
00248     Cap<void> const _o;
00249
00250 public:
00255     explicit Invalid_capability(Cap<void> const &o) noexcept : _o(o) {}
00256     template< typename T>
00257     explicit Invalid_capability(Cap<T> const &o) noexcept : _o(o.cap()) {}
00258     char const *str() const noexcept override { return "invalid object"; }
00259
00264     Cap<void> const &cap() const noexcept { return _o; }
00265     ~Invalid_capability() noexcept {}
00266 };
00267
00274 class Com_error : public Runtime_error
00275 {
00276 public:
00281     explicit Com_error(long err) noexcept : Runtime_error(err) {}
00282
00283     ~Com_error() noexcept {}
00284 };
00285
00289 class Bounds_error : public Runtime_error
00290 {
00291 public:
00292     explicit Bounds_error(char const *e = "") noexcept
00293         : Runtime_error(-L4_ERANGE, e) {}
00294     ~Bounds_error() noexcept {}
00295 };
00297 };
00298
00299 inline
00300 L4::BasicOStream &
00301 operator << (L4::BasicOStream &o, L4::Base_exception const &e)
00302 {
00303     o << "Exception: " << e.str() << ", backtrace ...\n";
00304     for (int i = 0; i < e.frame_count(); ++i)
00305         o << L4::n_hex(l4_addr_t(e.pc_array()[i])) << '\n';
00306
00307     return o;
00308 }
00309
00310 inline
00311 L4::BasicOStream &
00312 operator << (L4::BasicOStream &o, L4::Runtime_error const &e)
00313 {
00314     o << "Exception: " << e.str() << ": ";
00315     if (e.extra_str())
00316         o << e.extra_str() << ": ";
00317     o << "backtrace ...\n";
00318     for (int i = 0; i < e.frame_count(); ++i)
00319         o << L4::n_hex(l4_addr_t(e.pc_array()[i])) << '\n';
00320
00321     return o;
00322 }

```

## 16.164 hlist

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2011 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *

```

```

00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYINGING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019
00020 #pragma once
00021
00022 #include "bits/list_basics.h"
00023 #include "type_traits"
00024
00025 namespace cxx {
00026
00032 template<typename ELEM_TYPE>
00033 class H_list_item_t
00034 {
00035 public:
00041   H_list_item_t() : _n(0), _pn(0) {}
00048   ~H_list_item_t() noexcept { l_remove(); }
00049
00050 private:
00051   H_list_item_t(H_list_item_t const &) = delete;
00052
00053   template<typename T, typename P> friend class H_list;
00054   template<typename T, typename X> friend struct Bits::Basic_list_policy;
00055
00056   void l_remove() noexcept
00057   {
00058       if (!_pn)
00059           return;
00060
00061       *_pn = _n;
00062       if (_n)
00063           _n->_pn = _pn;
00064
00065       _pn = 0;
00066   }
00067
00068   H_list_item_t *_n, **_pn;
00069 };
00070
00072 typedef H_list_item_t<void> H_list_item;
00073
00079 template< typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item> >
00080 class H_list : public Bits::Basic_list<POLICY>
00081 {
00082 private:
00083     typedef typename POLICY::Item_type Item;
00084     typedef Bits::Basic_list<POLICY> Base;
00085     H_list(H_list const &);
00086     void operator = (H_list const &);
00087
00088 public:
00089     typedef typename Base::Iterator Iterator;
00090
00091     // BSS allocation
00092     explicit H_list(bool x) : Base(x) {}
00093     H_list() : Base() {}
00094
00104     static Iterator iter(T *c) { return Base::__iter(c->Item::_pn); }
00105
00107     static bool in_list(T const *e) { return e->Item::_pn; }
00108
00110     void add(T *e)
00111     {
00112         if (this->_f)
00113             this->_f->_pn = &e->Item::_n;
00114         e->Item::_n = this->_f;
00115         e->Item::_pn = &this->_f;
00116         this->_f = static_cast<Item*>(e);
00117     }
00118
00120     void push_front(T *e) { add(e); }
00121
00127     T *pop_front()
00128     {
00129         T *r = this->front();
00130         remove(r);
00131         return r;

```

```

00132     }
00133
00144     Iterator insert(T *e, Iterator const &pred)
00145     {
00146         Item **x = &this->_f;
00147         if (pred != Base::end())
00148             x = &(*pred)->_n;
00149
00150         e->Item::_n = *x;
00151
00152         if (*x)
00153             (*x)->_pn = &(e->Item::_n);
00154
00155         e->Item::_pn = x;
00156         *x = static_cast<Item*>(e);
00157         return iter(e);
00158     }
00159
00171     static Iterator insert_after(T *e, Iterator const &pred)
00172     {
00173         Item **x = &(*pred)->_n;
00174         e->Item::_n = *x;
00175
00176         if (*x)
00177             (*x)->_pn = &(e->Item::_n);
00178
00179         e->Item::_pn = x;
00180         *x = static_cast<Item*>(e);
00181         return iter(e);
00182     }
00183
00191     static void insert_before(T *e, Iterator const &succ)
00192     {
00193         Item **x = Base::__get_internal(succ);
00194
00195         e->Item::_n = *x;
00196         e->Item::_pn = x;
00197
00198         if (*x)
00199             (*x)->_pn = &e->Item::_n;
00200
00201         *x = static_cast<Item*>(e);
00202     }
00203
00215     static void replace(T *p, T *e)
00216     {
00217         e->Item::_n = p->Item::_n;
00218         e->Item::_pn = p->Item::_pn;
00219         *(p->Item::_pn) = static_cast<Item*>(e);
00220         if (e->Item::_n)
00221             e->Item::_n->_pn = &(e->Item::_n);
00222
00223         p->Item::_pn = 0;
00224     }
00225
00231     static void remove(T *e)
00232     { e->Item::l_remove(); }
00233
00247     static Iterator erase(Iterator const &e)
00248     { e->Item::l_remove(); return e; }
00249 };
00250
00258 template< typename T >
00259 struct H_list_t : H_list<T, Bits::Basic_list_policy< T, H_list_item_t<T> > >
00260 {
00261     H_list_t() = default;
00262     H_list_t(bool b)
00263     : H_list<T, Bits::Basic_list_policy< T, H_list_item_t<T> > >(b)
00264     {};
00265 };
00266
00267 template< typename T >
00268 class H_list_bss : public H_list<T>
00269 {
00270 public:
00271     H_list_bss() : H_list<T>(true) {}
00272 };
00273
00274 template< typename T >
00275 class H_list_t_bss : public H_list_t<T>
00276 {
00277 public:
00278     H_list_t_bss() : H_list_t<T>(true) {}
00279 };
00280
00281
00282 }

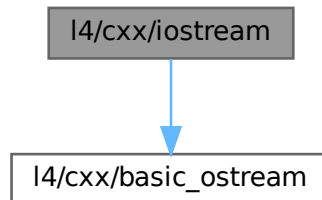
```

## 16.165 l4/cxx/iostream File Reference

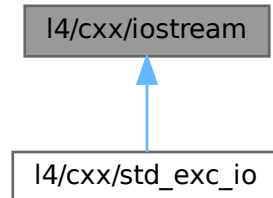
IO Stream.

```
#include <l4/cxx/basic_ostream>
```

Include dependency graph for iostream:



This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

### Variables

- BasicOStream **L4::cout**  
*Standard output stream.*
- BasicOStream **L4::cerr**  
*Standard error stream.*

## 16.165.1 Detailed Description

IO Stream.

Definition in file [iostream](#).

## 16.166 iostream

[Go to the documentation of this file.](#)

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/cxx/basic_ostream>
00028
00029 namespace L4 {
00030
00034     extern BasicOStream cout;
00035
00039     extern BasicOStream cerr;
00040
00041     extern void iostream_init();
00042
00043     static void __attribute__((used, constructor)) __iostream_init()
00044     { iostream_init(); }
00045 };
```

## 16.167 l4/cxx/ipc\_helper File Reference

IPC helper.

```
#include <l4/cxx/exceptions>
#include <l4/sys/utcb.h>
```





## 16.168 ipc\_helper

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/cxx/exceptions>
00027 #include <l4/sys/utcb.h>
00028
00033 namespace L4
00034 {
00035 #ifdef __EXCEPTIONS
00044     inline void
00045     throw_ipc_exception([[maybe_unused]] L4::Cap<void> const &o,
00046                        l4_msgtag_t const &err, l4_utcb_t *utcb)
00047     {
00048         if (err.has_error())
00049             throw (L4::Com_error(l4_error_u(err, utcb)));
00050     }
00051
00060     inline void
00061     throw_ipc_exception(void const *o, l4_msgtag_t const &err,
00062                        l4_utcb_t *utcb)
00063     { throw_ipc_exception(L4::Cap<void>(o), err, utcb); }
00064 #endif
00065
00066 }
```

## 16.169 l4/cxx/ipc\_server File Reference

IPC server loop.

```
#include <l4/sys/capability>
#include <l4/sys/typeinfo_svr>
#include <l4/sys/err.h>
#include <l4/cxx/ipc_stream>
#include <l4/sys/cxx/ipc_epiface>
#include <l4/sys/cxx/ipc_server_loop>
#include <l4/cxx/type_traits>
#include <l4/cxx/exceptions>
```



## 16.170 ipc\_server

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  * Alexander Warg <warg@os.inf.tu-dresden.de>,
00005  * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00006  * economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU General Public License 2.
00010  * Please see the COPYING-GPL-2 file for details.
00011  *
00012  * As a special exception, you may use this file as part of a free software
00013  * library without restriction. Specifically, if other files instantiate
00014  * templates or use macros or inline functions from this file, or you compile
00015  * this file and link it with other files to produce an executable, this
00016  * file does not by itself cause the resulting executable to be covered by
00017  * the GNU General Public License. This exception does not however
00018  * invalidate any other reasons why the executable file might be covered by
00019  * the GNU General Public License.
00020  */
00021 #pragma once
00022
00023 #include <l4/sys/capability>
00024 #include <l4/sys/typeinfo_svr>
00025 #include <l4/sys/err.h>
00026 #include <l4/cxx/ipc_stream>
00027 #include <l4/sys/cxx/ipc_epiface>
00028 #include <l4/sys/cxx/ipc_server_loop>
00029 #include <l4/cxx/type_traits>
00030 #include <l4/cxx/exceptions>
00031
00032 namespace L4 {
00033
00034 class Server_object : public Epiface
00035 {
00036 public:
00037     virtual int dispatch(unsigned long rights, Ipc::Iostream &ios) = 0;
00038
00039     l4_msgtag_t dispatch(l4_msgtag_t tag, unsigned rights, l4_utcb_t *utcb) override
00040     {
00041         L4::Ipc::Iostream ios(utcb);
00042         ios.tag() = tag;
00043         int r = dispatch(rights, ios);
00044         return ios.prepare_ipc(r);
00045     }
00046
00047     Cap<Kobject> obj_cap() const
00048     { return cap_cast<Kobject>(Epiface::obj_cap()); }
00049 };
00050
00051 template<typename IFACE, typename BASE = L4::Server_object>
00052 struct Server_object_t : BASE
00053 {
00054     typedef IFACE Interface;
00055
00056     typename BASE::Demand get_buffer_demand() const override
00057     { return typename L4::Kobject_typeid<IFACE>::Demand(); }
00058
00059     int dispatch_meta_request(L4::Ipc::Iostream &ios)
00060     { return L4::Util::handle_meta_request<IFACE>(ios); }
00061
00062     template<typename THIS>
00063     static int proto_dispatch(THIS *self, l4_umword_t rights, L4::Ipc::Iostream &ios)
00064     {
00065         l4_msgtag_t t;
00066         ios » t;
00067         return Kobject_typeid<IFACE>::proto_dispatch(self, t.label(), rights, ios);
00068     }
00069 };
00070
00071 template<typename Derived, typename IFACE, typename BASE = L4::Server_object>
00072 struct Server_object_x : Server_object_t<IFACE, BASE>
00073 {
00074     int dispatch(l4_umword_t r, L4::Ipc::Iostream &ios)
00075     {
00076         return Server_object_t<IFACE, BASE>::proto_dispatch(static_cast<Derived *>(this),
00077                                                             r, ios);
00078     }
00079 };
00080
00081 struct Irq_handler_object : Server_object_t<Kobject> {};
00082
00083 }

```

## 16.171 ipc\_server

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019 #pragma GCC system_header
00020
00021 #include <l4/sys/cxx/ipc_basics>
00022 #include <l4/sys/cxx/ipc_iface>
00023 #include <l4/sys/___typeinfo.h>
00024 #include <stddef.h>
00025
00026 namespace L4 {
00027 namespace Ipc {
00028 namespace Msg {
00029 namespace Detail {
00030
00031 template<typename T> struct Sizeof { enum { size = sizeof(T) }; };
00032 template<> struct Sizeof<void> { enum { size = 0 }; };
00033
00037 template<typename ...> struct Arg_pack
00038 {
00039     template<typename DIR>
00040     unsigned get(char *, unsigned offset, unsigned)
00041     { return offset; }
00042
00043     template<typename DIR>
00044     unsigned set(char *, unsigned offset, unsigned, long)
00045     { return offset; }
00046
00047     template<typename F, typename ...ARGS>
00048     long call(F f, ARGS ...args)
00049     { return f(args...); }
00050
00051     template<typename O, typename FUNC, typename ...ARGS>
00052     long obj_call(O *o, ARGS ...args)
00053     {
00054         typedef typename FUNC::template fwd<O> Fwd;
00055         return Fwd(o).template call<ARGS...>(args...);
00056         //return o->op_dispatch(args...);
00057     }
00058 };
00059
00063 template<typename T, typename SVR_TYPE, typename ...M>
00064 struct Svr_arg : Svr_xmit<T>, Arg_pack<M...>
00065 {
00066     typedef Arg_pack<M...> Base;
00067
00068     typedef SVR_TYPE svr_type;
00069     typedef typename _Elem<T>::svr_arg_type svr_arg_type;
00070
00071     svr_type v;
00072
00073     template<typename DIR>
00074     int get(char *msg, unsigned offset, unsigned limit)
00075     {
00076         typedef Svr_xmit<T> ct;
00077         int r = ct::to_svr(msg, offset, limit, this->v,
00078                             typename DIR::dir(), typename DIR::cls());
00079         if (L4_LIKELY(r >= 0))
00080             return Base::template get<DIR>(msg, r, limit);
00081
00082         if (_Elem<T>::Is_optional)
00083         {
00084             v = svr_type();
00085             return Base::template get<DIR>(msg, offset, limit);
00086         }
00087         return r;
00088     }
00089
00090     template<typename DIR>
00091     int set(char *msg, unsigned offset, unsigned limit, long ret)

```

```

00092 {
00093     typedef Svr_xmit<T> ct;
00094     int r = ct::from_svr(msg, offset, limit, ret, this->v,
00095                         typename DIR::dir(), typename DIR::cls());
00096     if (L4_UNLIKELY(r < 0))
00097         return r;
00098     return Base::template set<DIR>(msg, r, limit, ret);
00099 }
00100
00101 template<typename F, typename ...ARGS>
00102 long call(F f, ARGS ...args)
00103 {
00104     //As_arg<value_type> check;
00105     return Base::template
00106         call<F, ARGS..., svr_arg_type>(f, args..., this->v);
00107 }
00108
00109 template<typename O, typename FUNC, typename ...ARGS>
00110 long obj_call(O *o, ARGS ...args)
00111 {
00112     //As_arg<value_type> check;
00113     return Base::template
00114         obj_call<O, FUNC, ARGS..., svr_arg_type>(o, args..., this->v);
00115 }
00116 };
00117
00118 template<typename T, typename ...M>
00119 struct Svr_arg<T, void, M...> : Arg_pack<M...>
00120 {
00121     typedef Arg_pack<M...> Base;
00122
00123     template<typename DIR>
00124     int get(char *msg, unsigned offset, unsigned limit)
00125     { return Base::template get<DIR>(msg, offset, limit); }
00126
00127     template<typename DIR>
00128     int set(char *msg, unsigned offset, unsigned limit, long ret)
00129     { return Base::template set<DIR>(msg, offset, limit, ret); }
00130
00131     template<typename F, typename ...ARGS>
00132     long call(F f, ARGS ...args)
00133     {
00134         return Base::template call<F, ARGS...>(f, args...);
00135     }
00136
00137     template<typename O, typename FUNC, typename ...ARGS>
00138     long obj_call(O *o, ARGS ...args)
00139     {
00140         return Base::template obj_call<O, FUNC, ARGS...>(o, args...);
00141     }
00142 };
00143
00144 template<typename A, typename ...M>
00145 struct Arg_pack<A, M...> : Svr_arg<A, typename _Elem<A>::svr_type, M...>
00146 {};
00147
00148 } // namespace Detail
00149
00150 //-----
00151 template<typename IPC_TYPE> struct Svr_arg_pack;
00152
00153 template<typename R, typename ...ARGS>
00154 struct Svr_arg_pack<R (ARGS...)> : Detail::Arg_pack<ARGS...>
00155 {
00156     typedef Detail::Arg_pack<ARGS...> Base;
00157     template<typename DIR>
00158     int get(void *msg, unsigned offset, unsigned limit)
00159     {
00160         char *buf = static_cast<char *>(msg);
00161         return Base::template get<DIR>(buf, offset, limit);
00162     }
00163
00164     template<typename DIR>
00165     int set(void *msg, unsigned offset, unsigned limit, long ret)
00166     {
00167         char *buf = static_cast<char *>(msg);
00168         return Base::template set<DIR>(buf, offset, limit, ret);
00169     }
00170 };
00171
00172 template<typename IPC_TYPE, typename O, typename ...ARGS>
00173 static l4_msgtag_t
00174 handle_svr_obj_call(O *o, l4_utcb_t *utcb, l4_msgtag_t tag, ARGS ...args)
00175 {
00176     typedef Svr_arg_pack<typename IPC_TYPE::rpc::ipc_type> Pack;
00177     enum
00178     {

```

```

00186     Do_reply = IPC_TYPE::rpc::flags_type::Is_call,
00187     Short_err = Do_reply ? -L4_MSGTOOSHORT : -L4_ENOREPLY,
00188 };
00189
00190 // XXX: send a reply or just do not reply in case of a cheating client
00191 if (L4_UNLIKELY(tag.words() + tag.items() * Item_words > Mr_words))
00192     return l4_msgtag(Short_err, 0, 0, 0);
00193
00194 // our whole arguments data structure
00195 Pack pack;
00196 l4_msgregs_t *mrs = l4_utcb_mr_u(utcb);
00197
00198 int in_pos = Detail::Sizeof<typename IPC_TYPE::opcode_type>::size;
00199
00200 unsigned const in_bytes = tag.words() * Word_bytes;
00201
00202 in_pos = pack.template get<Do_in_data>(&mrs->mr[0], in_pos, in_bytes);
00203
00204 if (L4_UNLIKELY(in_pos < 0))
00205     return l4_msgtag(Short_err, 0, 0, 0);
00206
00207 if (L4_UNLIKELY(pack.template get<Do_out_data>(mrs->mr, 0, Mr_bytes) < 0))
00208     return l4_msgtag(Short_err, 0, 0, 0);
00209
00210
00211 in_pos = pack.template get<Do_in_items>(&mrs->mr[tag.words()], 0,
00212                                         tag.items() * Item_bytes);
00213
00214 if (L4_UNLIKELY(in_pos < 0))
00215     return l4_msgtag(Short_err, 0, 0, 0);
00216
00217 asm volatile (" : "=m" (mrs->mr));
00218
00219 // call the server function
00220 long ret = pack.template obj_call<0, typename IPC_TYPE::rpc, ARGS...>(o, args...);
00221
00222 if (!Do_reply)
00223     return l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00224
00225 // our convention says that negative return value means no
00226 // reply data
00227 if (L4_UNLIKELY(ret < 0))
00228     return l4_msgtag(ret, 0, 0, 0);
00229
00230 // reply with the reply data from the server function
00231 int bytes = pack.template set<Do_out_data>(mrs->mr, 0, Mr_bytes, ret);
00232 if (L4_UNLIKELY(bytes < 0))
00233     return l4_msgtag(-L4_MSGTOOLONG, 0, 0, 0);
00234
00235 unsigned words = (bytes + Word_bytes - 1) / Word_bytes;
00236 bytes = pack.template set<Do_out_items>(&mrs->mr[words], 0,
00237                                         Mr_bytes - words * Word_bytes,
00238                                         ret);
00239 if (L4_UNLIKELY(bytes < 0))
00240     return l4_msgtag(-L4_MSGTOOLONG, 0, 0, 0);
00241
00242 unsigned const items = bytes / Item_bytes;
00243 return l4_msgtag(ret, words, items, 0);
00244 }
00245
00246 //-----
00247
00248 template<typename RPCS, typename OPCODE_TYPE>
00249 struct Dispatch_call;
00250
00251 template<typename CLASS>
00252 struct Dispatch_call<L4::Typeid::Raw_ipc<CLASS>, void>
00253 {
00254     template<typename OBJ, typename ...ARGS>
00255     static l4_msgtag_t
00256     call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, ARGS ...a)
00257     {
00258         return o->op_dispatch(utcb, tag, a...);
00259     }
00260 };
00261
00262 template<typename RPCS>
00263 struct Dispatch_call<RPCS, void>
00264 {
00265     constexpr static unsigned rmask()
00266     { return RPCS::rpc::flags_type::Rights & 3UL; }
00267
00268     template<typename OBJ, typename ...ARGS>
00269     static l4_msgtag_t
00270     call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, unsigned rights, ARGS ...a)
00271     {
00272         if ((rights & rmask()) != rmask())

```

```

00273         return l4_msgtag(-L4_EPERM, 0, 0, 0);
00274
00275     typedef L4::Typeid::Rights<typename RPCS::rpc::class_type> Rights;
00276     return handle_svr_obj_call<RPCS>(o, utcb, tag,
00277                                     Rights(rights), a...);
00278
00279 }
00280 };
00281
00282 template<typename RPCS, typename OPCODE_TYPE>
00283 struct Dispatch_call
00284 {
00285     constexpr static unsigned rmask()
00286     { return RPCS::rpc::flags_type::Rights & 3UL; }
00287
00288     template<typename OBJ, typename ...ARGS>
00289     static l4_msgtag_t
00290     _call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, unsigned rights, OPCODE_TYPE op, ARGS ...a)
00291     {
00292         if (L4::Types::Same<typename RPCS::opcode_type, void>::value
00293             || RPCS::Opcode == op)
00294         {
00295             if ((rights & rmask()) != rmask())
00296                 return l4_msgtag(-L4_EPERM, 0, 0, 0);
00297
00298             typedef L4::Typeid::Rights<typename RPCS::rpc::class_type> Rights;
00299             return handle_svr_obj_call<RPCS>(o, utcb, tag,
00300                                             Rights(rights), a...);
00301         }
00302         return Dispatch_call<typename RPCS::next, OPCODE_TYPE>::template
00303             _call<OBJ, ARGS...>(o, utcb, tag, rights, op, a...);
00304     }
00305
00306     template<typename OBJ, typename ...ARGS>
00307     static l4_msgtag_t
00308     call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, unsigned rights, ARGS ...a)
00309     {
00310         OPCODE_TYPE op;
00311         unsigned limit = tag.words() * Word_bytes;
00312         typedef Svr_xmit<OPCODE_TYPE> S;
00313         int err = S::to_svr(reinterpret_cast<char *>(l4_utcb_mr_u(utcb)->mr), 0,
00314                             limit, op, Dir_in(), Cls_data());
00315         if (L4_UNLIKELY(err < 0))
00316             return l4_msgtag(-L4_EMMSGTOOSHORT, 0, 0, 0);
00317
00318         return _call<OBJ, ARGS...>(o, utcb, tag, rights, op, a...);
00319     }
00320 };
00321
00322 template<>
00323 struct Dispatch_call<Typeid::Detail::Rpc_end, void>
00324 {
00325     template<typename OBJ, typename ...ARGS>
00326     static l4_msgtag_t
00327     _call(OBJ *, l4_utcb_t *, l4_msgtag_t, unsigned, int, ARGS ...)
00328     { return l4_msgtag(-L4_ENOSYS, 0, 0, 0); }
00329
00330     template<typename OBJ, typename ...ARGS>
00331     static l4_msgtag_t
00332     call(OBJ *, l4_utcb_t *, l4_msgtag_t, unsigned, ARGS ...)
00333     { return l4_msgtag(-L4_ENOSYS, 0, 0, 0); }
00334 };
00335
00336 template<typename OPCODE_TYPE>
00337 struct Dispatch_call<Typeid::Detail::Rpc_end, OPCODE_TYPE> :
00338     Dispatch_call<Typeid::Detail::Rpc_end, void> {};
00339
00340 template<typename RPCS, typename OBJ, typename ...ARGS>
00341 static l4_msgtag_t
00342 dispatch_call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, unsigned rights, ARGS ...a)
00343 {
00344     return Dispatch_call<typename RPCS::type, typename RPCS::opcode_type>::template
00345         call<OBJ, ARGS...>(o, utcb, tag, rights, a...);
00346 }
00347
00348 } // namespace Msg
00349 } // namespace Ipc
00350 } // namespace L4

```

## 16.172 l4/cxx/ipc\_stream File Reference

IPC stream.





- Input stream for IPC unmarshalling.*
- class [L4::lpc::Ostream](#)  
*Output stream for IPC marshalling.*
- class [L4::lpc::Istream](#)  
*Input/Output stream for IPC [un]marshalling.*

## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*
- namespace [L4::lpc](#)  
*IPC related functionality.*

## Functions

- template<typename T >  
[Internal::Buf\\_cp\\_out](#)< T > [L4::lpc::buf\\_cp\\_out](#) (T const \*v, unsigned long size)  
*Insert an array into an [lpc::Ostream](#).*
- template<typename T >  
[Internal::Buf\\_cp\\_in](#)< T > [L4::lpc::buf\\_cp\\_in](#) (T \*v, unsigned long &size)  
*Extract an array from an [lpc::Istream](#).*
- template<typename T >  
[Str\\_cp\\_in](#)< T > [L4::lpc::str\\_cp\\_in](#) (T \*v, unsigned long &size)  
*Create a [Str\\_cp\\_in](#) for the given values.*
- template<typename T >  
[Msg\\_ptr](#)< T > [L4::lpc::msg\\_ptr](#) (T \*&p)  
*Create an [Msg\\_ptr](#) to adjust the given pointer.*
- template<typename T >  
[Internal::Buf\\_in](#)< T > [L4::lpc::buf\\_in](#) (T \*&v, unsigned long &size)  
*Return a pointer to stream array data.*
- [L4::lpc::Istream & operator>>](#) ([L4::lpc::Istream](#) &s, bool &v)  
*Extract one element of type T from the stream s.*
- [L4::lpc::Istream & operator>>](#) ([L4::lpc::Istream](#) &s, [l4\\_msgtag\\_t](#) &v)  
*Extract the L4 message tag from the stream s.*
- template<typename T >  
[L4::lpc::Istream & operator>>](#) ([L4::lpc::Istream](#) &s, [L4::lpc::Internal::Buf\\_in](#)< T > const &v)  
*Extract an array of T elements from the stream s.*
- template<typename T >  
[L4::lpc::Istream & operator>>](#) ([L4::lpc::Istream](#) &s, [L4::lpc::Msg\\_ptr](#)< T > const &v)  
*Extract an element of type T from the stream s.*
- template<typename T >  
[L4::lpc::Istream & operator>>](#) ([L4::lpc::Istream](#) &s, [L4::lpc::Internal::Buf\\_cp\\_in](#)< T > const &v)  
*Extract an array of T elements from the stream s.*
- template<typename T >  
[L4::lpc::Istream & operator>>](#) ([L4::lpc::Istream](#) &s, [L4::lpc::Str\\_cp\\_in](#)< T > const &v)  
*Extract a zero-terminated string from the stream.*
- [L4::lpc::Ostream & operator<<](#) ([L4::lpc::Ostream](#) &s, bool v)  
*Insert an element to type T into the stream s.*
- [L4::lpc::Ostream & operator<<](#) ([L4::lpc::Ostream](#) &s, [l4\\_msgtag\\_t](#) const &v)  
*Insert the L4 message tag into the stream s.*

- `template<typename T>`  
`L4::lpc::Ostream & operator<< (L4::lpc::Ostream &s, L4::lpc::Internal::Buf_cp_out< T > const &v)`  
*Insert an array with elements of type `T` into the stream `s`.*
- `L4::lpc::Ostream & operator<< (L4::lpc::Ostream &s, char const *v)`  
*Insert a zero terminated character string into the stream `s`.*
- `template<typename T>`  
`T L4::lpc::read (lstream &s)`  
*Read a value out of a stream.*

### 16.172.1 Detailed Description

IPC stream.

Definition in file [ipc\\_stream](#).

### 16.172.2 Function Documentation

#### 16.172.2.1 `operator<<()` [1/4]

```
L4::lpc::Ostream & operator<< (
    L4::lpc::Ostream & s,
    bool v ) [inline]
```

Insert an element to type `T` into the stream `s`.

##### Parameters

<code>s</code>	The stream to insert the element <code>v</code> .
<code>v</code>	The element to insert.

##### Returns

The stream `s`.

Definition at line 1208 of file [ipc\\_stream](#).

References [L4::lpc::Ostream::put\(\)](#).

Here is the call graph for this function:



**16.172.2.2 operator<<() [2/4]**

```
L4::Ipc::Ostream & operator<< (
    L4::Ipc::Ostream & s,
    char const * v ) [inline]
```

Insert a zero terminated character string into the stream *s*.

**Parameters**

<i>s</i>	The stream to insert the string <i>v</i> .
<i>v</i>	The string to insert.

**Returns**

The stream *s*.

This operator produces basically the same content as the array insertion, however the length of the array is calculated using `strlen(v) + 1`. The string is copied into the message including the trailing zero.

Definition at line 1280 of file `ipc_stream`.

References `L4::Ipc::Ostream::put()`.

Here is the call graph for this function:

**16.172.2.3 operator<<() [3/4]**

```
template<typename T >
L4::Ipc::Ostream & operator<< (
    L4::Ipc::Ostream & s,
    L4::Ipc::Internal::Buf_cp_out< T > const & v ) [inline]
```

Insert an array with elements of type *T* into the stream *s*.

**Parameters**

<i>s</i>	The stream to insert the array <i>v</i> .
<i>v</i>	The array to insert (see <code>Ipc::Buf_cp_out()</code> ).

**Returns**

The stream *s*.

Definition at line 1259 of file [ipc\\_stream](#).

References [L4::Ipc::Ostream::put\(\)](#).

Here is the call graph for this function:

**16.172.2.4 operator<<() [4/4]**

```

L4::Ipc::Ostream & operator<< (
    L4::Ipc::Ostream & s,
    l4_msgtag_t const & v ) [inline]
  
```

Insert the [L4](#) message tag into the stream *s*.

**Parameters**

<i>s</i>	The stream to insert the tag <i>v</i> .
<i>v</i>	The <a href="#">L4</a> message tag to insert.

**Returns**

The stream *s*.

**Note**

Only one message tag can be inserted into a stream. Multiple insertions simply overwrite previous insertions.

Definition at line 1243 of file [ipc\\_stream](#).

References [L4::Ipc::Ostream::tag\(\)](#).

Here is the call graph for this function:



**16.172.2.5 operator>>() [1/6]**

```
L4::Ipc::Istream & operator>> (
    L4::Ipc::Istream & s,
    bool & v ) [inline]
```

Extract one element of type T from the stream s.

**Parameters**

	<i>s</i>	The stream to extract from.
out	<i>v</i>	Extracted value.

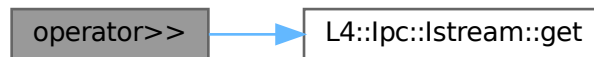
**Returns**

The stream s.

Definition at line 1055 of file [ipc\\_stream](#).

References [L4::Ipc::Istream::get\(\)](#).

Here is the call graph for this function:

**16.172.2.6 operator>>>() [2/6]**

```
template<typename T >
L4::Ipc::Istream & operator>>> (
    L4::Ipc::Istream & s,
    L4::Ipc::Internal::Buf_cp_in< T > const & v ) [inline]
```

Extract an array of T elements from the stream s.

**Parameters**

	<i>s</i>	The stream to extract from.
out	<i>v</i>	Buffer description to copy the array to (Ipc::Buf_cp_out()).

**Returns**

The stream s.

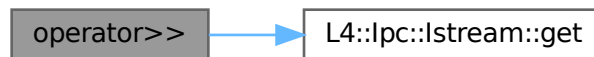
This operator does a copy out of the data into the given buffer.

See `lpc::Buf_in`, `lpc::Buf_cp_in`, and `lpc::Buf_cp_out`.

Definition at line 1162 of file `ipc_stream`.

References `L4::lpc::lstream::get()`.

Here is the call graph for this function:



### 16.172.2.7 operator>>() [3/6]

```

template<typename T >
L4::Ipc::Istream & operator>> (
    L4::Ipc::Istream & s,
    L4::Ipc::Internal::Buf_in< T > const & v ) [inline]
  
```

Extract an array of T elements from the stream `s`.

#### Parameters

	<code>s</code>	The stream to extract from.
<code>out</code>	<code>v</code>	Pointer to the extracted array ( <code>ipc_buf_in()</code> ).

#### Returns

The stream `s`.

This operator actually does not copy out the data in the array, but returns a pointer into the message buffer itself. This means that the data is only valid as long as there is no new data inserted into the stream.

#### Note

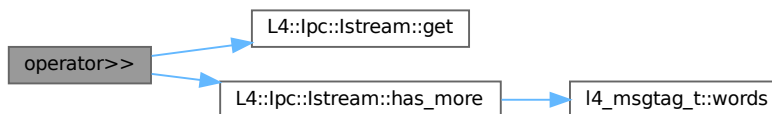
If array does not fit into transmitted words size will be set to zero. Client has to implement check against zero.

See `lpc::Buf_in`, `lpc::Buf_cp_in`, and `lpc::Buf_cp_out`.

Definition at line 1114 of file `ipc_stream`.

References `L4::lpc::lstream::get()`, and `L4::lpc::lstream::has_more()`.

Here is the call graph for this function:



### 16.172.2.8 operator>>() [4/6]

```

template<typename T >
L4::Ipc::Istream & operator>> (
    L4::Ipc::Istream & s,
    L4::Ipc::Msg_ptr< T > const & v ) [inline]
  
```

Extract an element of type T from the stream s.

#### Parameters

	s	The stream to extract from.
out	v	Pointer to the extracted element.

#### Returns

The stream s.

This operator actually does not copy out the data, but returns a pointer into the message buffer itself. This means that the data is only valid as long as there is no new data inserted into the stream.

See `Msg_ptr`.

Definition at line 1141 of file `ipc_stream`.

References [L4::ipc::Istream::get\(\)](#).

Here is the call graph for this function:



**16.172.2.9 operator>>() [5/6]**

```
template<typename T >
L4::Ipc::Istream & operator>> (
    L4::Ipc::Istream & s,
    L4::Ipc::Str_cp_in< T > const & v ) [inline]
```

Extract a zero-terminated string from the stream.

**Parameters**

	<i>s</i>	The stream to extract from.
out	<i>v</i>	Buffer description to copy the array to (lpc::Str_cp_out()).

**Returns**

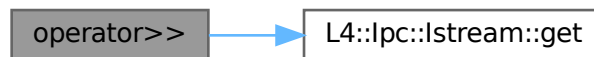
The stream *s*.

This operator does a copy out of the data into the given buffer.

Definition at line 1183 of file [ipc\\_stream](#).

References [L4::lpc::Istream::get\(\)](#).

Here is the call graph for this function:

**16.172.2.10 operator>>() [6/6]**

```
L4::Ipc::Istream & operator>> (
    L4::Ipc::Istream & s,
    l4_msgtag_t & v ) [inline]
```

Extract the [L4](#) message tag from the stream *s*.

**Parameters**

	<i>s</i>	The stream to extract from.
out	<i>v</i>	The extracted tag.



## Returns

The stream `s`.

Definition at line 1089 of file `ipc_stream`.

References `L4::ipc::Istream::tag()`.

Here is the call graph for this function:



## 16.173 ipc\_stream

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/sys/ipc.h>
00028 #include <l4/sys/capability>
00029 #include <l4/sys/cxx/ipc_types>
00030 #include <l4/sys/cxx/ipc_varg>
00031 #include <l4/cxx/type_traits>
00032 #include <l4/cxx/minmax>
00033
00034 namespace L4 {
00035 namespace Ipc {
00036
00037 class Ostream;
00038 class Istream;
00039
00040 namespace Internal {
00058 template< typename T >
00059 class Buf_cp_out
00060 {
00061 public:
00068   Buf_cp_out(T const *v, unsigned long size) : _v(v), _s(size) {}
00069
00077   unsigned long size() const { return _s; }
00078
00086   T const *buf() const { return _v; }
00087
00088 private:
00089   friend class Ostream;
00090   T const *_v;
  
```

```

00091     unsigned long _s;
00092 };
00093 }
00094
00110 template< typename T >
00111 Internal::Buf_cp_out<T> buf_cp_out(T const *v, unsigned long size)
00112 { return Internal::Buf_cp_out<T>(v, size); }
00113
00114
00115 namespace Internal {
00128 template< typename T >
00129 class Buf_cp_in
00130 {
00131 public:
00140     Buf_cp_in(T *v, unsigned long &size) : _v(v), _s(&size) {}
00141
00142     unsigned long &size() const { return *_s; }
00143     T *buf() const { return _v; }
00144
00145 private:
00146     friend class Istream;
00147     T *_v;
00148     unsigned long *_s;
00149 };
00150 }
00151
00169 template< typename T >
00170 Internal::Buf_cp_in<T> buf_cp_in(T *v, unsigned long &size)
00171 { return Internal::Buf_cp_in<T>(v, size); }
00172
00188 template< typename T >
00189 class Str_cp_in
00190 {
00191 public:
00200     Str_cp_in(T *v, unsigned long &size) : _v(v), _s(&size) {}
00201
00202     unsigned long &size() const { return *_s; }
00203     T *buf() const { return _v; }
00204
00205 private:
00206     friend class Istream;
00207     T *_v;
00208     unsigned long *_s;
00209 };
00210
00223 template< typename T >
00224 Str_cp_in<T> str_cp_in(T *v, unsigned long &size)
00225 { return Str_cp_in<T>(v, size); }
00226
00239 template< typename T >
00240 class Msg_ptr
00241 {
00242 private:
00243     T **_p;
00244 public:
00251     explicit Msg_ptr(T *&p) : _p(&p) {}
00252     void set(T *p) const { *_p = p; }
00253 };
00254
00262 template< typename T >
00263 Msg_ptr<T> msg_ptr(T *&p)
00264 { return Msg_ptr<T>(p); }
00265
00266
00267 namespace Internal {
00281 template< typename T >
00282 class Buf_in
00283 {
00284 public:
00291     Buf_in(T *&v, unsigned long &size) : _v(&v), _s(&size) {}
00292
00293     void set_size(unsigned long s) const { *_s = s; }
00294     T *&buf() const { return *_v; }
00295
00296 private:
00297     friend class Istream;
00298     T **_v;
00299     unsigned long *_s;
00300 };
00301 }
00302
00320 template< typename T >
00321 Internal::Buf_in<T> buf_in(T *&v, unsigned long &size)
00322 { return Internal::Buf_in<T>(v, size); }
00323
00324 namespace Utc_stream_check
00325 {

```

```

00326     static bool check_utcb_data_offset(unsigned sz)
00327     { return sz > sizeof(l4_umword_t) * L4_UTCB_GENERIC_DATA_SIZE; }
00328 }
00329
00330
00345 class Istream
00346 {
00347 public:
00359     Istream(l4_utcb_t *utcb)
00360     : _tag(), _utcb(utcb),
00361       _current_msg(reinterpret_cast<char*>(l4_utcb_mr_u(utcb)->mr)),
00362       _pos(0), _current_buf(0)
00363     {}
00364
00369     void reset()
00370     {
00371         _pos = 0;
00372         _current_buf = 0;
00373         _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00374     }
00375
00379     template< typename T >
00380     bool has_more(unsigned long count = 1)
00381     {
00382         auto const max_bytes = L4_UTCB_GENERIC_DATA_SIZE * sizeof(l4_umword_t);
00383         unsigned apos = cxx::Type_traits<T>::align(_pos);
00384         return (count <= max_bytes / sizeof(T))
00385             && (apos + (sizeof(T) * count)
00386                 <= _tag.words() * sizeof(l4_umword_t));
00387     }
00388
00393
00404     template< typename T >
00405     unsigned long get(T *buf, unsigned long elems)
00406     {
00407         if (L4_UNLIKELY(!has_more<T>(elems)))
00408             return 0;
00409
00410         unsigned long size = elems * sizeof(T);
00411         _pos = cxx::Type_traits<T>::align(_pos);
00412
00413         __builtin_memcpy(buf, _current_msg + _pos, size);
00414         _pos += size;
00415         return elems;
00416     }
00417
00418
00424     template< typename T >
00425     void skip(unsigned long elems)
00426     {
00427         if (L4_UNLIKELY(!has_more<T>(elems)))
00428             return;
00429
00430         unsigned long size = elems * sizeof(T);
00431         _pos = cxx::Type_traits<T>::align(_pos);
00432         _pos += size;
00433     }
00434
00449     template< typename T >
00450     unsigned long get(Msg_ptr<T> const &buf, unsigned long elems = 1)
00451     {
00452         if (L4_UNLIKELY(!has_more<T>(elems)))
00453             return 0;
00454
00455         unsigned long size = elems * sizeof(T);
00456         _pos = cxx::Type_traits<T>::align(_pos);
00457
00458         buf.set(reinterpret_cast<T*>(_current_msg + _pos));
00459         _pos += size;
00460         return elems;
00461     }
00462
00463
00474     template< typename T >
00475     bool get(T &v)
00476     {
00477         if (L4_UNLIKELY(!has_more<T>()))
00478         {
00479             v = T();
00480             return false;
00481         }
00482
00483         _pos = cxx::Type_traits<T>::align(_pos);
00484         v = *(reinterpret_cast<T*>(_current_msg + _pos));
00485         _pos += sizeof(T);
00486         return true;
00487     }

```

```

00488
00489
00490 bool get(Ipc::Varg *va)
00491 {
00492     Ipc::Varg::Tag t;
00493     if (!has_more<Ipc::Varg::Tag>())
00494     {
00495         va->tag(0);
00496         return 0;
00497     }
00498     get(t);
00499     va->tag(t);
00500     char const *d;
00501     get(msg_ptr(d), va->length());
00502     va->data(d);
00503
00504     return 1;
00505 }
00506
00516 l4_msgtag_t tag() const { return _tag; }
00517
00518
00528 l4_msgtag_t &tag() { return _tag; }
00529
00531
00536 inline bool put(Rcv_fpage const &);
00537
00542 inline bool put(Small_buf const &);
00543
00544
00549
00559 inline l4_msgtag_t wait(l4_umword_t *src)
00560 { return wait(src, L4_IPC_NEVER); }
00561
00572 inline l4_msgtag_t wait(l4_umword_t *src, l4_timeout_t timeout);
00573
00583 inline l4_msgtag_t receive(l4_cap_idx_t src)
00584 { return receive(src, L4_IPC_NEVER); }
00585
00585 inline l4_msgtag_t receive(l4_cap_idx_t src, l4_timeout_t timeout);
00586
00588
00592 inline l4_utcb_t *utcb() const { return _utcb; }
00593
00594 protected:
00595     l4_msgtag_t _tag;
00596     l4_utcb_t *_utcb;
00597     char *_current_msg;
00598     unsigned _pos;
00599     unsigned char _current_buf;
00600 };
00601
00602 class Istream_copy : public Istream
00603 {
00604 private:
00605     l4_msg_regs_t _mrs;
00606
00607 public:
00608     Istream_copy(Istream const &o) : Istream(o), _mrs(*l4_utcb_mr_u(o.utcb()))
00609     {
00610         // do some reverse mr to utcb trickery
00611         _utcb = reinterpret_cast<l4_utcb_t *>
00612             (reinterpret_cast<l4_addr_t>(&_mrs)
00613              - reinterpret_cast<l4_addr_t>(l4_utcb_mr_u(nullptr)));
00614         _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00615     }
00616
00617 };
00618
00634 class Ostream
00635 {
00636 public:
00640     Ostream(l4_utcb_t *utcb)
00641     : _tag(), _utcb(utcb),
00642       _current_msg(reinterpret_cast<char *>(l4_utcb_mr_u(_utcb)->mr)),
00643       _pos(0), _current_item(0)
00644     {}
00645
00649     void reset()
00650     {
00651         _pos = 0;
00652         _current_item = 0;
00653         _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00654     }
00655
00663
00670 template< typename T >
00671 bool put(T *buf, unsigned long size)

```

```

00672 {
00673     size *= sizeof(T);
00674     _pos = cxx::Type_traits<T>::align(_pos);
00675     if (Utcb_stream_check::check_utcb_data_offset(_pos + size))
00676         return false;
00677
00678     __builtin_memcpy(_current_msg + _pos, buf, size);
00679     _pos += size;
00680     return true;
00681 }
00682
00688 template< typename T >
00689 bool put(T const &v)
00690 {
00691     _pos = cxx::Type_traits<T>::align(_pos);
00692     if (Utcb_stream_check::check_utcb_data_offset(_pos + sizeof(T)))
00693         return false;
00694
00695     *(reinterpret_cast<T*>(_current_msg + _pos)) = v;
00696     _pos += sizeof(T);
00697     return true;
00698 }
00699
00700 int put(Varg const &va)
00701 {
00702     put(va.tag());
00703     put(va.data(), va.length());
00704
00705     return 0;
00706 }
00707
00708 template< typename T >
00709 int put(Varg_t<T> const &va)
00710 { return put(static_cast<Varg const &>(va)); }
00711
00717 l4_msgtag_t tag() const { return _tag; }
00718
00724 l4_msgtag_t &tag() { return _tag; }
00725
00727
00732 inline bool put_snd_item(Snd_fpage const &);
00733
00734
00739
00749 inline l4_msgtag_t send(l4_cap_idx_t dst, long proto = 0, unsigned flags = 0);
00750
00752
00756 inline l4_utcb_t *utcb() const { return _utcb; }
00757 #if 0
00761 unsigned long tell() const
00762 {
00763     unsigned w = l4_bytes_to_mwords(_pos) - _current_item * 2;
00764     _tag = l4_msgtag(0, w, _current_item, 0);
00765 }
00766 #endif
00767 public:
00768 l4_msgtag_t prepare_ipc(long proto = 0, unsigned flags = 0)
00769 {
00770     unsigned w = l4_bytes_to_mwords(_pos) - _current_item * 2;
00771     return l4_msgtag(proto, w, _current_item, flags);
00772 }
00773
00774 // XXX: this is a hack for <l4/sys/cxx/ipc_server> adaption
00775 void set_ipc_params(l4_msgtag_t tag)
00776 {
00777     _pos = (tag.words() + tag.items() * 2) * sizeof(l4_umword_t);
00778     _current_item = tag.items();
00779 }
00780 protected:
00781 l4_msgtag_t _tag;
00782 l4_utcb_t *_utcb;
00783 char *_current_msg;
00784 unsigned _pos;
00785 unsigned char _current_item;
00786 };
00787
00788
00800 class Iostream : public Istream, public Ostream
00801 {
00802 public:
00803
00812 explicit Iostream(l4_utcb_t *utcb)
00813 : Istream(utcb), Ostream(utcb)
00814 {}
00815
00816 // disambiguate those functions
00817 l4_msgtag_t tag() const { return Istream::tag(); }

```

```

00818     l4_msgtag_t &tag() { return Istream::tag(); }
00819     l4_utcb_t *utcb() const { return Istream::utcb(); }
00820
00826     void reset()
00827     {
00828         Istream::reset();
00829         Ostream::reset();
00830     }
00831
00832
00840
00841     using Istream::get;
00842     using Istream::put;
00843     using Ostream::put;
00844
00846
00851
00867     inline l4_msgtag_t call(l4_cap_idx_t dst, l4_timeout_t timeout, long proto = 0);
00868     inline l4_msgtag_t call(l4_cap_idx_t dst, long proto = 0);
00869
00885     inline l4_msgtag_t reply_and_wait(l4_umword_t *src_dst, long proto = 0)
00886     { return reply_and_wait(src_dst, L4_IPC_SEND_TIMEOUT_0, proto); }
00887
00888     inline l4_msgtag_t send_and_wait(l4_cap_idx_t dest, l4_umword_t *src,
00889                                     long proto = 0)
00890     { return send_and_wait(dest, src, L4_IPC_SEND_TIMEOUT_0, proto); }
00891
00908     inline l4_msgtag_t reply_and_wait(l4_umword_t *src_dst,
00909                                     l4_timeout_t timeout, long proto = 0);
00910     inline l4_msgtag_t send_and_wait(l4_cap_idx_t dest, l4_umword_t *src,
00911                                     l4_timeout_t timeout, long proto = 0);
00912     inline l4_msgtag_t reply(l4_timeout_t timeout, long proto = 0);
00913     inline l4_msgtag_t reply(long proto = 0)
00914     { return reply(L4_IPC_SEND_TIMEOUT_0, proto); }
00915
00917 };
00918
00919
00920 inline bool
00921 Ostream::put_snd_item(Snd_fpage const &v)
00922 {
00923     typedef Snd_fpage T;
00924     _pos = cxx::Type_traits<Snd_fpage>::align(_pos);
00925     if (Utcb_stream_check::check_utcb_data_offset(_pos + sizeof(T)))
00926         return false;
00927
00928     *(reinterpret_cast<T*>(_current_msg + _pos)) = v;
00929     _pos += sizeof(T);
00930     ++_current_item;
00931     return true;
00932 }
00933
00934
00935 inline bool
00936 Istream::put(Rcv_fpage const &item)
00937 {
00938     unsigned words = item.is_compound() ? 3 : 2;
00939     if (_current_buf >= L4_UTCB_GENERIC_BUFFERS_SIZE - words - 1)
00940         return false;
00941
00942     l4_utcb_br_u(_utcb)->bdr &= ~L4_BDR_OFFSET_MASK;
00943
00944     l4_umword_t *buf
00945         = reinterpret_cast<l4_umword_t *>(&l4_utcb_br_u(_utcb)->br[_current_buf]);
00946     *buf++ = item.base_x();
00947     *buf++ = item.data();
00948     if (item.is_compound())
00949         *buf++ = item.rcv_task();
00950     _current_buf += words;
00951     return true;
00952 }
00953
00954
00955 inline bool
00956 Istream::put(Small_buf const &item)
00957 {
00958     if (_current_buf >= L4_UTCB_GENERIC_BUFFERS_SIZE - 2)
00959         return false;
00960
00961     l4_utcb_br_u(_utcb)->bdr &= ~L4_BDR_OFFSET_MASK;
00962
00963     reinterpret_cast<Small_buf&>(l4_utcb_br_u(_utcb)->br[_current_buf]) = item;
00964     _current_buf += 1;
00965     return true;
00966 }
00967
00968

```

```

00969 inline l4_msgtag_t
00970 ostream::send(l4_cap_idx_t dst, long proto, unsigned flags)
00971 {
00972     l4_msgtag_t tag = prepare_ipc(proto, L4_MSGTAG_FLAGS & flags);
00973     return l4_ipc_send(dst, _utcb, tag, L4_IPC_NEVER);
00974 }
00975
00976 inline l4_msgtag_t
00977 ostream::call(l4_cap_idx_t dst, l4_timeout_t timeout, long label)
00978 {
00979     l4_msgtag_t tag = prepare_ipc(label);
00980     tag = l4_ipc_call(dst, ostream::_utcb, tag, timeout);
00981     istream::tag() = tag;
00982     istream::_pos = 0;
00983     return tag;
00984 }
00985
00986 inline l4_msgtag_t
00987 ostream::call(l4_cap_idx_t dst, long label)
00988 { return call(dst, L4_IPC_NEVER, label); }
00989
00990
00991 inline l4_msgtag_t
00992 ostream::reply_and_wait(l4_umword_t *src_dst, l4_timeout_t timeout, long proto)
00993 {
00994     l4_msgtag_t tag = prepare_ipc(proto);
00995     tag = l4_ipc_reply_and_wait(ostream::_utcb, tag, src_dst, timeout);
00996     istream::tag() = tag;
00997     istream::_pos = 0;
00998     return tag;
00999 }
01000
01001
01002 inline l4_msgtag_t
01003 ostream::send_and_wait(l4_cap_idx_t dest, l4_umword_t *src,
01004                       l4_timeout_t timeout, long proto)
01005 {
01006     l4_msgtag_t tag = prepare_ipc(proto);
01007     tag = l4_ipc_send_and_wait(dest, ostream::_utcb, tag, src, timeout);
01008     istream::tag() = tag;
01009     istream::_pos = 0;
01010     return tag;
01011 }
01012
01013 inline l4_msgtag_t
01014 ostream::reply(l4_timeout_t timeout, long proto)
01015 {
01016     l4_msgtag_t tag = prepare_ipc(proto);
01017     tag = l4_ipc_send(L4_INVALID_CAP | L4_SYSF_REPLY, ostream::_utcb, tag, timeout);
01018     istream::tag() = tag;
01019     istream::_pos = 0;
01020     return tag;
01021 }
01022
01023 inline l4_msgtag_t
01024 ostream::wait(l4_umword_t *src, l4_timeout_t timeout)
01025 {
01026     l4_msgtag_t res;
01027     res = l4_ipc_wait(_utcb, src, timeout);
01028     tag() = res;
01029     _pos = 0;
01030     return res;
01031 }
01032
01033
01034 inline l4_msgtag_t
01035 ostream::receive(l4_cap_idx_t src, l4_timeout_t timeout)
01036 {
01037     l4_msgtag_t res;
01038     res = l4_ipc_receive(src, _utcb, timeout);
01039     tag() = res;
01040     _pos = 0;
01041     return res;
01042 }
01043
01044 } // namespace Ipc
01045 } // namespace L4
01046
01055 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, bool &v) { s.get(v); return s; }
01056 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, int &v) { s.get(v); return s; }
01057 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, long int &v) { s.get(v); return s; }
01058 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, long long int &v) { s.get(v); return s; }
01059 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned int &v) { s.get(v); return s; }
01060 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned long int &v) { s.get(v); return s; }
01061 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned long long int &v) { s.get(v);
    return s; }

```

```

01062 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, short int &v) { s.get(v); return s; }
01063 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned short int &v) { s.get(v); return s; }
01064 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, char &v) { s.get(v); return s; }
01065 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned char &v) { s.get(v); return s; }
01066 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, signed char &v) { s.get(v); return s; }
01067 inline L4::Ipc::Istream &operator « (L4::Ipc::Istream &s, L4::Ipc::Rcv_fpage const &v) { s.put(v);
    return s; }
01068 inline L4::Ipc::Istream &operator « (L4::Ipc::Istream &s, L4::Ipc::Small_buf const &v) { s.put(v);
    return s; }
01069 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, L4::Ipc::Snd_fpage &v)
01070 {
01071     l4_umword_t b, d;
01072     s » b » d;
01073     v = L4::Ipc::Snd_fpage(b, d);
01074     return s;
01075 }
01076 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, L4::Ipc::Varg &v)
01077 { s.get(&v); return s; }
01078
01079
01088 inline
01089 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, l4_msgtag_t &v)
01090 {
01091     v = s.tag();
01092     return s;
01093 }
01094
01112 template< typename T >
01113 inline
01114 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
01115                             L4::Ipc::Internal::Buf_in<T> const &v)
01116 {
01117     unsigned long si;
01118     if (s.get(si) && s.has_more<T>(si))
01119         v.set_size(s.get(L4::Ipc::Msg_ptr<T>(v.buf()), si));
01120     else
01121         v.set_size(0);
01122     return s;
01123 }
01124
01139 template< typename T >
01140 inline
01141 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
01142                             L4::Ipc::Msg_ptr<T> const &v)
01143 {
01144     s.get(v);
01145     return s;
01146 }
01147
01160 template< typename T >
01161 inline
01162 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
01163                             L4::Ipc::Internal::Buf_cp_in<T> const &v)
01164 {
01165     unsigned long sz;
01166     s.get(sz);
01167     v.size() = s.get(v.buf(), cxx::min(v.size(), sz));
01168     return s;
01169 }
01170
01181 template< typename T >
01182 inline
01183 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
01184                             L4::Ipc::Str_cp_in<T> const &v)
01185 {
01186     unsigned long sz;
01187     s.get(sz);
01188     unsigned long rsz = s.get(v.buf(), cxx::min(v.size(), sz));
01189     if (rsz < v.size() && v.buf()[rsz - 1])
01190         ++rsz; // add the zero termination behind the received data
01191
01192     if (rsz != 0)
01193         v.buf()[rsz - 1] = 0;
01194
01195     v.size() = rsz;
01196     return s;
01197 }
01198
01199
01208 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, bool v) { s.put(v); return s; }
01209 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, int v) { s.put(v); return s; }
01210 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, long int v) { s.put(v); return s; }
01211 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, long long int v) { s.put(v); return s; }
01212 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned int v) { s.put(v); return s; }
01213 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned long int v) { s.put(v); return s; }
01214 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned long long int v) { s.put(v); return

```



```

    s; }
01215 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, short int v) { s.put(v); return s; }
01216 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned short int v) { s.put(v); return s; }
    }
01217 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, char v) { s.put(v); return s; }
01218 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned char v) { s.put(v); return s; }
01219 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, signed char v) { s.put(v); return s; }
01220 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Ipc::Snd_fpage const &v) {
    s.put_snd_item(v); return s; }
01221 template< typename T >
01222 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Cap<T> const &v)
01223 { s « L4::Ipc::Snd_fpage(v.fpage()); return s; }
01224
01225 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Ipc::Varg const &v)
01226 { s.put(v); return s; }
01227 template< typename T >
01228 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Ipc::Varg_t<T> const &v)
01229 { s.put(v); return s; }
01230
01242 inline
01243 L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, l4_msgtag_t const &v)
01244 {
01245     s.tag() = v;
01246     return s;
01247 }
01248
01257 template< typename T >
01258 inline
01259 L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s,
01260                             L4::Ipc::Internal::Buf_cp_out<T> const &v)
01261 {
01262     s.put(v.size());
01263     s.put(v.buf(), v.size());
01264     return s;
01265 }
01266
01279 inline
01280 L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, char const *v)
01281 {
01282     unsigned long l = __builtin_strlen(v) + 1;
01283     s.put(l);
01284     s.put(v, l);
01285     return s;
01286 }
01287
01288 namespace L4 { namespace Ipc {
01298 template< typename T >
01299 inline
01300 T read(Istream &s) { T t; s » t; return t; }
01301
01302 } // namespace Ipc
01303 } // namespace L4

```

## 16.174 ipc\_timeout\_queue

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00004  *
00005  * This file is licensed under the terms of the GNU General Public License 2.
00006  * Please see the COPYING-GPL-2 file for details.
00007  *
00008  * As a special exception, you may use this file as part of a free software
00009  * library without restriction. Specifically, if other files instantiate
00010  * templates or use macros or inline functions from this file, or you compile
00011  * this file and link it with other files to produce an executable, this file
00012  * does not by itself cause the resulting executable to be covered by the GNU
00013  * General Public License. This exception does not however invalidate any other
00014  * reasons why the executable file might be covered by the GNU General Public
00015  * License.
00016  */
00017 #pragma once
00018
00019 #include <l4/cxx/hlist>
00020 #include <l4/sys/cxx/ipc_server_loop>
00021
00022 namespace L4 { namespace Ipc_svr {
00023
00028 class Timeout : public cxx::H_list_item
00029 {
00030     friend class Timeout_queue;
00031 public:
00033     Timeout() : _timeout(0) {}

```

```

00034
00036     virtual ~Timeout() = 0;
00037
00044     virtual void expired() = 0;
00045
00052     l4_kernel_clock_t timeout() const
00053     { return _timeout; }
00054
00055 private:
00056     l4_kernel_clock_t _timeout;
00057 };
00058
00059 inline Timeout::~Timeout() {}
00060
00065 class Timeout_queue
00066 {
00067 public:
00069     typedef L4::Ipc_svr::Timeout Timeout;
00070
00075     l4_kernel_clock_t next_timeout() const
00076     {
00077         if (auto e = _timeouts.front())
00078             return e->timeout();
00079
00080         return 0;
00081     }
00082
00091     bool timeout_expired(l4_kernel_clock_t now) const
00092     {
00093         l4_kernel_clock_t next = next_timeout();
00094         return (next != 0) && (next <= now);
00095     }
00096
00101     void handle_expired_timeouts(l4_kernel_clock_t now)
00102     {
00103         while (!_timeouts.empty())
00104         {
00105             Queue::Iterator top = _timeouts.begin();
00106             if ((*top)->_timeout > now)
00107                 return;
00108
00109             Timeout *t = *top;
00110             top = _timeouts.erase(top);
00111             t->expired();
00112         }
00113     }
00114
00121     void add(Timeout *timeout, l4_kernel_clock_t time)
00122     {
00123         timeout->_timeout = time;
00124         Queue::Iterator i = _timeouts.begin();
00125         while (i != _timeouts.end() && (*i)->timeout() < time)
00126             ++i;
00127
00128         _timeouts.insert_before(timeout, i);
00129     }
00130
00136     void remove(Timeout *timeout)
00137     {
00138         _timeouts.remove(timeout);
00139     }
00140
00141 private:
00142     typedef cxx::H_list<Timeout> Queue;
00143     Queue _timeouts;
00144 };
00145
00159 template< typename HOOKS, typename BR_MAN = Br_manager_no_buffers >
00160 class Timeout_queue_hooks : public BR_MAN
00161 {
00162     l4_kernel_clock_t _now()
00163     { return static_cast<HOOKS*>(this)->now(); }
00164
00165     unsigned _timeout_br()
00166     { return this->first_free_br(); }
00167
00168 public:
00170     l4_timeout_t timeout()
00171     {
00172         l4_kernel_clock_t t = queue.next_timeout();
00173         if (t)
00174             return l4_timeout(L4_IPC_TIMEOUT_0, l4_timeout_abs(t, _timeout_br()));
00175         return L4_IPC_SEND_TIMEOUT_0;
00176     }
00177
00179     void setup_wait(l4_utcb_t *utcb, L4::Ipc_svr::Reply_mode mode)
00180     {

```

```

00181     // we must handle the timer only when called after a possible reply
00182     // otherwise we probably destroy the reply message.
00183     if (mode == L4::Ipc_svr::Reply_separate)
00184     {
00185         l4_kernel_clock_t now = _now();
00186         if (queue.timeout_expired(now))
00187             queue.handle_expired_timeouts(now);
00188     }
00189
00190     BR_MAN::setup_wait(utcb, mode);
00191 }
00192
00194 L4::Ipc_svr::Reply_mode before_reply(l4_msgtag_t, l4_utcb_t *)
00195 {
00196     // split up reply and wait when a timeout has expired
00197     if (queue.timeout_expired(_now()))
00198         return L4::Ipc_svr::Reply_separate;
00199     return L4::Ipc_svr::Reply_compound;
00200 }
00201
00212 int add_timeout(Timeout *timeout, l4_kernel_clock_t time) override
00213 {
00214     queue.add(timeout, time);
00215     return 0;
00216 }
00217
00225 int remove_timeout(Timeout *timeout) override
00226 {
00227     queue.remove(timeout);
00228     return 0;
00229 }
00230
00231 Timeout_queue queue;
00232 };
00233
00234 }}

```

## 16.175 l4/cxx/l4iostream File Reference

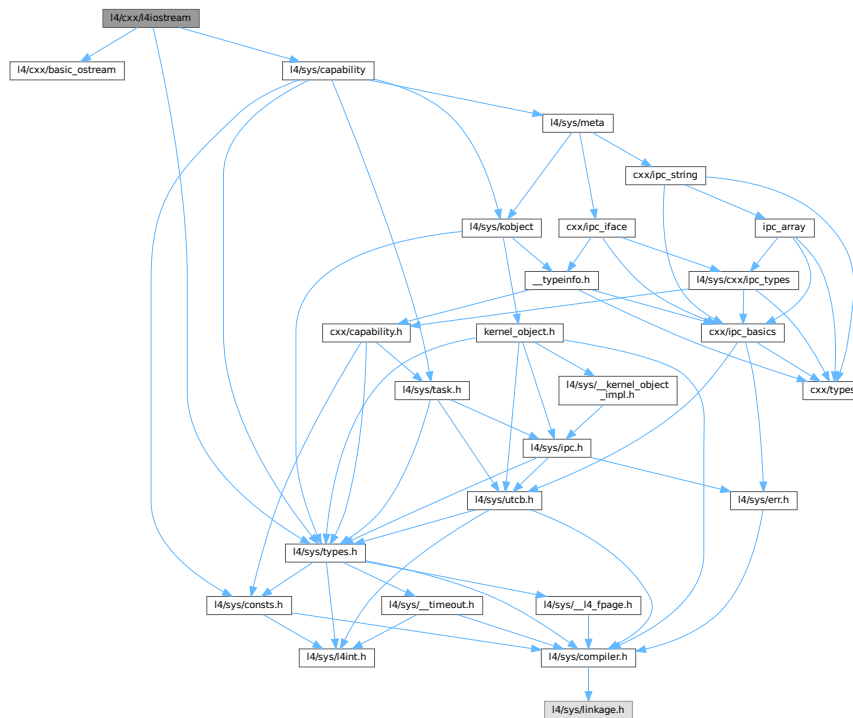
[L4](#) IO stream.

```

#include <l4/cxx/basic_ostream>
#include <l4/sys/types.h>
#include <l4/sys/capability>

```

Include dependency graph for `l4iostream`:



## 16.175.1 Detailed Description

[L4](#) IO stream.

Definition in file [l4iostream](#).

## 16.176 l4iostream

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/cxx/basic_ostream>
00027 #include <l4/sys/types.h>
00028 #include <l4/sys/capability>

```

```

00029
00030 inline
00031 L4::BasicOStream &operator « (L4::BasicOStream &o, l4_msgtag_t const &tag)
00032 {
00033     L4::IOBackend::Mode m = o.be_mode();
00034     o « "[l=" « L4::dec « tag.label() « "; w=" « tag.words() « "; i="
00035       « tag.items() « "];";
00036     o.be_mode(m);
00037     return o;
00038 }
00039
00040 template<typename T>
00041 inline
00042 L4::BasicOStream &operator « (L4::BasicOStream &o, L4::Cap<T> const &cap)
00043 {
00044     o « "[C:" « L4::n_hex(cap.cap()) « "];";
00045     return o;
00046 }

```

## 16.177 l4/cxx/l4types.h File Reference

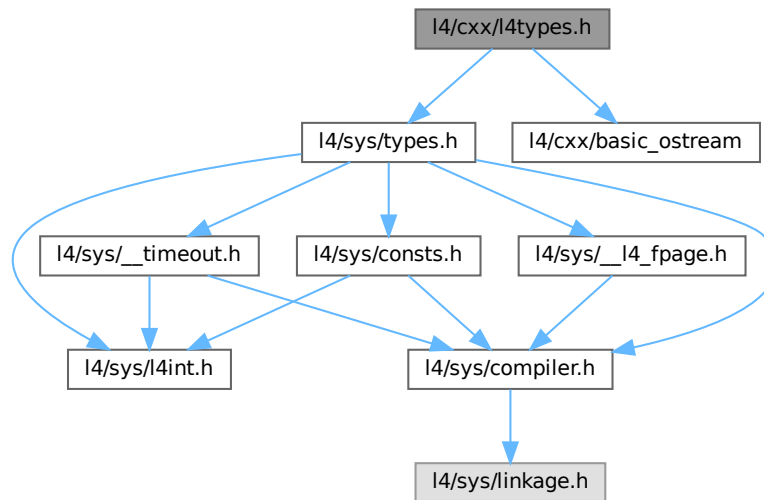
[L4 Types](#).

```

#include <l4/sys/types.h>
#include <l4/cxx/basic_ostream>

```

Include dependency graph for l4types.h:





```

00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019
00020 #pragma once
00021
00022 #include <l4/cxx/type_traits>
00023 #include <l4/cxx/std_alloc>
00024 #include <l4/cxx/std_ops>
00025
00026 namespace cxx {
00027 /*
00028  * Classes: List_item, List<D, Alloc>
00029  */
00030
00037 class List_item
00038 {
00039 public:
00045     class Iter
00046     {
00047     public:
00048         Iter(List_item *c, List_item *f) noexcept : _c(c), _f(f) {}
00049         Iter(List_item *f = 0) noexcept : _c(f), _f(f) {}
00050
00051         List_item *operator * () const noexcept { return _c; }
00052         List_item *operator -> () const noexcept { return _c; }
00053         Iter &operator ++ () noexcept
00054         {
00055             if (!_f)
00056                 _c = 0;
00057             else
00058                 _c = _c->get_next_item();
00059
00060             if (_c == _f)
00061                 _c = 0;
00062
00063             return *this;
00064         }
00065
00066         Iter operator ++ (int) noexcept
00067         { Iter o = *this; operator ++ (); return o; }
00068
00069         Iter &operator -- () noexcept
00070         {
00071             if (!_f)
00072                 _c = 0;
00073             else
00074                 _c = _c->get_prev_item();
00075
00076             if (_c == _f)
00077                 _c = 0;
00078
00079             return *this;
00080         }
00081
00082         Iter operator -- (int) noexcept
00083         { Iter o = *this; operator -- (); return o; }
00084
00086         List_item *remove_me() noexcept
00087         {
00088             if (!_c)
00089                 return 0;
00090
00091             List_item *l = _c;
00092             operator ++ ();
00093             l->remove_me();
00094
00095             if (_f == l)
00096                 _f = _c;
00097
00098             return l;
00099         }
00100
00101     private:
00102         List_item *_c, *_f;
00103     };
00104
00118     template< typename T, bool Poly = false>
00119     class T_iter : public Iter
00120     {
00121     private:
00122         static bool const P = !Conversion<const T*, const List_item *>::exists
00123             || Poly;
00124     };

```

```

00125     static List_item *cast_to_li(T *i, Int_to_type<true>) noexcept
00126     { return dynamic_cast<List_item*>(i); }
00127
00128     static List_item *cast_to_li(T *i, Int_to_type<false>) noexcept
00129     { return i; }
00130
00131     static T *cast_to_type(List_item *i, Int_to_type<true>) noexcept
00132     { return dynamic_cast<T*>(i); }
00133
00134     static T *cast_to_type(List_item *i, Int_to_type<false>) noexcept
00135     { return static_cast<T*>(i); }
00136
00137 public:
00138
00139     template< typename O >
00140     explicit T_iter(T_iter<O> const &o) noexcept
00141     : Iter(o) { dynamic_cast<T*>(&o); }
00142
00143     //TIter(CListItem *f) : Iter(f) {}
00144     T_iter(T *f = 0) noexcept : Iter(cast_to_li(f, Int_to_type<P>())) {}
00145     T_iter(T *c, T *f) noexcept
00146     : Iter(cast_to_li(c, Int_to_type<P>()),
00147     cast_to_li(f, Int_to_type<P>()))
00148     {}
00149
00150     inline T *operator * () const noexcept
00151     { return cast_to_type(Iter::operator * (),Int_to_type<P>()); }
00152     inline T *operator -> () const noexcept
00153     { return operator * (); }
00154
00155     T_iter<T, Poly> operator ++ (int) noexcept
00156     { T_iter<T, Poly> o = *this; Iter::operator ++ (); return o; }
00157     T_iter<T, Poly> operator -- (int) noexcept
00158     { T_iter<T, Poly> o = *this; Iter::operator -- (); return o; }
00159     T_iter<T, Poly> &operator ++ () noexcept
00160     { Iter::operator ++ (); return *this; }
00161     T_iter<T, Poly> &operator -- () noexcept
00162     { Iter::operator -- (); return *this; }
00163     inline T *remove_me() noexcept;
00164 };
00165
00166 List_item() noexcept : _n(this), _p(this) {}
00167
00168 protected:
00169 List_item(List_item const &) noexcept : _n(this), _p(this) {}
00170
00171 public:
00172 List_item *get_prev_item() const noexcept { return _p; }
00173
00174 List_item *get_next_item() const noexcept { return _n; }
00175
00176 void insert_prev_item(List_item *p) noexcept
00177 {
00178     p->_p->_n = this;
00179     List_item *pr = p->_p;
00180     p->_p = _p;
00181     _p->_n = p;
00182     _p = pr;
00183 }
00184
00185 void insert_next_item(List_item *p) noexcept
00186 {
00187     p->_p->_n = _n;
00188     p->_p = this;
00189     _n->_p = p;
00190     _n = p;
00191 }
00192
00193 void remove_me() noexcept
00194 {
00195     if (_p != this)
00196     {
00197         _p->_n = _n;
00198         _n->_p = _p;
00199     }
00200     _p = _n = this;
00201 }
00202
00203 template< typename C, typename N >
00204 static inline C *push_back(C *head, N *p) noexcept;
00205
00206 template< typename C, typename N >
00207 static inline C *push_front(C *head, N *p) noexcept;
00208
00209 template< typename C, typename N >
00210 static inline C *remove(C *head, N *p) noexcept;
00211
00212

```



```

00241 private:
00242     List_item *_n, *_p;
00243 };
00244
00245
00246 /* IMPLEMENTATION -----*/
00247 template< typename C, typename N >
00248 C *List_item::push_back(C *h, N *p) noexcept
00249 {
00250     if (!p)
00251         return h;
00252     if (!h)
00253         return p;
00254     h->insert_prev_item(p);
00255     return h;
00256 }
00257
00258 template< typename C, typename N >
00259 C *List_item::push_front(C *h, N *p) noexcept
00260 {
00261     if (!p)
00262         return h;
00263     if (h)
00264         h->insert_prev_item(p);
00265     return p;
00266 }
00267
00268 template< typename C, typename N >
00269 C *List_item::remove(C *h, N *p) noexcept
00270 {
00271     if (!p)
00272         return h;
00273     if (!h)
00274         return 0;
00275     if (h == p)
00276     {
00277         if (p == p->_n)
00278             h = 0;
00279         else
00280             h = static_cast<C*>(p->_n);
00281     }
00282     p->remove_me();
00283
00284     return h;
00285 }
00286
00287 template< typename T, bool Poly >
00288 inline
00289 T *List_item::T_iter<T, Poly>::remove_me() noexcept
00290 { return cast_to_type(Iter::remove_me(), Int_to_type<P>()); }
00291
00292
00293 template< typename T >
00294 class T_list_item : public List_item
00295 {
00296 public:
00297     T *next() const { return static_cast<T*>(List_item::get_next_item()); }
00298     T *prev() const { return static_cast<T*>(List_item::get_prev_item()); }
00299 };
00300
00301
00302 template< typename LI >
00303 class L_list
00304 {
00305 private:
00306     LI *_h;
00307
00308 public:
00309     L_list() : _h(0) {}
00310
00311     void push_front(LI *e) { _h = LI::push_front(_h, e); }
00312     void push_back(LI *e) { _h = LI::push_back(_h, e); }
00313     void insert_before(LI *e, LI *p)
00314     {
00315         p->insert_prev_item(e);
00316         if (_h == p)
00317             _h = e;
00318     }
00319     void insert_after(LI *e, LI *p) { p->insert_next_item(e); }
00320
00321     void remove(LI *e)
00322     { _h = LI::remove(_h, e); }
00323
00324     LI *head() const { return _h; }
00325 };
00326
00327

```

```

00333 template< typename D, template<typename A> class Alloc = New_allocator >
00334 class List
00335 {
00336 private:
00337     class E : public List_item
00338     {
00339     public:
00340         E(D const &d) noexcept : data(d) {}
00341         D data;
00342     };
00343
00344 public:
00345     class Node : private E
00346     {};
00347
00348     typedef Alloc<Node> Node_alloc;
00349
00354     class Iter
00355     {
00356     private:
00357         List_item::T_iter<E> _i;
00358
00359     public:
00360         Iter(E *e) noexcept : _i(e) {}
00361
00362         D &operator * () const noexcept { return (*_i)->data; }
00363         D &operator -> () const noexcept { return (*_i)->data; }
00364
00365         Iter operator ++ (int) noexcept
00366         { Iter o = *this; operator ++ (); return o; }
00367         Iter operator -- (int) noexcept
00368         { Iter o = *this; operator -- (); return o; }
00369         Iter &operator ++ () noexcept { ++_i; return *this; }
00370         Iter &operator -- () noexcept { --_i; return *this; }
00371
00373         operator E* () const noexcept { return *_i; }
00374     };
00375
00376     List(Alloc<Node> const &a = Alloc<Node>()) noexcept : _h(0), _l(0), _a(a) {}
00377
00379     void push_back(D const &d) noexcept
00380     {
00381         void *n = _a.alloc();
00382         if (!n) return;
00383         _h = E::push_back(_h, new (n) E(d));
00384         ++_l;
00385     }
00386
00388     void push_front(D const &d) noexcept
00389     {
00390         void *n = _a.alloc();
00391         if (!n) return;
00392         _h = E::push_front(_h, new (n) E(d));
00393         ++_l;
00394     }
00395
00397     void remove(Iter const &i) noexcept
00398     { E *e = i; _h = E::remove(_h, e); --_l; _a.free(e); }
00399
00401     unsigned long size() const noexcept { return _l; }
00402
00404     D const &operator [] (unsigned long idx) const noexcept
00405     { Iter i = _h; for (; idx && *i; ++i, --idx) { } return *i; }
00406
00408     D &operator [] (unsigned long idx) noexcept
00409     { Iter i = _h; for (; idx && *i; ++i, --idx) { } return *i; }
00410
00412     Iter items() noexcept { return Iter(_h); }
00413
00414 private:
00415     E *_h;
00416     unsigned _l;
00417     Alloc<Node> _a;
00418 };
00419
00420
00421 };
00422

```

## 16.180 list\_alloc

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*

```

```

00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *          Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *          economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020
00021 #pragma once
00022
00023 #include <l4/cxx/arith>
00024 #include <l4/cxx/minmax>
00025 #include <l4/sys/consts.h>
00026
00027 namespace cxx {
00028
00032 class List_alloc
00033 {
00034 private:
00035     friend class List_alloc_sanity_guard;
00036
00037     struct Mem_block
00038     {
00039         Mem_block *next;
00040         unsigned long size;
00041     };
00042
00043     Mem_block *_first;
00044
00045     inline void check_overlap(void *, unsigned long );
00046     inline void sanity_check_list(char const *, char const *);
00047     inline void merge();
00048
00049 public:
00050
00057     List_alloc() : _first(0) {}
00058
00071     inline void free(void *block, unsigned long size, bool initial_free = false);
00072
00087     inline void *alloc(unsigned long size, unsigned long align,
00088                       unsigned long lower = 0, unsigned long upper = ~0UL);
00089
00110     inline void *alloc_max(unsigned long min, unsigned long *max,
00111                           unsigned long align, unsigned granularity,
00112                           unsigned long lower = 0, unsigned long upper = ~0UL);
00113
00119     inline unsigned long avail();
00120
00121     template <typename DBG>
00122     void dump_free_list(DBG &out);
00123 };
00124
00125 #if !defined (CXX_LIST_ALLOC_SANITY)
00126 class List_alloc_sanity_guard
00127 {
00128 public:
00129     List_alloc_sanity_guard(List_alloc *, char const *)
00130     {}
00131
00132 };
00133
00134
00135 void
00136 List_alloc::check_overlap(void *, unsigned long )
00137 {}
00138
00139 void
00140 List_alloc::sanity_check_list(char const *, char const *)
00141 {}
00142
00143 #else
00144
00145 class List_alloc_sanity_guard
00146 {
00147 private:
00148     List_alloc *a;
00149     char const *func;

```

```

00150
00151 public:
00152     List_alloc_sanity_guard(List_alloc *a, char const *func)
00153     : a(a), func(func)
00154     { a->sanity_check_list(func, "entry"); }
00155
00156     ~List_alloc_sanity_guard()
00157     { a->sanity_check_list(func, "exit"); }
00158 };
00159
00160 void
00161 List_alloc::check_overlap(void *b, unsigned long s)
00162 {
00163     unsigned long const mb_align = (1UL << arith::Ld<sizeof(Mem_block)>::value) - 1;
00164     if ((unsigned long)b & mb_align)
00165     {
00166         L4::cerr << "List_alloc(FATAL): trying to free unaligned memory: "
00167                 << b << " align=" << arith::Ld<sizeof(Mem_block)>::value << "\n";
00168     }
00169
00170     Mem_block *c = _first;
00171     for (; c ; c = c->next)
00172     {
00173         unsigned long x_s = (unsigned long)b;
00174         unsigned long x_e = x_s + s;
00175         unsigned long b_s = (unsigned long)c;
00176         unsigned long b_e = b_s + c->size;
00177
00178         if ((x_s >= b_s && x_s < b_e)
00179             || (x_e > b_s && x_e <= b_e)
00180             || (b_s >= x_s && b_s < x_e)
00181             || (b_e > x_s && b_e <= x_e))
00182         {
00183             L4::cerr << "List_alloc(FATAL): trying to free memory that "
00184                     << "is already free: \n ["
00185                     << (void*)x_s << '-' << (void*)x_e << "] overlaps ["
00186                     << (void*)b_s << '-' << (void*)b_e << "]\n";
00187         }
00188     }
00189 }
00190
00191 void
00192 List_alloc::sanity_check_list(char const *func, char const *info)
00193 {
00194     Mem_block *c = _first;
00195     for (; c ; c = c->next)
00196     {
00197         if (c->next)
00198         {
00199             if (c >= c->next)
00200             {
00201                 L4::cerr << "List_alloc(FATAL): " << func << ' (' << info
00202                         << "): list order violation\n";
00203             }
00204
00205             if (((unsigned long)c) + c->size > (unsigned long)c->next)
00206             {
00207                 L4::cerr << "List_alloc(FATAL): " << func << ' (' << info
00208                         << "): list order violation\n";
00209             }
00210         }
00211     }
00212 }
00213
00214 #endif
00215
00216 void
00217 List_alloc::merge()
00218 {
00219     List_alloc_sanity_guard __attribute__((unused)) guard(this, __func__);
00220     Mem_block *c = _first;
00221     while (c && c->next)
00222     {
00223         unsigned long f_start = reinterpret_cast<unsigned long>(c);
00224         unsigned long f_end   = f_start + c->size;
00225         unsigned long n_start = reinterpret_cast<unsigned long>(c->next);
00226
00227         if (f_end == n_start)
00228         {
00229             c->size += c->next->size;
00230             c->next = c->next->next;
00231             continue;
00232         }
00233
00234         c = c->next;
00235     }
00236 }

```

```

00237
00238 void
00239 List_alloc::free(void *block, unsigned long size, bool initial_free)
00240 {
00241     List_alloc_sanity_guard __attribute__((unused)) guard(this, __func__);
00242
00243     unsigned long const mb_align = (1UL « arith::Ld<sizeof(Mem_block)>::value) - 1;
00244
00245     if (initial_free)
00246     {
00247         // enforce alignment constraint on initial memory
00248         unsigned long nblock = (reinterpret_cast<unsigned long>(block) + mb_align)
00249             & ~mb_align;
00250         size = (size - (nblock - reinterpret_cast<unsigned long>(block)))
00251             & ~mb_align;
00252         block = reinterpret_cast<void*>(nblock);
00253     }
00254     else
00255         // blow up size to the minimum aligned size
00256         size = (size + mb_align) & ~mb_align;
00257
00258     check_overlap(block, size);
00259
00260     Mem_block **c = &_first;
00261     Mem_block *next = 0;
00262
00263     if (*c)
00264     {
00265         while (*c && *c < block)
00266             c = &(*c)->next;
00267
00268         next = *c;
00269     }
00270
00271     *c = reinterpret_cast<Mem_block*>(block);
00272
00273     (*c)->next = next;
00274     (*c)->size = size;
00275
00276     merge();
00277 }
00278
00279 void *
00280 List_alloc::alloc_max(unsigned long min, unsigned long *max, unsigned long align,
00281                     unsigned granularity, unsigned long lower,
00282                     unsigned long upper)
00283 {
00284     List_alloc_sanity_guard __attribute__((unused)) guard(this, __func__);
00285
00286     unsigned char const mb_bits = arith::Ld<sizeof(Mem_block)>::value;
00287     unsigned long const mb_align = (1UL « mb_bits) - 1;
00288
00289     // blow minimum up to at least the minimum aligned size of a Mem_block
00290     min = l4_round_size(min, mb_bits);
00291     // truncate maximum to at least the size of a Mem_block
00292     *max = l4_trunc_size(*max, mb_bits);
00293     // truncate maximum size according to granularity
00294     *max = *max & ~(granularity - 1UL);
00295
00296     if (min > *max)
00297         return 0;
00298
00299     unsigned long almask = align ? (align - 1UL) : 0;
00300
00301     // minimum alignment is given by the size of a Mem_block
00302     if (almask < mb_align)
00303         almask = mb_align;
00304
00305     Mem_block **c = &_first;
00306     Mem_block **fit = 0;
00307     unsigned long max_fit = 0;
00308     unsigned long a_lower = (lower + almask) & ~almask;
00309
00310     for (; *c; c = &(*c)->next)
00311     {
00312         // address of free memory block
00313         unsigned long n_start = reinterpret_cast<unsigned long>(*c);
00314
00315         // block too small, next
00316         // XXX: maybe we can skip this and just do the test below
00317         if ((*c)->size < min)
00318             continue;
00319
00320         // block outside region, next
00321         if (upper < n_start || a_lower > n_start + (*c)->size)
00322             continue;
00323

```

```

00324 // aligned start address within the free block
00325 unsigned long a_start = (n_start + almask) & ~almask;
00326
00327 // check if aligned start address is behind the block, next
00328 if (a_start - n_start >= (*c)->size)
00329     continue;
00330
00331 a_start = a_start < a_lower ? a_lower : a_start;
00332
00333 // end address would overflow, next
00334 if (min > ~0UL - a_start)
00335     continue;
00336
00337 // block outside region, next
00338 if (a_start + min - 1UL > upper)
00339     continue;
00340
00341 // remaining size after subtracting the padding for the alignment
00342 unsigned long r_size = (*c)->size - a_start + n_start;
00343
00344 // upper limit can limit maximum size
00345 if (a_start + r_size - 1UL > upper)
00346     r_size = upper - a_start + 1UL;
00347
00348 // round down according to granularity
00349 r_size &= ~(granularity - 1UL);
00350
00351 // block too small
00352 if (r_size < min)
00353     continue;
00354
00355 if (r_size >= *max)
00356 {
00357     fit = c;
00358     max_fit = *max;
00359     break;
00360 }
00361
00362 if (r_size > max_fit)
00363 {
00364     max_fit = r_size;
00365     fit = c;
00366 }
00367 }
00368
00369 if (fit)
00370 {
00371     unsigned long n_start = reinterpret_cast<unsigned long>(*fit);
00372     unsigned long a_lower = (lower + almask) & ~almask;
00373     unsigned long a_start = (n_start + almask) & ~almask;
00374     a_start = a_start < a_lower ? a_lower : a_start;
00375     unsigned long r_size = (*fit)->size - a_start + n_start;
00376
00377     if (a_start > n_start)
00378     {
00379         (*fit)->size -= r_size;
00380         fit = &(*fit)->next;
00381     }
00382     else
00383         *fit = (*fit)->next;
00384
00385     *max = max_fit;
00386     if (r_size == max_fit)
00387         return reinterpret_cast<void *>(a_start);
00388
00389     Mem_block *m = reinterpret_cast<Mem_block*>(a_start + max_fit);
00390     m->next = *fit;
00391     m->size = r_size - max_fit;
00392     *fit = m;
00393     return reinterpret_cast<void *>(a_start);
00394 }
00395
00396 return 0;
00397 }
00398
00399 void *
00400 List_alloc::alloc(unsigned long size, unsigned long align, unsigned long lower,
00401                  unsigned long upper)
00402 {
00403     List_alloc_sanity_guard __attribute__((unused)) guard(this, __func__);
00404
00405     unsigned long const mb_align
00406         = (1UL « arith::Ld<sizeof(Mem_block)>::value) - 1;
00407
00408     // blow up size to the minimum aligned size
00409     size = (size + mb_align) & ~mb_align;
00410

```

```

00411 unsigned long almask = align ? (align - 1UL) : 0;
00412
00413 // minimum alignment is given by the size of a Mem_block
00414 if (almask < mb_align)
00415     almask = mb_align;
00416
00417 Mem_block **c = &_first;
00418 unsigned long a_lower = (lower + almask) & ~almask;
00419
00420 for (; *c; c=(*c)->next)
00421 {
00422     // address of free memory block
00423     unsigned long n_start = reinterpret_cast<unsigned long>(*c);
00424
00425     // block too small, next
00426     // XXX: maybe we can skip this and just do the test below
00427     if ((*c)->size < size)
00428         continue;
00429
00430     // block outside region, next
00431     if (upper < n_start || a_lower > n_start + (*c)->size)
00432         continue;
00433
00434     // aligned start address within the free block
00435     unsigned long a_start = (n_start + almask) & ~almask;
00436
00437     // block too small after alignment, next
00438     if (a_start - n_start >= (*c)->size)
00439         continue;
00440
00441     a_start = a_start < a_lower ? a_lower : a_start;
00442
00443     // end address would overflow, next
00444     if (size > ~0UL - a_start)
00445         continue;
00446
00447     // block outside region, next
00448     if (a_start + size - 1UL > upper)
00449         continue;
00450
00451     // remaining size after subtracting the padding
00452     // for the alignment
00453     unsigned long r_size = (*c)->size - a_start + n_start;
00454
00455     // block too small
00456     if (r_size < size)
00457         continue;
00458
00459     if (a_start > n_start)
00460     {
00461         // have free space before the allocated block
00462         // shrink the block and set c to the next pointer of that
00463         // block
00464         (*c)->size -= r_size;
00465         c = &(*c)->next;
00466     }
00467     else
00468     {
00469         // drop the block, c remains the next pointer of the
00470         // previous block
00471         *c = (*c)->next;
00472     }
00473
00474     // allocated the whole remaining space
00475     if (r_size == size)
00476         return reinterpret_cast<void*>(a_start);
00477
00478     // add a new free block behind the allocated block
00479     Mem_block *m = reinterpret_cast<Mem_block*>(a_start + size);
00480     m->next = *c;
00481     m->size = r_size - size;
00482     *c = m;
00483     return reinterpret_cast<void *>(a_start);
00484 }
00485 return 0;
00486 }
00487 unsigned long
00488 List_alloc::avail()
00489 {
00490     List_alloc_sanity_guard __attribute__((unused)) guard(this, __FUNCTION__);
00491     Mem_block *c = _first;
00492     unsigned long a = 0;
00493     while (c)
00494     {
00495         a += c->size;
00496         c = c->next;
00497     }

```

```
00498
00499     return a;
00500 }
00501
00502 template <typename DBG>
00503 void
00504 List_alloc::dump_free_list(DBG &out)
00505 {
00506     Mem_block *c = _first;
00507     while (c)
00508     {
00509         unsigned sz;
00510         const char *unit;
00511
00512         if (c->size < 1024)
00513         {
00514             sz = c->size;
00515             unit = "Byte";
00516         }
00517         else if (c->size < 1 « 20)
00518         {
00519             sz = c->size » 10;
00520             unit = "kB";
00521         }
00522         else
00523         {
00524             sz = c->size » 20;
00525             unit = "MB";
00526         }
00527
00528         out.printf("%12p - %12p (%u %s)\n", c,
00529                 reinterpret_cast<char *>(c) + c->size - 1, sz, unit);
00530
00531         c = c->next;
00532     }
00533 }
00534
00535 }
```

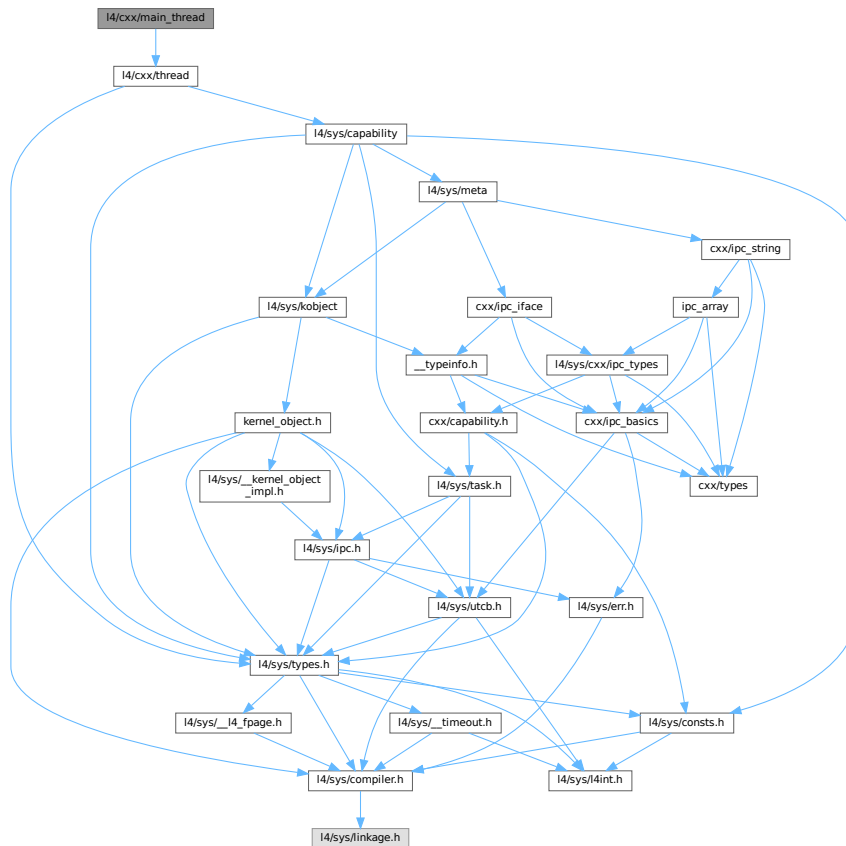
## 16.181 I4/cxx/main\_thread File Reference

Main thread.



```
#include <l4/cxx/thread>
```

Include dependency graph for main\_thread:



## Namespaces

- namespace `cxx`  
Our C++ library.

## 16.181.1 Detailed Description

Main thread.

Definition in file [main\\_thread](#).

## 16.182 main\_thread

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2004-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
```

```

00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023
00024 #ifndef L4_CXX_MAIN_THREAD_H__
00025 #define L4_CXX_MAIN_THREAD_H__
00026
00027 #include <l4/cxx/thread>
00028
00029 namespace cxx {
00030     class MainThread : public Thread
00031     {
00032     public:
00033         MainThread() : Thread(true)
00034         {}
00035     };
00036 };
00037
00038 #endif /* L4_CXX_MAIN_THREAD_H__ */

```

## 16.183 minmax

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019 #pragma once
00020
00021 #include "type_traits"
00022
00023 namespace cxx
00024 {
00025     // trivial, used to terminate the variadic recursion
00026     template<typename A>
00027     constexpr A const &
00028     min(A const &a)
00029     { return a; }
00030
00031     template<typename A, typename ...ARGS>
00032     constexpr A const &
00033     min(A const &a1, A const &a2, ARGS const &...a)
00034     {
00035         return min((a1 <= a2) ? a1 : a2, a...);
00036     }
00037
00038     template<typename A, typename ...ARGS>
00039     constexpr A const &
00040     min(cxx::identity_t<A> const &a1,
00041         cxx::identity_t<A> const &a2,
00042         ARGS const &...a)
00043     {
00044         return min<A>((a1 <= a2) ? a1 : a2, a...);
00045     }
00046
00047     // trivial, used to terminate the variadic recursion
00048     template<typename A>
00049     constexpr A const &
00050     max(A const &a)
00051     { return a; }
00052
00053     template<typename A, typename ...ARGS>
00054     constexpr A const &
00055     max(A const &a1, A const &a2, ARGS const &...a)
00056     {
00057         return max((a1 <= a2) ? a1 : a2, a...);
00058     }
00059
00060     template<typename A, typename ...ARGS>
00061     constexpr A const &
00062     max(cxx::identity_t<A> const &a1,
00063         cxx::identity_t<A> const &a2,
00064         ARGS const &...a)
00065     {
00066         return max<A>((a1 <= a2) ? a1 : a2, a...);
00067     }
00068 }

```

```

00087     template<typename A, typename ...ARGS>
00088     constexpr A const &
00089     max(A const &a1, A const &a2, ARGS const &...a)
00090     { return max((a1 >= a2) ? a1 : a2, a...); }
00091
00102     template<typename A, typename ...ARGS>
00103     constexpr A const &
00104     max(cxx::identity_t<A> const &a1,
00105         cxx::identity_t<A> const &a2,
00106         ARGS const &...a)
00107     {
00108         return max<A>((a1 >= a2) ? a1 : a2, a...);
00109     }
00110
00118     template< typename T1 >
00119     inline
00120     T1 clamp(T1 v, T1 lo, T1 hi)
00121     { return min(hi, max(lo, v)); }
00122 };

```

## 16.184 observer

```

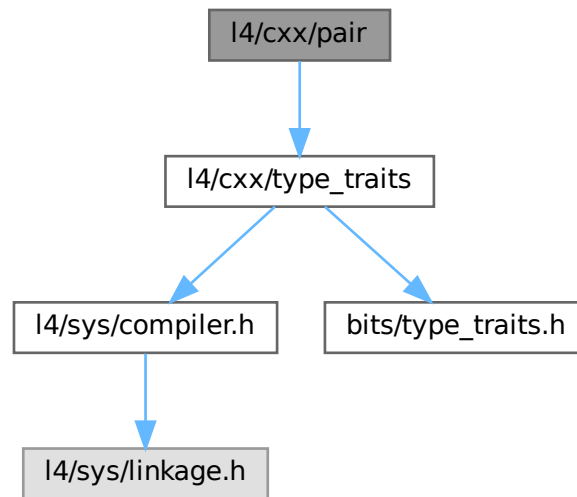
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  */
00010 #pragma once
00011
00012 #include <l4/cxx/hlist>
00013
00014 namespace cxx {
00015
00016     class Observer : public H_list_item
00017     {
00018     public:
00019         virtual void notify() = 0;
00020     };
00021
00022     class Notifier : public H_list<Observer>
00023     {
00024     public:
00025         void notify()
00026         {
00027             for (Iterator i = begin(); i != end(); ++i)
00028                 i->notify();
00029         }
00030     };
00031
00032 }
00033
00034

```

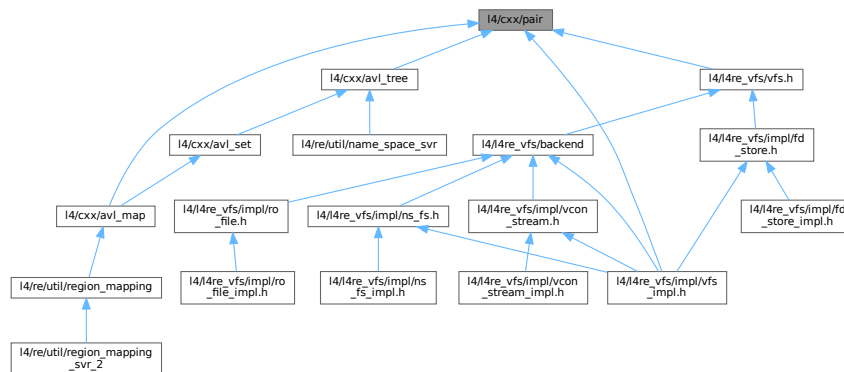
## 16.185 l4/cxx/pair File Reference

Pair implementation.

```
#include <l4/cxx/type_traits>
Include dependency graph for pair:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `cxx::Pair< First, Second >`  
*Pair of two values.*
- class `cxx::Pair_first_compare< Cmp, Typ >`  
*Comparison functor for [Pair](#).*

## Namespaces

- namespace `cxx`  
*Our C++ library.*

## 16.185.1 Detailed Description

Pair implementation.

Definition in file [pair](#).

## 16.186 pair

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 #include <l4/cxx/type_traits>
00026
00027 namespace cxx {
00028
00037 template< typename First, typename Second >
00038 struct Pair
00039 {
00041     typedef First First_type;
00043     typedef Second Second_type;
00044
00046     First first;
00048     Second second;
00049
00055     template<typename A1, typename A2>
00056     Pair(A1 &&first, A2 &&second)
00057     : first(cxx::forward<A1>(first)), second(cxx::forward<A2>(second)) {}
00058
00063     template<typename A1>
00064     Pair(A1 &&first)
00065     : first(cxx::forward<A1>(first)), second() {}
00066
00068     Pair() = default;
00069 };
00070
00071 template< typename F, typename S >
00072 Pair<F,S> pair(F const &f, S const &s)
00073 { return cxx::Pair<F,S>(f,s); }
00074
00075
00084 template< typename Cmp, typename Typ >
00085 class Pair_first_compare
00086 {
00087 private:
00088     Cmp const &_cmp;
00089
00090 public:
00095     Pair_first_compare(Cmp const &cmp = Cmp()) : _cmp(cmp) {}
00096
00102     bool operator () (Typ const &l, Typ const &r) const
00103     { return _cmp(l.first,r.first); }
00104 };
00105
00106 }
00107
00108 template< typename OS, typename A, typename B >
00109 inline
00110 OS &operator << (OS &os, cxx::Pair<A,B> const &p)
00111 {
00112     os << p.first << ' ' << p.second;
00113     return os;
00114 }
00115
```

## 16.187 ref\_ptr

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019
00020 #pragma once
00021
00022 #include "type_traits"
00023 #include <cstddef>
00024 #include <lib/sys/compiler.h>
00025
00026 namespace cxx {
00027
00028 template< typename T >
00029 struct Default_ref_counter
00030 {
00031     void h_drop_ref(T *p) noexcept
00032     {
00033         if (p->remove_ref() == 0)
00034             delete p;
00035     }
00036
00037     void h_take_ref(T *p) noexcept
00038     {
00039         p->add_ref();
00040     }
00041 };
00042
00043 struct Ref_ptr_base
00044 {
00045     enum Default_value
00046     { Nil = 0 };
00047 };
00048
00049 template<typename T, template< typename X > class CNT = Default_ref_counter>
00050 class Weak_ptr;
00051
00052 template <
00053     typename T = void,
00054     template< typename X > class CNT = Default_ref_counter
00055 >
00056 class Ref_ptr : public Ref_ptr_base, private CNT<T>
00057 {
00058 private:
00059     typedef decltype(nullptr) Null_type;
00060     typedef Weak_ptr<T, CNT> Wp;
00061
00062 public:
00063     Ref_ptr() noexcept : _p(0) {}
00064
00065     Ref_ptr(Ref_ptr_base::Default_value v)
00066     : _p(reinterpret_cast<T*>(static_cast<unsigned long>(v))) {}
00067
00068     Ref_ptr(Wp const &o) noexcept : _p(o.ptr())
00069     { __take_ref(); }
00070
00071     Ref_ptr(decltype(nullptr) n) noexcept : _p(n) {}
00072
00073     template<typename X>
00074     explicit Ref_ptr(X *o) noexcept : _p(o)
00075     { __take_ref(); }
00076
00077     Ref_ptr(T *o, [[maybe_unused]] bool d) noexcept : _p(o) {}
00078
00079     T *get() const noexcept __attribute__((pure))
00080     {
00081         return _p;
00082     }
00083
00084     T *ptr() const noexcept __attribute__((pure))
00085     {

```

```

00140     return _p;
00141 }
00142
00149 T *release() noexcept
00150 {
00151     T *p = _p;
00152     _p = 0;
00153     return p;
00154 }
00155
00156 ~Ref_ptr() noexcept
00157 { __drop_ref(); }
00158
00159 template<typename OT>
00160 Ref_ptr(Ref_ptr<OT, CNT> const &o) noexcept
00161 {
00162     _p = o.ptr();
00163     __take_ref();
00164 }
00165
00166 Ref_ptr(Ref_ptr<T> const &o) noexcept
00167 {
00168     _p = o._p;
00169     __take_ref();
00170 }
00171
00172 template< typename OT >
00173 void operator = (Ref_ptr<OT> const &o) noexcept
00174 {
00175     __drop_ref();
00176     _p = o.ptr();
00177     __take_ref();
00178 }
00179
00180 void operator = (Ref_ptr<T> const &o) noexcept
00181 {
00182     if (&o == this)
00183         return;
00184
00185     __drop_ref();
00186     _p = o._p;
00187     __take_ref();
00188 }
00189
00190 void operator = (Null_type) noexcept
00191 {
00192     __drop_ref();
00193     _p = 0;
00194 }
00195
00196 template<typename OT>
00197 Ref_ptr(Ref_ptr<OT, CNT> &&o) noexcept
00198 { _p = o.release(); }
00199
00200 Ref_ptr(Ref_ptr<T> &&o) noexcept
00201 { _p = o.release(); }
00202
00203 template< typename OT >
00204 void operator = (Ref_ptr<OT> &&o) noexcept
00205 {
00206     __drop_ref();
00207     _p = o.release();
00208 }
00209
00210 void operator = (Ref_ptr<T> &&o) noexcept
00211 {
00212     if (&o == this)
00213         return;
00214
00215     __drop_ref();
00216     _p = o.release();
00217 }
00218
00219 explicit operator bool () const noexcept { return _p; }
00220
00221 T *operator -> () const noexcept
00222 { return _p; }
00223
00224 bool operator == (Ref_ptr const &o) const noexcept
00225 { return _p == o._p; }
00226
00227 bool operator != (Ref_ptr const &o) const noexcept
00228 { return _p != o._p; }
00229
00230 bool operator < (Ref_ptr const &o) const noexcept
00231 { return _p < o._p; }
00232

```

```

00233 bool operator <= (Ref_ptr const &o) const noexcept
00234 { return _p <= o._p; }
00235
00236 bool operator > (Ref_ptr const &o) const noexcept
00237 { return _p > o._p; }
00238
00239 bool operator >= (Ref_ptr const &o) const noexcept
00240 { return _p >= o._p; }
00241
00242 bool operator == (T const *o) const noexcept
00243 { return _p == o; }
00244
00245 bool operator < (T const *o) const noexcept
00246 { return _p < o; }
00247
00248 bool operator <= (T const *o) const noexcept
00249 { return _p <= o; }
00250
00251 bool operator > (T const *o) const noexcept
00252 { return _p > o; }
00253
00254 bool operator >= (T const *o) const noexcept
00255 { return _p >= o; }
00256
00257 private:
00258 void __drop_ref() noexcept
00259 {
00260     if (_p)
00261         static_cast<CNT<T>*>(this)->h_drop_ref(_p);
00262 }
00263
00264 void __take_ref() noexcept
00265 {
00266     if (_p)
00267         static_cast<CNT<T>*>(this)->h_take_ref(_p);
00268 }
00269
00270 T *_p;
00271 };
00272
00273
00274 template<typename T, template< typename X > class CNT>
00275 class Weak_ptr
00276 {
00277 private:
00278     struct Null_type;
00279     typedef Ref_ptr<T, CNT> Rp;
00280
00281 public:
00282     Weak_ptr() = default;
00283     Weak_ptr(decltype(nullptr)) : _p(nullptr) {}
00284     // backwards 0 ctor
00285     explicit Weak_ptr(int x) noexcept
00286         L4_DEPRECATED("Use initialization from 'nullptr'")
00287         : _p(nullptr)
00288         { if (x != 0) __builtin_trap(); }
00289
00290     Weak_ptr(Rp const &o) noexcept : _p(o.ptr()) {}
00291     explicit Weak_ptr(T *o) noexcept : _p(o) {}
00292
00293     template<typename OT>
00294     Weak_ptr(Weak_ptr<OT, CNT> const &o) noexcept : _p(o.ptr()) {}
00295
00296     Weak_ptr(Weak_ptr<T, CNT> const &o) noexcept : _p(o._p) {}
00297
00298     Weak_ptr<T, CNT> &operator = (const Weak_ptr<T, CNT> &o) = default;
00299
00300     T *get() const noexcept { return _p; }
00301     T *ptr() const noexcept { return _p; }
00302
00303     T *operator -> () const noexcept { return _p; }
00304     operator Null_type const * () const noexcept
00305     { return reinterpret_cast<Null_type const*>(_p); }
00306
00307 private:
00308     T *_p;
00309 };
00310
00311 template<typename OT, typename T> inline
00312 Ref_ptr<OT> ref_ptr_static_cast(Ref_ptr<T> const &o)
00313 { return ref_ptr(static_cast<OT*>(o.ptr())); }
00314
00315 template< typename T >
00316 inline Ref_ptr<T> ref_ptr(T *t)
00317 { return Ref_ptr<T>(t); }
00318
00319 template< typename T >

```



```

00320 inline Weak_ptr<T> weak_ptr(T *t)
00321 { return Weak_ptr<T>(t); }
00322
00323
00324 class Ref_obj
00325 {
00326 private:
00327     mutable int _ref_cnt;
00328
00329 public:
00330     Ref_obj() : _ref_cnt(0) {}
00331     void add_ref() const noexcept { ++_ref_cnt; }
00332     int remove_ref() const noexcept { return --_ref_cnt; }
00333 };
00334
00335 template< typename T, typename... Args >
00336 Ref_ptr<T>
00337 make_ref_obj(Args &&... args)
00338 { return cxx::Ref_ptr<T>(new T(cxx::forward<Args>(args)...)); }
00339
00340 template<typename T, typename U>
00341 Ref_ptr<T>
00342 dynamic_pointer_cast(Ref_ptr<U> const &p) noexcept
00343 {
00344     // our constructor from a naked pointer increments the counter
00345     return Ref_ptr<T>(dynamic_cast<T *>(p.get()));
00346 }
00347
00348 template<typename T, typename U>
00349 Ref_ptr<T>
00350 static_pointer_cast(Ref_ptr<U> const &p) noexcept
00351 {
00352     // our constructor from a naked pointer increments the counter
00353     return Ref_ptr<T>(static_cast<T *>(p.get()));
00354 }
00355
00356 }

```

## 16.188 l4/cxx/ref\_ptr\_list File Reference

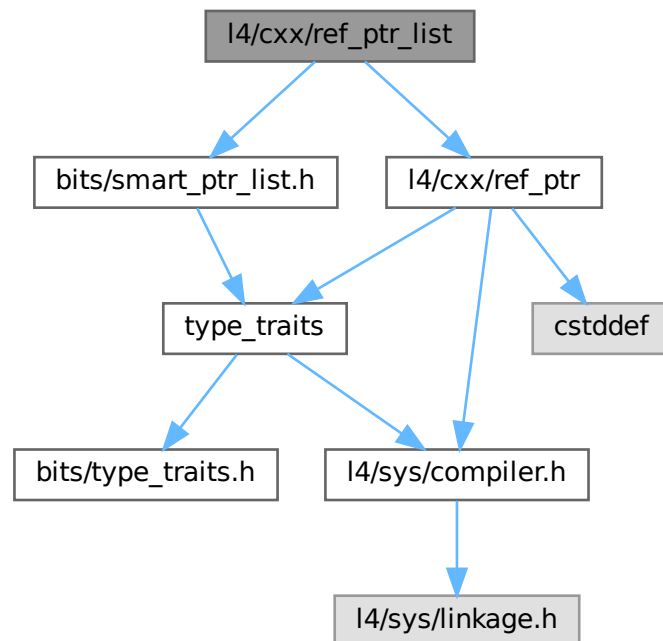
Implementation of a list of ref-ptr-managed objects.

```

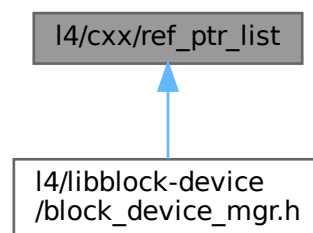
#include <l4/cxx/ref_ptr>
#include "bits/smart_ptr_list.h"

```

Include dependency graph for `ref_ptr_list`:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `cxx::Ref_obj_list_item< T >`  
Item for list linked via `cxx::Ref_ptr` with default reference counting.

## Namespaces

- namespace `cxx`  
Our C++ library.

## Typedefs

- `template<typename T>`  
`using cxx::Ref_ptr_list_item = Bits::Smart_ptr_list_item< T, cxx::Ref_ptr< T> >`  
*Item for list linked with `cxx::Ref_ptr`.*
- `template<typename T>`  
`using cxx::Ref_ptr_list = Bits::Smart_ptr_list< Ref_ptr_list_item< T> >`  
*Single-linked list where elements are connected via a `cxx::Ref_ptr`.*

### 16.188.1 Detailed Description

Implementation of a list of ref-ptr-managed objects.

Definition in file [ref\\_ptr\\_list](#).

## 16.189 ref\_ptr\_list

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * Copyright (C) 2018, 2022 Kernkonzept GmbH.
00008  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00009  *
00010  * This file is distributed under the terms of the GNU General Public
00011  * License, version 2. Please see the COPYING-GPL-2 file for details.
00012  */
00013 #pragma once
00014
00015 #include <l4/cxx/ref_ptr>
00016
00017 #include "bits/smart_ptr_list.h"
00018
00019 namespace cxx {
00020
00022 template <typename T>
00023 using Ref_ptr_list_item = Bits::Smart_ptr_list_item<T, cxx::Ref_ptr<T> >;
00024
00026 template <typename T>
00027 struct Ref_obj_list_item : public Ref_ptr_list_item<T>, public cxx::Ref_obj {};
00028
00032 template <typename T>
00033 using Ref_ptr_list = Bits::Smart_ptr_list<Ref_ptr_list_item<T> >;
00034
00035 }
```

## 16.190 slab\_alloc

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020
```

```

00021 #pragma once
00022
00023 #include <l4/cxx/std_alloc>
00024 #include <l4/cxx/hlist>
00025 #include <l4/sys/consts.h>
00026
00027 namespace cxx {
00028
00040 template< int Obj_size, int Slab_size = L4_PAGE_SIZE,
00041         int Max_free = 2, template<typename A> class Alloc = New_allocator >
00042 class Base_slab
00043 {
00044 private:
00045     struct Free_o
00046     {
00047         Free_o *next;
00048     };
00049
00050 protected:
00051     struct Slab_i;
00052
00053 private:
00054     struct Slab_head : public H_list_item
00055     {
00056         unsigned num_free;
00057         Free_o *free;
00061         Base_slab<Obj_size, Slab_size, Max_free, Alloc> *cache;
00062
00063         inline Slab_head() noexcept : num_free(0), free(0), cache(0)
00064         {}
00065     };
00066
00067     // In an empty or partially filled slab, each free object stores a pointer to
00068     // the next free object. Thus, the size of an object needs to be at least the
00069     // size of a pointer.
00070     static_assert(Obj_size >= sizeof(void *),
00071         "Object size must be at least the size of a pointer.");
00072     static_assert(Obj_size <= Slab_size - sizeof(Slab_head),
00073         "Object_size exceeds slab capability.");
00074
00075 public:
00076     enum
00077     {
00079         object_size      = Obj_size,
00081         slab_size        = Slab_size,
00083         objects_per_slab = (Slab_size - sizeof(Slab_head)) / object_size,
00085         max_free_slabs   = Max_free,
00086     };
00087
00088 protected:
00089     struct Slab_store
00090     {
00091         char _o[slab_size - sizeof(Slab_head)];
00092         Free_o *object(unsigned obj) noexcept
00093         { return reinterpret_cast<Free_o*>(_o + object_size * obj); }
00094     };
00095
00097     struct Slab_i : public Slab_store, public Slab_head
00098     {};
00099
00100 public:
00102     typedef Alloc<Slab_i> Slab_alloc;
00103
00104     typedef void Obj_type;
00105
00106 private:
00108     Slab_alloc _alloc;
00110     unsigned _num_free;
00112     unsigned _num_slabs;
00114     H_list<Slab_i> _full_slabs;
00116     H_list<Slab_i> _partial_slabs;
00118     H_list<Slab_i> _empty_slabs;
00119
00121     void add_slab(Slab_i *s) noexcept
00122     {
00123         s->num_free = objects_per_slab;
00124         s->cache = this;
00125
00126         //L4::cerr << "Slab: " << this << "->add_slab(" << s << ", size="
00127         // << slab_size << ")." << " f=" << s->object(0) << '\n';
00128
00129         // initialize free list
00130         Free_o *f = s->free = s->object(0);
00131         for (unsigned i = 1; i < objects_per_slab; ++i)
00132         {
00133             f->next = s->object(i);
00134             f = f->next;

```

```

00135     }
00136     f->next = 0;
00137
00138     // insert slab into cache's list
00139     _empty_slabs.push_front(s);
00140     ++_num_slabs;
00141     ++_num_free;
00142 }
00143
00144 bool grow() noexcept
00145 {
00146     Slab_i *s = _alloc.alloc();
00147     if (!s)
00148         return false;
00149
00150     new (s, cxx::Nothrow()) Slab_i();
00151
00152     add_slab(s);
00153     return true;
00154 }
00155
00156 void shrink() noexcept
00157 {
00158     if (!_alloc.can_free)
00159         return;
00160
00161     while (!_empty_slabs.empty() && _num_free > max_free_slabs)
00162     {
00163         Slab_i *s = _empty_slabs.front();
00164         _empty_slabs.remove(s);
00165         --_num_free;
00166         --_num_slabs;
00167         _alloc.free(s);
00168     }
00169 }
00170
00171 public:
00172 Base_slab(Slab_alloc const &alloc = Slab_alloc()) noexcept
00173 : _alloc(alloc), _num_free(0), _num_slabs(0), _full_slabs(),
00174   _partial_slabs(), _empty_slabs()
00175 {}
00176
00177 ~Base_slab() noexcept
00178 {
00179     while (!_empty_slabs.empty())
00180     {
00181         Slab_i *o = _empty_slabs.front();
00182         _empty_slabs.remove(o);
00183         _alloc.free(o);
00184     }
00185     while (!_partial_slabs.empty())
00186     {
00187         Slab_i *o = _partial_slabs.front();
00188         _partial_slabs.remove(o);
00189         _alloc.free(o);
00190     }
00191     while (!_full_slabs.empty())
00192     {
00193         Slab_i *o = _full_slabs.front();
00194         _full_slabs.remove(o);
00195         _alloc.free(o);
00196     }
00197 }
00198
00199 void *alloc() noexcept
00200 {
00201     H_list<Slab_i> *free = &_partial_slabs;
00202     if (free->empty())
00203         free = &_empty_slabs;
00204
00205     if (free->empty() && !grow())
00206         return 0;
00207
00208     Slab_i *s = free->front();
00209     Free_o *o = s->free;
00210     s->free = o->next;
00211
00212     if (free == &_empty_slabs)
00213     {
00214         _empty_slabs.remove(s);
00215         --_num_free;
00216     }
00217
00218     --(s->num_free);
00219
00220     if (!s->free)
00221     {

```

```

00241 _partial_slabs.remove(s);
00242 _full_slabs.push_front(s);
00243 }
00244 else if (free == &_empty_slabs)
00245 _partial_slabs.push_front(s);
00246
00247 //L4::cerr << this << "->alloc(): " << o << ", of " << s << '\n';
00248
00249 return o;
00250 }
00251
00257 void free(void *_o) noexcept
00258 {
00259     if (!_o)
00260         return;
00261
00262     unsigned long addr = reinterpret_cast<unsigned long>(_o);
00263
00264     // find out the slab the object is in
00265     addr = (addr / slab_size) * slab_size;
00266     Slab_i *s = reinterpret_cast<Slab_i*>(addr);
00267
00268     if (s->cache != this)
00269         return;
00270
00271     Free_o *o = reinterpret_cast<Free_o*>(_o);
00272
00273     o->next = s->free;
00274     s->free = o;
00275
00276     bool was_full = false;
00277
00278     if (!s->num_free)
00279     {
00280         _full_slabs.remove(s);
00281         was_full = true;
00282     }
00283
00284     ++(s->num_free);
00285
00286     if (s->num_free == objects_per_slab)
00287     {
00288         if (!was_full)
00289             _partial_slabs.remove(s);
00290
00291         _empty_slabs.push_front(s);
00292         ++_num_free;
00293         if (_num_free > max_free_slabs)
00294             shrink();
00295
00296         was_full = false;
00297     }
00298     else if (was_full)
00299         _partial_slabs.push_front(s);
00300
00301     //L4::cerr << this << "->free(" << _o << "): of " << s << '\n';
00302 }
00303
00310 unsigned total_objects() const noexcept
00311 { return _num_slabs * objects_per_slab; }
00312
00319 unsigned free_objects() const noexcept
00320 {
00321     unsigned count = 0;
00322
00323     /* count partial slabs first */
00324     for (typename H_list<Slab_i>::Const_iterator s = _partial_slabs.begin();
00325          s != _partial_slabs.end(); ++s)
00326         count += s->num_free;
00327
00328     /* add empty slabs */
00329     count += _num_free * objects_per_slab;
00330
00331     return count;
00332 }
00333 };
00334
00344 template<typename Type, int Slab_size = L4_PAGESIZE,
00345         int Max_free = 2, template<typename A> class Alloc = New_allocator >
00346 class Slab : public Base_slab<sizeof(Type), Slab_size, Max_free, Alloc>
00347 {
00348 private:
00349     typedef Base_slab<sizeof(Type), Slab_size, Max_free, Alloc> Base_type;
00350 public:
00351
00352     typedef Type Obj_type;
00353

```

```

00354     Slab(typename Base_type::Slab_alloc const &alloc
00355         = typename Base_type::Slab_alloc()) noexcept
00356         : Base_slab<sizeof(Type), Slab_size, Max_free, Alloc>(alloc) {}
00357
00358
00366     Type *alloc() noexcept
00367     {
00368         return reinterpret_cast<Type *>(Base_type::alloc());
00369     }
00370
00377     void free(Type *o) noexcept
00378     { Base_slab<sizeof(Type), Slab_size, Max_free, Alloc>::free(o); }
00379 };
00380
00381
00397 template< int Obj_size, int Slab_size = L4_PAGESIZE,
00398           int Max_free = 2, template<typename A> class Alloc = New_allocator >
00399 class Base_slab_static
00400 {
00401 private:
00402     typedef Base_slab<Obj_size, Slab_size, Max_free, Alloc> _A;
00403     static _A _a;
00404 public:
00405     typedef void Obj_type;
00406     enum
00407     {
00409         object_size      = Obj_size,
00411         slab_size        = Slab_size,
00413         objects_per_slab = _A::objects_per_slab,
00415         max_free_slabs   = Max_free,
00416     };
00417
00423     void *alloc() noexcept { return _a.alloc(); }
00424
00431     void free(void *p) noexcept { _a.free(p); }
00432
00441     unsigned total_objects() const noexcept { return _a.total_objects(); }
00442
00451     unsigned free_objects() const noexcept { return _a.free_objects(); }
00452 };
00453
00454
00455 template< int _O, int _S, int _M, template<typename A> class Alloc >
00456 typename Base_slab_static<_O,_S,_M,Alloc>::_A
00457     Base_slab_static<_O,_S,_M,Alloc>::_a;
00458
00474 template<typename Type, int Slab_size = L4_PAGESIZE,
00475           int Max_free = 2, template<typename A> class Alloc = New_allocator >
00476 class Slab_static
00477 : public Base_slab_static<sizeof(Type), Slab_size, Max_free, Alloc>
00478 {
00479 public:
00480
00481     typedef Type Obj_type;
00489     Type *alloc() noexcept
00490     {
00491         return reinterpret_cast<Type *>(
00492             Base_slab_static<sizeof(Type), Slab_size, Max_free, Alloc>::alloc());
00493     }
00494 };
00495
00496

```

## 16.191 slist

```

00001 // vi:set ft=c++: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2011 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */

```

```

00019
00020 #pragma once
00021
00022 #include "bits/list_basics.h"
00023
00024 namespace cxx {
00025
00026 class S_list_item
00027 {
00028 public:
00029     S_list_item() : _n(0) {}
00030     // BSS allocation
00031     explicit S_list_item(bool) {}
00032
00033 private:
00034     template<typename T, typename P> friend class S_list;
00035     template<typename T, typename P> friend class S_list_tail;
00036     template<typename T, typename X> friend struct Bits::Basic_list_policy;
00037
00038     S_list_item(S_list_item const &);
00039     void operator = (S_list_item const &);
00040
00041     S_list_item *_n;
00042 };
00043
00050 template< typename T, typename POLICY = Bits::Basic_list_policy< T, S_list_item > >
00051 class S_list : public Bits::Basic_list<POLICY>
00052 {
00053     S_list(S_list const &) = delete;
00054     void operator = (S_list const &) = delete;
00055
00056 private:
00057     typedef typename Bits::Basic_list<POLICY> Base;
00058
00059 public:
00060     typedef typename Base::Iterator Iterator;
00061
00062     S_list(S_list &&o) : Base(static_cast<Base&&>(o)) {}
00063
00064     S_list &operator = (S_list &&o)
00065     {
00066         Base::operator = (static_cast<Base&&>(o));
00067         return *this;
00068     }
00069
00070     // BSS allocation
00071     explicit S_list(bool x) : Base(x) {}
00072
00073     S_list() : Base() {}
00074
00076 void add(T *e)
00077 {
00078     e->_n = this->_f;
00079     this->_f = e;
00080 }
00081
00082 template< typename CAS >
00083 void add(T *e, CAS const &c)
00084 {
00085     do
00086     {
00087         e->_n = this->_f;
00088     }
00089     while (!c(&this->_f, e->_n, e));
00090 }
00091
00093 void push_front(T *e) { add(e); }
00094
00100 T *pop_front()
00101 {
00102     T *r = this->front();
00103     if (this->_f)
00104         this->_f = this->_f->_n;
00105     return r;
00106 }
00107
00108 void insert(T *e, Iterator const &pred)
00109 {
00110     S_list_item *p = *pred;
00111     e->_n = p->_n;
00112     p->_n = e;
00113 }
00114
00115 static void insert_before(T *e, Iterator const &succ)
00116 {
00117     S_list_item **x = Base::__get_internal(succ);
00118

```



```

00119     e->_n = *x;
00120     *x = e;
00121 }
00122
00123 static void replace(Iterator const &p, T*e)
00124 {
00125     S_list_item **x = Base::__get_internal(p);
00126     e->_n = (*x)->_n;
00127     *x = e;
00128 }
00129
00130 static Iterator erase(Iterator const &e)
00131 {
00132     S_list_item **x = Base::__get_internal(e);
00133     *x = (*x)->_n;
00134     return e;
00135 }
00136
00137 };
00138
00139
00140 template< typename T >
00141 class S_list_bss : public S_list<T>
00142 {
00143 public:
00144     S_list_bss() : S_list<T>(true) {}
00145 };
00146
00147 template< typename T, typename POLICY = Bits::Basic_list_policy< T, S_list_item > >
00148 class S_list_tail : public S_list<T, POLICY>
00149 {
00150 private:
00151     typedef S_list<T, POLICY> Base;
00152     void add(T *e) = delete;
00153
00154 public:
00155     using Iterator = typename Base::Iterator;
00156     S_list_tail() : Base(), _tail(&this->_f) {}
00157
00158     S_list_tail(S_list_tail &t)
00159     : Base(static_cast<Base&&>(t)), _tail(t.empty() ? &this->_f : t._tail)
00160     {
00161         t._tail = &t._f;
00162     }
00163
00164     S_list_tail &operator = (S_list_tail &t)
00165     {
00166         if (&t == this)
00167             return *this;
00168
00169         Base::operator = (static_cast<Base &&>(t));
00170         _tail = t.empty() ? &this->_f : t._tail;
00171         t._tail = &t._f;
00172         return *this;
00173     }
00174
00175     void push_front(T *e)
00176     {
00177         if (Base::empty())
00178             _tail = &e->_n;
00179
00180         Base::push_front(e);
00181     }
00182
00183     void push_back(T *e)
00184     {
00185         e->_n = 0;
00186         *_tail = e;
00187         _tail = &e->_n;
00188     }
00189
00190     void clear()
00191     {
00192         Base::clear();
00193         _tail = &this->_f;
00194     }
00195
00196     void append(S_list_tail &o)
00197     {
00198         T *x = o.front();
00199         *_tail = x;
00200         if (x)
00201             _tail = o._tail;
00202         o.clear();
00203     }
00204
00205     T *pop_front()

```

```

00206 {
00207     T *t = Base::pop_front();
00208     if (t && Base::empty())
00209         _tail = &this->_f;
00210     return t;
00211 }
00212
00213 private:
00214     static void insert(T *e, Iterator const &pred);
00215     static void insert_before(T *e, Iterator const &succ);
00216     static void replace(Iterator const &p, T*e);
00217     static Iterator erase(Iterator const &e);
00218
00219 private:
00220     S_list_item **_tail;
00221 };
00222
00223 }

```

## 16.192 static\_container

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002
00003 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00004 /*
00005  * Copyright (C) 2012-2013 Technische Universität Dresden.
00006  * Copyright (C) 2016-2017, 2020 Kernkonzept GmbH.
00007  */
00008
00009 #pragma once
00010
00011 #include <l4/cxx/type_traits>
00012 #include <stddef.h>
00013
00014 namespace cxx {
00015
00016 template< typename T >
00017 class Static_container
00018 {
00019 private:
00020     struct X : T
00021     {
00022         void *operator new (size_t, void *p) noexcept { return p; }
00023         void operator delete (void *) {}
00024         X() = default;
00025         template<typename ...Args>
00026         X(Args && ...a) : T(cxx::forward<Args>(a)...) {}
00027     };
00028
00029 public:
00030     void operator = (Static_container const &) = delete;
00031     Static_container(Static_container const &) = delete;
00032     Static_container() = default;
00033
00034     T *get() { return reinterpret_cast<X*>(_s); }
00035     T *operator -> () { return get(); }
00036     T &operator * () { return *get(); }
00037     operator T* () { return get(); }
00038
00039     void construct()
00040     { new (reinterpret_cast<void*>(_s)) X; }
00041
00042     template< typename ...Args >
00043     void construct(Args && ...args)
00044     { new (reinterpret_cast<void*>(_s)) X(cxx::forward<Args>(args)...) ; }
00045
00046 private:
00047     char _s[sizeof(X)] __attribute__((aligned(__alignof(X))));
00048 };
00049
00050 }
00051
00052

```

## 16.193 static\_vector

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002
00003 #pragma once

```

```

00004
00005 #include "type_traits"
00006
00007 namespace cxx {
00008
00015 template<typename T, typename IDX = unsigned>
00016 class static_vector
00017 {
00018 private:
00019     template<typename X, typename IDX2> friend class static_vector;
00020     T *_v;
00021     IDX _l;
00022
00023 public:
00024     typedef T value_type;
00025     typedef IDX index_type;
00026
00027     static_vector() = default;
00028     static_vector(value_type *v, index_type length) : _v(v), _l(length) {}
00029
00030     template<typename Z,
00031             typename = enable_if_t<is_same<remove_extent_t<Z>, T>::value>
00032     constexpr static_vector(Z &v) : _v(v), _l(array_size(v))
00033     {}
00034
00036     template<typename X,
00037             typename = enable_if_t<is_convertible<X, T>::value>
00038     static_vector(static_vector<X, IDX> const &o) : _v(o._v), _l(o._l) {}
00039
00040     index_type size() const { return _l; }
00041     bool empty() const { return _l == 0; }
00042
00043     value_type &operator [] (index_type idx) { return _v[idx]; }
00044     value_type const &operator [] (index_type idx) const { return _v[idx]; }
00045
00046     value_type *begin() { return _v; }
00047     value_type *end() { return _v + _l; }
00048     value_type const *begin() const { return _v; }
00049     value_type const *end() const { return _v + _l; }
00050     value_type const *cbegin() const { return _v; }
00051     value_type const *cend() const { return _v + _l; }
00052
00054     index_type index(value_type const *o) const { return o - _v; }
00055     index_type index(value_type const &o) const { return &o - _v; }
00056 };
00057
00058 }

```

## 16.194 std\_alloc

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020
00021 #pragma once
00022
00023 #include <stddef.h>
00024 namespace cxx {
00030 class Nothrow {};
00031 }
00032
00039 inline void *operator new (size_t, void *mem, cxx::Nothrow const &) noexcept
00040 { return mem; }
00041
00046 void *operator new (size_t, cxx::Nothrow const &) noexcept;
00047
00053 void operator delete (void *, cxx::Nothrow const &) noexcept;

```

```

00054
00055 namespace cxx {
00056
00057
00066 template< typename _Type >
00067 class New_allocator
00068 {
00069 public:
00070     enum { can_free = true };
00071
00072     New_allocator() noexcept {}
00073     New_allocator(New_allocator const &) noexcept {}
00074
00075     ~New_allocator() noexcept {}
00076
00077     _Type *alloc() noexcept
00078     { return static_cast<_Type*> (::operator new(sizeof (_Type), cxx::Nothrow())); }
00079
00080     void free(_Type *t) noexcept
00081     { ::operator delete(t, cxx::Nothrow()); }
00082 };
00083
00084 }
00085

```

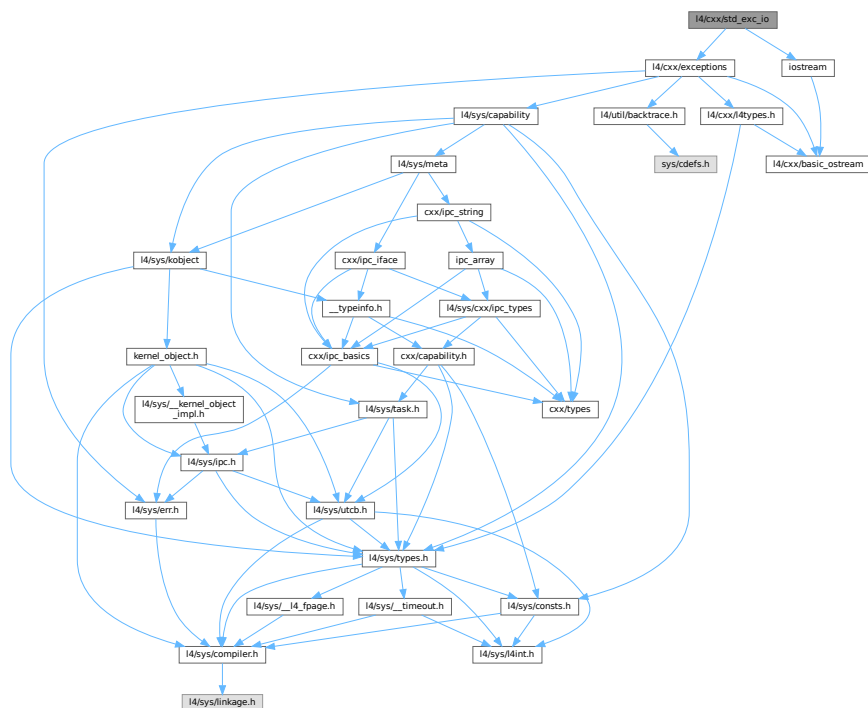
## 16.195 l4/cxx/std\_exc\_io File Reference

Base exceptions std stream operator.

```
#include <l4/cxx/exceptions>
```

```
#include <iostream>
```

Include dependency graph for std\_exc\_io:



### 16.195.1 Detailed Description

Base exceptions std stream operator.

Definition in file [std\\_exc\\_io](#).

## 16.196 std\_exc\_io

[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020 #pragma once
00021
00022 #include <l4/cxx/exceptions>
00023 #include <iostream>
00024
00025 inline
00026 std::ostream &
00027 operator << (std::ostream &o, L4::Base_exception const &e)
00028 {
00029     o << "Exception: " << e.str() << ", backtrace ...\n";
00030     for (int i = 0; i < e.frame_count(); ++i)
00031         o << (void *) (e.pc_array()[i]) << '\n';
00032     return o;
00033 }
00034
00035 inline
00036 std::ostream &
00037 operator << (std::ostream &o, L4::Runtime_error const &e)
00038 {
00039     o << "Exception: " << e.str() << ": ";
00040     if (e.extra_str())
00041         o << e.extra_str() << ": ";
00042     o << "backtrace ...\n";
00043     for (int i = 0; i < e.frame_count(); ++i)
00044         o << (void *) (e.pc_array()[i]) << '\n';
00045     return o;
00046 }
00047 }
```

## 16.197 std\_ops

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020 #pragma once
00021
00022 namespace cxx {
00023
00024 template< typename Obj >
00025 struct Lt_functor
00026 {
```

```

00031     bool operator () (Obj const &l, Obj const &r) const
00032     { return l < r; }
00033 };
00034
00035 };
00036

```

## 16.198 string

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019
00023 #pragma once
00024
00025 #include <l4/cxx/minmax>
00026 #include <l4/cxx/basic_ostream>
00027
00028
00029 namespace cxx {
00030
00041 class String
00042 {
00043 public:
00044
00046     typedef char const *Index;
00047
00049     String(char const *s) noexcept : _start(s), _len(__builtin_strlen(s)) {}
00051     String(char const *s, unsigned long len) noexcept : _start(s), _len(len) {}
00052
00059     String(char const *s, char const *e) noexcept : _start(s), _len(e - s) {}
00060
00062     String() : _start(0), _len(0) {}
00063
00065     Index start() const { return _start; }
00067     Index end() const { return _start + _len; }
00069     int len() const { return _len; }
00070
00072     void start(char const *s) { _start = s; }
00074     void len(unsigned long len) { _len = len; }
00076     bool empty() const { return !_len; }
00077
00079     String head(Index end) const
00080     {
00081         if (end < _start)
00082             return String();
00083
00084         if (eof(end))
00085             return *this;
00086
00087         return String(_start, end - _start);
00088     }
00089
00091     String head(unsigned long end) const
00092     { return head(start() + end); }
00093
00095     String substr(unsigned long idx, unsigned long len = ~0UL) const
00096     {
00097         if (idx >= _len)
00098             return String(end(), 0UL);
00099
00100         return String(_start + idx, cxx::min(len, _len - idx));
00101     }
00102
00104     String substr(char const *start, unsigned long len = 0) const
00105     {
00106         if (start >= _start && !eof(start))
00107             {

```

```

00108 unsigned long nlen = _start + _len - start;
00109 if (len != 0)
00110     nlen = cxx::min(nlen, len);
00111 return String(start, nlen);
00112 }
00113
00114     return String(end(), OUL);
00115 }
00116
00117 template< typename F >
00118 char const *find_match(F &&match) const
00119 {
00120     String::Index s = _start;
00121     while (1)
00122     {
00123         if (eof(s))
00124             return s;
00125         if (match(*s))
00126             return s;
00127         ++s;
00128     }
00129 }
00130
00131 char const *find(char const *c) const
00132 { return find(c, start()); }
00133
00134 char const *find(int c) const
00135 { return find(c, start()); }
00136
00137 char const *rfind(char const *c) const
00138 {
00139     if (!_len)
00140         return end();
00141     char const *p = end();
00142     --p;
00143     while (p >= _start)
00144     {
00145         if (*p == *c)
00146             return p;
00147         --p;
00148     }
00149     return end();
00150 }
00151
00152 Index starts_with(cxx::String const &c) const
00153 {
00154     unsigned long i;
00155     for (i = 0; i < c._len && i < _len; ++i)
00156         if (_start[i] != c[i])
00157             return 0;
00158     return i == c._len ? start() + i : 0;
00159 }
00160
00161 char const *find(int c, char const *s) const
00162 {
00163     if (s < _start)
00164         return end();
00165     while (1)
00166     {
00167         if (eof(s))
00168             return s;
00169         if (*s == c)
00170             return s;
00171         ++s;
00172     }
00173 }
00174
00175 char const *find(char const *c, char const *s) const
00176 {
00177     if (s < _start)
00178         return end();
00179     while (1)
00180     {
00181         if (eof(s))
00182             return s;
00183         for (char const *x = c; *x; ++x)
00184             if (*s == *x)
00185                 return s;
00186     }
00187 }

```

```

00215
00216     ++s;
00217   }
00218 }
00219
00221 char const &operator [] (unsigned long idx) const { return _start[idx]; }
00223 char const &operator [] (int idx) const { return _start[idx]; }
00225 char const &operator [] (Index idx) const { return *idx; }
00226
00228 bool eof(char const *s) const { return s >= _start + _len || !*s; }
00229
00238 template<typename INT>
00239 int from_dec(INT *v) const
00240 {
00241     *v = 0;
00242     Index c;
00243     for (c = start(); !eof(c); ++c)
00244     {
00245         unsigned char n;
00246         if (*c >= '0' && *c <= '9')
00247             n = *c - '0';
00248         else
00249             return c - start();
00250
00251         *v *= 10;
00252         *v += n;
00253     }
00254     return c - start();
00255 }
00256
00267 template<typename INT>
00268 int from_hex(INT *v) const
00269 {
00270     *v = 0;
00271     unsigned shift = 0;
00272     Index c;
00273     for (c = start(); !eof(c); ++c)
00274     {
00275         shift += 4;
00276         if (shift > sizeof(INT) * 8)
00277             return -1;
00278         unsigned char n;
00279         if (*c >= '0' && *c <= '9')
00280             n = *c - '0';
00281         else if (*c >= 'A' && *c <= 'F')
00282             n = *c - 'A' + 10;
00283         else if (*c >= 'a' && *c <= 'f')
00284             n = *c - 'a' + 10;
00285         else
00286             return c - start();
00287
00288         *v <<= 4;
00289         *v |= n;
00290     }
00291     return c - start();
00292 }
00293
00295 bool operator == (String const &o) const
00296 {
00297     if (len() != o.len())
00298         return false;
00299
00300     for (unsigned long i = 0; i < _len; ++i)
00301         if (_start[i] != o._start[i])
00302             return false;
00303
00304     return true;
00305 }
00306
00308 bool operator != (String const &o) const
00309 { return ! (operator == (o)); }
00310
00311 private:
00312     char const *_start;
00313     unsigned long _len;
00314 };
00315
00316 }
00317
00319 inline
00320 L4::BasicOStream &operator << (L4::BasicOStream &s, cxx::String const &str)
00321 {
00322     s.write(str.start(), str.len());
00323     return s;
00324 }

```

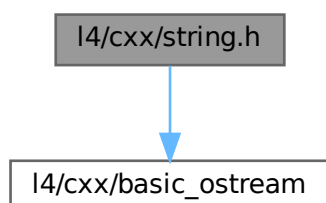


## 16.199 l4/cxx/string.h File Reference

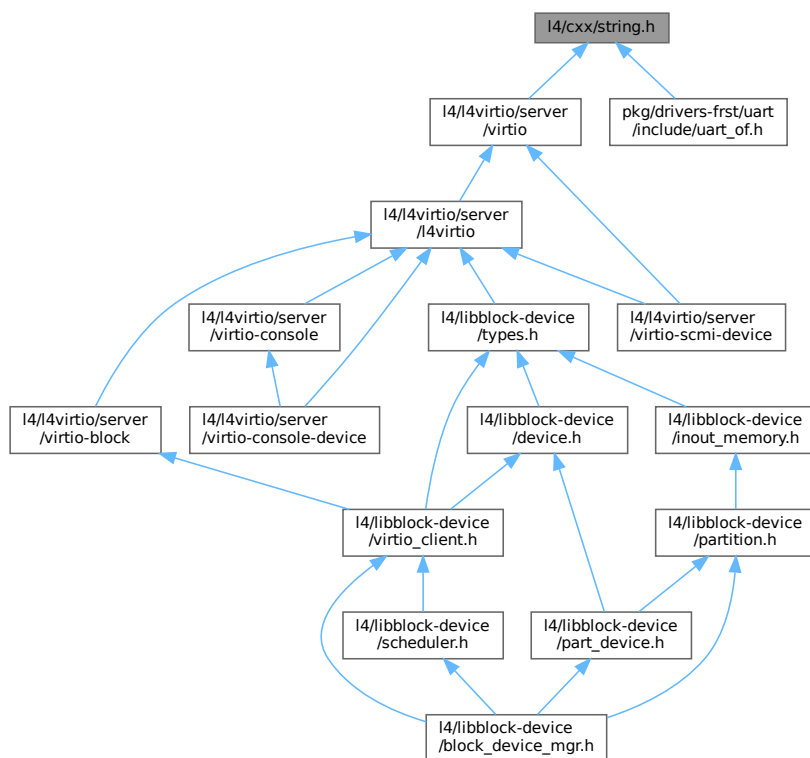
String.

```
#include <l4/cxx/basic_ostream>
```

Include dependency graph for string.h:



This graph shows which files directly or indirectly include this file:



### Data Structures

- class [L4::String](#)  
A null-terminated string container class.

## Namespaces

- namespace [L4](#)  
*[L4](#) low-level kernel interface.*

### 16.199.1 Detailed Description

String.

Definition in file [string.h](#).

## 16.200 string.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00007  *                      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 #include <l4/cxx/basic_ostream>
00026
00027 namespace L4 {
00028
00033     class String
00034     {
00035     public:
00036         String(char const *str = "") : _str(str)
00037         {}
00038
00039         unsigned length() const
00040         {
00041             unsigned l;
00042             for (l = 0; _str[l]; l++)
00043                 ;
00044             return l;
00045         }
00046
00047         char const *p_str() const { return _str; }
00048
00049     private:
00050         char const *_str;
00051     };
00052 }
00053
00054 inline
00055 L4::BasicOStream &operator << (L4::BasicOStream &o, L4::String const &s)
00056 {
00057     o << s.p_str();
00058     return o;
00059 }

```

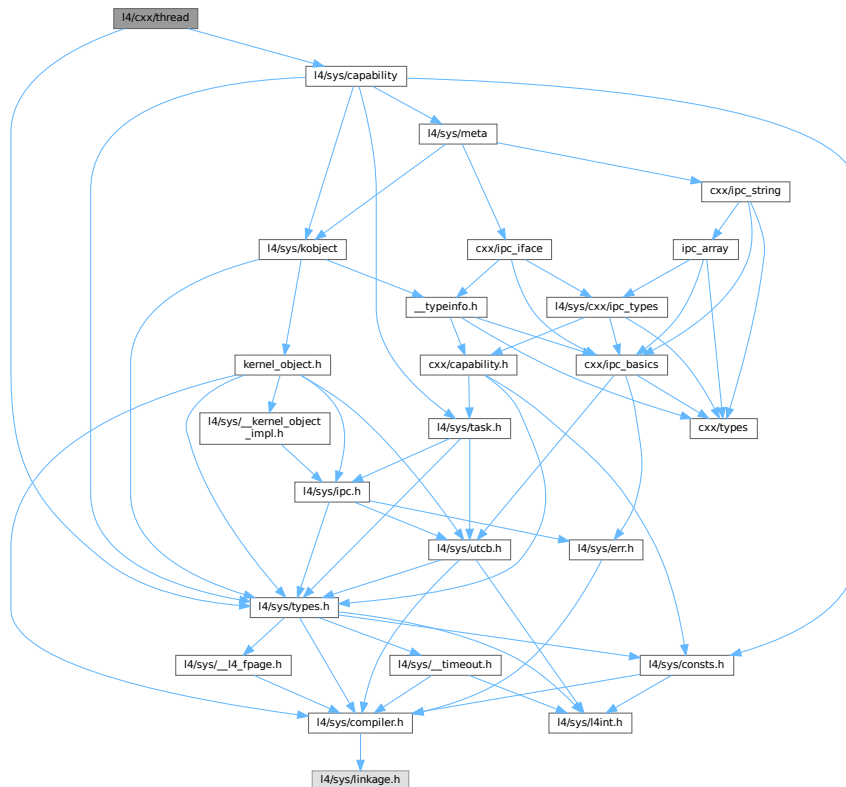
## 16.201 l4/cxx/thread File Reference

Thread implementation.

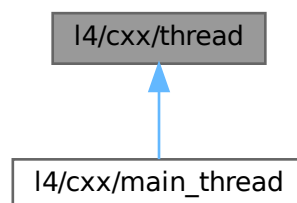
```
#include <l4/sys/capability>
```

```
#include <l4/sys/types.h>
```

Include dependency graph for thread:



This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace **cxx**  
Our C++ library.

## 16.201.1 Detailed Description

Thread implementation.

Definition in file [thread](#).

## 16.202 thread

[Go to the documentation of this file.](#)

```

00001
00002 /*
00003  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU Lesser General Public License 2.1.
00007  * Please see the COPYING-LGPL-2.1 file for details.
00008  */
00009
00010 #ifndef CXX_THREAD_H__
00011 #define CXX_THREAD_H__
00012
00013 #include <l4/sys/capability>
00014 #include <l4/sys/types.h>
00015
00016 namespace cxx {
00017
00018     class Thread
00019     {
00020     public:
00021
00022         enum State
00023         {
00024             Dead      = 0,
00025             Running   = 1,
00026             Stopped   = 2,
00027         };
00028
00029         Thread(bool initiate);
00030         Thread(void *stack);
00031         Thread(void *stack, L4::Cap<L4::Thread> const &cap);
00032         virtual ~Thread();
00033         void execute() asm ("L4_Thread_execute");
00034         virtual void run() = 0;
00035         virtual void shutdown() asm ("L4_Thread_shutdown");
00036         void start();
00037         void stop();
00038
00039         L4::Cap<L4::Thread> self() const throw()
00040         { return _cap; }
00041
00042         State state() const
00043         { return _state; }
00044
00045         static void start_cxx_thread(Thread *_this)
00046             asm ("L4_Thread_start_cxx_thread");
00047
00048         static void kill_cxx_thread(Thread *_this)
00049             asm ("L4_Thread_kill_cxx_thread");
00050
00051         static void set_pager(L4::Cap<void> const &p) throw()
00052         { _pager = p; }
00053
00054     private:
00055         int create();
00056
00057         L4::Cap<L4::Thread> _cap;
00058         State _state;
00059
00060     protected:
00061         void *_stack;
00062
00063     private:
00064         static L4::Cap<void> _pager;
00065         static L4::Cap<void> _master;
00066     };
00067
00068 };
00069
00070 #endif /* CXX_THREAD_H__ */

```

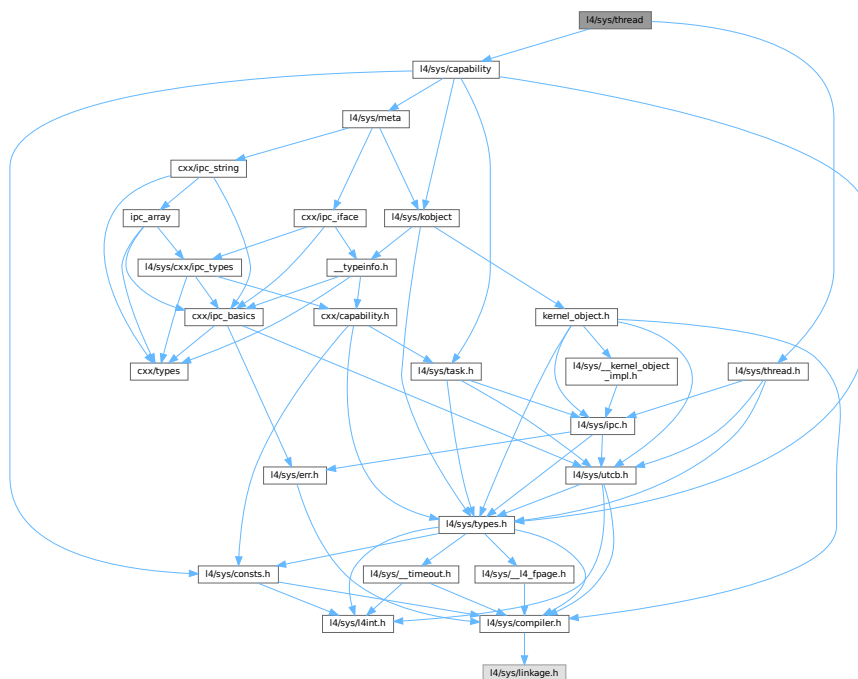
## 16.203 I4/sys/thread File Reference

## Common thread related definitions.

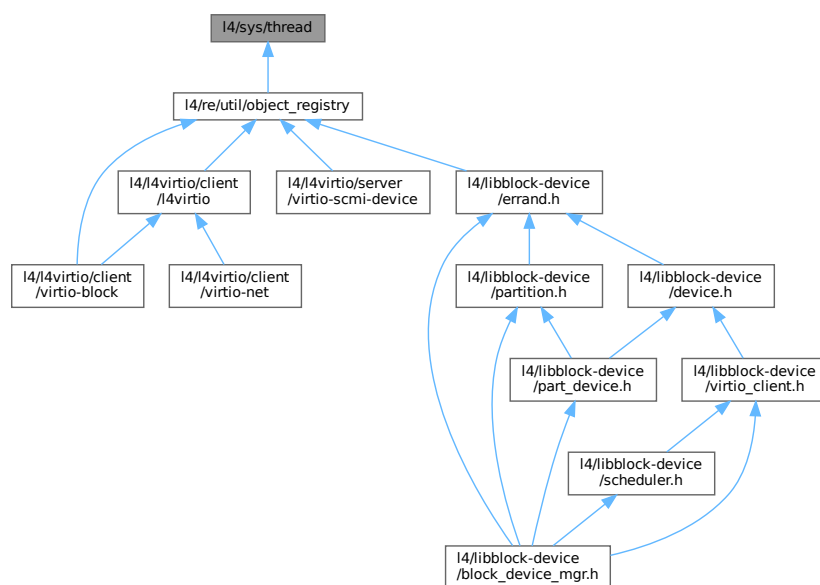
```
#include <linux/sys/capability>
```

```
#include <pthread.h>
```

Include dependency graph for thread:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::Thread](#)  
*C++ L4 kernel thread interface, see [Thread](#) for the C interface.*
- class [L4::Thread::Attr](#)  
*Thread attributes used for [control\(\)](#).*
- class [L4::Thread::Modify\\_senders](#)  
*Class wrapping a list of rules which modify the sender label of IPC messages inbound to this thread.*

## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

### 16.203.1 Detailed Description

Common thread related definitions.

Definition in file [thread](#).

## 16.204 thread

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include <l4/sys/capability>
00028 #include <l4/sys/thread.h>
00029
00030 namespace L4 {
00031
00059 class Thread :
00060     public Kobject_t<Thread, Kobject, L4_PROTO_THREAD,
00061         Type_info::Demand_t<1> >
00062 {
00063 public:
00090     l4_msgtag_t ex_regs(l4_addr_t ip, l4_addr_t sp,
00091         l4_umword_t flags,
00092         l4_utcb_t *utcb = l4_utcb()) noexcept
00093     { return l4_thread_ex_regs_u(cap(), ip, sp, flags, utcb); }
00094
00123     l4_msgtag_t ex_regs(l4_addr_t *ip, l4_addr_t *sp,
00124         l4_umword_t *flags,
00125         l4_utcb_t *utcb = l4_utcb()) noexcept
00126     { return l4_thread_ex_regs_ret_u(cap(), ip, sp, flags, utcb); }
00127
00128
```

```

00141 class Attr
00142 {
00143 private:
00144     friend class L4::Thread;
00145     l4_utcb_t *_u;
00146
00147 public:
00155     explicit Attr(l4_utcb_t *utcb = l4_utcb()) noexcept : _u(utcb)
00156     { l4_thread_control_start_u(utcb); }
00157
00165     void pager(Cap<void> const &pager) noexcept
00166     { l4_thread_control_pager_u(pager.cap(), _u); }
00167
00174     Cap<void> pager() noexcept
00175     { return Cap<void>(l4_utcb_mr_u(_u)->mr[1]); }
00176
00184     void exc_handler(Cap<void> const &exc_handler) noexcept
00185     { l4_thread_control_exc_handler_u(exc_handler.cap(), _u); }
00186
00193     Cap<void> exc_handler() noexcept
00194     { return Cap<void>(l4_utcb_mr_u(_u)->mr[2]); }
00195
00217     void bind(l4_utcb_t *thread_utcb, Cap<Task> const &task) noexcept
00218     { l4_thread_control_bind_u(thread_utcb, task.cap(), _u); }
00219
00223     void alien(int on) noexcept
00224     { l4_thread_control_alien_u(_u, on); }
00225
00231     void ux_host_syscall(int on) noexcept
00232     { l4_thread_control_ux_host_syscall_u(_u, on); }
00233
00234 };
00235
00249 l4_msgtag_t control(Attr const &attr) noexcept
00250 { return l4_thread_control_commit_u(cap(), attr._u); }
00251
00259 l4_msgtag_t switch_to(l4_utcb_t *utcb = l4_utcb()) noexcept
00260 { return l4_thread_switch_u(cap(), utcb); }
00261
00270 l4_msgtag_t stats_time(l4_kernel_clock_t *us,
00271                        l4_utcb_t *utcb = l4_utcb()) noexcept
00272 { return l4_thread_stats_time_u(cap(), us, utcb); }
00273
00289 l4_msgtag_t vcpu_resume_start(l4_utcb_t *utcb = l4_utcb()) noexcept
00290 { return l4_thread_vcpu_resume_start_u(utcb); }
00291
00330 l4_msgtag_t vcpu_resume_commit(l4_msgtag_t tag,
00331                                l4_utcb_t *utcb = l4_utcb()) noexcept
00332 { return l4_thread_vcpu_resume_commit_u(cap(), tag, utcb); }
00333
00354 l4_msgtag_t vcpu_control(l4_addr_t vcpu_state, l4_utcb_t *utcb = l4_utcb())
00355     noexcept
00356 { return l4_thread_vcpu_control_u(cap(), vcpu_state, utcb); }
00357
00387 l4_msgtag_t vcpu_control_ext(l4_addr_t ext_vcpu_state,
00388                              l4_utcb_t *utcb = l4_utcb()) noexcept
00389 { return l4_thread_vcpu_control_ext_u(cap(), ext_vcpu_state, utcb); }
00390
00414 l4_msgtag_t register_del_irq(Cap<Irq> irq, l4_utcb_t *u = l4_utcb()) noexcept
00415 { return l4_thread_register_del_irq_u(cap(), irq.cap(), u); }
00416
00435 class Modify_senders
00436 {
00437 private:
00438     friend class Thread;
00439     l4_utcb_t *utcb;
00440     unsigned cnt;
00441
00442 public:
00443     explicit Modify_senders(l4_utcb_t *u = l4_utcb()) noexcept
00444     : utcb(u), cnt(1)
00445     {
00446         l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_MODIFY_SENDER_OP;
00447     }
00448
00468 int add(l4_umword_t match_mask, l4_umword_t match,
00469         l4_umword_t del_bits, l4_umword_t add_bits) noexcept
00470 {
00471     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00472     if (cnt >= L4_UTCB_GENERIC_DATA_SIZE - 4)
00473         return -L4_ENOMEM;
00474     m->mr[cnt++] = match_mask;
00475     m->mr[cnt++] = match;
00476     m->mr[cnt++] = del_bits;
00477     m->mr[cnt++] = add_bits;
00478     return 0;
00479 }

```

```

00480     };
00481
00501     l4_msgtag_t modify_senders(Modify_senders const &todo) noexcept
00502     {
00503         return l4_ipc_call(cap(), todo.utcb, l4_msgtag(L4_PROTO_THREAD, todo.cnt, 0, 0), L4_IPC_NEVER);
00504     }
00505 };
00506 }

```

## 16.205 type\_list

```

00001 // vi:set ft=c++: -- Mode: C++ --
00002 #pragma once
00003
00004 /*
00005  * (c) 2012 Alexander Warg <warg@os.inf.tu-dresden.de>,
00006  *     economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU General Public License 2.
00010  * Please see the COPYING-GPL-2 file for details.
00011  *
00012  * As a special exception, you may use this file as part of a free software
00013  * library without restriction. Specifically, if other files instantiate
00014  * templates or use macros or inline functions from this file, or you compile
00015  * this file and link it with other files to produce an executable, this
00016  * file does not by itself cause the resulting executable to be covered by
00017  * the GNU General Public License. This exception does not however
00018  * invalidate any other reasons why the executable file might be covered by
00019  * the GNU General Public License.
00020  */
00021
00022 #include "type_traits"
00023
00024 namespace cxx {
00025
00026     template<typename ...T >
00027     struct type_list;
00028
00029     template<>
00030     struct type_list<>
00031     {
00032         typedef false_type head;
00033         typedef false_type tail;
00034     };
00035
00036     template<typename HEAD, typename ...TAIL>
00037     struct type_list<HEAD, TAIL...>
00038     {
00039         typedef HEAD head;
00040         typedef type_list<TAIL...> tail;
00041     };
00042
00043     template<typename TYPELIST, template <typename T> class PREDICATE>
00044     struct find_type;
00045
00046     template<template <typename T> class PREDICATE>
00047     struct find_type<type_list<>, PREDICATE>
00048     {
00049         typedef false_type type;
00050     };
00051
00052     template<typename TYPELIST, template <typename T> class PREDICATE>
00053     struct find_type
00054     {
00055         typedef typename conditional<PREDICATE<typename TYPELIST::head>::value,
00056                                     typename TYPELIST::head,
00057                                     typename find_type<typename TYPELIST::tail, PREDICATE>::type>::type
00058         type;
00059     };
00060
00061     template<typename TYPELIST, template <typename T> class PREDICATE>
00062     using find_type_t = typename find_type<TYPELIST, PREDICATE>::type;
00063
00064 }
00065

```

## 16.206 type\_traits

```

00001 // vi:set ft=c++: -- Mode: C++ --

```



```

00002
00003 /*
00004  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00005  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00006  *      economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU General Public License 2.
00010  * Please see the COPYING-GPL-2 file for details.
00011  *
00012  * As a special exception, you may use this file as part of a free software
00013  * library without restriction. Specifically, if other files instantiate
00014  * templates or use macros or inline functions from this file, or you compile
00015  * this file and link it with other files to produce an executable, this
00016  * file does not by itself cause the resulting executable to be covered by
00017  * the GNU General Public License. This exception does not however
00018  * invalidate any other reasons why the executable file might be covered by
00019  * the GNU General Public License.
00020  */
00021
00022
00023 #pragma once
00024
00025 #pragma GCC system_header
00026
00027 #include <14/sys/compiler.h>
00028 #include "bits/type_traits.h"
00029
00030 namespace cxx {
00031
00032 template< typename T, T V >
00033 struct integral_constant
00034 {
00035     static T const value = V;
00036     typedef T value_type;
00037     typedef integral_constant<T, V> type;
00038 };
00039
00040 typedef integral_constant<bool, true> true_type;
00041 typedef integral_constant<bool, false> false_type;
00042
00043 template< typename T > struct remove_reference;
00044
00045 template< typename T > struct identity { typedef T type; };
00046 template< typename T > using identity_t = typename identity<T>::type;
00047
00048 template< typename T1, typename T2 > struct is_same;
00049
00050 template< typename T > struct remove_const;
00051
00052 template< typename T > struct remove_volatile;
00053
00054 template< typename T > struct remove_cv;
00055
00056 template< typename T > struct remove_pointer;
00057
00058 template< typename T > struct remove_extent;
00059
00060 template< typename T > struct remove_all_extents;
00061
00062
00063
00064 template< typename, typename >
00065 struct is_same : false_type {};
00066
00067 template< typename T >
00068 struct is_same<T, T> : true_type {};
00069
00070 template< typename T1, typename T2 >
00071 inline constexpr bool is_same_v = is_same<T1, T2>::value;
00072
00073 template< typename T >
00074 struct remove_reference { typedef T type; };
00075
00076 template< typename T >
00077 struct remove_reference<T &> { typedef T type; };
00078
00079 template< typename T >
00080 struct remove_reference<T &&> { typedef T type; };
00081
00082 template< typename T >
00083 using remove_reference_t = typename remove_reference<T>::type;
00084
00085 template< typename T > struct remove_const { typedef T type; };
00086 template< typename T > struct remove_const<T const> { typedef T type; };
00087 template< typename T > using remove_const_t = typename remove_const<T>::type;
00088

```

```

00089 template< typename T > struct remove_volatile { typedef T type; };
00090 template< typename T > struct remove_volatile<T volatile> { typedef T type; };
00091 template< typename T > using remove_volatile_t = typename remove_volatile<T>::type;
00092
00093 template< typename T >
00094 struct remove_cv { typedef remove_const_t<remove_volatile_t<T> type; };
00095
00096 template< typename T >
00097 using remove_cv_t = typename remove_cv<T>::type;
00098
00099 template<class T>
00100 struct remove_cvref { using type = remove_cv_t<remove_reference_t<T> }; };
00101
00102 template< typename T >
00103 using remove_cvref_t = typename remove_cvref<T>::type;
00104
00105 template< typename T, typename >
00106 struct __remove_pointer_h { typedef T type; };
00107
00108 template< typename T, typename I >
00109 struct __remove_pointer_h<T, I*> { typedef I type; };
00110
00111 template< typename T >
00112 struct remove_pointer : __remove_pointer_h<T, remove_cv_t<T> {};
00113
00114 template< typename T >
00115 using remove_pointer_t = typename remove_pointer<T>::type;
00116
00117
00118 template< typename T >
00119 struct remove_extent { typedef T type; };
00120
00121 template< typename T >
00122 struct remove_extent<T[]> { typedef T type; };
00123
00124 template< typename T, unsigned long N >
00125 struct remove_extent<T[N]> { typedef T type; };
00126
00127 template< typename T >
00128 using remove_extent_t = typename remove_extent<T>::type;
00129
00130
00131 template< typename T >
00132 struct remove_all_extents { typedef T type; };
00133
00134 template< typename T >
00135 struct remove_all_extents<T[]> { typedef typename remove_all_extents<T>::type type; };
00136
00137 template< typename T, unsigned long N >
00138 struct remove_all_extents<T[N]> { typedef typename remove_all_extents<T>::type type; };
00139
00140 template< typename T >
00141 using remove_all_extents_t = typename remove_all_extents<T>::type;
00142
00143 template< typename T >
00144 constexpr T &&
00145 forward(cxx::remove_reference_t<T> &t)
00146 { return static_cast<T &&>(t); }
00147
00148 template< typename T >
00149 constexpr T &&
00150 forward(cxx::remove_reference_t<T> &&t)
00151 { return static_cast<T &&>(t); }
00152
00153 template< typename T >
00154 constexpr cxx::remove_reference_t<T> &&
00155 move(T &&t) { return static_cast<cxx::remove_reference_t<T> &&>(t); }
00156
00157 template< bool, typename T = void >
00158 struct enable_if {};
00159
00160 template< typename T >
00161 struct enable_if<true, T> { typedef T type; };
00162
00163 template< bool C, typename T = void >
00164 using enable_if_t = typename enable_if<C, T>::type;
00165
00166 template< typename T >
00167 struct is_const : false_type {};
00168
00169 template< typename T >
00170 struct is_const<T const> : true_type {};
00171
00172 template< typename T >
00173 inline constexpr bool is_const_v = is_const<T>::value;
00174
00175 template< typename T >

```

```

00176 struct is_volatile : false_type {};
00177
00178 template< typename T >
00179 struct is_volatile<T volatile> : true_type {};
00180
00181 template< typename T >
00182 inline constexpr bool is_volatile_v = is_volatile<T>::value;
00183
00184 template< typename T >
00185 struct is_pointer : false_type {};
00186
00187 template< typename T >
00188 struct is_pointer<T *> : true_type {};
00189
00190 template< typename T >
00191 inline constexpr bool is_pointer_v = is_pointer<T>::value;
00192
00193 template<class T>
00194 inline constexpr bool is_null_pointer_v = is_same_v<decltype(nullptr), remove_cv_t<T>;
00195
00196 template< typename T >
00197 struct is_reference : false_type {};
00198
00199 template< typename T >
00200 struct is_reference<T &> : true_type {};
00201
00202 template< typename T >
00203 struct is_reference<T &&> : true_type {};
00204
00205 template< typename T >
00206 inline constexpr bool is_reference_v = is_reference<T>::value;
00207
00208 template< bool, typename, typename >
00209 struct conditional;
00210
00211 template< bool C, typename T_TRUE, typename T_FALSE >
00212 struct conditional { typedef T_TRUE type; };
00213
00214 template< typename T_TRUE, typename T_FALSE >
00215 struct conditional< false, T_TRUE, T_FALSE > { typedef T_FALSE type; };
00216
00217 template< bool C, typename T_TRUE, typename T_FALSE >
00218 using conditional_t = typename conditional<C, T_TRUE, T_FALSE>::type;
00219
00220 template<typename T>
00221 struct is_enum : integral_constant<bool, __is_enum(T)> {};
00222
00223 template< typename T >
00224 inline constexpr bool is_enum_v = is_enum<T>::value;
00225
00226 template<typename T>
00227 struct is_polymorphic : cxx::integral_constant<bool, __is_polymorphic(T)> {};
00228
00229 template< typename T > struct is_integral : false_type {};
00230
00231 template<> struct is_integral<bool> : true_type {};
00232
00233 template<> struct is_integral<char> : true_type {};
00234 template<> struct is_integral<signed char> : true_type {};
00235 template<> struct is_integral<unsigned char> : true_type {};
00236 template<> struct is_integral<short> : true_type {};
00237 template<> struct is_integral<unsigned short> : true_type {};
00238 template<> struct is_integral<int> : true_type {};
00239 template<> struct is_integral<unsigned int> : true_type {};
00240 template<> struct is_integral<long> : true_type {};
00241 template<> struct is_integral<unsigned long> : true_type {};
00242 template<> struct is_integral<long long> : true_type {};
00243 template<> struct is_integral<unsigned long long> : true_type {};
00244
00245 template< typename T >
00246 inline constexpr bool is_integral_v = is_integral<T>::value;
00247
00248 template< typename T, bool = is_integral_v<T> || is_enum_v<T> >
00249 struct __is_signed_helper : integral_constant<bool, static_cast<bool>(T(-1) < T(0))> {};
00250
00251 template< typename T >
00252 struct __is_signed_helper<T, false> : integral_constant<bool, false> {};
00253
00254 template< typename T >
00255 struct is_signed : __is_signed_helper<T> {};
00256
00257 template< typename T >
00258 inline constexpr bool is_signed_v = is_signed<T>::value;
00259
00260
00261 template< typename >
00262 struct is_array : false_type {};

```

```

00263
00264 template< typename T >
00265 struct is_array<T[]> : true_type {};
00266
00267 template< typename T, unsigned long N >
00268 struct is_array<T[N]> : true_type {};
00269
00270 template< typename T >
00271 inline constexpr bool is_array_v = is_array<T>::value;
00272
00273 template< typename T, unsigned N >
00274 constexpr unsigned array_size(T const (&)[N]) { return N; }
00275
00276 template< int SIZE, bool SIGN = false, bool = true > struct int_type_for_size;
00277
00278 template<> struct int_type_for_size<sizeof(char), true, true>
00279 { typedef signed char type; };
00280
00281 template<> struct int_type_for_size<sizeof(char), false, true>
00282 { typedef unsigned char type; };
00283
00284 template<> struct int_type_for_size<sizeof(short), true, (sizeof(short) > sizeof(char))>
00285 { typedef short type; };
00286
00287 template<> struct int_type_for_size<sizeof(short), false, (sizeof(short) > sizeof(char))>
00288 { typedef unsigned short type; };
00289
00290 template<> struct int_type_for_size<sizeof(int), true, (sizeof(int) > sizeof(short))>
00291 { typedef int type; };
00292
00293 template<> struct int_type_for_size<sizeof(int), false, (sizeof(int) > sizeof(short))>
00294 { typedef unsigned int type; };
00295
00296 template<> struct int_type_for_size<sizeof(long), true, (sizeof(long) > sizeof(int))>
00297 { typedef long type; };
00298
00299 template<> struct int_type_for_size<sizeof(long), false, (sizeof(long) > sizeof(int))>
00300 { typedef unsigned long type; };
00301
00302 template<> struct int_type_for_size<sizeof(long long), true, (sizeof(long long) > sizeof(long))>
00303 { typedef long long type; };
00304
00305 template<> struct int_type_for_size<sizeof(long long), false, (sizeof(long long) > sizeof(long))>
00306 { typedef unsigned long long type; };
00307
00308 template< int SIZE, bool SIGN = false>
00309 using int_type_for_size_t = typename int_type_for_size<SIZE, SIGN>::type;
00310
00311 template< typename T, class Enable = void > struct underlying_type {};
00312
00313 template< typename T >
00314 struct underlying_type<T, typename enable_if<is_enum_v<T>>::type >
00315 {
00316     typedef int_type_for_size_t<sizeof(T), is_signed_v<T> type;
00317 };
00318
00319 template< typename T >
00320 using underlying_type_t = typename underlying_type<T>::type;
00321
00322 template< typename T > struct make_signed;
00323 template<> struct make_signed<char> { typedef signed char type; };
00324 template<> struct make_signed<unsigned char> { typedef signed char type; };
00325 template<> struct make_signed<signed char> { typedef signed char type; };
00326 template<> struct make_signed<unsigned int> { typedef signed int type; };
00327 template<> struct make_signed<signed int> { typedef signed int type; };
00328 template<> struct make_signed<unsigned long int> { typedef signed long int type; };
00329 template<> struct make_signed<signed long int> { typedef signed long int type; };
00330 template<> struct make_signed<unsigned long long int> { typedef signed long long int type; };
00331 template<> struct make_signed<signed long long int> { typedef signed long long int type; };
00332 template< typename T > using make_signed_t = typename make_signed<T>::type;
00333
00334 template< typename T > struct make_unsigned;
00335 template<> struct make_unsigned<char> { typedef unsigned char type; };
00336 template<> struct make_unsigned<unsigned char> { typedef unsigned char type; };
00337 template<> struct make_unsigned<signed char> { typedef unsigned char type; };
00338 template<> struct make_unsigned<unsigned int> { typedef unsigned int type; };
00339 template<> struct make_unsigned<signed int> { typedef unsigned int type; };
00340 template<> struct make_unsigned<unsigned long int> { typedef unsigned long int type; };
00341 template<> struct make_unsigned<signed long int> { typedef unsigned long int type; };
00342 template<> struct make_unsigned<unsigned long long int> { typedef unsigned long long int type; };
00343 template<> struct make_unsigned<signed long long int> { typedef unsigned long long int type; };
00344 template< typename T > using make_unsigned_t = typename make_unsigned<T>::type;
00345
00346
00347 template<typename From, typename To>
00348 struct is_convertible
00349 {

```

```

00350 private:
00351     struct _true { char x[2]; };
00352     struct _false {};
00353
00354     static _true _helper(To const *);
00355     static _false _helper(...);
00356 public:
00357     enum
00358     {
00359         value = sizeof(_true) == sizeof(_helper(static_cast<From*>(0)))
00360             ? true : false
00361     };
00362
00363     typedef bool value_type;
00364 };
00365
00366 template<typename From, typename To>
00367 inline constexpr bool is_convertible_v = is_convertible<From, To>::value;
00368
00369 template<typename T>
00370 struct is_empty : integral_constant<bool, __is_empty(T)> {};
00371
00372 template<typename T>
00373 inline constexpr bool is_empty_v = is_empty<T>::value;
00374
00375
00376 #if L4_HAS_BUILTIN(__is_function)
00377     template < typename T >
00378     struct is_function : integral_constant<bool, __is_function(T)> {};
00379 #else
00380     template < typename T >
00381     struct is_function : integral_constant<bool, !is_reference_v<T>
00382         && !is_const_v<const T> {}> {};
00383 #endif
00384
00385 template<typename T>
00386 inline constexpr bool is_function_v = is_function<T>::value;
00387
00388 }
00389

```

## 16.207 unique\_ptr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002
00003 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00004 /*
00005  * Copyright (C) 2013 Technische Universität Dresden.
00006  * Copyright (C) 2014-2017, 2020 Kernkonzept GmbH.
00007  */
00008
00009 #pragma once
00010
00011 #include "type_traits"
00012
00013 namespace cxx
00014 {
00015
00016     template<typename T>
00017     struct default_delete
00018     {
00019         default_delete() {}
00020
00021         template<typename U>
00022         default_delete(default_delete<U> const &) {}
00023
00024         void operator () (T *p) const
00025         { delete p; }
00026     };
00027
00028     template<typename T>
00029     struct default_delete<T[]>
00030     {
00031         default_delete() {}
00032
00033         void operator () (T *p)
00034         { delete [] p; }
00035     };
00036
00037     template<typename T, typename C>
00038     struct unique_ptr_index_op {};
00039
00040     template<typename T, typename C>

```

```

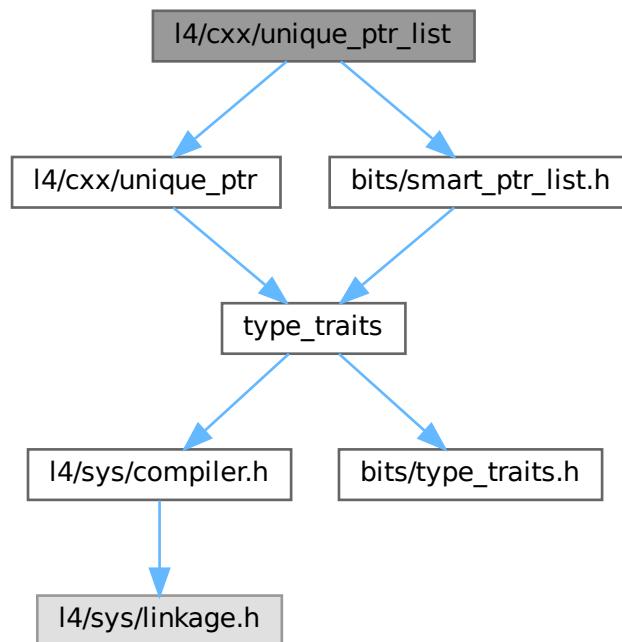
00041 struct unique_ptr_index_op<T[], C>
00042 {
00043     typedef T &reference;
00044     reference operator [] (int idx) const
00045     { return static_cast<C const *>(this)->get()[idx]; }
00046 };
00047
00048 template< typename T, typename T_Del = default_delete<T> >
00049 class unique_ptr : public unique_ptr_index_op<T, unique_ptr<T, T_Del> >
00050 {
00051 private:
00052     struct _unspec;
00053     typedef _unspec* _unspec_ptr_type;
00054
00055 public:
00056     typedef cxx::remove_extent_t<T> element_type;
00057     typedef element_type *pointer;
00058     typedef element_type &reference;
00059     typedef T_Del deleter_type;
00060
00061     unique_ptr() : _ptr(pointer()) {}
00062
00063     explicit unique_ptr(pointer p) : _ptr(p) {}
00064
00065     unique_ptr(unique_ptr &&o) : _ptr(o.release()) {}
00066
00067     ~unique_ptr() { reset(); }
00068
00069     unique_ptr &operator = (unique_ptr &&o)
00070     {
00071         reset(o.release());
00072         return *this;
00073     }
00074
00075     unique_ptr &operator = (_unspec_ptr_type)
00076     {
00077         reset();
00078         return *this;
00079     }
00080
00081     element_type &operator * () const { return *get(); }
00082     pointer operator -> () const { return get(); }
00083
00084     pointer get() const { return _ptr; }
00085
00086     operator _unspec_ptr_type () const
00087     { return reinterpret_cast<_unspec_ptr_type>(get()); }
00088
00089     pointer release()
00090     {
00091         pointer r = _ptr;
00092         _ptr = 0;
00093         return r;
00094     }
00095
00096     void reset(pointer p = pointer())
00097     {
00098         if (p != get())
00099         {
00100             deleter_type()(get());
00101             _ptr = p;
00102         }
00103     }
00104
00105     unique_ptr(unique_ptr const &) = delete;
00106     unique_ptr &operator = (unique_ptr const &) = delete;
00107
00108 private:
00109     pointer _ptr;
00110 };
00111
00112 template< typename T >
00113 unique_ptr<T>
00114 make_unique_ptr(T *p)
00115 { return unique_ptr<T>(p); }
00116
00117 template< typename T >
00118 cxx::enable_if_t<cxx::is_array<T>::value, unique_ptr<T>
00119 make_unique(unsigned long size)
00120 { return cxx::unique_ptr<T>(new cxx::remove_extent_t<T>[size]()); }
00121
00122 template< typename T, typename... Args >
00123 cxx::enable_if_t<!cxx::is_array<T>::value, unique_ptr<T>
00124 make_unique(Args &&... args)
00125 { return cxx::unique_ptr<T>(new T(cxx::forward<Args>(args)...)); }
00126
00127 }

```

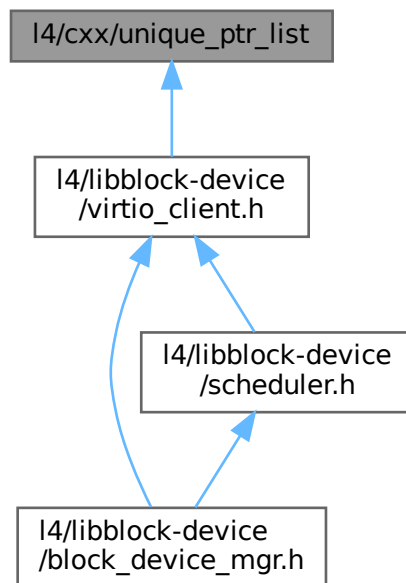
## 16.208 l4/cxx/unique\_ptr\_list File Reference

Implementation of a list of unique-ptr-managed objects.

```
#include <l4/cxx/unique_ptr>  
#include "bits/smart_ptr_list.h"  
Include dependency graph for unique_ptr_list:
```



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [cxx](#)  
*Our C++ library.*

## Typedefs

- `template<typename T>`  
using `cxx::Unique\_ptr\_list\_item = Bits::Smart\_ptr\_list\_item< T, cxx::unique\_ptr< T > >`  
*Item for list linked with `cxx::unique\_ptr`.*
- `template<typename T>`  
using `cxx::Unique\_ptr\_list = Bits::Smart\_ptr\_list< Unique\_ptr\_list\_item< T > >`  
*Single-linked list where elements are connected with a `cxx::unique\_ptr`.*

### 16.208.1 Detailed Description

Implementation of a list of unique-ptr-managed objects.

Definition in file [unique\\_ptr\\_list](#).



## 16.209 unique\_ptr\_list

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * Copyright (C) 2018-2019, 2022 Kernkonzept GmbH.
00008  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00009  *
00010  * This file is distributed under the terms of the GNU General Public
00011  * License, version 2. Please see the COPYING-GPL-2 file for details.
00012  */
00013 #pragma once
00014
00015 #include <14/cxx/unique_ptr>
00016
00017 #include "bits/smart_ptr_list.h"
00018
00019 namespace cxx {
00020
00022 template <typename T>
00023 using Unique_ptr_list_item = Bits::Smart_ptr_list_item<T, cxx::unique_ptr<T> >;
00024
00028 template <typename T>
00029 using Unique_ptr_list = Bits::Smart_ptr_list<Unique_ptr_list_item<T> >;
00030
00031 }
```

## 16.210 utils

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00003 /*
00004  * Copyright (C) 2013 Technische Universität Dresden.
00005  */
00006
00007 #pragma once
00008
00009 namespace cxx {
00010
00038 template< typename T > inline
00039 T access_once(T const *a)
00040 {
00041     if 1
00042         __asm__ __volatile__ ( "" : "=m"(*const_cast<T*>(a));
00043         T tmp = *a;
00044         __asm__ __volatile__ ( "" : "=m"(*const_cast<T*>(a));
00045         return tmp;
00046     else
00047         return *static_cast<T const volatile *>(a);
00048 #endif
00049 }
00050
00069 template< typename T, typename VAL > inline
00070 void write_now(T *a, VAL &&val)
00071 {
00072     __asm__ __volatile__ ( "" : "=m"(*a));
00073     *a = val;
00074     __asm__ __volatile__ ( "" : : "m"(*a));
00075 }
00076
00077
00078 }
00079
```

## 16.211 weak\_ref

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2015, 2017 Kernkonzept GmbH.
00004  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00005  *           Alwexander Warg <alexander.warg@kernkonzept.com>
00006  *
00007  * This file is distributed under the terms of the GNU General Public
00008  * License, version 2. Please see the COPYING-GPL-2 file for details.
00009  */
00010 #pragma once
00011
```

```

00012 #include "hlist"
00013
00014 namespace cxx {
00015
00025 class Weak_ref_base : public H_list_item_t<Weak_ref_base>
00026 {
00027 protected:
00028     Weak_ref_base(void const *ptr = nullptr) : _obj(ptr) {}
00029     void reset_hard() { _obj = nullptr; }
00030     void const *_obj;
00031
00032 public:
00039     struct List : H_list_t<Weak_ref_base>
00040     {
00041         void reset()
00042         {
00043             while (!empty())
00044                 pop_front()->reset_hard();
00045         }
00046
00047         ~List()
00048         { reset(); }
00049     };
00050
00051     explicit operator bool () const
00052     { return _obj ? true : false; }
00053 };
00054
00055
00095 template <typename T>
00096 class Weak_ref : public Weak_ref_base
00097 {
00098 public:
00099     T *get() const
00100     { return reinterpret_cast<T*>(const_cast<void *>(_obj)); }
00101
00102     T *reset(T *n)
00103     {
00104         T *r = get();
00105         if (r)
00106             r->remove_weak_ref(this);
00107
00108         _obj = n;
00109         if (n)
00110             n->add_weak_ref(this);
00111
00112         return r;
00113     }
00114
00115     Weak_ref(T *s = nullptr) : Weak_ref_base(s)
00116     {
00117         if (s)
00118             s->add_weak_ref(this);
00119     }
00120
00121     ~Weak_ref() { reset(0); }
00122
00123     void operator = (T *n)
00124     { reset(n); }
00125
00126     Weak_ref(Weak_ref const &o) : Weak_ref_base(o._obj)
00127     {
00128         if (T *x = get())
00129             x->add_weak_ref(this);
00130     }
00131
00132     Weak_ref &operator = (Weak_ref const &o)
00133     {
00134         if (&o == this)
00135             return *this;
00136
00137         reset(o.get());
00138         return *this;
00139     }
00140
00141     T &operator * () const { return get(); }
00142     T *operator -> () const { return get(); }
00143 };
00144
00145 class Weak_ref_obj
00146 {
00147 protected:
00148     template <typename T> friend class Weak_ref;
00149     mutable Weak_ref_base::List weak_references;
00150
00151     void add_weak_ref(Weak_ref_base *ref) const
00152     { weak_references.push_front(ref); }

```

```

00153
00154 void remove_weak_ref(Weak_ref_base *ref) const
00155 { weak_references.remove(ref); }
00156 };
00157
00158 }

```

## 16.212 backend

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020 #pragma once
00021
00022 #include <l4/l4re_vfs/vfs.h>
00023 #include <l4/crtn/initpriorities.h>
00024
00025 namespace L4Re { namespace Vfs {
00026
00028 extern L4Re::Vfs::Ops *vfs_ops asm ("l4re_env_posix_vfs_ops");
00029
00030 class Mount_tree;
00031
00033 class Be_file : public File
00034 {
00035 public:
00036 void *operator new (size_t size) noexcept
00037 { return vfs_ops->malloc(size); }
00038
00039 void *operator new (size_t, void *m) noexcept
00040 { return m; }
00041
00042 void operator delete (void *m)
00043 { vfs_ops->free(m); }
00044
00045 // used in close, to unlock all locks of a file (as POSIX says)
00046 int unlock_all_locks() noexcept override
00047 { return 0; }
00048
00049 // for mmap
00050 L4::Cap<L4Re::Dataspace> data_space() noexcept override
00051 { return L4::Cap<L4Re::Dataspace>::Invalid; }
00052
00053 ssize_t readv(const struct iovec*, int) noexcept override
00054 { return -EINVAL; }
00055
00056 ssize_t writev(const struct iovec*, int) noexcept override
00057 { return -EINVAL; }
00058
00059 ssize_t pwritev(const struct iovec*, int, off64_t) noexcept override
00060 { return -EINVAL; }
00061
00062 ssize_t preadv(const struct iovec*, int, off64_t) noexcept override
00063 { return -EINVAL; }
00064
00065 off64_t lseek64(off64_t, int) noexcept override
00066 { return -ESPIPE; }
00067
00068 int ftruncate64(off64_t) noexcept override
00069 { return -EINVAL; }
00070
00071 int fsync() const noexcept override
00072 { return -EINVAL; }
00073
00074 int fdatsync() const noexcept override
00075 { return -EINVAL; }
00076
00077
00078
00079
00080
00081
00082
00083
00084
00085
00086
00087
00088
00089
00090

```

```

00092 int ioctl(unsigned long, va_list) noexcept override
00093 { return -EINVAL; }
00094
00095 int fstat64(struct stat64 *) const noexcept override
00096 { return -EINVAL; }
00097
00099 int fchmod(mode_t) noexcept override
00100 { return -EINVAL; }
00101
00103 int get_status_flags() const noexcept override
00104 { return 0; }
00105
00107 int set_status_flags(long) noexcept override
00108 { return 0; }
00109
00111 int get_lock(struct flock64 *) noexcept override
00112 { return -ENOLCK; }
00113
00115 int set_lock(struct flock64 *, bool) noexcept override
00116 { return -ENOLCK; }
00117
00119 int faccessat(const char *, int, int) noexcept override
00120 { return -ENOTDIR; }
00121
00123 int fchmodat(const char *, mode_t, int) noexcept override
00124 { return -ENOTDIR; }
00125
00127 int utime(const struct utimbuf *) noexcept override
00128 { return -EROFS; }
00129
00131 int utimes(const struct timeval [2]) noexcept override
00132 { return -EROFS; }
00133
00135 int utimensat(const char *, const struct timespec [2], int) noexcept override
00136 { return -EROFS; }
00137
00139 int mkdir(const char *, mode_t) noexcept override
00140 { return -ENOTDIR; }
00141
00143 int unlink(const char *) noexcept override
00144 { return -ENOTDIR; }
00145
00147 int rename(const char *, const char *) noexcept override
00148 { return -ENOTDIR; }
00149
00151 int link(const char *, const char *) noexcept override
00152 { return -ENOTDIR; }
00153
00155 int symlink(const char *, const char *) noexcept override
00156 { return -EPERM; }
00157
00159 int rmdir(const char *) noexcept override
00160 { return -ENOTDIR; }
00161
00163 ssize_t readlink(char *, size_t) override
00164 { return -EINVAL; }
00165
00166 ssize_t getdents(char *, size_t) noexcept override
00167 { return -ENOTDIR; }
00168
00169
00170
00171 // Socket interface
00172 int bind(sockaddr const *, socklen_t) noexcept override
00173 { return -ENOTSOCK; }
00174
00175 int connect(sockaddr const *, socklen_t) noexcept override
00176 { return -ENOTSOCK; }
00177
00178 ssize_t send(void const *, size_t, int) noexcept override
00179 { return -ENOTSOCK; }
00180
00181 ssize_t recv(void *, size_t, int) noexcept override
00182 { return -ENOTSOCK; }
00183
00184 ssize_t sendto(void const *, size_t, int, sockaddr const *, socklen_t) noexcept
00185 override
00186 { return -ENOTSOCK; }
00187
00188 ssize_t recvfrom(void *, size_t, int, sockaddr *, socklen_t *) noexcept override
00189 { return -ENOTSOCK; }
00190
00191 ssize_t sendmsg(msghdr const *, int) noexcept override
00192 { return -ENOTSOCK; }
00193
00194 ssize_t recvmsg(msghdr *, int) noexcept override
00195 { return -ENOTSOCK; }

```

```

00196
00197 int getsockopt(int, int, void *, socklen_t *) noexcept override
00198 { return -ENOTSOCK; }
00199
00200 int setsockopt(int, int, void const *, socklen_t) noexcept override
00201 { return -ENOTSOCK; }
00202
00203 int listen(int) noexcept override
00204 { return -ENOTSOCK; }
00205
00206 int accept(sockaddr *, socklen_t *) noexcept override
00207 { return -ENOTSOCK; }
00208
00209 int shutdown(int) noexcept override
00210 { return -ENOTSOCK; }
00211
00212 int getsockname(sockaddr *, socklen_t *) noexcept override
00213 { return -ENOTSOCK; }
00214
00215 int getpeername(sockaddr *, socklen_t *) noexcept override
00216 { return -ENOTSOCK; }
00217
00218 ~Be_file() noexcept = 0;
00219
00220 int select(int, fd_set *, fd_set *, fd_set *, struct timeval *) throw()
00221 { return -EOPNOTSUPP; }
00222
00223 private:
00225 int get_entry(const char *, int, mode_t, cxx::Ref_ptr<File> *) noexcept override
00226 { return -ENOTDIR; }
00227
00228 protected:
00229 const char *get_mount(const char *path, cxx::Ref_ptr<File> *dir) noexcept;
00230 };
00231
00232 inline
00233 Be_file::~Be_file() noexcept {}
00234
00235 class Be_file_pos : public Be_file
00236 {
00237 public:
00238 Be_file_pos() noexcept : Be_file(), _pos(0) {}
00239
00240 virtual off64_t size() const noexcept = 0;
00241
00242 ssize_t readv(const struct iovec *v, int iovcnt) noexcept override
00243 {
00244     ssize_t r = preadv(v, iovcnt, _pos);
00245     if (r > 0)
00246         _pos += r;
00247     return r;
00248 }
00249
00250 ssize_t writev(const struct iovec *v, int iovcnt) noexcept override
00251 {
00252     ssize_t r = pwritev(v, iovcnt, _pos);
00253     if (r > 0)
00254         _pos += r;
00255     return r;
00256 }
00257
00258 ssize_t preadv(const struct iovec *v, int iovcnt, off64_t offset) noexcept override = 0;
00259 ssize_t pwritev(const struct iovec *v, int iovcnt, off64_t offset) noexcept override = 0;
00260
00261 off64_t lseek64(off64_t offset, int whence) noexcept override
00262 {
00263     off64_t r;
00264     switch (whence)
00265     {
00266     case SEEK_SET: r = offset; break;
00267     case SEEK_CUR: r = _pos + offset; break;
00268     case SEEK_END: r = size() + offset; break;
00269     default: return -EINVAL;
00270     };
00271
00272     if (r < 0)
00273         return -EINVAL;
00274
00275     _pos = r;
00276     return _pos;
00277 }
00278
00279 ~Be_file_pos() noexcept = 0;
00280
00281 protected:
00282 off64_t pos() const noexcept { return _pos; }
00283

```

```

00284 private:
00285     off64_t _pos;
00286 };
00287
00288 inline Be_file_pos::~~Be_file_pos() noexcept {}
00289
00290 class Be_file_stream : public Be_file
00291 {
00292 public:
00293     ssize_t preadv(const struct iovec *v, int iovcnt, off64_t) noexcept override
00294     { return readv(v, iovcnt); }
00295
00296     ssize_t pwritev(const struct iovec *v, int iovcnt, off64_t) noexcept override
00297     { return writev(v, iovcnt); }
00298
00299     ~Be_file_stream() noexcept = 0;
00300 };
00301
00302 inline Be_file_stream::~~Be_file_stream() noexcept {}
00303
00311 class Be_file_system : public File_system
00312 {
00313 private:
00314     char const *const _fstype;
00315
00316 public:
00317
00325     explicit Be_file_system(char const *fstype) noexcept
00326     : File_system(), _fstype(fstype)
00327     {
00328         vfs_ops->register_file_system(this);
00329     }
00330
00337     ~Be_file_system() noexcept
00338     {
00339         vfs_ops->unregister_file_system(this);
00340     }
00341
00347     char const *type() const noexcept override { return _fstype; }
00348 };
00349
00350 /* Make sure filesystems can register before the constructor of libmount
00351  * runs */
00352 #define L4RE_VFS_FILE_SYSTEM_ATTRIBUTE \
00353     __attribute__((init_priority(INIT_PRIO_LATE)))
00354
00355 }

```

## 16.213 default\_ops\_impl.h

```

00001 // vi:ft=cpp
00002 /*
00003  * Copyright (C) 2016 Kernkonzept GmbH.
00004  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00005  *
00006  * This file is distributed under the terms of the GNU General Public
00007  * License, version 2. Please see the COPYING-GPL-2 file for details.
00008  */
00009 #include <l4/cxx/static_container>
00010
00011 namespace {
00012 struct Vfs_init
00013 {
00014     // The Static_containers are used to prevent automatic destruction during
00015     // program shutdown. At least the `vfs` object must never be destructed
00016     // because any later attempt to do any kind of file-descriptor access in
00017     // the program would crash, and we could not be sure that the destructor
00018     // would really be executed after each possible operation using files or file
00019     // descriptors.
00020     cxx::Static_container<Vfs> vfs;
00021
00022     // The Static_containers below are just for providing ordering. The factories
00023     // must be initialized after the `vfs` object.
00024     cxx::Static_container<L4Re::Vfs::File_factory_t<L4Re::Dataspace, L4Re::Core::Ro_file> > ro_file;
00025     cxx::Static_container<L4Re::Vfs::File_factory_t<L4Re::Namespace, L4Re::Core::Ns_dir> > ns_dir;
00026     cxx::Static_container<L4Re::Vfs::File_factory_t<L4Re::Vcon, L4Re::Core::Vcon_stream> > vcon_stream;
00027
00028     Vfs_init()
00029     {
00030         vfs.construct();
00031         __rtld_l4re_env_posix_vfs_ops = vfs;
00032         ns_dir.construct();

```

```

00033     auto ns_ptr = cxx::ref_ptr(ns_dir.get());
00034     vfs->register_file_factory(ns_ptr);
00035     ns_ptr.release(); // prevent deletion of static object
00036
00037     ro_file.construct();
00038     auto ro_ptr = cxx::ref_ptr(ro_file.get());
00039     vfs->register_file_factory(ro_ptr);
00040     ro_ptr.release(); // prevent deletion of static object
00041
00042     vcon_stream.construct();
00043     auto vcon_ptr = cxx::ref_ptr(vcon_stream.get());
00044     vfs->register_file_factory(vcon_ptr);
00045     vcon_ptr.release(); // prevent deletion of static object
00046 }
00047 };
00048
00049 static Vfs_init __vfs_init __attribute__((init_priority(INIT_PRIO_VFS_INIT)));
00050
00051 };

```

## 16.214 fd\_store.h

```

00001 /*
00002  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019
00020 #include <l4/l4re_vfs/vfs.h>
00021
00022 namespace L4Re { namespace Core {
00023
00024     using cxx::Ref_ptr;
00025
00026     class Fd_store
00027     {
00028     public:
00029         enum { MAX_FILES = 50 };
00030
00031         Fd_store() noexcept : _fd_hint(0) {}
00032
00033         int alloc() noexcept;
00034         void free(int fd) noexcept;
00035         bool check_fd(int fd) noexcept;
00036         Ref_ptr<L4Re::Vfs::File> get(int fd) noexcept;
00037         void set(int fd, Ref_ptr<L4Re::Vfs::File> const &f) noexcept;
00038
00039     private:
00040         int _fd_hint;
00041         Ref_ptr<L4Re::Vfs::File> _files[MAX_FILES];
00042     };
00043
00044     inline
00045     bool
00046     Fd_store::check_fd(int fd) noexcept
00047     {
00048         return fd >= 0 && fd < MAX_FILES;
00049     }
00050
00051     inline
00052     Ref_ptr<L4Re::Vfs::File>
00053     Fd_store::get(int fd) noexcept
00054     {
00055         if (check_fd(fd))
00056             return _files[fd];
00057         return Ref_ptr<>::Nil;
00058     }
00059
00060     inline
00061

```

```

00062 void
00063 Fd_store::set(int fd, Ref_ptr<L4Re::Vfs::File> const &f) noexcept
00064 {
00065     _files[fd] = f;
00066 }
00067
00068 }}

```

## 16.215 fd\_store\_impl.h

```

00001 /*
00002  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #include "fd_store.h"
00019
00020 namespace L4Re { namespace Core {
00021
00022     int
00023     Fd_store::alloc() noexcept
00024     {
00025         for (int i = _fd_hint; i < MAX_FILES; ++i)
00026         {
00027             if (!_files[i])
00028             {
00029                 _fd_hint = i + 1;
00030                 return i;
00031             }
00032         }
00033         return -1;
00034     }
00035 }
00036
00037 void
00038 Fd_store::free(int fd) noexcept
00039 {
00040     _files[fd] = 0;
00041     if (fd < _fd_hint)
00042         _fd_hint = fd;
00043 }
00044
00045 }}
00046

```

## 16.216 ns\_fs.h

```

00001 /*
00002  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *     Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019 #pragma once
00020

```



```

00021 #include <l4/l4re_vfs/backend>
00022 #include <l4/sys/capability>
00023 #include <l4/re/namespace>
00024 #include <l4/re/unique_cap>
00025
00026 namespace L4Re { namespace Core {
00027
00028     using cxx::Ref_ptr;
00029
00030     class Env_dir : public L4Re::Vfs::Be_file
00031     {
00032     public:
00033         explicit Env_dir(L4Re::Env const *env)
00034             : _env(env), _current_cap_entry(env->initial_caps())
00035         {}
00036
00037         ssize_t readv(const struct iovec*, int) noexcept override { return -EISDIR; }
00038         ssize_t writev(const struct iovec*, int) noexcept override { return -EISDIR; }
00039         ssize_t preadv(const struct iovec*, int, off64_t) noexcept override { return -EISDIR; }
00040         ssize_t pwritev(const struct iovec*, int, off64_t) noexcept override { return -EISDIR; }
00041         int fstat64(struct stat64 *) const noexcept override;
00042         int faccessat(const char *path, int mode, int flags) noexcept override;
00043         int get_entry(const char *path, int flags, mode_t mode,
00044                     Ref_ptr<L4Re::Vfs::File> *) noexcept override;
00045         ssize_t getdents(char *, size_t) noexcept override;
00046
00047         ~Env_dir() noexcept {}
00048
00049     private:
00050         int get_ds(const char *path, L4Re::Unique_cap<L4Re::Dataspace> *ds) noexcept;
00051         bool check_type(Env::Cap_entry const *e, long protocol) noexcept;
00052
00053         L4Re::Env const *_env;
00054         Env::Cap_entry const *_current_cap_entry;
00055     };
00056
00057     class Ns_dir : public L4Re::Vfs::Be_file
00058     {
00059     public:
00060         explicit Ns_dir(L4::Cap<L4Re::Namespace> ns)
00061             : _ns(ns), _current_dir_pos(0)
00062         {}
00063
00064         ssize_t readv(const struct iovec*, int) noexcept override { return -EISDIR; }
00065         ssize_t writev(const struct iovec*, int) noexcept override { return -EISDIR; }
00066         ssize_t preadv(const struct iovec*, int, off64_t) noexcept override { return -EISDIR; }
00067         ssize_t pwritev(const struct iovec*, int, off64_t) noexcept override { return -EISDIR; }
00068         int fstat64(struct stat64 *) const noexcept override;
00069         int faccessat(const char *path, int mode, int flags) noexcept override;
00070         int get_entry(const char *path, int flags, mode_t mode,
00071                     Ref_ptr<L4Re::Vfs::File> *) noexcept override;
00072         ssize_t getdents(char *, size_t) noexcept override;
00073
00074         ~Ns_dir() noexcept {}
00075
00076     private:
00077         int get_ds(const char *path, L4Re::Unique_cap<L4Re::Dataspace> *ds) noexcept;
00078
00079         L4::Cap<L4Re::Namespace> _ns;
00080         size_t _current_dir_pos;
00081     };
00082
00083 }

```

## 16.217 ns\_fs\_impl.h

```

00001 /*
00002  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.

```

```

00018  */
00019  #include "ns_fs.h"
00020
00021  #include <l4/re/dataspace>
00022  #include <l4/re/util/env_ns>
00023  #include <l4/re/unique_cap>
00024  #include <dirent.h>
00025
00026  namespace L4Re { namespace Core {
00027
00028  static
00029  Ref_ptr<L4Re::Vfs::File>
00030  cap_to_vfs_object(L4::Cap<void> o, int *err)
00031  {
00032      L4::Cap<L4::Meta> m = L4::cap_reinterpret_cast<L4::Meta>(o);
00033      long proto = 0;
00034      char name_buf[256];
00035      L4::Ipc::String<char> name(sizeof(name_buf), name_buf);
00036      int r = l4_error(m->interface(0, &proto, &name));
00037      *err = -ENOPROTOPT;
00038      if (r < 0)
00039          // could not get type of object so bail out
00040          return Ref_ptr<L4Re::Vfs::File>();
00041
00042      *err = -EPROTO;
00043      Ref_ptr<L4Re::Vfs::File_factory> factory;
00044
00045      if (proto != 0)
00046          factory = L4Re::Vfs::vfs_ops->get_file_factory(proto);
00047
00048      if (!factory)
00049          factory = L4Re::Vfs::vfs_ops->get_file_factory(name.data);
00050
00051      if (!factory)
00052          return Ref_ptr<L4Re::Vfs::File>();
00053
00054      *err = -ENOMEM;
00055      return factory->create(o);
00056  }
00057
00058  int
00059  Ns_dir::get_ds(const char *path, L4Re::Unique_cap<L4Re::Dataspace> *ds) noexcept
00060  {
00061      auto file = L4Re::make_unique_cap<L4Re::Dataspace>(L4Re::virt_cap_alloc);
00062
00063      if (!file.is_valid())
00064          return -ENOMEM;
00065
00066      int err = _ns->query(path, file.get());
00067
00068      if (err < 0)
00069          return -ENOENT;
00070
00071      *ds = cxx::move(file);
00072      return err;
00073  }
00074
00075  int
00076  Ns_dir::get_entry(const char *path, int /*flags*/, mode_t /*mode*/,
00077                  Ref_ptr<L4Re::Vfs::File> *f) noexcept
00078  {
00079      if (!*path)
00080      {
00081          *f = cxx::ref_ptr(this);
00082          return 0;
00083      }
00084
00085      L4Re::Unique_cap<Dataspace> file;
00086      int err = get_ds(path, &file);
00087
00088      if (err < 0)
00089          return -ENOENT;
00090
00091      cxx::Ref_ptr<L4Re::Vfs::File> fi = cap_to_vfs_object(file.get(), &err);
00092      if (!fi)
00093          return err;
00094
00095      file.release();
00096      *f = cxx::move(fi);
00097      return 0;
00098  }
00099
00100  int
00101  Ns_dir::faccessat(const char *path, int mode, int /*flags*/) noexcept
00102  {
00103      auto tmpcap = L4Re::make_unique_cap<void>(L4Re::virt_cap_alloc);

```

```

00105
00106     if (!tmpcap.is_valid())
00107         return -ENOMEM;
00108
00109     if (_ns->query(path, tmpcap.get()))
00110         return -ENOENT;
00111
00112     if (mode & W_OK)
00113         return -EACCES;
00114
00115     return 0;
00116 }
00117
00118 int
00119 Ns_dir::fststat64(struct stat64 *b) const noexcept
00120 {
00121     b->st_dev = 1;
00122     b->st_ino = 1;
00123     b->st_mode = S_IRWXU | S_IFDIR;
00124     b->st_nlink = 0;
00125     b->st_uid = 0;
00126     b->st_gid = 0;
00127     b->st_rdev = 0;
00128     b->st_size = 0;
00129     b->st_blksize = 0;
00130     b->st_blocks = 0;
00131     b->st_atime = 0;
00132     b->st_mtime = 0;
00133     b->st_ctime = 0;
00134     return 0;
00135 }
00136
00137 ssize_t
00138 Ns_dir::getdents(char *buf, size_t dest_sz) noexcept
00139 {
00140     struct dirent64 *dest = reinterpret_cast<struct dirent64 *>(buf);
00141     ssize_t ret = 0;
00142     L4_addr_t infoaddr;
00143     size_t infosz;
00144
00145     L4Re::Unique_cap<Dataspace> dirinfofile;
00146     int err = get_ds(".dirinfo", &dirinfofile);
00147     if (err)
00148         return 0;
00149
00150     infosz = dirinfofile->size();
00151     if (infosz <= 0)
00152         return 0;
00153
00154     infoaddr = L4_PAGESIZE;
00155     err = L4Re::Env::env()->rm()->attach(&infoaddr, infosz,
00156                                           Rm::F::Search_addr | Rm::F::R,
00157                                           dirinfofile.get(), 0);
00158     if (err < 0)
00159         return 0;
00160
00161     char *p = reinterpret_cast<char *>(infoaddr) + _current_dir_pos;
00162     char *end = reinterpret_cast<char *>(infoaddr) + infosz;
00163
00164     char *current_dirinfo_entry = p;
00165     while (dest && p < end)
00166     {
00167         // parse lines of dirinfofile
00168         long len = 0;
00169         for (; p < end && *p >= '0' && *p <= '9'; ++p)
00170         {
00171             len += 10;
00172             len += *p - '0';
00173         }
00174
00175         if (len == 0)
00176             break;
00177
00178         if (p == end)
00179             break;
00180
00181         if (*p != ':')
00182             break;
00183         p++; // skip colon
00184
00185         if (p + len >= end)
00186             break;
00187
00188         unsigned l = len + 1;
00189         if (l > sizeof(dest->d_name))
00190             l = sizeof(dest->d_name);
00191

```

```

00192     unsigned n = offsetof (struct dirent64, d_name) + 1;
00193     n = (n + sizeof(long) - 1) & ~(sizeof(long) - 1);
00194
00195     if (n > dest_sz)
00196         break;
00197
00198     dest->d_ino = 1;
00199     dest->d_off = 0;
00200     memcpy(dest->d_name, p, 1 - 1);
00201     dest->d_name[1 - 1] = 0;
00202     dest->d_reclen = n;
00203     dest->d_type   = DT_UNKNOWN;
00204     ret += n;
00205     dest_sz -= n;
00206
00207     // next entry
00208     dest = reinterpret_cast<struct dirent64 *>
00209         (reinterpret_cast<unsigned long>(dest) + n);
00210
00211     // next infodirfile line
00212     p += len;
00213     while (p < end && *p && (*p == '\n' || *p == '\r'))
00214         p++;
00215
00216     current_dirinfo_entry = p;
00217 }
00218
00219 _current_dir_pos = current_dirinfo_entry - reinterpret_cast<char *>(infoaddr);
00220
00221 if (!ret) // hack since we should only reset this at open times
00222     _current_dir_pos = 0;
00223
00224 L4Re::Env::env()->rm()->detach(infoaddr, 0);
00225
00226 return ret;
00227 }
00228
00229 int
00230 Env_dir::get_ds(const char *path, L4Re::Unique_cap<L4Re::Dataspace> *ds) noexcept
00231 {
00232     Vfs::Path p(path);
00233     Vfs::Path first = p.strip_first();
00234
00235     if (first.empty())
00236         return -ENOENT;
00237
00238     L4::Cap<L4Re::Namespace>
00239         c = _env->get_cap<L4Re::Namespace>(first.path(), first.length());
00240
00241     if (!c.is_valid())
00242         return -ENOENT;
00243
00244     if (p.empty())
00245     {
00246         *ds = L4Re::Unique_cap<L4Re::Dataspace>(L4::cap_reinterpret_cast<L4Re::Dataspace>(c));
00247         return 0;
00248     }
00249
00250     auto file = L4Re::make_unique_cap<L4Re::Dataspace>(L4Re::virt_cap_alloc);
00251
00252     if (!file.is_valid())
00253         return -ENOMEM;
00254
00255     int err = c->query(p.path(), p.length(), file.get());
00256
00257     if (err < 0)
00258         return -ENOENT;
00259
00260     *ds = cxx::move(file);
00261     return err;
00262 }
00263
00264 int
00265 Env_dir::get_entry(const char *path, int /*flags*/, mode_t /*mode*/,
00266                   Ref_ptr<L4Re::Vfs::File> *f) noexcept
00267 {
00268     if (!*path)
00269     {
00270         *f = cxx::ref_ptr(this);
00271         return 0;
00272     }
00273
00274     L4Re::Unique_cap<Dataspace> file;
00275     int err = get_ds(path, &file);
00276
00277     if (err < 0)
00278         return -ENOENT;

```

```

00279
00280     cxx::Ref_ptr<L4Re::Vfs::File> fi = cap_to_vfs_object(file.get(), &err);
00281     if (!fi)
00282         return err;
00283
00284     file.release();
00285     *f = cxx::move(fi);
00286     return 0;
00287 }
00288
00289 int
00290 Env_dir::faccessat(const char *path, int mode, int /*flags*/) noexcept
00291 {
00292     Vfs::Path p(path);
00293     Vfs::Path first = p.strip_first();
00294
00295     if (first.empty())
00296         return -ENOENT;
00297
00298     L4::Cap<L4Re::Namespace>
00299     c = _env->get_cap<L4Re::Namespace>(first.path(), first.length());
00300
00301     if (!c.is_valid())
00302         return -ENOENT;
00303
00304     if (p.empty())
00305     {
00306         if (mode & W_OK)
00307             return -EACCES;
00308
00309         return 0;
00310     }
00311
00312     auto tmpcap = L4Re::make_unique_cap<void>(L4Re::virt_cap_alloc);
00313
00314     if (!tmpcap.is_valid())
00315         return -ENOMEM;
00316
00317     if (c->query(p.path(), p.length(), tmpcap.get()))
00318         return -ENOENT;
00319
00320     if (mode & W_OK)
00321         return -EACCES;
00322
00323     return 0;
00324 }
00325
00326 bool
00327 Env_dir::check_type(Env::Cap_entry const *e, long protocol) noexcept
00328 {
00329     L4::Cap<L4::Meta> m(e->cap);
00330     return m->supports(protocol).label();
00331 }
00332
00333 int
00334 Env_dir::fstat64(struct stat64 *b) const noexcept
00335 {
00336     b->st_dev = 1;
00337     b->st_ino = 1;
00338     b->st_mode = S_IRWXU | S_IFDIR;
00339     b->st_nlink = 0;
00340     b->st_uid = 0;
00341     b->st_gid = 0;
00342     b->st_rdev = 0;
00343     b->st_size = 0;
00344     b->st_blksize = 0;
00345     b->st_blocks = 0;
00346     b->st_atime = 0;
00347     b->st_mtime = 0;
00348     b->st_ctime = 0;
00349     return 0;
00350 }
00351
00352 ssize_t
00353 Env_dir::getdents(char *buf, size_t sz) noexcept
00354 {
00355     struct dirent64 *d = reinterpret_cast<struct dirent64 *>(buf);
00356     ssize_t ret = 0;
00357
00358     while (d
00359         && _current_cap_entry
00360         && _current_cap_entry->flags != ~0UL)
00361     {
00362         unsigned l = strlen(_current_cap_entry->name) + 1;
00363         if (l > sizeof(d->d_name))
00364             l = sizeof(d->d_name);
00365     }

```

```

00366     unsigned n = offsetof (struct dirent64, d_name) + 1;
00367     n = (n + sizeof(long) - 1) & ~(sizeof(long) - 1);
00368
00369     if (n <= sz)
00370     {
00371         d->d_ino = 1;
00372         d->d_off = 0;
00373         memcpy(d->d_name, _current_cap_entry->name, 1);
00374         d->d_name[1 - 1] = 0;
00375         d->d_reclen = n;
00376         if (check_type(_current_cap_entry, L4Re::Namespace::Protocol))
00377             d->d_type = DT_DIR;
00378         else if (check_type(_current_cap_entry, L4Re::Dataspace::Protocol))
00379             d->d_type = DT_REG;
00380         else
00381             d->d_type = DT_UNKNOWN;
00382         ret += n;
00383         sz -= n;
00384         d = reinterpret_cast<struct dirent64 *>
00385             (reinterpret_cast<unsigned long>(d) + n);
00386         _current_cap_entry++;
00387     }
00388     else
00389         return ret;
00390 }
00391
00392 // bit of a hack because we should only (re)set this when opening the dir
00393 if (!ret)
00394     _current_cap_entry = _env->initial_caps();
00395
00396 return ret;
00397 }
00398
00399 }

```

## 16.218 ro\_file.h

```

00001 /*
00002  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019 #pragma once
00020
00021 #include <l4/l4re_vfs/backend>
00022
00023 namespace L4Re { namespace Core {
00024
00025     class Ro_file : public L4Re::Vfs::Be_file_pos
00026     {
00027     private:
00028         L4::Cap<L4Re::Dataspace> _ds;
00029         off64_t _size;
00030         char const *_addr;
00031
00032     public:
00033         explicit Ro_file(L4::Cap<L4Re::Dataspace> ds) noexcept
00034             : Be_file_pos(), _ds(ds), _addr(0)
00035         {
00036             _size = _ds->size();
00037         }
00038
00039         L4::Cap<L4Re::Dataspace> data_space() noexcept override { return _ds; }
00040
00041         int fstat64(struct stat64 *buf) const noexcept override;
00042
00043         int ioctl1(unsigned long, va_list) noexcept override;
00044
00045         off64_t size() const noexcept override { return _size; }
00046     }

```

```

00047 int get_status_flags() const noexcept override
00048 { return O_RDONLY; }
00049
00050 int set_status_flags(long) noexcept override
00051 { return 0; }
00052
00053 ~Ro_file() noexcept;
00054
00055 private:
00056 ssize_t read_single(const struct iovec*, off64_t) noexcept;
00057 ssize_t preadv(const struct iovec *, int, off64_t) noexcept override;
00058 ssize_t pwritev(const struct iovec *, int , off64_t) noexcept override;
00059 };
00060
00061
00062 }

```

## 16.219 ro\_file\_impl.h

```

00001 /*
00002  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019
00020 #include "ro_file.h"
00021
00022 #include <sys/ioctl.h>
00023
00024 #include <L4/re/env>
00025
00026 namespace L4Re { namespace Core {
00027
00028 Ro_file::~Ro_file() noexcept
00029 {
00030     if (_addr)
00031         L4Re::Env::env()->rm()->detach(L4_addr_t(_addr), 0);
00032     L4Re::virt_cap_alloc->release(_ds);
00033 }
00034
00035 int
00036 Ro_file::fstat64(struct stat64 *buf) const noexcept
00037 {
00038     static int fake = 0;
00039
00040     memset(buf, 0, sizeof(*buf));
00041     buf->st_size = _size;
00042     buf->st_mode = S_IFREG | 0644;
00043     buf->st_dev = _ds.cap();
00044     buf->st_ino = ++fake;
00045     buf->st_blksize = L4_PAGESIZE;
00046     buf->st_blocks = L4_round_page(_size);
00047     return 0;
00048 }
00049
00050
00051 ssize_t
00052 Ro_file::read_single(const struct iovec *vec, off64_t pos) noexcept
00053 {
00054     off64_t l = vec->iiov_len;
00055     if (_size - pos < l)
00056         l = _size - pos;
00057
00058     if (l > 0)
00059     {
00060         Vfs_config::memcpy(vec->iiov_base, _addr + pos, l);
00061         return l;
00062     }
00063
00064     return 0;

```

```

00065 }
00066
00067 ssize_t
00068 Ro_file::preadv(const struct iovec *vec, int cnt, off64_t offset) noexcept
00069 {
00070     if (!_addr)
00071     {
00072         void const *file = reinterpret_cast<void*>(L4_PAGESIZE);
00073         long err = L4Re::Env::env()->rm()->attach(&file, _size,
00074                                                  Rm::F::Search_addr | Rm::F::R,
00075                                                  _ds, 0);
00076
00077         if (err < 0)
00078             return err;
00079
00080         _addr = static_cast<char const *>(file);
00081     }
00082
00083     ssize_t l = 0;
00084
00085     while (cnt > 0)
00086     {
00087         ssize_t r = read_single(vec, offset);
00088         offset += r;
00089         l += r;
00090
00091         if (static_cast<size_t>(r) < vec->iiov_len)
00092             return l;
00093
00094         ++vec;
00095         --cnt;
00096     }
00097     return l;
00098 }
00099
00100 ssize_t
00101 Ro_file::pwritev(const struct iovec *, int, off64_t) noexcept
00102 {
00103     return -EROFS;
00104 }
00105
00106 int
00107 Ro_file::ioctl(unsigned long v, va_list args) noexcept
00108 {
00109     switch (v)
00110     {
00111         case FIONREAD: // return amount of data still available
00112             int *available = va_arg(args, int *);
00113             *available = _size - pos();
00114             return 0;
00115     };
00116     return -ENOTTY;
00117 }
00118
00119 }}

```

## 16.220 vcon\_stream.h

```

00001 /*
00002  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019
00020 #include <l4/sys/capability>
00021 #include <l4/sys/vcon>
00022 #include <l4/sys/semaphore>
00023
00024 #include <l4/l4re_vfs/backend>
00025

```



```

00026 namespace L4Re { namespace Core {
00027
00028 class Vcon_stream : public L4Re::Vfs::Be_file_stream
00029 {
00030 private:
00031     L4::Cap<L4::Vcon> _s;
00032     L4::Cap<L4::Semaphore> _irq;
00033     unsigned _irq_bound;
00034
00035 public:
00036     explicit Vcon_stream(L4::Cap<L4::Vcon> s) noexcept;
00037
00038     ssize_t readv(const struct iovec*, int iovcnt) noexcept override;
00039     ssize_t writev(const struct iovec*, int iovcnt) noexcept override;
00040     int fstat64(struct stat64 *buf) const noexcept override;
00041     int get_status_flags() const noexcept override { return O_RDWR; }
00042     int set_status_flags(long) noexcept override { return 0; }
00043     int ioctl(unsigned long request, va_list args) noexcept override;
00044
00045     ~Vcon_stream() noexcept {}
00046     void operator delete (void *) {}
00047 };
00048
00049 }

```

## 16.221 vcon\_stream\_impl.h

```

00001 /*
00002  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018
00019 #include <l4/re/env>
00020 #include <l4/sys/factory>
00021 #include <l4/cxx/minmax>
00022
00023 #include "vcon_stream.h"
00024
00025 #include <termios.h>
00026 #include <unistd.h>
00027 #include <sys/ioctl.h>
00028 #include <sys/ttydefaults.h>
00029
00030 namespace L4Re { namespace Core {
00031     Vcon_stream::Vcon_stream(L4::Cap<L4::Vcon> s) noexcept
00032     : Be_file_stream(),
00033       _s(s), _irq(L4Re::virt_cap_alloc->alloc<L4::Semaphore>()), _irq_bound(false)
00034     {
00035         // [[maybe_unused]] int res =
00036         l4_error(L4Re::Env::env()->factory()->create(_irq));
00037         // (void)res; // handle errors!
00038     }
00039
00040     ssize_t
00041     Vcon_stream::readv(const struct iovec *iovec, int iovcnt) noexcept
00042     {
00043         if (iovcnt < 0)
00044             return -EINVAL;
00045
00046         if (!_irq_bound)
00047         {
00048             bool was_bound = __atomic_exchange_n(&_irq_bound, true, __ATOMIC_SEQ_CST);
00049             if (!was_bound)
00050                 if (l4_error(_s->bind(0, _irq)) < 0)
00051                     return -EIO;
00052         }
00053
00054         ssize_t bytes = 0;
00055         for (; iovcnt > 0; --iovcnt, ++iovec)
00056         {

```

```

00057     size_t len = cxx::min<size_t>(iovec->iov_len, SSIZE_MAX - bytes);
00058     if (len == 0)
00059         continue;
00060
00061     char *buf = static_cast<char *>(iovec->iov_base);
00062
00063     while (1)
00064     {
00065         size_t l = cxx::min<size_t>(L4_VCON_READ_SIZE, len);
00066         int ret = _s->read(buf, l);
00067
00068         if (ret > static_cast<int>(1))
00069             ret = 1;
00070
00071         if (ret < 0)
00072             return ret;
00073         else if (ret == 0)
00074         {
00075             if (bytes)
00076                 return bytes;
00077
00078             ret = _s->read(buf, l);
00079             if (ret < 0)
00080                 return ret;
00081             else if (ret == 0)
00082             {
00083                 _irq->down();
00084                 continue;
00085             }
00086         }
00087
00088         bytes += ret;
00089         len -= ret;
00090         buf += ret;
00091
00092         if (len == 0)
00093             break;
00094     }
00095 }
00096
00097 return bytes;
00098 }
00099
00100 ssize_t
00101 Vcon_stream::writev(const struct iovec *iovec, int iovcnt) noexcept
00102 {
00103     l4_msg_regs_t store;
00104     l4_msg_regs_t *mr = l4_utcb_mr();
00105
00106     if (iovcnt < 0)
00107         return -EINVAL;
00108
00109     Vfs_config::memcpy(&store, mr, sizeof(store));
00110
00111     ssize_t written = 0;
00112     while (iovcnt)
00113     {
00114         size_t sl = cxx::min<size_t>(iovec->iov_len, SSIZE_MAX - written);
00115         char const *b = static_cast<char const *>(iovec->iov_base);
00116
00117         for (; sl > L4_VCON_WRITE_SIZE
00118             ; sl -= L4_VCON_WRITE_SIZE, b += L4_VCON_WRITE_SIZE,
00119             written += L4_VCON_WRITE_SIZE)
00120             _s->send(b, L4_VCON_WRITE_SIZE);
00121
00122         _s->send(b, sl);
00123
00124         written += sl;
00125
00126         ++iovec;
00127         --iovcnt;
00128     }
00129     Vfs_config::memcpy(mr, &store, sizeof(store));
00130     return written;
00131 }
00132
00133 int
00134 Vcon_stream::fstat64(struct stat64 *buf) const noexcept
00135 {
00136     buf->st_size = 0;
00137     buf->st_mode = 0666;
00138     buf->st_dev = _s.cap();
00139     buf->st_ino = 0;
00140     return 0;
00141 }
00142
00143 int

```

```

00144 Vcon_stream::ioctl(unsigned long request, va_list args) noexcept
00145 {
00146     switch (request) {
00147         case TCGETS:
00148             {
00149                 //vt100_tcgetattr(term, (struct termios *)argp);
00150
00151                 struct termios *t = va_arg(args, struct termios *);
00152
00153                 l4_vcon_attr_t l4a;
00154                 if (!l4_error(_s->get_attr(&l4a)))
00155                     {
00156                         t->c_iflag = l4a.i_flags;
00157                         t->c_oflag = l4a.o_flags; // output flags
00158                         t->c_cflag = 0; // control flags
00159                         t->c_lflag = l4a.l_flags; // local flags
00160                     }
00161                 else
00162                     t->c_iflag = t->c_oflag = t->c_cflag = t->c_lflag = 0;
00163             #if 0
00164                 //t->c_lflag |= ECHO; // if term->echo
00165                 t->c_lflag |= ICANON; // if term->term_mode == VT100MODE_COOKED
00166             #endif
00167
00168                 t->c_cc[VEOF] = CEOF;
00169                 t->c_cc[VEOL] = _POSIX_VDISABLE;
00170                 t->c_cc[VEOL2] = _POSIX_VDISABLE;
00171                 t->c_cc[VERASE] = CERASE;
00172                 t->c_cc[VWERASE] = CWERASE;
00173                 t->c_cc[VKILL] = CKILL;
00174                 t->c_cc[VREPRINT] = CREPRINT;
00175                 t->c_cc[VINTR] = CINTR;
00176                 t->c_cc[VQUIT] = _POSIX_VDISABLE;
00177                 t->c_cc[VSUSP] = CSUSP;
00178                 t->c_cc[VSTART] = CSTART;
00179                 t->c_cc[VSTOP] = CSTOP;
00180                 t->c_cc[VLNEXT] = CLNEXT;
00181                 t->c_cc[VDISCARD] = CDISCARD;
00182                 t->c_cc[VMIN] = CMIN;
00183                 t->c_cc[VTIME] = 0;
00184             }
00185
00186             return 0;
00187
00188         case TCSETS:
00189         case TCSETSW:
00190         case TCSETSF:
00191             {
00192                 //vt100_tcsetattr(term, (struct termios *)argp);
00193                 struct termios const *t = va_arg(args, struct termios const *);
00194
00195                 // XXX: well, we're cheating, get this from the other side!
00196
00197                 l4_vcon_attr_t l4a;
00198                 l4a.i_flags = t->c_iflag;
00199                 l4a.o_flags = t->c_oflag; // output flags
00200                 l4a.l_flags = t->c_lflag; // local flags
00201                 _s->set_attr(&l4a);
00202             }
00203             return 0;
00204
00205         default:
00206             break;
00207     };
00208     return -ENOTTY;
00209 }
00210 }
00211
00212 }

```

## 16.222 vfs\_impl.h

```

00001 /*
00002  * (c) 2008-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software

```

```

00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020
00021 #include "fd_store.h"
00022 #include "vcon_stream.h"
00023 #include "ns_fs.h"
00024
00025 #include <l4/bid_config.h>
00026 #include <l4/re/env>
00027 #include <l4/re/rm>
00028 #include <l4/re/dataspace>
00029 #include <l4/sys/assert.h>
00030 #include <l4/cxx/hlist>
00031 #include <l4/cxx/pair>
00032 #include <l4/cxx/std_alloc>
00033
00034 #include <l4/l4re_vfs/backend>
00035 #include <l4/re/shared_cap>
00036
00037 #include <unistd.h>
00038 #include <cstdarg>
00039 #include <errno.h>
00040 #include <sys/uio.h>
00041
00042 #if 0
00043 #include <l4/sys/kdebug.h>
00044 static int debug_mmap = 1;
00045 #define DEBUG_LOG(level, dbg...) do { if (level) dbg } while (0)
00046 #else
00047 #define DEBUG_LOG(level, dbg...) do { } while (0)
00048 #endif
00049
00055 #define USE_BIG_ANON_DS
00056
00057 using L4Re::Rm;
00058
00059 namespace {
00060
00061 using cxx::Ref_ptr;
00062
00063 class Fd_store : public L4Re::Core::Fd_store
00064 {
00065 public:
00066     Fd_store() noexcept;
00067 };
00068
00069 // for internal Vcon_streams we want to have a placement new operator, so
00070 // inherit and add one
00071 class Std_stream : public L4Re::Core::Vcon_stream
00072 {
00073 public:
00074     Std_stream(L4::Cap<L4::Vcon> c) : L4Re::Core::Vcon_stream(c) {}
00075 };
00076
00077 Fd_store::Fd_store() noexcept
00078 {
00079     // use this strange way to prevent deletion of the stdio object
00080     // this depends on Fd_store to being a singleton !!!
00081     static char m[sizeof(Std_stream)] __attribute__((aligned(sizeof(long))));
00082     Std_stream *s = new (m) Std_stream(L4Re::Env::env()->log());
00083     // make sure that we never delete the static io stream thing
00084     s->add_ref();
00085     set(0, cxx::ref_ptr(s)); // stdin
00086     set(1, cxx::ref_ptr(s)); // stdout
00087     set(2, cxx::ref_ptr(s)); // stderr
00088 }
00089
00090 class Root_mount_tree : public L4Re::Vfs::Mount_tree
00091 {
00092 public:
00093     Root_mount_tree() : L4Re::Vfs::Mount_tree(0) {}
00094     void operator delete (void *) {}
00095 };
00096
00097 class Vfs : public L4Re::Vfs::Ops
00098 {
00099 private:
00100     bool _early_oom;
00101
00102 public:
00103     Vfs()

```

```

00104 : _early_oom(true), _root_mount(), _root(L4Re::Env::env())
00105 {
00106     _root_mount.add_ref();
00107     _root.add_ref();
00108     _root_mount.mount(cxx::ref_ptr(&_root));
00109     _cwd = cxx::ref_ptr(&_root);
00110
00111 #if 0
00112     Ref_ptr<L4Re::Vfs::File> rom;
00113     _root.openat("rom", 0, 0, &rom);
00114
00115     _root_mount.create_tree("lib/foo", rom);
00116
00117     _root.openat("lib", 0, 0, &_cwd);
00118
00119 #endif
00120 }
00121
00122 int alloc_fd(Ref_ptr<L4Re::Vfs::File> const &f) noexcept override;
00123 Ref_ptr<L4Re::Vfs::File> free_fd(int fd) noexcept override;
00124 Ref_ptr<L4Re::Vfs::File> get_root() noexcept override;
00125 Ref_ptr<L4Re::Vfs::File> get_cwd() noexcept override;
00126 void set_cwd(Ref_ptr<L4Re::Vfs::File> const &dir) noexcept override;
00127 Ref_ptr<L4Re::Vfs::File> get_file(int fd) noexcept override;
00128 cxx::Pair<Ref_ptr<L4Re::Vfs::File>, int>
00129     set_fd(int fd, Ref_ptr<L4Re::Vfs::File> const &f = Ref_ptr<>::Nil) noexcept
00130     override;
00131
00132 int mmap2(void *start, size_t len, int prot, int flags, int fd,
00133     off_t offset, void **ptr) noexcept override;
00134
00135 int munmap(void *start, size_t len) noexcept override;
00136 int mremap(void *old, size_t old_sz, size_t new_sz, int flags,
00137     void **new_addr) noexcept override;
00138 int mprotect(const void *a, size_t sz, int prot) noexcept override;
00139 int msync(void *addr, size_t len, int flags) noexcept override;
00140 int madvise(void *addr, size_t len, int advice) noexcept override;
00141
00142 int register_file_system(L4Re::Vfs::File_system *f) noexcept override;
00143 int unregister_file_system(L4Re::Vfs::File_system *f) noexcept override;
00144 L4Re::Vfs::File_system *get_file_system(char const *fstype) noexcept override;
00145 L4Re::Vfs::File_system_list file_system_list() noexcept override;
00146
00147 int register_file_factory(cxx::Ref_ptr<L4Re::Vfs::File_factory> f) noexcept override;
00148 int unregister_file_factory(cxx::Ref_ptr<L4Re::Vfs::File_factory> f) noexcept override;
00149 Ref_ptr<L4Re::Vfs::File_factory> get_file_factory(int proto) noexcept override;
00150 Ref_ptr<L4Re::Vfs::File_factory> get_file_factory(char const *proto_name) noexcept override;
00151 int mount(char const *path, cxx::Ref_ptr<L4Re::Vfs::File> const &dir) noexcept override;
00152
00153 void operator delete (void *) {}
00154
00155 void *malloc(size_t size) noexcept override { return Vfs_config::malloc(size); }
00156 void free(void *m) noexcept override { Vfs_config::free(m); }
00157
00158 private:
00159     Root_mount_tree _root_mount;
00160     L4Re::Core::Env_dir _root;
00161     Ref_ptr<L4Re::Vfs::File> _cwd;
00162     Fd_store fds;
00163
00164     L4Re::Vfs::File_system *_fs_registry;
00165
00166     struct File_factory_item : cxx::H_list_item_t<File_factory_item>
00167     {
00168         cxx::Ref_ptr<L4Re::Vfs::File_factory> f;
00169         explicit File_factory_item(cxx::Ref_ptr<L4Re::Vfs::File_factory> const &f)
00170             : f(f) {};
00171
00172         File_factory_item() = default;
00173         File_factory_item(File_factory_item const &) = delete;
00174         File_factory_item &operator = (File_factory_item const &) = delete;
00175     };
00176
00177     cxx::H_list_t<File_factory_item> _file_factories;
00178
00179     l4_addr_t _anon_offset;
00180     L4Re::Shared_cap<L4Re::Dataspace> _anon_ds;
00181
00182     int alloc_ds(unsigned long size, L4Re::Shared_cap<L4Re::Dataspace> *ds);
00183     int alloc_anon_mem(l4_umword_t size, L4Re::Shared_cap<L4Re::Dataspace> *ds,
00184         l4_addr_t *offset);
00185
00186     void align_mmap_start_and_length(void **start, size_t *length);
00187     int munmap_regions(void *start, size_t len);
00188
00189     L4Re::Vfs::File_system *find_fs_from_type(char const *fstype) noexcept;
00190 };

```

```

00191
00192 static inline bool strequal(char const *a, char const *b)
00193 {
00194     for (;*a && *a == *b; ++a, ++b)
00195         ;
00196     return *a == *b;
00197 }
00198
00199 int
00200 Vfs::register_file_system(L4Re::Vfs::File_system *f) noexcept
00201 {
00202     using L4Re::Vfs::File_system;
00203
00204     if (!f)
00205         return -EINVAL;
00206
00207     for (File_system *c = _fs_registry; c; c = c->next())
00208         if (strequal(c->type(), f->type()))
00209             return -EEXIST;
00210
00211     f->next(_fs_registry);
00212     _fs_registry = f;
00213
00214     return 0;
00215 }
00216
00217 int
00218 Vfs::unregister_file_system(L4Re::Vfs::File_system *f) noexcept
00219 {
00220     using L4Re::Vfs::File_system;
00221
00222     if (!f)
00223         return -EINVAL;
00224
00225     File_system **p = &_fs_registry;
00226
00227     for (; *p; p = &(*p)->next())
00228         if (*p == f)
00229         {
00230             *p = f->next();
00231             f->next() = 0;
00232             return 0;
00233         }
00234
00235     return -ENOENT;
00236 }
00237
00238 L4Re::Vfs::File_system *
00239 Vfs::find_fs_from_type(char const *fstype) noexcept
00240 {
00241     L4Re::Vfs::File_system_list fsl(_fs_registry);
00242     for (L4Re::Vfs::File_system_list::Iterator c = fsl.begin();
00243          c != fsl.end(); ++c)
00244         if (strequal(c->type(), fstype))
00245             return *c;
00246     return 0;
00247 }
00248
00249 L4Re::Vfs::File_system_list
00250 Vfs::file_system_list() noexcept
00251 {
00252     return L4Re::Vfs::File_system_list(_fs_registry);
00253 }
00254
00255 L4Re::Vfs::File_system *
00256 Vfs::get_file_system(char const *fstype) noexcept
00257 {
00258     L4Re::Vfs::File_system *fs;
00259     if ((fs = find_fs_from_type(fstype)))
00260         return fs;
00261
00262     // Try to load a file system module dynamically
00263     int res = Vfs_config::load_module(fstype);
00264     if (res < 0)
00265         return 0;
00266
00267     // Try again
00268     return find_fs_from_type(fstype);
00269 }
00270
00271 int
00272 Vfs::register_file_factory(cxx::Ref_ptr<L4Re::Vfs::File_factory> f) noexcept
00273 {
00274     if (!f)
00275         return -EINVAL;
00276
00277     void *x = this->malloc(sizeof(File_factory_item));

```

```

00278     if (!x)
00279         return -ENOMEM;
00280
00281     auto ff = new (x, cxx::Nothrow()) File_factory_item(f);
00282     _file_factories.push_front(ff);
00283     return 0;
00284 }
00285
00286 int
00287 Vfs::unregister_file_factory(cxx::Ref_ptr<L4Re::Vfs::File_factory> f) noexcept
00288 {
00289     for (auto p: _file_factories)
00290     {
00291         if (p->f == f)
00292         {
00293             _file_factories.remove(p);
00294             p->~File_factory_item();
00295             this->free(p);
00296             return 0;
00297         }
00298     }
00299     return -ENOENT;
00300 }
00301
00302 Ref_ptr<L4Re::Vfs::File_factory>
00303 Vfs::get_file_factory(int proto) noexcept
00304 {
00305     for (auto p: _file_factories)
00306         if (p->f->proto() == proto)
00307             return p->f;
00308
00309     return Ref_ptr<L4Re::Vfs::File_factory>();
00310 }
00311
00312 Ref_ptr<L4Re::Vfs::File_factory>
00313 Vfs::get_file_factory(char const *proto_name) noexcept
00314 {
00315     for (auto p: _file_factories)
00316     {
00317         auto n = p->f->proto_name();
00318         if (n)
00319         {
00320             char const *a = n;
00321             char const *b = proto_name;
00322             for (; *a && *b && *a == *b; ++a, ++b)
00323                 ;
00324
00325             if ((*a == 0) && (*b == 0))
00326                 return p->f;
00327         }
00328     }
00329
00330     return Ref_ptr<L4Re::Vfs::File_factory>();
00331 }
00332
00333 int
00334 Vfs::alloc_fd(Ref_ptr<L4Re::Vfs::File> const &f) noexcept
00335 {
00336     int fd = fds.alloc();
00337     if (fd < 0)
00338         return -EMFILE;
00339
00340     if (f)
00341         fds.set(fd, f);
00342
00343     return fd;
00344 }
00345
00346 Ref_ptr<L4Re::Vfs::File>
00347 Vfs::free_fd(int fd) noexcept
00348 {
00349     Ref_ptr<L4Re::Vfs::File> f = fds.get(fd);
00350
00351     if (!f)
00352         return Ref_ptr<>::Nil;
00353
00354     fds.free(fd);
00355     return f;
00356 }
00357
00358
00359 Ref_ptr<L4Re::Vfs::File>
00360 Vfs::get_root() noexcept
00361 {
00362     return cxx::ref_ptr(&_root);
00363 }
00364

```

```

00365 Ref_ptr<L4Re::Vfs::File>
00366 Vfs::get_cwd() noexcept
00367 {
00368     return _cwd;
00369 }
00370
00371 void
00372 Vfs::set_cwd(Ref_ptr<L4Re::Vfs::File> const &dir) noexcept
00373 {
00374     // FIXME: check for is dir
00375     if (dir)
00376         _cwd = dir;
00377 }
00378
00379 Ref_ptr<L4Re::Vfs::File>
00380 Vfs::get_file(int fd) noexcept
00381 {
00382     return fds.get(fd);
00383 }
00384
00385 cxx::Pair<Ref_ptr<L4Re::Vfs::File>, int>
00386 Vfs::set_fd(int fd, Ref_ptr<L4Re::Vfs::File> const &f) noexcept
00387 {
00388     if (!fds.check_fd(fd))
00389         return cxx::pair(Ref_ptr<L4Re::Vfs::File>(Ref_ptr<>::Nil), EBADF);
00390
00391     Ref_ptr<L4Re::Vfs::File> old = fds.get(fd);
00392     fds.set(fd, f);
00393     return cxx::pair(old, 0);
00394 }
00395
00396
00397 #define GET_FILE_DBG(fd, err) \
00398     Ref_ptr<L4Re::Vfs::File> fi = fds.get(fd); \
00399     if (!fi) \
00400     { \
00401         return -err; \
00402     }
00403
00404 #define GET_FILE(fd, err) \
00405     Ref_ptr<L4Re::Vfs::File> fi = fds.get(fd); \
00406     if (!fi) \
00407         return -err;
00408
00409 void
00410 Vfs::align_mmap_start_and_length(void **start, size_t *length)
00411 {
00412     l4_addr_t const s = reinterpret_cast<l4_addr_t>(*start);
00413     size_t const o = s & (L4_PAGESIZE - 1);
00414
00415     *start = reinterpret_cast<void *>(l4_trunc_page(s));
00416     *length = l4_round_page(*length + o);
00417 }
00418
00419 int
00420 Vfs::munmap_regions(void *start, size_t len)
00421 {
00422     using namespace L4;
00423     using namespace L4Re;
00424
00425     int err;
00426     Cap<Dataspace> ds;
00427     Cap<Rm> r = Env::env()->rm();
00428
00429     if (l4_addr_t(start) & (L4_PAGESIZE - 1))
00430         return -EINVAL;
00431
00432     align_mmap_start_and_length(&start, &len);
00433
00434     while (1)
00435     {
00436         DEBUG_LOG(debug_mmap, {
00437             outstring("DETACH: start = 0x");
00438             outhex32(l4_addr_t(start));
00439             outstring(" len = 0x");
00440             outhex32(len);
00441             outstring("\n");
00442         });
00443         err = r->detach(l4_addr_t(start), len, &ds, This_task);
00444         if (err < 0)
00445             return err;
00446
00447         switch (err & Rm::Detach_result_mask)
00448         {
00449             case Rm::Split_ds:
00450                 if (ds.is_valid())
00451                     L4Re::virt_cap_alloc->take(ds);

```



```

00452         return 0;
00453     case Rm::Detached_ds:
00454         if (ds.is_valid())
00455             L4Re::virt_cap_alloc->release(ds);
00456         break;
00457     default:
00458         break;
00459     }
00460
00461     if (!(err & Rm::Detach_again))
00462         return 0;
00463 }
00464 }
00465
00466 int
00467 Vfs::munmap(void *start, size_t len) L4_NOTHROW
00468 {
00469     using namespace L4;
00470     using namespace L4Re;
00471
00472     int err = 0;
00473     Cap<Rm> r = Env::env()->rm();
00474
00475     // Fields for obtaining a list of areas for the calling process
00476     long area_cnt = -1;           // No. of areas in this process
00477     Rm::Area const *area_array;
00478     bool matches_area = false;    // true if unmap parameters match an area
00479
00480     // First check if there are any areas matching the munmap request. Those
00481     // might have been created by an mmap call using PROT_NONE as protection
00482     // modifier.
00483
00484     area_cnt = r->get_areas((l4_addr_t) start, &area_array);
00485
00486     // It is enough to check for the very first entry, since get_areas will
00487     // only return areas with a starting address equal or greater to <start>.
00488     // However, we intend to unmap at most the area starting exactly at
00489     // <start>.
00490     if (area_cnt > 0)
00491     {
00492         size_t area_size = area_array[0].end - area_array[0].start + 1;
00493
00494         // Only free the area if the munmap parameters describe it exactly.
00495         if (area_array[0].start == (l4_addr_t) start && area_size == len)
00496         {
00497             r->free_area((l4_addr_t) start);
00498             matches_area = true;
00499         }
00500     }
00501
00502     // After clearing possible area reservations from PROT_NONE mappings, clear
00503     // any regions in the address range specified. Note that errors shall be
00504     // suppressed if an area was freed but no regions were found.
00505     err = munmap_regions(start, len);
00506     if (err == -ENOENT && matches_area)
00507         return 0;
00508
00509     return err;
00510 }
00511
00512 int
00513 Vfs::alloc_ds(unsigned long size, L4Re::Shared_cap<L4Re::Dataspace> *ds)
00514 {
00515     *ds = L4Re::make_shared_cap<L4Re::Dataspace>(L4Re::virt_cap_alloc);
00516
00517     if (!ds->is_valid())
00518         return -ENOMEM;
00519
00520     int err;
00521     if ((err = Vfs_config::allocator()->alloc(size, ds->get())) < 0)
00522         return err;
00523
00524     DEBUG_LOG(debug_mmap, {
00525         outstring("ANON DS ALLOCATED: size=");
00526         outhex32(size);
00527         outstring(" cap = 0x");
00528         outhex32(ds->cap());
00529         outstring("\n");
00530     });
00531
00532     return 0;
00533 }
00534
00535 int
00536 Vfs::alloc_anon_mem(l4_umword_t size, L4Re::Shared_cap<L4Re::Dataspace> *ds,
00537                     l4_addr_t *offset)
00538 {

```

```

00539 #if !defined(CONFIG_MMU)
00540 // Small values for !MMU systems. These platforms do not have much memory
00541 // typically and the memory must be instantly allocated.
00542 enum
00543 {
00544     ANON_MEM_DS_POOL_SIZE = 256UL « 10, // size of a pool dataspace used for anon memory
00545     ANON_MEM_MAX_SIZE      = 32UL « 10, // chunk size that will be allocate a dataspace
00546 };
00547 #elif defined(USE_BIG_ANON_DS)
00548 enum
00549 {
00550     ANON_MEM_DS_POOL_SIZE = 256UL « 20, // size of a pool dataspace used for anon memory
00551     ANON_MEM_MAX_SIZE      = 32UL « 20, // chunk size that will be allocate a dataspace
00552 };
00553 #else
00554 enum
00555 {
00556     ANON_MEM_DS_POOL_SIZE = 256UL « 20, // size of a pool dataspace used for anon memory
00557     ANON_MEM_MAX_SIZE      = 0UL « 20,  // chunk size that will be allocate a dataspace
00558 };
00559 #endif
00560
00561 if (size >= ANON_MEM_MAX_SIZE)
00562 {
00563     int err;
00564     if ((err = alloc_ds(size, ds)) < 0)
00565         return err;
00566
00567     *offset = 0;
00568
00569     if (!_early_oom)
00570         return err;
00571
00572     return (*ds)->allocate(0, size);
00573 }
00574
00575 if (!_anon_ds.is_valid() || _anon_offset + size >= ANON_MEM_DS_POOL_SIZE)
00576 {
00577     int err;
00578     if ((err = alloc_ds(ANON_MEM_DS_POOL_SIZE, ds)) < 0)
00579         return err;
00580
00581     _anon_offset = 0;
00582     _anon_ds = *ds;
00583 }
00584 else
00585     *ds = _anon_ds;
00586
00587 if (_early_oom)
00588 {
00589     if (int err = (*ds)->allocate(_anon_offset, size))
00590         return err;
00591 }
00592
00593 *offset = _anon_offset;
00594 _anon_offset += size;
00595 return 0;
00596 }
00597
00598 int
00599 Vfs::mmap2(void *start, size_t len, int prot, int flags, int fd, off_t page4k_offset,
00600            void **resptr) L4_NOTHROW
00601 {
00602     DEBUG_LOG(debug_mmap, {
00603         outstring("MMAP params: ");
00604         outstring("start = 0x");
00605         outhex32(l4_addr_t(start));
00606         outstring(", len = 0x");
00607         outhex32(len);
00608         outstring(", prot = 0x");
00609         outhex32(prot);
00610         outstring(", flags = 0x");
00611         outhex32(flags);
00612         outstring(", offset = 0x");
00613         outhex32(page4k_offset);
00614         outstring("\n");
00615     });
00616
00617     using namespace L4Re;
00618     off64_t offset = l4_trunc_page(page4k_offset « 12);
00619
00620     if (flags & MAP_FIXED)
00621         if (l4_addr_t(start) & (L4_PAGESIZE - 1))
00622             return -EINVAL;
00623
00624     align_mmap_start_and_length(&start, &len);
00625

```

```

00626 // special code to just reserve an area of the virtual address space
00627 // Same behavior should be exposed when mapping with PROT_NONE. Mind that
00628 // PROT_NONE can only be specified exclusively, since it is defined to 0x0.
00629 if ((flags & 0x1000000) || (prot == PROT_NONE))
00630 {
00631     int err;
00632     L4::Cap<Rm> r = Env::env()->rm();
00633     l4_addr_t area = reinterpret_cast<l4_addr_t>(start);
00634     err = r->reserve_area(&area, len, L4Re::Rm::F::Search_addr);
00635     if (err < 0)
00636         return err;
00637
00638     *resptr = reinterpret_cast<void*>(area);
00639
00640     DEBUG_LOG(debug_mmap, {
00641         outstring(" MMAP reserved area: 0x");
00642         outhex32(area);
00643         outstring(" length= 0x");
00644         outhex32(len);
00645         outstring("\n");
00646     });
00647
00648     return 0;
00649 }
00650
00651 L4Re::Shared_cap<L4Re::Dataspace> ds;
00652 l4_addr_t anon_offset = 0;
00653 L4Re::Rm::Flags rm_flags(0);
00654
00655 if (flags & (MAP_ANONYMOUS | MAP_PRIVATE))
00656 {
00657     rm_flags |= L4Re::Rm::F::Detach_free;
00658
00659     int err = alloc_anon_mem(len, &ds, &anon_offset);
00660     if (err)
00661         return err;
00662
00663     DEBUG_LOG(debug_mmap, {
00664         outstring(" USE ANON MEM: 0x");
00665         outhex32(ds.cap());
00666         outstring(" offs = 0x");
00667         outhex32(anon_offset);
00668         outstring("\n");
00669     });
00670 }
00671
00672 if (!(flags & MAP_ANONYMOUS))
00673 {
00674     Ref_ptr<L4Re::Vfs::File> fi = fds.get(fd);
00675     if (!fi)
00676         return -EBADF;
00677
00678     L4::Cap<L4Re::Dataspace> fds = fi->data_space();
00679
00680     if (!fds.is_valid())
00681         return -EINVAL;
00682
00683     if (len + offset > l4_round_page(fds->size()))
00684         return -EINVAL;
00685
00686     if (flags & MAP_PRIVATE)
00687     {
00688         DEBUG_LOG(debug_mmap, outstring("COW\n"));
00689         int err = ds->copy_in(anon_offset, fds, offset, len);
00690         if (err == -L4_EINVAL)
00691         {
00692             L4::Cap<Rm> r = Env::env()->rm();
00693             Rm::Unique_region<char*> src;
00694             Rm::Unique_region<char*> dst;
00695             err = r->attach(&src, len,
00696                 L4Re::Rm::F::Search_addr | L4Re::Rm::F::R,
00697                 fds, offset);
00698             if (err < 0)
00699                 return err;
00700
00701             err = r->attach(&dst, len,
00702                 L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00703                 ds.get(), anon_offset);
00704             if (err < 0)
00705                 return err;
00706
00707             memcpy(dst.get(), src.get(), len);
00708         }
00709     }
00710     else if (err)
00711         return err;
00712
00713     offset = anon_offset;

```

```

00713     }
00714     else
00715     {
00716         L4Re::virt_cap_alloc->take(fds);
00717         ds = L4Re::Shared_cap<L4Re::Dataspace>(fds, L4Re::virt_cap_alloc);
00718     }
00719 }
00720 else
00721     offset = anon_offset;
00722
00723
00724 if (!(flags & MAP_FIXED) && start == 0)
00725     start = reinterpret_cast<void*>(L4_PAGESIZE);
00726
00727 char *data = static_cast<char *>(start);
00728 L4::Cap<Rm> r = Env::env()->rm();
00729 l4_addr_t overmap_area = L4_INVALID_ADDR;
00730
00731 int err;
00732 if (flags & MAP_FIXED)
00733 {
00734     overmap_area = l4_addr_t(start);
00735
00736     err = r->reserve_area(&overmap_area, len);
00737     if (err < 0)
00738         overmap_area = L4_INVALID_ADDR;
00739
00740     rm_flags |= Rm::F::In_area;
00741
00742     // Make sure to remove old mappings residing at the respective address
00743     // range. If none exists, we are fine as well, allowing us to ignore
00744     // ENOENT here.
00745     err = munmap_regions(start, len);
00746     if (err && err != -ENOENT)
00747         return err;
00748 }
00749
00750 if (!(flags & MAP_FIXED))
00751     rm_flags |= Rm::F::Search_addr;
00752 if (prot & PROT_READ)
00753     rm_flags |= Rm::F::R;
00754 if (prot & PROT_WRITE)
00755     rm_flags |= Rm::F::W;
00756 if (prot & PROT_EXEC)
00757     rm_flags |= Rm::F::X;
00758
00759 err = r->attach(&data, len, rm_flags,
00760               L4::Ipc::make_cap(ds.get(), (prot & PROT_WRITE)
00761                                ? L4_CAP_FPAGE_RW
00762                                : L4_CAP_FPAGE_RO),
00763               offset);
00764
00765 DEBUG_LOG(debug_mmap, {
00766     outstring("  MAPPED: 0x");
00767     outhex32(ds.cap());
00768     outstring("  addr: 0x");
00769     outhex32(l4_addr_t(data));
00770     outstring("  bytes: 0x");
00771     outhex32(len);
00772     outstring("  offset: 0x");
00773     outhex32(offset);
00774     outstring("  err = ");
00775     outdec(err);
00776     outstring("\n");
00777 });
00778
00779
00780 if (overmap_area != L4_INVALID_ADDR)
00781     r->free_area(overmap_area);
00782
00783 if (err < 0)
00784     return err;
00785
00786 l4_assert (!(start && !data));
00787
00788 // release ownership of the attached DS
00789 ds.release();
00790 *resptr = data;
00791
00792 return 0;
00793 }
00794
00795 namespace {
00796 class Auto_area
00797 {
00798 public:
00799     L4::Cap<L4Re::Rm> r;

```

```

00800     l4_addr_t a;
00801
00802     explicit Auto_area(L4::Cap<L4Re::Rm> r, l4_addr_t a = L4_INVALID_ADDR)
00803     : r(r), a(a) {}
00804
00805     int reserve(l4_addr_t _a, l4_size_t sz, L4Re::Rm::Flags flags)
00806     {
00807         free();
00808         a = _a;
00809         int e = r->reserve_area(&a, sz, flags);
00810         if (e)
00811             a = L4_INVALID_ADDR;
00812         return e;
00813     }
00814
00815     void free()
00816     {
00817         if (is_valid())
00818         {
00819             r->free_area(a);
00820             a = L4_INVALID_ADDR;
00821         }
00822     }
00823
00824     bool is_valid() const { return a != L4_INVALID_ADDR; }
00825
00826     ~Auto_area() { free(); }
00827 };
00828
00829
00830 int
00831 Vfs::mremap(void *old_addr, size_t old_size, size_t new_size, int flags,
00832             void **new_addr) L4_NOTHROW
00833 {
00834     using namespace L4Re;
00835
00836     DEBUG_LOG(debug_mmap, {
00837         outstring("Mremap: addr = 0x");
00838         outhex32((l4_umword_t)old_addr);
00839         outstring(" old_size = 0x");
00840         outhex32(old_size);
00841         outstring(" new_size = 0x");
00842         outhex32(new_size);
00843         outstring("\n");
00844     });
00845
00846     if (flags & MREMAP_FIXED && !(flags & MREMAP_MAYMOVE))
00847         return -EINVAL;
00848
00849     l4_addr_t oa = l4_trunc_page(reinterpret_cast<l4_addr_t>(old_addr));
00850     if (oa != reinterpret_cast<l4_addr_t>(old_addr))
00851         return -EINVAL;
00852
00853     bool const fixed = flags & MREMAP_FIXED;
00854     bool const maymove = flags & MREMAP_MAYMOVE;
00855
00856     L4::Cap<Rm> r = Env::env()->rm();
00857
00858     // sanitize input parameters to multiples of pages
00859     old_size = l4_round_page(old_size);
00860     new_size = l4_round_page(new_size);
00861
00862     if (!fixed)
00863     {
00864         if (new_size < old_size)
00865         {
00866             *new_addr = old_addr;
00867             return munmap(reinterpret_cast<void*>(oa + new_size),
00868                          old_size - new_size);
00869         }
00870
00871         if (new_size == old_size)
00872         {
00873             *new_addr = old_addr;
00874             return 0;
00875         }
00876     }
00877
00878     Auto_area old_area(r);
00879     int err = old_area.reserve(oa, old_size, L4Re::Rm::Flags(0));
00880     if (err < 0)
00881         return -EINVAL;
00882
00883     l4_addr_t pad_addr;
00884     Auto_area new_area(r);
00885     if (fixed)
00886     {

```

```

00887     l4_addr_t na = l4_trunc_page(reinterpret_cast<l4_addr_t>(*new_addr));
00888     if (na != reinterpret_cast<l4_addr_t>(*new_addr))
00889         return -EINVAL;
00890
00891     // check if the current virtual memory area can be expanded
00892     int err = new_area.reserve(na, new_size, L4Re::Rm::Flags(0));
00893     if (err < 0)
00894         return err;
00895
00896     pad_addr = na;
00897     // unmap all stuff and remap ours ....
00898 }
00899 else
00900 {
00901     l4_addr_t ta = oa + old_size;
00902     unsigned long ts = new_size - old_size;
00903     // check if the current virtual memory area can be expanded
00904     long err = new_area.reserve(ta, ts, L4Re::Rm::Flags(0));
00905     if (!maymove && err)
00906         return -ENOMEM;
00907
00908     L4Re::Rm::Offset toffs;
00909     L4Re::Rm::Flags tflags;
00910     L4::Cap<L4Re::Dataspace> tds;
00911
00912     err = r->find(&ta, &ts, &toffs, &tflags, &tds);
00913
00914     // there is enough space to expand the mapping in place
00915     if (err == -ENOENT || (err == 0 && (tflags & Rm::F::In_area)))
00916     {
00917         old_area.free(); // pad at the original address
00918         pad_addr = oa + old_size;
00919         *new_addr = old_addr;
00920     }
00921     else if (!maymove)
00922         return -ENOMEM;
00923     else
00924     {
00925         // search for a new area to remap
00926         err = new_area.reserve(0, new_size, Rm::F::Search_addr);
00927         if (err < 0)
00928             return -ENOMEM;
00929
00930         pad_addr = new_area.a + old_size;
00931         *new_addr = reinterpret_cast<void *>(new_area.a);
00932     }
00933 }
00934
00935 if (old_area.is_valid())
00936 {
00937     unsigned long size = old_size;
00938
00939     l4_addr_t a = old_area.a;
00940     unsigned long s = 1;
00941     L4Re::Rm::Offset o;
00942     L4Re::Rm::Flags f;
00943     L4::Cap<L4Re::Dataspace> ds;
00944
00945     while (r->find(&a, &s, &o, &f, &ds) >= 0 && !(f & Rm::F::In_area))
00946     {
00947         if (a < old_area.a)
00948         {
00949             auto d = old_area.a - a;
00950             a = old_area.a;
00951             s -= d;
00952             o += d;
00953         }
00954
00955         if (a + s > old_area.a + old_size)
00956             s = old_area.a + old_size - a;
00957
00958         l4_addr_t x = a - old_area.a + new_area.a;
00959
00960         int err = r->attach(&x, s, Rm::F::In_area | f,
00961                             L4::Ipc::make_cap(ds, f.cap_rights()), o);
00962         if (err < 0)
00963             return err;
00964
00965         // count the new attached ds reference
00966         L4Re::virt_cap_alloc->take(ds);
00967
00968         err = r->detach(a, s, &ds, This_task,
00969                        Rm::Detach_exact | Rm::Detach_keep);
00970         if (err < 0)
00971             return err;
00972
00973         switch (err & Rm::Detach_result_mask)

```

```

00974         {
00975             case Rm::Split_ds:
00976                 // add a reference as we split up a mapping
00977                 if (ds.is_valid())
00978                     L4Re::virt_cap_alloc->take(ds);
00979                 break;
00980             case Rm::Detached_ds:
00981                 if (ds.is_valid())
00982                     L4Re::virt_cap_alloc->release(ds);
00983                 break;
00984             default:
00985                 break;
00986         }
00987
00988         if (size <= s)
00989             break;
00990         a += s;
00991         size -= s;
00992         s = 1;
00993     }
00994
00995     old_area.free();
00996 }
00997
00998 if (old_size < new_size)
00999 {
01000     l4_addr_t const pad_sz = new_size - old_size;
01001     l4_addr_t toffs;
01002     L4Re::Shared_cap<L4Re::Dataspace> tds;
01003     int err = alloc_anon_mem(pad_sz, &tds, &toffs);
01004     if (err)
01005         return err;
01006
01007     // FIXME: must get the protection rights from the old
01008     // mapping and use the same here, for now just use RWX
01009     err = r->attach(&pad_addr, pad_sz,
01010                  Rm::F::In_area | Rm::F::Detach_free | Rm::F::RWX,
01011                  L4::Ipc::make_cap_rw(tds.get()), toffs);
01012     if (err < 0)
01013         return err;
01014
01015     // release ownership of tds, the region map is now the new owner
01016     tds.release();
01017 }
01018
01019 return 0;
01020 }
01021
01022 int
01023 Vfs::mprotect(const void * /* a */, size_t /* sz */, int prot) L4_NOTHROW
01024 {
01025     return (prot & PROT_WRITE) ? -1 : 0;
01026 }
01027
01028 int
01029 Vfs::msync(void *, size_t, int) L4_NOTHROW
01030 { return 0; }
01031
01032 int
01033 Vfs::madvise(void *, size_t, int) L4_NOTHROW
01034 { return 0; }
01035
01036 }
01037
01038 L4Re::Vfs::Ops *__rtld_l4re_env_posix_vfs_ops;
01039 extern void *l4re_env_posix_vfs_ops __attribute__((alias("__rtld_l4re_env_posix_vfs_ops"),
01040 visibility("default")));
01041
01042 namespace {
01043     class Real_mount_tree : public L4Re::Vfs::Mount_tree
01044     {
01045     public:
01046         explicit Real_mount_tree(char *n) : Mount_tree(n) {}
01047
01048         void *operator new (size_t size)
01049         { return __rtld_l4re_env_posix_vfs_ops->malloc(size); }
01050
01051         void operator delete (void *mem)
01052         { __rtld_l4re_env_posix_vfs_ops->free(mem); }
01053     };
01054
01055 int
01056 Vfs::mount(char const *path, cxx::Ref_ptr<L4Re::Vfs::File> const &dir) noexcept
01057 {
01058     using L4Re::Vfs::File;
01059     using L4Re::Vfs::Mount_tree;

```

```

01061     using L4Re::Vfs::Path;
01062
01063     cxx::Ref_ptr<Mount_tree> root = get_root()->mount_tree();
01064     if (!root)
01065         return -EINVAL;
01066
01067     cxx::Ref_ptr<Mount_tree> base;
01068     Path p = root->lookup(Path(path), &base);
01069
01070     while (!p.empty())
01071     {
01072         Path f = p.strip_first();
01073
01074         if (f.empty())
01075             return -EEXIST;
01076
01077         char *name = __rtld_l4re_env_posix_vfs_ops->strndup(f.path(), f.length());
01078         if (!name)
01079             return -ENOMEM;
01080
01081         auto nt = cxx::make_ref_obj<Real_mount_tree>(name);
01082         if (!nt)
01083         {
01084             __rtld_l4re_env_posix_vfs_ops->free(name);
01085             return -ENOMEM;
01086         }
01087
01088         base->add_child_node(nt);
01089         base = nt;
01090
01091         if (p.empty())
01092         {
01093             nt->mount(dir);
01094             return 0;
01095         }
01096     }
01097
01098     return -EINVAL;
01099 }
01100
01101 #undef DEBUG_LOG
01102 #undef GET_FILE_DBG
01103 #undef GET_FILE

```

## 16.223 vfs.h

```

00001 /*
00002  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019 #pragma once
00020
00021 #include <l4/sys/compiler.h>
00022
00023 #include <unistd.h>
00024 #include <stdarg.h>
00025 #include <fcntl.h>
00026 #include <sys/stat.h>
00027 #include <sys/mman.h>
00028 #include <sys/socket.h>
00029 #include <utime.h>
00030 #include <errno.h>
00031
00032 #ifndef AT_FDCWD
00033 #define AT_FDCWD -100
00034 #endif
00035
00036 #ifdef __cplusplus
00037

```



```

00038 #include <l4/sys/capability>
00039 #include <l4/re/cap_alloc>
00040 #include <l4/re/dataspace>
00041 #include <l4/cxx/pair>
00042 #include <l4/cxx/ref_ptr>
00043
00044 namespace L4Re {
00048 namespace Vfs {
00049
00050 class Mount_tree;
00051 class File;
00052
00062 class Generic_file
00063 {
00064 public:
00065     virtual ~Generic_file() noexcept = 0;
00077     virtual int unlock_all_locks() noexcept = 0;
00078
00087     virtual int fstat64(struct stat64 *buf) const noexcept = 0;
00088
00094     virtual int fchmod(mode_t) noexcept = 0;
00095
00105     virtual int get_status_flags() const noexcept = 0;
00106
00122     virtual int set_status_flags(long flags) noexcept = 0;
00123
00124     virtual int utime(const struct utimbuf *) noexcept = 0;
00125     virtual int utimes(const struct timeval [2]) noexcept = 0;
00126     virtual ssize_t readlink(char *, size_t) = 0;
00127 };
00128
00129 inline
00130 Generic_file::~Generic_file() noexcept
00131 {}
00132
00140 class Directory
00141 {
00142 public:
00143     virtual ~Directory() noexcept = 0;
00144
00158     virtual int faccessat(const char *path, int mode, int flags) noexcept = 0;
00159
00172     virtual int mkdir(const char *path, mode_t mode) noexcept = 0;
00173
00184     virtual int unlink(const char *path) noexcept = 0;
00185
00199     virtual int rename(const char *src_path, const char *dst_path) noexcept = 0;
00200
00214     virtual int link(const char *src_path, const char *dst_path) noexcept = 0;
00215
00228     virtual int symlink(const char *src_path, const char *dst_path) noexcept = 0;
00229
00240     virtual int rmdir(const char *path) noexcept = 0;
00241     virtual int openat(const char *path, int flags, mode_t mode,
00242                       cxx::Ref_ptr<File> *f) noexcept = 0;
00243
00244     virtual ssize_t getdents(char *buf, size_t sizebytes) noexcept = 0;
00245
00246     virtual int fchmodat(const char *pathname,
00247                         mode_t mode, int flags) noexcept = 0;
00248
00249     virtual int utimensat(const char *pathname,
00250                          const struct timespec times[2], int flags) noexcept = 0;
00251
00255     virtual int get_entry(const char *, int, mode_t, cxx::Ref_ptr<File> *) noexcept = 0;
00256 };
00257
00258 inline
00259 Directory::~Directory() noexcept
00260 {}
00261
00267 class Regular_file
00268 {
00269 public:
00270     virtual ~Regular_file() noexcept = 0;
00271
00282     virtual L4::Cap<L4Re::Dataspace> data_space() noexcept = 0;
00283
00293     virtual ssize_t readv(const struct iovec*, int iovcnt) noexcept = 0;
00294
00305     virtual ssize_t writev(const struct iovec*, int iovcnt) noexcept = 0;
00306
00307     virtual ssize_t preadv(const struct iovec *iov, int iovcnt, off64_t offset) noexcept = 0;
00308     virtual ssize_t pwritev(const struct iovec *iov, int iovcnt, off64_t offset) noexcept = 0;
00309
00317     virtual off64_t lseek64(off64_t, int) noexcept = 0;
00318

```

```

00319
00327 virtual int ftruncate64(off64_t pos) noexcept = 0;
00328
00334 virtual int fsync() const noexcept = 0;
00335
00341 virtual int fdatsync() const noexcept = 0;
00342
00352 virtual int get_lock(struct flock64 *lock) noexcept = 0;
00353
00362 virtual int set_lock(struct flock64 *lock, bool wait) noexcept = 0;
00363 };
00364
00365 inline
00366 Regular_file::~Regular_file() noexcept
00367 {}
00368
00369 class Socket
00370 {
00371 public:
00372     virtual ~Socket() noexcept = 0;
00373     virtual int bind(sockaddr const *, socklen_t) noexcept = 0;
00374     virtual int connect(sockaddr const *, socklen_t) noexcept = 0;
00375     virtual ssize_t send(void const *, size_t, int) noexcept = 0;
00376     virtual ssize_t recv(void *, size_t, int) noexcept = 0;
00377     virtual ssize_t sendto(void const *, size_t, int, sockaddr const *, socklen_t) noexcept = 0;
00378     virtual ssize_t recvfrom(void *, size_t, int, sockaddr *, socklen_t *) noexcept = 0;
00379     virtual ssize_t sendmsg(msghdr const *, int) noexcept = 0;
00380     virtual ssize_t recvmsg(msghdr *, int) noexcept = 0;
00381     virtual int getsockopt(int level, int opt, void *, socklen_t *) noexcept = 0;
00382     virtual int setsockopt(int level, int opt, void const *, socklen_t) noexcept = 0;
00383     virtual int listen(int) noexcept = 0;
00384     virtual int accept(sockaddr *addr, socklen_t *) noexcept = 0;
00385     virtual int shutdown(int) noexcept = 0;
00386
00387     virtual int getsockname(sockaddr *, socklen_t *) noexcept = 0;
00388     virtual int getpeername(sockaddr *, socklen_t *) noexcept = 0;
00389 };
00390
00391 inline
00392 Socket::~Socket() noexcept
00393 {}
00394
00400 class Special_file
00401 {
00402 public:
00403     virtual ~Special_file() noexcept = 0;
00404
00415     virtual int ioctl(unsigned long cmd, va_list args) noexcept = 0;
00416 };
00417
00418 inline
00419 Special_file::~Special_file() noexcept
00420 {}
00421
00435 class File :
00436     public Generic_file,
00437     public Regular_file,
00438     public Directory,
00439     public Special_file,
00440     public Socket
00441 {
00442     friend class Mount_tree;
00443
00444 private:
00445     void operator = (File const &);
00446
00447 protected:
00448     File() noexcept : _ref_cnt(0) {}
00449     File(File const &)
00450     : Generic_file(), Regular_file(), Directory(), Special_file(), _ref_cnt(0)
00451     {}
00452
00453 public:
00454
00455     const char *get_mount(const char *path, cxx::Ref_ptr<File> *dir,
00456                           cxx::Ref_ptr<Mount_tree> *mt = 0) noexcept;
00457
00458     int openat(const char *path, int flags, mode_t mode,
00459               cxx::Ref_ptr<File> *f) noexcept override;
00460
00461     void add_ref() noexcept { ++_ref_cnt; }
00462     int remove_ref() noexcept { return --_ref_cnt; }
00463
00464     virtual ~File() noexcept = 0;
00465
00466     cxx::Ref_ptr<Mount_tree> mount_tree() const noexcept
00467     { return _mount_tree; }

```

```

00468
00469     virtual int select(int nfd, fd_set *readfds, fd_set *writefds,
00470                       fd_set *exceptfds, struct timeval *timeout) throw() = 0;
00471
00472 private:
00473     int _ref_cnt;
00474     cxx::Ref_ptr<Mount_tree> _mount_tree;
00475
00476 };
00477
00478 inline
00479 File::~File() noexcept
00480 {}
00481
00482 class Path
00483 {
00484 private:
00485     char const *_p;
00486     unsigned _l;
00487 public:
00488     Path() noexcept : _p(0), _l(0) {}
00489
00490     explicit Path(char const *p) noexcept : _p(p)
00491     { for (_l = 0; *p; ++p, ++_l) ; }
00492
00493     Path(char const *p, unsigned l) noexcept : _p(p), _l(l)
00494     {}
00495
00496     static bool __is_sep(char s) noexcept;
00497
00498     Path cmp_path(char const *prefix) const noexcept;
00499
00500     struct Invalid_ptr;
00501     operator Invalid_ptr const * () const
00502     { return reinterpret_cast<Invalid_ptr const *>(_p); }
00503
00504     unsigned length() const { return _l; }
00505     char const *path() const { return _p; }
00506
00507     bool empty() const { return _l == 0; }
00508
00509     bool is_sep(unsigned offset) const { return __is_sep(_p[offset]); }
00510
00511     bool strip_sep()
00512     {
00513         bool s = false;
00514         for (; __is_sep(_p) && _l; ++_p, --_l)
00515             s = true;
00516         return s;
00517     }
00518
00519     Path first() const
00520     {
00521         unsigned i;
00522         for (i = 0; i < _l && !is_sep(i); ++i)
00523             ;
00524
00525         return Path(_p, i);
00526     }
00527
00528     Path strip_first()
00529     {
00530         Path r = first();
00531         _p += r.length();
00532         _l -= r.length();
00533         strip_sep();
00534         return r;
00535     }
00536
00537 };
00538
00539
00540
00541 class Mount_tree
00542 {
00543 public:
00544     explicit Mount_tree(char *n) noexcept;
00545
00546     Path lookup(Path const &path, cxx::Ref_ptr<Mount_tree> *mt,
00547               cxx::Ref_ptr<Mount_tree> *mp = 0) noexcept;
00548
00549     Path find(Path const &p, cxx::Ref_ptr<Mount_tree> *t) noexcept;
00550
00551     cxx::Ref_ptr<File> mount() const
00552     { return _mount; }
00553
00554
00555

```

```

00561 void mount(cxx::Ref_ptr<File> const &m)
00562 {
00563     m->_mount_tree = cxx::ref_ptr(this);
00564     _mount = m;
00565 }
00566
00567 static int create_tree(cxx::Ref_ptr<Mount_tree> const &root,
00568                       char const *path,
00569                       cxx::Ref_ptr<File> const &dir) noexcept;
00570
00571 void add_child_node(cxx::Ref_ptr<Mount_tree> const &cld);
00572
00573 virtual ~Mount_tree() noexcept = 0;
00574
00575 void add_ref() noexcept { ++_ref_cnt; }
00576 int remove_ref() noexcept { return --_ref_cnt; }
00577
00578 private:
00579     friend class Real_mount_tree;
00580
00581     int _ref_cnt;
00582     char *_name;
00583     cxx::Ref_ptr<Mount_tree> _cld;
00584     cxx::Ref_ptr<Mount_tree> _sib;
00585     cxx::Ref_ptr<File> _mount;
00586 };
00587
00588 inline
00589 Mount_tree::~Mount_tree() noexcept
00590 {}
00591
00592 inline bool
00593 Path::__is_sep(char s) noexcept
00594 { return s == '/'; }
00595
00596 inline Path
00597 Path::cmp_path(char const *n) const noexcept
00598 {
00599     char const *p = _p;
00600     for (; *p && !__is_sep(*p) && *n; ++p, ++n)
00601         if (*p != *n)
00602             return Path();
00603
00604     if (*n || (*p && !__is_sep(*p)))
00605         return Path();
00606
00607     return Path(p, _l - (p - _p));
00608 }
00609
00610 inline
00611 Mount_tree::Mount_tree(char *n) noexcept
00612 : _ref_cnt(0), _name(n)
00613 {}
00614
00615 inline Path
00616 Mount_tree::find(Path const &p, cxx::Ref_ptr<Mount_tree> *t) noexcept
00617 {
00618     if (!_cld)
00619         return Path();
00620
00621     for (cxx::Ref_ptr<Mount_tree> x = _cld; x; x = x->_sib)
00622     {
00623         Path const r = p.cmp_path(x->_name);
00624         if (r)
00625         {
00626             *t = x;
00627             return r;
00628         }
00629     }
00630
00631     return Path();
00632 }
00633
00634 inline Path
00635 Mount_tree::lookup(Path const &path, cxx::Ref_ptr<Mount_tree> *mt,
00636                   cxx::Ref_ptr<Mount_tree> *mp) noexcept
00637 {
00638     cxx::Ref_ptr<Mount_tree> x(this);
00639     Path p = path;
00640
00641     if (p.first().cmp_path("."))
00642         p.strip_first();
00643
00644     Path last_mp = p;
00645
00646     if (mp)
00647         *mp = x;;

```

```

00648
00649     while (1)
00650     {
00651         Path r = x->find(p, &x);
00652
00653         if (!r)
00654         {
00655             if (mp)
00656                 return last_mp;
00657
00658             if (mt)
00659                 *mt = x;
00660
00661             return p;
00662         }
00663
00664         r.strip_sep();
00665
00666         if (mp && x->_mount)
00667         {
00668             last_mp = r;
00669             *mp = x;
00670         }
00671
00672         if (r.empty())
00673         {
00674             if (mt)
00675                 *mt = x;
00676
00677             if (mp)
00678                 return last_mp;
00679             else
00680                 return r;
00681         }
00682
00683         p = r;
00684     }
00685 }
00686
00687 inline
00688 void
00689 Mount_tree::add_child_node(cxx::Ref_ptr<Mount_tree> const &cld)
00690 {
00691     cld->_sib = _cld;
00692     _cld = cld;
00693 }
00694
00695 inline
00696 const char *
00697 File::get_mount(const char *path, cxx::Ref_ptr<File> *dir,
00698                 cxx::Ref_ptr<Mount_tree> *mt) noexcept
00699 {
00700     if (!_mount_tree)
00701     {
00702         *dir = cxx::ref_ptr(this);
00703         return path;
00704     }
00705
00706     cxx::Ref_ptr<Mount_tree> mp;
00707     Path p = _mount_tree->lookup(Path(path), mt, &mp);
00708     if (mp->mount())
00709     {
00710         *dir = mp->mount();
00711         return p.path();
00712     }
00713     else
00714     {
00715         *dir = cxx::ref_ptr(this);
00716         return path;
00717     }
00718 }
00719
00720 inline int
00721 File::openat(const char *path, int flags, mode_t mode,
00722              cxx::Ref_ptr<File> *f) noexcept
00723 {
00724     cxx::Ref_ptr<File> dir;
00725     cxx::Ref_ptr<Mount_tree> mt;
00726     path = get_mount(path, &dir, &mt);
00727
00728     int res = dir->get_entry(path, flags, mode, f);
00729
00730     if (res < 0)
00731         return res;
00732
00733     if (!(*f)->_mount_tree && mt)
00734         (*f)->_mount_tree = mt;

```

```

00735
00736     return res;
00737 }
00738
00747 class Mman
00748 {
00749 public:
00751     virtual int mmap2(void *start, size_t len, int prot, int flags, int fd,
00752                      off_t offset, void **ptr) noexcept = 0;
00753
00755     virtual int munmap(void *start, size_t len) noexcept = 0;
00756
00758     virtual int mremap(void *old, size_t old_sz, size_t new_sz, int flags,
00759                      void **new_addr) noexcept = 0;
00760
00762     virtual int mprotect(const void *a, size_t sz, int prot) noexcept = 0;
00763
00765     virtual int msync(void *addr, size_t len, int flags) noexcept = 0;
00766
00768     virtual int madvise(void *addr, size_t len, int advice) noexcept = 0;
00769
00770     virtual ~Mman() noexcept = 0;
00771 };
00772
00773 inline
00774 Mman::~Mman() noexcept {}
00775
00776 class File_factory
00777 {
00778 private:
00779     int _ref_cnt = 0;
00780     int _proto = 0;
00781     char const *_proto_name = 0;
00782
00783     template<typename T> friend struct cxx::Default_ref_counter;
00784     void add_ref() noexcept { ++_ref_cnt; }
00785     int remove_ref() noexcept { return --_ref_cnt; }
00786
00787 public:
00788     explicit File_factory(int proto) : _proto(proto) {}
00789     explicit File_factory(char const *proto_name) : _proto_name(proto_name) {}
00790     File_factory(int proto, char const *proto_name)
00791     : _proto(proto), _proto_name(proto_name)
00792     {}
00793
00794     File_factory(File_factory const &) = delete;
00795     File_factory &operator = (File_factory const &) = delete;
00796
00797     char const *proto_name() const { return _proto_name; }
00798     int proto() const { return _proto; }
00799
00800     virtual ~File_factory() noexcept = 0;
00801     virtual cxx::Ref_ptr<File> create(L4::Cap<void> file) = 0;
00802 };
00803
00804 inline File_factory::~File_factory() noexcept {}
00805
00806 template<typename IFACE, typename IMPL>
00807 class File_factory_t : public File_factory
00808 {
00809 public:
00810     File_factory_t()
00811     : File_factory(IFACE::Protocol, L4::kobject_typeid<IFACE>()->name())
00812     {}
00813
00814     cxx::Ref_ptr<File> create(L4::Cap<void> file) override
00815     { return cxx::make_ref_obj<IMPL>(L4::cap_cast<IFACE>(file)); }
00816 };
00817
00831 class File_system
00832 {
00833 protected:
00834     File_system *_next;
00835
00836 public:
00837     File_system() noexcept : _next(0) {}
00843     virtual char const *type() const noexcept = 0;
00844
00861     virtual int mount(char const *source, unsigned long mountflags,
00862                     void const *data, cxx::Ref_ptr<File> *dir) noexcept = 0;
00863
00864     virtual ~File_system() noexcept = 0;
00865
00870     File_system *next() const noexcept { return _next; }
00871     File_system *&next() noexcept { return _next; }
00872     void next(File_system *n) noexcept { _next = n; }
00873 };

```

```

00874
00875 inline
00876 File_system::~File_system() noexcept
00877 {}
00878
00879 class File_system_list
00880 {
00881 public:
00882     class Iterator
00883     {
00884     public:
00885         explicit constexpr Iterator(File_system *c = nullptr) : _c(c) {}
00886
00887         Iterator &operator++()
00888         {
00889             if (_c)
00890                 _c = _c->next();
00891             return *this;
00892         }
00893
00894         bool operator==(Iterator const &other) const { return _c == other._c; }
00895         bool operator!=(Iterator const &other) const { return _c != other._c; }
00896         File_system *operator*() const { return _c; }
00897         File_system *operator->() const { return _c; }
00898
00899     private:
00900         File_system *_c;
00901     };
00902
00903     File_system_list(File_system *head) : _head(head) {}
00904
00905     constexpr Iterator begin() const
00906     { return Iterator(_head); }
00907
00908     constexpr Iterator end() const
00909     { return Iterator(); }
00910
00911     private:
00912         File_system *_head;
00913 };
00914
00920 class Fs
00921 {
00922 public:
00928     virtual cxx::Ref_ptr<File> get_file(int fd) noexcept = 0;
00929
00931     virtual cxx::Ref_ptr<File> get_root() noexcept = 0;
00932
00934     virtual cxx::Ref_ptr<File> get_cwd() noexcept { return get_root(); }
00935
00937     virtual void set_cwd(cxx::Ref_ptr<File> const &) noexcept {}
00938
00944     virtual int alloc_fd(cxx::Ref_ptr<File> const &f = cxx::Ref_ptr<>::Nil) noexcept = 0;
00945
00956     virtual cxx::Pair<cxx::Ref_ptr<File>, int>
00957         set_fd(int fd, cxx::Ref_ptr<File> const &f = cxx::Ref_ptr<>::Nil) noexcept = 0;
00958
00964     virtual cxx::Ref_ptr<File> free_fd(int fd) noexcept = 0;
00965
00973     virtual int mount(char const *path, cxx::Ref_ptr<File> const &dir) noexcept = 0;
00974
00982     virtual int register_file_system(File_system *f) noexcept = 0;
00983
00991     virtual int unregister_file_system(File_system *f) noexcept = 0;
00992
01000     virtual File_system *get_file_system(char const *fstype) noexcept = 0;
01001
01010     virtual File_system_list file_system_list() noexcept = 0;
01011
01015     int mount(char const *source, char const *target,
01016               char const *fstype, unsigned long mountflags,
01017               void const *data) noexcept;
01018
01019     virtual int register_file_factory(cxx::Ref_ptr<File_factory> f) noexcept = 0;
01020     virtual int unregister_file_factory(cxx::Ref_ptr<File_factory> f) noexcept = 0;
01021     virtual cxx::Ref_ptr<File_factory> get_file_factory(int proto) noexcept = 0;
01022     virtual cxx::Ref_ptr<File_factory> get_file_factory(char const *proto_name) noexcept = 0;
01023
01024     virtual ~Fs() = 0;
01025
01026     private:
01027         int mount_one(char const *source, char const *target,
01028                       File_system *fs, unsigned long mountflags,
01029                       void const *data) noexcept;
01030 };
01031
01032 inline int

```

```

01033 Fs::mount_one(char const *source, char const *target,
01034               File_system *fs, unsigned long mountflags,
01035               void const *data) noexcept
01036 {
01037     cxx::Ref_ptr<File> dir;
01038     int res = fs->mount(source, mountflags, data, &dir);
01039     if (res < 0)
01040         return res;
01041     return mount(target, dir);
01042 }
01043
01044 inline int
01045 Fs::mount(char const *source, char const *target,
01046           char const *fstype, unsigned long mountflags,
01047           void const *data) noexcept
01048 {
01049     if (fstype[0] == 'a'
01050         && fstype[1] == 'u'
01051         && fstype[2] == 't'
01052         && fstype[3] == 'o'
01053         && fstype[4] == 0)
01054     {
01055         File_system_list fsl = file_system_list();
01056         for (File_system_list::Iterator c = fsl.begin(); c != fsl.end(); ++c)
01057             if (mount_one(source, target, *c, mountflags, data) == 0)
01058                 return 0;
01059         return -ENODEV;
01060     }
01061     File_system *fs = get_file_system(fstype);
01062     if (!fs)
01063         return -ENODEV;
01064     return mount_one(source, target, fs, mountflags, data);
01065 }
01066
01067 inline
01068 Fs::~Fs()
01069 {}
01070
01071 class Ops : public Mman, public Fs
01072 {
01073 public:
01074     virtual void *malloc(size_t bytes) noexcept = 0;
01075     virtual void free(void *mem) noexcept = 0;
01076     virtual ~Ops() noexcept = 0;
01077
01078     char *strndup(char const *str, unsigned l) noexcept
01079     {
01080         unsigned len;
01081         for (len = 0; str[len] && len < l; ++len)
01082             ;
01083         if (len == 0)
01084             return nullptr;
01085         ++len;
01086         char *b = static_cast<char *>(this->malloc(len));
01087         if (b == nullptr)
01088             return nullptr;
01089         char *r = b;
01090         for (; len - 1 > 0 && *str; --len, ++b, ++str)
01091             *b = *str;
01092         *b = 0;
01093         return r;
01094     }
01095 };
01096
01097 inline
01098 Ops::~Ops() noexcept
01099 {}
01100
01101 #endif
01102

```



## 16.224 virtio-block

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2015-2020, 2022 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *           Manuel von Oltersdorff-Kalettka <manuel.kalettka@kernkonzept.com>
00007  *
00008  */
00009 #pragma once
00010
00011 #include <l4/sys/factory>
00012 #include <l4/sys/semaphore>
00013 #include <l4/re/dataspace>
00014 #include <l4/re/env>
00015 #include <l4/re/util/unique_cap>
00016 #include <l4/re/util/object_registry>
00017 #include <l4/re/error_helper>
00018
00019 #include <l4/util/atomic.h>
00020 #include <l4/util/bitops.h>
00021 #include <l4/l4virtio/client/l4virtio>
00022 #include <l4/l4virtio/l4virtio>
00023 #include <l4/l4virtio/virtqueue>
00024 #include <l4/l4virtio/virtio_block.h>
00025 #include <l4/sys/consts.h>
00026
00027 #include <cstring>
00028 #include <vector>
00029 #include <functional>
00030
00031 namespace L4virtio { namespace Driver {
00032
00036 class Block_device : public Device
00037 {
00038 public:
00039     typedef std::function<void(unsigned char)> Callback;
00040
00041 private:
00042     enum { Header_size = sizeof(l4virtio_block_header_t) };
00043
00044     struct Request
00045     {
00046         l4_uint16_t tail;
00047         Callback callback;
00048
00049         Request() : tail(Virtqueue::Eq), callback(0) {}
00050     };
00051
00052 public:
00056 class Handle
00057 {
00058     friend Block_device;
00059     l4_uint16_t head;
00060
00061     explicit Handle(l4_uint16_t descno) : head(descno) {}
00062
00063 public:
00064     Handle() : head(Virtqueue::Eq) {}
00065     bool valid() const { return head != Virtqueue::Eq; }
00066 };
00067
00092 void setup_device(L4::Cap<L4virtio::Device> srvcap, l4_size_t usermem,
00093                 void **userdata, Ptr<void> &user_devaddr,
00094                 L4::Cap<L4Re::Dataspace> qds = L4::Cap<L4Re::Dataspace>(),
00095                 l4_uint32_t fmask0 = -1U, l4_uint32_t fmask1 = -1U)
00096 {
00097     // Contact device.
00098     driver_connect(srvcap);
00099
00100     if (_config->device != L4VIRTIO_ID_BLOCK)
00101         L4Re::chksys(-L4_ENODEV, "Device is not a block device.");
00102
00103     if (_config->num_queues != 1)
00104         L4Re::chksys(-L4_EINVAL, "Invalid number of queues reported.");
00105
00106     // Memory is shared in one large dataspace which contains queues,
00107     // space for header/status and additional user-defined memory.
00108     unsigned queuesz = max_queue_size(0);
00109     l4_size_t totalsz = l4_round_page(usermem);
00110
00111     l4_uint64_t const header_offset =
00112         l4_round_size(_queue.total_size(queuesz),
00113                     l4util_bsr(aligned(l4virtio_block_header_t)));
00114     l4_uint64_t const status_offset = header_offset + queuesz * Header_size;
00115     l4_uint64_t const usermem_offset = l4_round_page(status_offset + queuesz);

```

```

00116
00117 // reserve space for one header/status per descriptor
00118 // TODO Should be reduced to 1/3 but this way no freelist is needed.
00119 totalsz += usermem_offset;
00120
00121 auto *e = L4Re::Env::env();
00122 if (!qds.is_valid())
00123 {
00124     _ds = L4Re::chkcap(L4Re::Util::make_unique_cap<L4Re::Dataspace>(),
00125                       "Allocate queue dataspace capability");
00126     L4Re::chksys(e->mem_alloc()->alloc(totalsz, _ds.get(),
00127                                       L4Re::Mem_alloc::Continuous
00128                                       | L4Re::Mem_alloc::Pinned),
00129                 "Allocate memory for virtio structures");
00130     _queue_ds = _ds.get();
00131 }
00132 else
00133 {
00134     if (qds->size() < totalsz)
00135         L4Re::chksys(-L4_EINVAL, "External queue dataspace too small.");
00136     _queue_ds = qds;
00137 }
00138
00139 // Now sort out which region goes where in the dataspace.
00140 L4Re::chksys(e->rm()->attach(&_queue_region, totalsz,
00141                             L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00142                             L4::Ipc::make_cap_rw(_queue_ds), 0,
00143                             L4_PAGESHIFT),
00144             "Attach dataspace for virtio structures");
00145
00146 l4_uint64_t devaddr;
00147 L4Re::chksys(register_ds(_queue_ds, 0, totalsz, &devaddr),
00148             "Register queue dataspace with device");
00149
00150 _queue.init_queue(queuesz, _queue_region.get());
00151
00152 config_queue(0, queuesz, devaddr, devaddr + _queue.avail_offset(),
00153             devaddr + _queue.used_offset());
00154
00155 _header_addr = devaddr + header_offset;
00156 _headers = reinterpret_cast<l4virtio_block_header_t *>(_queue_region.get()
00157                                                         + header_offset);
00158
00159 _status_addr = devaddr + status_offset;
00160 _status = _queue_region.get() + status_offset;
00161
00162 user_devaddr = Ptr<void>(devaddr + usermem_offset);
00163 if (userdata)
00164     *userdata = _queue_region.get() + usermem_offset;
00165
00166 // setup the callback mechanism
00167 _pending.assign(queuesz, Request());
00168
00169 // Finish handshake with device.
00170 _config->driver_features_map[0] = fmask0;
00171 _config->driver_features_map[1] = fmask1;
00172 driver_acknowledge();
00173 }
00174
00175 l4virtio_block_config_t const &device_config() const
00176 {
00177     return *_config->device_config<l4virtio_block_config_t>();
00178 }
00179
00180
00181
00182
00191 Handle start_request(l4_uint64_t sector, l4_uint32_t type,
00192                     Callback callback)
00193 {
00194     l4_uint16_t descno = _queue.alloc_descriptor();
00195     if (descno == Virtqueue::Eoq)
00196         return Handle(Virtqueue::Eoq);
00197
00198     L4virtio::Virtqueue::Desc &desc = _queue.desc(descno);
00199     Request &req = _pending[descno];
00200
00201     // setup the header
00202     l4virtio_block_header_t &head = _headers[descno];
00203     head.type = type;
00204     head.ioprio = 0;
00205     head.sector = sector;
00206
00207     // and put it in the descriptor
00208     desc.addr = Ptr<void>(_header_addr + descno * Header_size);
00209     desc.len = Header_size;
00210     desc.flags.raw = 0; // no write, no indirect
00211
00212     req.tail = descno;
00213     req.callback = callback;

```

```

00214
00215     return Handle(descno);
00216 }
00217
00229 int add_block(Handle handle, Ptr<void> addr, l4_uint32_t size)
00230 {
00231     l4_uint16_t descno = _queue.alloc_descriptor();
00232     if (descno == Virtqueue::Eoq)
00233         return -L4_EAGAIN;
00234
00235     Request &req = _pending[handle.head];
00236     L4virtio::Virtqueue::Desc &desc = _queue.desc(descno);
00237     L4virtio::Virtqueue::Desc &prev = _queue.desc(req.tail);
00238
00239     prev.next = descno;
00240     prev.flags.next() = true;
00241
00242     desc.addr = addr;
00243     desc.len = size;
00244     desc.flags.raw = 0;
00245     if (_headers[handle.head].type > 0) // write or flush request
00246         desc.flags.write() = true;
00247
00248     req.tail = descno;
00249
00250     return L4_EOK;
00251 }
00252
00265 int send_request(Handle handle)
00266 {
00267     // add the status bit
00268     auto descno = _queue.alloc_descriptor();
00269     if (descno == Virtqueue::Eoq)
00270         return -L4_EAGAIN;
00271
00272     Request &req = _pending[handle.head];
00273     L4virtio::Virtqueue::Desc &desc = _queue.desc(descno);
00274     L4virtio::Virtqueue::Desc &prev = _queue.desc(req.tail);
00275
00276     prev.next = descno;
00277     prev.flags.next() = true;
00278
00279     desc.addr = Ptr<void>(_status_addr + descno);
00280     desc.len = 1;
00281     desc.flags.raw = 0;
00282     desc.flags.write() = true;
00283
00284     req.tail = descno;
00285
00286     send(_queue, handle.head);
00287
00288     return L4_EOK;
00289 }
00290
00306 int process_request(Handle handle)
00307 {
00308     // add the status bit
00309     auto descno = _queue.alloc_descriptor();
00310     if (descno == Virtqueue::Eoq)
00311         return -L4_EAGAIN;
00312
00313     L4virtio::Virtqueue::Desc &desc = _queue.desc(descno);
00314     L4virtio::Virtqueue::Desc &prev = _queue.desc(_pending[handle.head].tail);
00315
00316     prev.next = descno;
00317     prev.flags.next() = true;
00318
00319     desc.addr = Ptr<void>(_status_addr + descno);
00320     desc.len = 1;
00321     desc.flags.raw = 0;
00322     desc.flags.write() = true;
00323
00324     _pending[handle.head].tail = descno;
00325
00326     int ret = send_and_wait(_queue, handle.head);
00327     unsigned char status = _status[descno];
00328     free_request(handle);
00329
00330     if (ret < 0)
00331         return ret;
00332
00333     switch (status)
00334     {
00335     case L4VIRTIO_BLOCK_S_OK: return L4_EOK;
00336     case L4VIRTIO_BLOCK_S_IOERR: return -L4_EIO;
00337     case L4VIRTIO_BLOCK_S_UNSUPP: return -L4_ENOSYS;
00338     }

```

```

00339
00340     return -L4_EINVAL;
00341 }
00342
00343 void free_request(Handle handle)
00344 {
00345     if (handle.head != Virtqueue::Eq
00346         && _pending[handle.head].tail != Virtqueue::Eq)
00347         _queue.free_descriptor(handle.head, _pending[handle.head].tail);
00348     _pending[handle.head].tail = Virtqueue::Eq;
00349 }
00350
00351 void process_used_queue()
00352 {
00353     for (l4_uint16_t descno = _queue.find_next_used();
00354          descno != Virtqueue::Eq;
00355          descno = _queue.find_next_used()
00356              )
00357     {
00358         if (descno >= _queue.num() || _pending[descno].tail == Virtqueue::Eq)
00359             L4Re::chksys(-L4_ENOSYS, "Bad descriptor number");
00360
00361         unsigned char status = _status[descno];
00362         free_request(Handle(descno));
00363
00364         if (_pending[descno].callback)
00365             _pending[descno].callback(status);
00366     }
00367 }
00368
00369 protected:
00370     L4Re::Util::Unique_cap<L4Re::Dataspace> _ds;
00371     L4::Cap<L4Re::Dataspace> _queue_ds;
00372
00373 private:
00374     L4Re::Rm::Unique_region<unsigned char *> _queue_region;
00375     l4virtio_block_header_t *_headers;
00376     unsigned char *_status;
00377     l4_uint64_t _header_addr;
00378     l4_uint64_t _status_addr;
00379     Virtqueue _queue;
00380     std::vector<Request> _pending;
00381 };
00382
00383 }
00384 }

```

## 16.225 virtio-block

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2017-2021 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  */
00007 #pragma once
00008
00009 #include <l4/cxx/unique_ptr>
00010 #include <l4/re/util/unique_cap>
00011
00012 #include <climits>
00013
00014 #include <l4/l4virtio/virtio.h>
00015 #include <l4/l4virtio/virtio_block.h>
00016 #include <l4/l4virtio/server/l4virtio>
00017 #include <l4/sys/cxx/ipc_epiface>
00018
00019 namespace L4virtio { namespace Svr {
00020
00021     template <typename Ds_data> class Block_dev_base;
00022
00023     template <typename Ds_data>
00024     class Block_request
00025     {
00026     public:
00027         friend class Block_dev_base<Ds_data>;
00028         enum { Header_size = sizeof(l4virtio_block_header_t) };
00029
00030     public:
00031         struct Data_block
00032         {
00033             Driver_mem_region_t<Ds_data> *mem;
00034             void *addr;
00035             l4_uint32_t len;
00036         };
00037     };
00038 }
00039 }

```

```

00042
00043     Data_block() = default;
00044
00045     Data_block(Driver_mem_region_t<Ds_data> *m, Virtqueue::Desc const &desc,
00046               Request_processor const *)
00047     : mem(m), addr(m->local(desc.addr)), len(desc.len)
00048     {}
00049 };
00050
00051
00052
00063 unsigned data_size() const
00064 {
00065     Request_processor rp;
00066     Data_block data;
00067
00068     rp.start(_mem_list, _request, &data);
00069
00070     unsigned total = data.len;
00071
00072     try
00073     {
00074         while (rp.has_more())
00075         {
00076             rp.next(_mem_list, &data);
00077             total += data.len;
00078         }
00079     }
00080     catch (Bad_descriptor const &e)
00081     {
00082         // need to convert the exception because e contains a raw pointer to rp
00083         throw L4::Runtime_error(-L4_EIO, "bad virtio descriptor");
00084     }
00085
00086     if (total < Header_size + 1)
00087         throw L4::Runtime_error(-L4_EIO, "virtio request too short");
00088
00089     return total - Header_size - 1;
00090 }
00091
00095 bool has_more()
00096 {
00097     // peek into the remaining data
00098     while (_data.len == 0 && _rp.has_more())
00099         _rp.next(_mem_list, &_data);
00100
00101     // there always must be one byte left for status
00102     return (_data.len > 1 || _rp.has_more());
00103 }
00104
00113 Data_block next_block()
00114 {
00115     Data_block out;
00116
00117     if (_data.len == 0)
00118     {
00119         if (!_rp.has_more())
00120             throw L4::Runtime_error(-L4_EEXIST,
00121                                     "No more data blocks in virtio request");
00122
00123         if (_todo_blocks == 0)
00124             throw Bad_descriptor(&_rp, Bad_descriptor::Bad_size);
00125         --_todo_blocks;
00126
00127         _rp.next(_mem_list, &_data);
00128     }
00129
00130     if (_data.len > _max_block_size)
00131         throw Bad_descriptor(&_rp, Bad_descriptor::Bad_size);
00132
00133     out = _data;
00134
00135     if (!_rp.has_more())
00136     {
00137         --(out.len);
00138         _data.len = 1;
00139         _data.addr = static_cast<char *>(_data.addr) + out.len;
00140     }
00141     else
00142         _data.len = 0; // is consumed
00143
00144     return out;
00145 }
00146
00148 l4virtio_block_header_t const &header() const
00149 { return _header; }
00150

```

```

00151 private:
00152     Block_request(Virtqueue::Request req, Driver_mem_list_t<Ds_data> *mem_list,
00153                   unsigned max_blocks, l4_uint32_t max_block_size)
00154     : _mem_list(mem_list),
00155       _request(req),
00156       _todo_blocks(max_blocks),
00157       _max_block_size(max_block_size)
00158     {
00159         // read header which should be in the first block
00160         _rp.start(mem_list, _request, &_data);
00161         --_todo_blocks;
00162
00163         if (_data.len < Header_size)
00164             throw Bad_descriptor(&_rp, Bad_descriptor::Bad_size);
00165
00166         _header = *(static_cast<l4virtio_block_header_t *>(_data.addr));
00167         _data.addr = static_cast<char *>(_data.addr) + Header_size;
00168         _data.len -= Header_size;
00169
00170         // if there is no space for status bit we cannot really recover
00171         if (!_rp.has_more() && _data.len == 0)
00172             throw Bad_descriptor(&_rp, Bad_descriptor::Bad_size);
00173     }
00174
00175 int release_request(Virtqueue *queue, l4_uint8_t status, unsigned sz)
00176 {
00177     // write back status
00178     // If there was an error on the way or the status byte is in its
00179     // own block, fast-forward to the last block.
00180     if (_rp.has_more())
00181     {
00182         while (_rp.next(_mem_list, &_data) && _todo_blocks > 0)
00183             --_todo_blocks;
00184
00185         if (_todo_blocks > 0 && _data.len > 0)
00186             *(static_cast<l4_uint8_t *>(_data.addr) + _data.len - 1) = status;
00187         else
00188             return -L4_EIO; // too many data blocks
00189     }
00190     else if (_data.len > 0)
00191         *(static_cast<l4_uint8_t *>(_data.addr)) = status;
00192     else
00193         return -L4_EIO; // no space for final status byte
00194
00195     // now release the head
00196     queue->consumed(_request, sz);
00197
00198     return L4_EOK;
00199 }
00200
00201 Driver_mem_list_t<Ds_data> *_mem_list;
00202 l4virtio_block_header_t _header;
00203 Request_processor _rp;
00204 Data_block _data;
00205
00206 Virtqueue::Request _request;
00207 unsigned _todo_blocks;
00208 l4_uint32_t _max_block_size;
00209 };
00210
00211 struct Block_features : public Dev_config::Features
00212 {
00213     Block_features() = default;
00214     Block_features(l4_uint32_t raw) : Dev_config::Features(raw) {}
00215
00216     CXX_BITFIELD_MEMBER( 1, 1, size_max, raw);
00217     CXX_BITFIELD_MEMBER( 2, 2, seg_max, raw);
00218     CXX_BITFIELD_MEMBER( 4, 4, geometry, raw);
00219     CXX_BITFIELD_MEMBER( 5, 5, ro, raw);
00220     CXX_BITFIELD_MEMBER( 6, 6, blk_size, raw);
00221     CXX_BITFIELD_MEMBER( 9, 9, flush, raw);
00222     CXX_BITFIELD_MEMBER(10, 10, topology, raw);
00223     CXX_BITFIELD_MEMBER(11, 11, config_wce, raw);
00224     CXX_BITFIELD_MEMBER(13, 13, discard, raw);
00225     CXX_BITFIELD_MEMBER(14, 14, write_zeroes, raw);
00226 };
00227
00228 template <typename Ds_data>
00229 class Block_dev_base : public L4virtio::Svr::Device_t<Ds_data>
00230 {
00231 private:
00232     L4Re::Util::Unique_cap<L4::Irq> _kick_guest_irq;
00233     Virtqueue _queue;
00234     unsigned _vq_max;
00235     l4_uint32_t _max_block_size = UINT_MAX;

```

```

00264     Dev_config_t<l4virtio_block_config_t> _dev_config;
00265
00266 public:
00267     typedef Block_request<Ds_data> Request;
00268
00269 protected:
00270     Block_features negotiated_features() const
00271     { return _dev_config.negotiated_features(0); }
00272
00273     Block_features device_features() const
00274     { return _dev_config.host_features(0); }
00275
00276     void set_device_features(Block_features df)
00277     { _dev_config.host_features(0) = df.raw; }
00278
00288     void set_size_max(l4_uint32_t sz)
00289     {
00290         _dev_config.priv_config()->size_max = sz;
00291         Block_features df = device_features();
00292         df.size_max() = true;
00293         set_device_features(df);
00294
00295         _max_block_size = sz;
00296     }
00297
00302     void set_seg_max(l4_uint32_t sz)
00303     {
00304         _dev_config.priv_config()->seg_max = sz;
00305         Block_features df = device_features();
00306         df.seg_max() = true;
00307         set_device_features(df);
00308     }
00309
00313     void set_geometry(l4_uint16_t cylinders, l4_uint8_t heads, l4_uint8_t sectors)
00314     {
00315         l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00316         pc->geometry.cylinders = cylinders;
00317         pc->geometry.heads = heads;
00318         pc->geometry.sectors = sectors;
00319         Block_features df = device_features();
00320         df.geometry() = true;
00321         set_device_features(df);
00322     }
00323
00330     void set_blk_size(l4_uint32_t sz)
00331     {
00332         _dev_config.priv_config()->blk_size = sz;
00333         Block_features df = device_features();
00334         df.blk_size() = true;
00335         set_device_features(df);
00336     }
00337
00346     void set_topology(l4_uint8_t physical_block_exp,
00347                     l4_uint8_t alignment_offset,
00348                     l4_uint32_t min_io_size,
00349                     l4_uint32_t opt_io_size)
00350     {
00351         l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00352         pc->topology.physical_block_exp = physical_block_exp;
00353         pc->topology.alignment_offset = alignment_offset;
00354         pc->topology.min_io_size = min_io_size;
00355         pc->topology.opt_io_size = opt_io_size;
00356         Block_features df = device_features();
00357         df.topology() = true;
00358         set_device_features(df);
00359     }
00360
00362     void set_flush()
00363     {
00364         Block_features df = device_features();
00365         df.flush() = true;
00366         set_device_features(df);
00367     }
00368
00373     void set_config_wce(l4_uint8_t writeback)
00374     {
00375         l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00376         pc->writeback = writeback;
00377         Block_features df = device_features();
00378         df.config_wce() = true;
00379         set_device_features(df);
00380     }
00381
00386     l4_uint8_t get_writeback()
00387     {
00388         l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00389         return pc->writeback;

```

```

00390     }
00391
00400 void set_discard(l4_uint32_t max_discard_sectors, l4_uint32_t max_discard_seg,
00401                 l4_uint32_t discard_sector_alignment)
00402 {
00403     l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00404     pc->max_discard_sectors = max_discard_sectors;
00405     pc->max_discard_seg = max_discard_seg;
00406     pc->discard_sector_alignment = discard_sector_alignment;
00407     Block_features df = device_features();
00408     df.discard() = true;
00409     set_device_features(df);
00410 }
00411
00420 void set_write_zeroes(l4_uint32_t max_write_zeroes_sectors,
00421                      l4_uint32_t max_write_zeroes_seg,
00422                      l4_uint8_t write_zeroes_may_unmap)
00423 {
00424     l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00425     pc->max_write_zeroes_sectors = max_write_zeroes_sectors;
00426     pc->max_write_zeroes_seg = max_write_zeroes_seg;
00427     pc->write_zeroes_may_unmap = write_zeroes_may_unmap;
00428     Block_features df = device_features();
00429     df.write_zeroes() = true;
00430     set_device_features(df);
00431 }
00432
00433 public:
00442 Block_dev_base(l4_uint32_t vendor, unsigned queue_size, l4_uint64_t capacity,
00443               bool read_only)
00444 : L4virtio::Svr::Device_t<Ds_data>(&_dev_config),
00445   _vq_max(queue_size),
00446   _dev_config(vendor, L4VIRTIO_ID_BLOCK, 1)
00447 {
00448     this->reset_queue_config(0, queue_size);
00449
00450     Block_features df(0);
00451     df.ring_indirect_desc() = true;
00452     df.ro() = read_only;
00453     set_device_features(df);
00454
00455     _dev_config.set_host_feature(L4VIRTIO_FEATURE_VERSION_1);
00456
00457     _dev_config.priv_config()->capacity = capacity;
00458 }
00459
00463 virtual void reset_device() = 0;
00464
00468 virtual bool queue_stopped() = 0;
00469
00481 void finalize_request(cxx::unique_ptr<Request> req, unsigned sz,
00482                     l4_uint8_t status = L4VIRTIO_BLOCK_S_OK)
00483 {
00484     if (_dev_config.status().fail_state() || !_queue.ready())
00485         return;
00486
00487     if (req->release_request(&_queue, status, sz) < 0)
00488         this->device_error();
00489
00490     if (_queue.no_notify_guest())
00491         return;
00492
00493     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00494     _kick_guest_irq->trigger();
00495
00496     // Request can be dropped here.
00497 }
00498
00499 int reconfig_queue(unsigned idx) override
00500 {
00501     if (idx == 0 && this->setup_queue(&_queue, 0, _vq_max))
00502         return 0;
00503
00504     return -L4_EINVAL;
00505 }
00506
00507 void reset() override
00508 {
00509     _queue.disable();
00510     _dev_config.reset_queue(0, _vq_max);
00511     _dev_config.reset_hdr();
00512     reset_device();
00513 }
00514
00515 protected:
00516 bool check_for_new_requests()
00517 {

```



```

00518     if (!_queue.ready() || queue_stopped())
00519         return false;
00520
00521     if (_dev_config.status().fail_state())
00522         return false;
00523
00524     return _queue.desc_avail();
00525 }
00526
00528 cxx::unique_ptr<Request> get_request()
00529 {
00530     cxx::unique_ptr<Request> req;
00531
00532     if (!_queue.ready() || queue_stopped())
00533         return req;
00534
00535     if (_dev_config.status().fail_state())
00536         return req;
00537
00538     auto r = _queue.next_avail();
00539     if (!r)
00540         return req;
00541
00542     try
00543     {
00544         cxx::unique_ptr<Request> cur{
00545             new Request(r, &(this->_mem_info), _vq_max, _max_block_size)};
00546
00547         req = cxx::move(cur);
00548     }
00549     catch (Bad_descriptor const &e)
00550     {
00551         this->device_error();
00552         return req;
00553     }
00554
00555     return req;
00556 }
00557
00558 private:
00559 void register_single_driver_irq() override
00560 {
00561     _kick_guest_irq = L4Re::Util::Unique_cap<L4::Irq>(
00562         L4Re::chkcapi(this->server_iface()->template rcv_cap<L4::Irq>(0)));
00563
00564     L4Re::chksys(this->server_iface()->realloc_rcv_cap(0));
00565 }
00566
00567 void trigger_driver_config_irq() override
00568 {
00569     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_CONFIG);
00570     _kick_guest_irq->trigger();
00571 }
00572
00573 bool check_queues() override
00574 {
00575     if (!_queue.ready())
00576     {
00577         reset();
00578         return false;
00579     }
00580
00581     return true;
00582 }
00583 };
00584
00585 template <typename Ds_data>
00586 struct Block_dev
00587 : Block_dev_base<Ds_data>,
00588   L4::Epiface_t<Block_dev<Ds_data>, L4virtio::Device>
00589 {
00590 private:
00591     class Irq_object : public L4::Irqep_t<Irq_object>
00592     {
00593     public:
00594         Irq_object(Block_dev<Ds_data> *parent) : _parent(parent) {}
00595
00596         void handle_irq()
00597         {
00598             _parent->kick();
00599         }
00600
00601     private:
00602         Block_dev<Ds_data> *_parent;
00603     };
00604     Irq_object _irq_handler;
00605

```

```

00606 protected:
00607     L4::Epiface *irq_iface()
00608     { return &_irq_handler; }
00609
00610 public:
00611     Block_dev(l4_uint32_t vendor, unsigned queue_size, l4_uint64_t capacity,
00612              bool read_only)
00613     : Block_dev_base<Ds_data>(vendor, queue_size, capacity, read_only),
00614       _irq_handler(this)
00615     {}
00616
00627     L4::Cap<void> register_obj(L4::Registry_iface *registry,
00628                               char const *service = 0)
00629     {
00630         L4Re::chkcap(registry->register_irq_obj(this->irq_iface()));
00631         L4::Cap<void> ret;
00632         if (service)
00633             ret = registry->register_obj(this, service);
00634         else
00635             ret = registry->register_obj(this);
00636         L4Re::chkcap(ret);
00637
00638         return ret;
00639     }
00640
00641     L4::Cap<void> register_obj(L4::Registry_iface *registry,
00642                               L4::Cap<L4::Rcv_endpoint> ep)
00643     {
00644         L4Re::chkcap(registry->register_irq_obj(this->irq_iface()));
00645
00646         return L4Re::chkcap(registry->register_obj(this, ep));
00647     }
00648
00649     typedef Block_request<Ds_data> Request;
00664     virtual bool process_request(cxx::unique_ptr<Request> &&req) = 0;
00665
00666 protected:
00667     L4::Ipc_svr::Server_iface *server_iface() const override
00668     {
00669         return this->L4::Epiface::server_iface();
00670     }
00671
00672     void kick()
00673     {
00674         for (;;)
00675         {
00676             auto req = this->get_request();
00677             if (!req)
00678                 return;
00679             if (!this->process_request(cxx::move(req)))
00680                 return;
00681         }
00682     }
00683
00684 private:
00685     L4::Cap<L4::Irq> device_notify_irq() const override
00686     {
00687         return L4::cap_cast<L4::Irq>(_irq_handler.obj_cap());
00688     }
00689 };
00690
00691 } }

```

## 16.226 virtio-net

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2022 Kernkonzept GmbH.
00005  * Author(s): Stephan Gerhold <stephan.gerhold@kernkonzept.com>
00006  */
00007 #pragma once
00008
00009 #include <cstring>
00010 #include <functional>
00011
00012 #include <l4/cxx/exceptions>
00013 #include <l4/cxx/minmax>
00014 #include <l4/re/dataspace>
00015 #include <l4/re/env>
00016 #include <l4/re/error_helper>
00017 #include <l4/re/util/unique_cap>
00018 #include <l4/sys/consts.h>

```

```

00019
00020 #include <l4/l4virtio/client/l4virtio>
00021 #include <l4/l4virtio/l4virtio>
00022 #include <l4/l4virtio/virtio_net.h>
00023 #include <l4/l4virtio/virtqueue>
00024
00025 namespace L4virtio { namespace Driver {
00026
00030 class Virtio_net_device : public L4virtio::Driver::Device
00031 {
00032 public:
00037 struct Packet
00038 {
00039     l4virtio_net_header_t hdr;
00040     l4_uint8_t data[1500 + 14]; /* MTU + Ethernet header */
00041 };
00042
00047 int rx_queue_size() const
00048 { return max_queue_size(0); }
00049
00054 int tx_queue_size() const
00055 { return max_queue_size(1); }
00056
00066 void setup_device(L4::Cap<L4virtio::Device> srvcap)
00067 {
00068     // Contact device.
00069     driver_connect(srvcap);
00070
00071     if (_config->device != L4VIRTIO_ID_NET)
00072         L4Re::chksys(-L4_ENODEV, "Device is not a network device.");
00073
00074     if (_config->num_queues < 2)
00075         L4Re::chksys(-L4_EINVAL, "Invalid number of queues reported.");
00076
00077     auto rxqsz = rx_queue_size();
00078     auto txqsz = tx_queue_size();
00079
00080     // Allocate memory for RX/TX queue and RX/TX packet buffers
00081     auto rxqoff = 0;
00082     auto txqoff = l4_round_size(rxqoff + rxqsz * _rxq.total_size(rxqsz),
00083                                L4virtio::Virtqueue::Desc_align);
00084     auto rxpktoff = l4_round_size(txqoff + txqsz * _txq.total_size(txqsz),
00085                                L4virtio::Virtqueue::Desc_align);
00086     auto txpktoff = rxpktoff + rxqsz * sizeof(Packet);
00087     auto totalsz = txpktoff + txqsz * sizeof(Packet);
00088
00089     _queue_ds = L4Re::chkcapi(L4Re::Util::make_unique_cap<L4Re::Dataspace>(),
00090                              "Allocate queue dataspace capability");
00091     auto *e = L4Re::Env::env();
00092     L4Re::chksys(e->mem_alloc()->alloc(totalsz, _queue_ds.get(),
00093                                       L4Re::Mem_alloc::Continuous
00094                                       | L4Re::Mem_alloc::Pinned),
00095                 "Allocate memory for virtio structures");
00096
00097     L4Re::chksys(e->rm()->attach(&_queue_region, totalsz,
00098                                L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00099                                L4::Ipc::make_cap_rw(_queue_ds.get(), 0,
00100                                L4_PAGESHIFT),
00101                                "Attach dataspace for virtio structures");
00102
00103     l4_uint64_t devaddr;
00104     L4Re::chksys(register_ds(_queue_ds.get(), 0, totalsz, &devaddr),
00105                 "Register queue dataspace with device");
00106
00107     _rxq.init_queue(rxqsz, _queue_region.get() + rxqoff);
00108     _txq.init_queue(txqsz, _queue_region.get() + txqoff);
00109
00110     config_queue(0, rxqsz, devaddr + rxqoff,
00111                 devaddr + rxqoff + _rxq.avail_offset(),
00112                 devaddr + rxqoff + _rxq.used_offset());
00113     config_queue(1, txqsz, devaddr + txqoff,
00114                 devaddr + txqoff + _txq.avail_offset(),
00115                 devaddr + txqoff + _txq.used_offset());
00116
00117     _rxpkts = reinterpret_cast<Packet*>(_queue_region.get() + rxpktoff);
00118     _txpkts = reinterpret_cast<Packet*>(_queue_region.get() + txpktoff);
00119
00120     // Prepare descriptors to save work later
00121     for (l4_uint16_t descno = 0; descno < rxqsz; ++descno)
00122     {
00123         auto &desc = _rxq.desc(descno);
00124         desc.addr = L4virtio::Ptr<void>(devaddr + rxpktoff +
00125                                         descno * sizeof(Packet));
00126         desc.len = sizeof(Packet);
00127         desc.flags.write() = 1;
00128     }
00129     for (l4_uint16_t descno = 0; descno < txqsz; ++descno)

```

```

00130     {
00131         auto &desc = _txq.desc(descno);
00132         desc.addr = L4virtio::Ptr<void>(devaddr + txpktoff +
00133                                         descno * sizeof(Packet));
00134         desc.len = sizeof(Packet);
00135     }
00136
00137     // Finish handshake with device
00138     l4virtio_set_feature(_config->driver_features_map,
00139                         L4VIRTIO_FEATURE_VERSION_1);
00140     l4virtio_set_feature(_config->driver_features_map, L4VIRTIO_NET_F_MAC);
00141     driver_acknowledge();
00142 }
00143
00147 l4virtio_net_config_t const &device_config() const
00148 {
00149     return *_config->device_config<l4virtio_net_config_t>();
00150 }
00151
00158 Packet &rx_pkt(l4_uint16_t descno)
00159 {
00160     if (descno >= _rxq.num())
00161         throw L4::Bounds_error("Invalid used descriptor number in RX queue");
00162     return _rxpkts[descno];
00163 }
00164
00176 l4_uint16_t wait_rx(l4_uint32_t *len = nullptr)
00177 {
00178     auto descno = L4Re::chksys(wait_for_next_used(_rxq, len), "Wait for RX");
00179     if (len)
00180         // Ensure that the length provided by the device in wait_for_next_used()
00181         // is not larger than the buffer and subtract the length of the header.
00182         *len = cxx::min(*len - sizeof(_rxpkts[0].hdr), sizeof(_rxpkts[0].data));
00183     return descno;
00184 }
00185
00194 void finish_rx(l4_uint16_t descno)
00195 {
00196     _rxq.free_descriptor(descno, descno);
00197 }
00198
00202 void queue_rx()
00203 {
00204     l4_uint16_t descno;
00205     while ((descno = _rxq.alloc_descriptor()) != Virtqueue::Eoq)
00206         _rxq.enqueue_descriptor(descno);
00207     notify(_rxq);
00208 }
00209
00224 bool tx(std::function<l4_uint32_t(Packet&)> prepare)
00225 {
00226     auto descno = _txq.alloc_descriptor();
00227     if (descno == Virtqueue::Eoq)
00228     {
00229         // Try again after cleaning old descriptors that have already been used
00230         free_used_tx_descriptors();
00231         descno = _txq.alloc_descriptor();
00232         if (descno == Virtqueue::Eoq)
00233             return false;
00234     }
00235
00236     auto &pkt = _txpkts[descno];
00237     auto &desc = _txq.desc(descno);
00238     desc.len = sizeof(pkt.hdr) + prepare(pkt);
00239     send(_txq, descno);
00240     return true;
00241 }
00242
00243 private:
00244 void free_used_tx_descriptors()
00245 {
00246     l4_uint16_t used;
00247     while ((used = _txq.find_next_used()) != Virtqueue::Eoq)
00248     {
00249         if (used >= _txq.num())
00250             throw L4::Bounds_error("Invalid used descriptor number in TX queue");
00251         _txq.free_descriptor(used, used);
00252     }
00253 }
00254
00255 private:
00256 L4Re::Util::Unique_cap<L4Re::Dataspace> _queue_ds;
00257 L4Re::Rm::Unique_region<l4_uint8_t *> _queue_region;
00258 L4virtio::Driver::Virtqueue _rxq, _txq;
00259 Packet *_rxpkts, *_txpkts;
00260 };
00261

```

```
00262 } }
```

## 16.227 l4virtio

```
00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2015-2020, 2022 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *
00007  */
00008 #pragma once
00009
00010 #include <l4/sys/factory>
00011 #include <l4/sys/semaphore>
00012 #include <l4/re/dataspace>
00013 #include <l4/re/env>
00014 #include <l4/re/util/unique_cap>
00015 #include <l4/re/util/object_registry>
00016 #include <l4/re/error_helper>
00017
00018 #include <l4/util/atomic.h>
00019 #include <l4/util/bitops.h>
00020 #include <l4/l4virtio/l4virtio>
00021 #include <l4/l4virtio/virtqueue>
00022 #include <l4/sys/consts.h>
00023
00024 #include <cstring>
00025
00026 namespace L4virtio { namespace Driver {
00027
00031 class Device
00032 {
00033 public:
00056 void driver_connect(L4::Cap<L4virtio::Device> srvcap, bool manage_notify = true)
00057 {
00058     _device = srvcap;
00059
00060     _next_devaddr = L4_SUPERPAGESIZE;
00061
00062     auto *e = L4Re::Env::env();
00063
00064     // Set up the virtio configuration page.
00065
00066     _config_cap = L4Re::chkcap(L4Re::Util::make_unique_cap<L4Re::Dataspace>(),
00067                               "Allocate config dataspace capability");
00068
00069     l4_addr_t ds_offset;
00070     L4Re::chksys(_device->device_config(_config_cap.get(), &ds_offset),
00071                 "Request virtio config page");
00072
00073     if (ds_offset & ~L4_PAGEMASK)
00074         L4Re::chksys(-L4_EINVAL, "Virtio config page is page aligned.");
00075
00076     L4Re::chksys(e->rm()->attach(&_config, L4_PAGESIZE,
00077                                L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00078                                L4::Ipc::make_cap_rw(_config_cap.get(), ds_offset,
00079                                L4_PAGESHIFT),
00080                                "Attach config dataspace");
00081
00082     if (memcmp(&_config->magic, "virt", 4) != 0)
00083         L4Re::chksys(-L4_ENODEV, "Device config has wrong magic value");
00084
00085     if (_config->version != 2)
00086         L4Re::chksys(-L4_ENODEV, "Invalid virtio version, must be 2");
00087
00088     _device->set_status(0); // reset
00089     int status = L4VIRTIO_STATUS_ACKNOWLEDGE;
00090     _device->set_status(status);
00091
00092     status |= L4VIRTIO_STATUS_DRIVER;
00093     _device->set_status(status);
00094
00095     if (_config->fail_state())
00096         L4Re::chksys(-L4_EIO, "Device failure during initialisation.");
00097
00098     // Set up the interrupt used to notify the device about events.
00099     // (only supporting one interrupt with index 0 at the moment)
00100
00101     _host_irq = L4Re::chkcap(L4Re::Util::make_unique_cap<L4::Irq>(),
00102                             "Allocate host IRQ capability");
00103
00104     L4Re::chksys(_device->device_notification_irq(0, _host_irq.get()),
```

```

00105         "Request device notification interrupt.");
00106
00107     // Set up the interrupt to get notifications from the device.
00108     // (only supporting one interrupt with index 0 at the moment)
00109     if (manage_notify)
00110     {
00111         _driver_notification =
00112             L4Re::chkcap(L4Re::Util::make_unique_cap<L4::Semaphore>(),
00113                 "Allocate notification capability");
00114
00115         L4Re::chksys(l4_error(e->factory()->create(_driver_notification.get())),
00116             "Create semaphore for notifications from device");
00117
00118         L4Re::chksys(_device->bind(0, _driver_notification.get()),
00119             "Bind driver notification interrupt");
00120     }
00121 }
00122
00129 int bind_notification_irq(unsigned index, L4::Cap<L4::Triggerable> irq) const
00130 { return l4_error(_device->bind(index, irq)); }
00131
00133 bool fail_state() const { return _config->fail_state(); }
00134
00145 bool feature_negotiated(unsigned int feat) const
00146 { return l4virtio_get_feature(_config->driver_features_map, feat); }
00147
00156 int driver_acknowledge()
00157 {
00158     if (!l4virtio_get_feature(_config->dev_features_map,
00159         L4VIRTIO_FEATURE_VERSION_1))
00160         L4Re::chksys(-L4_ENODEV,
00161             "Require Virtio 1.0 device; Legacy device not supported.");
00162
00163     _config->driver_features_map[0] &= _config->dev_features_map[0];
00164     _config->driver_features_map[1] &= _config->dev_features_map[1];
00165
00166     _device->set_status(_config->status | L4VIRTIO_STATUS_FEATURES_OK);
00167
00168     if (!(_config->status & L4VIRTIO_STATUS_FEATURES_OK))
00169         L4Re::chksys(-L4_EINVAL, "Negotiation of device features.");
00170
00171     _device->set_status(_config->status | L4VIRTIO_STATUS_DRIVER_OK);
00172
00173     if (_config->fail_state())
00174         return -L4_EIO;
00175
00176     return L4_EOK;
00177 }
00178
00196 int register_ds(L4::Cap<L4Re::Dataspace> ds, l4_umword_t offset,
00197     l4_umword_t size, l4_uint64_t *devaddr)
00198 {
00199     *devaddr = next_device_address(size);
00200     return _device->register_ds(L4::Ipc::make_cap_rw(ds), *devaddr, offset, size);
00201 }
00202
00212 int config_queue(int num, unsigned size, l4_uint64_t desc_addr,
00213     l4_uint64_t avail_addr, l4_uint64_t used_addr)
00214 {
00215     auto *queueconf = &_config->queues()[num];
00216     queueconf->num = size;
00217     queueconf->desc_addr = desc_addr;
00218     queueconf->avail_addr = avail_addr;
00219     queueconf->used_addr = used_addr;
00220     queueconf->ready = 1;
00221
00222     return _device->config_queue(num);
00223 }
00224
00230 int max_queue_size(int num) const
00231 {
00232     return _config->queues()[num].num_max;
00233 }
00234
00247 int send_and_wait(Virtqueue &queue, l4_uint16_t descno)
00248 {
00249     send(queue, descno);
00250
00251     // wait for a reply, we assume that no other
00252     // request will get in the way.
00253     auto head = wait_for_next_used(queue);
00254
00255     if (head < 0)
00256         return head;
00257
00258     return (head == descno) ? L4_EOK : -L4_EINVAL;
00259 }

```

```

00260
00268 int wait(int index) const
00269 {
00270     if (index != 0)
00271         return -L4_EEXIST;
00272
00273     return l4_ipc_error(_driver_notification->down(), l4_utcb());
00274 }
00275
00291 int wait_for_next_used(Virtqueue &queue, l4_uint32_t *len = nullptr) const
00292 {
00293     while (true)
00294     {
00295         int err = wait(0);
00296
00297         if (err < 0)
00298             return err;
00299
00300         auto head = queue.find_next_used(len);
00301         if (head != Virtqueue::Eq) // spurious interrupt?
00302             return head;
00303     }
00304 }
00305
00312 void send(Virtqueue &queue, l4_uint16_t descno)
00313 {
00314     queue.enqueue_descriptor(descno);
00315     notify(queue);
00316 }
00317
00318 void notify(Virtqueue &queue)
00319 {
00320     if (!queue.no_notify_host())
00321         _host_irq->trigger();
00322 }
00323
00324 private:
00325     l4_uint64_t next_device_address(l4_umword_t size)
00326     {
00327         l4_umword_t ret;
00328         size = l4_round_page(size);
00329         do
00330         {
00341             ret = _next_devaddr;
00342             if (l4_umword_t(~0) - ret < size)
00343                 L4Re::chksys(-L4_ENOMEM, "Out of device address space.");
00344         }
00345         while (!l4util_cmpxchg(&_next_devaddr, ret, ret + size));
00346
00347         return ret;
00348     }
00349
00350 protected:
00351     L4::Cap<L4virtio::Device> _device;
00352     L4Re::Rm::Unique_region<L4virtio::Device::Config_hdr *> _config;
00353     l4_umword_t _next_devaddr;
00354     L4Re::Util::Unique_cap<L4::Semaphore> _driver_notification;
00355
00356 private:
00357     L4Re::Util::Unique_cap<L4::Irq> _host_irq;
00358     L4Re::Util::Unique_cap<L4Re::Dataspace> _config_cap;
00359 };
00360
00361 } }

```

## 16.228 l4virtio

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2013-2022 Kernkonzept GmbH.
00005  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00006  *             Matthias Lange <matthias.lange@kernkonzept.com>
00007  *
00008  */
00009 #pragma once
00010
00011 #include "virtio.h"
00012 #include <l4/sys/capability>
00013 #include <l4/sys/cxx/ipc_client>
00014 #include <l4/re/dataspace>
00015 #include <l4/sys/irq>
00016 #include <l4/cxx/utis>

```

```

00017
00018 namespace L4virtio {
00039 class Device :
00040     public L4::Kobject_t<Device, L4::Icu, L4VIRTIO_PROTOCOL,
00041         L4::Type_info::Demand_t<1> >
00042 {
00043 public:
00044     typedef l4virtio_config_queue_t Config_queue;
00045     struct Config_hdr : l4virtio_config_hdr_t
00046     {
00047         Config_queue *queues() const
00048         { return l4virtio_config_queues(this); }
00049
00050         template <typename T>
00051         T *device_config() const
00052         {
00053             return static_cast<T*>(l4virtio_device_config(this));
00054         }
00055
00056         int config_queue(unsigned num, L4::Cap<L4::Triggerable> out_notify,
00057             L4::Cap<L4::Triggerable> in_notify,
00058             l4_timeout_s to = L4_IPC_TIMEOUT_NEVER)
00059         {
00060             return send_cmd(L4VIRTIO_CMD_CFG_QUEUE | num,
00061                 out_notify, in_notify, to);
00062         }
00063
00064         int notify_queue(unsigned num, L4::Cap<L4::Triggerable> out_notify,
00065             L4::Cap<L4::Triggerable> in_notify,
00066             l4_timeout_s to = L4_IPC_TIMEOUT_NEVER)
00067         {
00068             return send_cmd(L4VIRTIO_CMD_NOTIFY_QUEUE | num,
00069                 out_notify, in_notify, to);
00070         }
00071
00072         bool fail_state() const
00073         {
00074             auto cfg_status = cxx::access_once(&status);
00075             return cfg_status
00076                 & (L4VIRTIO_STATUS_FAILED | L4VIRTIO_STATUS_DEVICE_NEEDS_RESET);
00077         }
00078
00079         int set_status(unsigned new_status, L4::Cap<L4::Triggerable> out_notify,
00080             L4::Cap<L4::Triggerable> in_notify,
00081             l4_timeout_s to = L4_IPC_TIMEOUT_NEVER)
00082         {
00083             return send_cmd(L4VIRTIO_CMD_SET_STATUS | new_status,
00084                 out_notify, in_notify, to);
00085         }
00086
00087         int cfg_changed(unsigned reg, L4::Cap<L4::Triggerable> out_notify,
00088             L4::Cap<L4::Triggerable> in_notify,
00089             l4_timeout_s to = L4_IPC_TIMEOUT_NEVER)
00090         {
00091             return send_cmd(L4VIRTIO_CMD_CFG_CHANGED | reg,
00092                 out_notify, in_notify, to);
00093         }
00094
00095         int send_cmd(unsigned command, L4::Cap<L4::Triggerable> out_notify,
00096             L4::Cap<L4::Triggerable> in_notify,
00097             l4_timeout_s to = L4_IPC_TIMEOUT_NEVER)
00098         {
00099             cxx::write_now(&cmd, command);
00100
00101             if (out_notify)
00102                 out_notify->trigger();
00103
00104             auto utcb = l4_utcb();
00105             auto ipc_to = l4_timeout(L4_IPC_TIMEOUT_0, to);
00106
00107             do
00108             {
00109                 if (in_notify)
00110                     if (l4_ipc_error(l4_ipc_receive(in_notify.cap(), utcb, ipc_to),
00111                         utcb) == L4_IPC_RETIMEOUT)
00112                         break;
00113             }
00114             while (cxx::access_once(&cmd));
00115
00116             return cxx::access_once(&cmd) ? -L4_EBUSY : L4_EOK;
00117         }
00118     };
00119 };
00120
00121 L4_INLINE_RPC_OP(L4VIRTIO_OP_SET_STATUS, long,
00122     set_status, (unsigned status));
00123
00124 L4_INLINE_RPC_OP(L4VIRTIO_OP_CONFIG_QUEUE, long,

```



```

00156         config_queue, (unsigned queue));
00157
00181     L4_INLINE_RPC_OP(L4VIRTIO_OP_REGISTER_DS, long,
00182         register_ds, (L4::Ipc::Cap<L4Re::Dataspace> ds_cap,
00183             l4_uint64_t base, l4_umword_t offset,
00184             l4_umword_t size));
00185
00194     L4_INLINE_RPC_OP(L4VIRTIO_OP_DEVICE_CONFIG, long, device_config,
00195         (L4::Ipc::Out<L4::Cap<L4Re::Dataspace> > config_ds,
00196             l4_addr_t *ds_offset));
00197
00216     L4_INLINE_RPC_OP(L4VIRTIO_OP_GET_DEVICE_IRQ, long, device_notification_irq,
00217         (unsigned index, L4::Ipc::Out<L4::Cap<L4::Triggerable> > irq));
00218
00219
00220     typedef L4::Typeid::Rpcs<set_status_t, config_queue_t, register_ds_t,
00221         device_config_t, device_notification_irq_t>
00222         Rpcs;
00223 };
00224
00225 }

```

## 16.229 l4virtio

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2014-2022 Kernkonzept GmbH.
00005  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00006  *           Manuel von Oltersdorff-Kalettkka <manuel.kalettkka@kernkonzept.com>
00007  *
00008  */
00009 #pragma once
00010
00011 #include <algorithm>
00012 #include <limits.h>
00013 #include <memory>
00014 #include <vector>
00015
00016 #include <l4/re/dataspace>
00017 #include <l4/re/util/debug>
00018 #include <l4/re/env>
00019 #include <l4/re/error_helper>
00020 #include <l4/re/rm>
00021 #include <l4/re/util/cap_alloc>
00022 #include <l4/re/util/shared_cap>
00023 #include <l4/re/util/unique_cap>
00024
00025 #include <l4/sys/types.h>
00026 #include <l4/re/util/meta>
00027
00028 #include <l4/cxx/bitfield>
00029 #include <l4/cxx/utills>
00030 #include <l4/cxx/unique_ptr>
00031
00032 #include <l4/sys/cxx/ipc_legacy>
00033
00034 #include "../l4virtio"
00035 #include "virtio"
00036
00040 namespace L4virtio {
00041 namespace Svr {
00042
00052 class Dev_config
00053 {
00054 public:
00055     typedef Dev_status Status;
00056     typedef Dev_features Features;
00057
00058 private:
00059     typedef L4Re::Rm::Unique_region< l4virtio_config_hdr_t*> Cfg_region;
00060     typedef L4Re::Util::Shared_cap<L4Re::Dataspace> Cfg_cap;
00061
00062     l4_uint32_t _vendor, _device, _qoffset, _nqueues;
00063     l4_uint32_t _host_features[sizeof(l4virtio_config_hdr_t::dev_features_map)
00064         / sizeof(l4_uint32_t)];
00065     Cfg_cap _ds;
00066     Cfg_region _config;
00067     l4_addr_t _ds_offset = 0;
00068
00069     Status _status{0}; // status shadow, can be trusted by the device model
00070
00071     static l4_uint32_t align(l4_uint32_t x)

```

```

00072 { return (x + 0xfU) & ~0xfU; }
00073
00074 void attach_n_init_cfg(Cfg_cap const &cfg, l4_addr_t offset)
00075 {
00076     L4Re::chksys(L4Re::Env::env()->rm()->attach(&_config, L4_PAGESIZE,
00077         L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00078         L4::Ipc::make_cap_rw(cfg.get()),
00079         offset),
00080     "Attach config space to local address space.");
00081
00082     _config->generation = 0;
00083     memset(_config->driver_features_map, 0, sizeof(_config->driver_features_map));
00084     memset(_host_features, 0, sizeof(_host_features));
00085     set_host_feature(L4VIRTIO_FEATURE_VERSION_1);
00086     reset_hdr();
00087
00088     _ds = cfg;
00089     _ds_offset = offset;
00090 }
00091
00092 protected:
00093 void volatile *get_priv_config() const
00094 {
00095     return l4virtio_device_config(_config.get());
00096 }
00097
00098 public:
00099
00100 Dev_config(l4_uint32_t vendor, l4_uint32_t device,
00101     unsigned cfg_size, l4_uint32_t num_queues = 0)
00102 : _vendor(vendor), _device(device),
00103   _qoffset(0x100 + align(cfg_size)),
00104   _nqueues(num_queues)
00105 {
00106     using L4Re::Dataspace;
00107     using L4Re::chkcapi;
00108     using L4Re::chksys;
00109
00110     if (sizeof(l4virtio_config_queue_t) * _nqueues + _qoffset > L4_PAGESIZE)
00111     {
00112         // too many queues does not fit into our page
00113         _qoffset = 0;
00114         _nqueues = 0;
00115     }
00116
00117     auto cfg = chkcapi(L4Re::Util::make_shared_cap<Dataspace>());
00118     chksys(L4Re::Env::env()->mem_alloc()->alloc(L4_PAGESIZE, cfg.get()));
00119
00120     attach_n_init_cfg(cfg, 0);
00121 }
00122
00123 Dev_config(Cfg_cap const &cfg, l4_addr_t cfg_offset,
00124     l4_uint32_t vendor, l4_uint32_t device,
00125     unsigned cfg_size, l4_uint32_t num_queues = 0)
00126 : _vendor(vendor), _device(device),
00127   _qoffset(0x100 + align(cfg_size)),
00128   _nqueues(num_queues)
00129 {
00130     if (sizeof(l4virtio_config_queue_t) * _nqueues + _qoffset > L4_PAGESIZE)
00131     {
00132         // too many queues does not fit into our page
00133         _qoffset = 0;
00134         _nqueues = 0;
00135     }
00136
00137     attach_n_init_cfg(cfg, cfg_offset);
00138 }
00139
00140 void set_host_feature(unsigned feature)
00141 { l4virtio_set_feature(_host_features, feature); }
00142
00143 void clear_host_feature(unsigned feature)
00144 { l4virtio_clear_feature(_host_features, feature); }
00145
00146 bool get_host_feature(unsigned feature)
00147 { return l4virtio_get_feature(_host_features, feature); }
00148
00149 bool get_guest_feature(unsigned feature)
00150 { return l4virtio_get_feature(_config->driver_features_map, feature); }
00151
00152 l4_uint32_t &host_features(unsigned idx)
00153 { return _host_features[idx]; }
00154
00155 l4_uint32_t host_features(unsigned idx) const
00156 { return _host_features[idx]; }
00157
00158 l4_uint32_t num_queues() const

```

```

00185     { return _nqueues; }
00186
00198     l4_uint32_t guest_features(unsigned idx) const
00199     { return _config->driver_features_map[idx]; }
00200
00212     l4_uint32_t negotiated_features(unsigned idx) const
00213     { return _config->driver_features_map[idx] & _host_features[idx]; }
00214
00222     Status status() const { return _status; }
00223
00230     l4_uint32_t get_cmd() const
00231     {
00232         return hdr()->cmd;
00233     }
00234
00241     void reset_cmd()
00242     {
00243         const_cast<l4_uint32_t volatile &>(hdr()->cmd) = 0;
00244     }
00245
00253     void set_status(Status status)
00254     {
00255         _status = status;
00256         const_cast<l4_uint32_t volatile &>(hdr()->status) = status.raw;
00257     }
00258
00265     void add_irq_status(l4_uint32_t status)
00266     {
00267         const_cast<l4_uint32_t volatile &>(hdr()->irq_status) |= status;
00268     }
00269
00276     void set_device_needs_reset()
00277     {
00278         _status.device_needs_reset() = 1;
00279         const_cast<l4_uint32_t volatile &>(hdr()->status) = _status.raw;
00280     }
00281
00286     bool change_queue_config(l4_uint32_t num_queues)
00287     {
00288         if (sizeof(l4virtio_config_queue_t) * num_queues + _qoffset > L4_PAGESIZE)
00289             // too many queues does not fit into our page
00290             return false;
00291
00292         _nqueues = num_queues;
00293         reset_hdr(true);
00294         return true;
00295     }
00296
00303     l4virtio_config_queue_t volatile const *qconfig(unsigned index) const
00304     {
00305         if (L4_UNLIKELY(_qoffset < sizeof (l4virtio_config_hdr_t)))
00306             return 0;
00307
00308         if (L4_UNLIKELY(index >= _nqueues))
00309             return 0;
00310
00311         return reinterpret_cast<l4virtio_config_queue_t const *>
00312             (reinterpret_cast<char *>(_config.get()) + _qoffset) + index;
00313     }
00314
00318     void reset_hdr(bool inc_generation = false) const
00319     {
00320         _config->magic = L4VIRTIO_MAGIC;
00321         _config->version = 2;
00322         _config->device = _device;
00323         _config->vendor = _vendor;
00324         _config->status = 0;
00325         _config->irq_status = 0;
00326         _config->num_queues = _nqueues;
00327         _config->queues_offset = _qoffset;
00328
00329         memcpy(_config->dev_features_map, _host_features,
00330             sizeof(_config->dev_features_map));
00331         wmb();
00332         if (inc_generation)
00333             ++_config->generation;
00334     }
00335
00336
00345     bool reset_queue(unsigned index, unsigned num_max,
00346         bool inc_generation = false) const
00347     {
00348         l4virtio_config_queue_t volatile *qc;
00349         // this function is allowed to write to the device config
00350         qc = const_cast<l4virtio_config_queue_t volatile *>(qconfig(index));
00351         if (L4_UNLIKELY(qc == 0))
00352             return false;

```

```

00353
00354     qc->num_max = num_max;
00355     qc->num = 0;
00356     qc->ready = 0;
00357     wmb();
00358     if (inc_generation)
00359         ++_config->generation;
00360
00361     return true;
00362 }
00363
00364 l4virtio_config_hdr_t const volatile *hdr() const
00365 { return _config.get(); }
00366
00367 L4::Cap<L4Re::Dataspace> ds() const { return _ds.get(); }
00368
00369 l4_addr_t ds_offset() const
00370 { return _ds_offset; }
00371 };
00372
00373 template<typename PRIV_CONFIG>
00374 class Dev_config_t : public Dev_config
00375 {
00376 public:
00377     typedef PRIV_CONFIG Priv_config;
00378
00379     Dev_config_t(l4_uint32_t vendor, l4_uint32_t device,
00380                 l4_uint32_t num_queues = 0)
00381         : Dev_config(vendor, device, sizeof(PRIV_CONFIG), num_queues)
00382     {}
00383
00384     Dev_config_t(L4Re::Util::Shared_cap<L4Re::Dataspace> const &cfg,
00385                 l4_addr_t cfg_offset, l4_uint32_t vendor, l4_uint32_t device,
00386                 l4_uint32_t num_queues = 0)
00387         : Dev_config(cfg, cfg_offset, vendor, device, sizeof(PRIV_CONFIG),
00388                     num_queues)
00389     {}
00390
00391     Priv_config volatile *priv_config() const
00392     {
00393         return static_cast<Priv_config volatile *>(get_priv_config());
00394     }
00395 };
00396
00397 struct No_custom_data {};
00398
00399 template <typename DATA>
00400 class Driver_mem_region_t : public DATA
00401 {
00402 public:
00403     struct Flags
00404     {
00405         Flags() = default;
00406         explicit Flags(l4_uint32_t raw) : raw(raw) {}
00407
00408         l4_uint32_t raw;
00409         CXX_BITFIELD_MEMBER(0, 0, rw, raw);
00410     };
00411
00412 private:
00413     typedef L4Re::Util::Unique_cap<L4Re::Dataspace> Ds_cap;
00414
00415     l4_uint64_t _drv_base;
00416     l4_uint64_t _trans_offset;
00417     l4_umword_t _size;
00418     Flags      _flags;
00419
00420     Ds_cap      _ds;
00421     l4_addr_t    _ds_offset;
00422
00423     L4Re::Rm::Unique_region<l4_addr_t> _local_base;
00424
00425     template<typename T>
00426     T _local(l4_uint64_t addr) const
00427     {
00428         return reinterpret_cast<T>(addr - _trans_offset);
00429     }
00430
00431 public:
00432     Driver_mem_region_t() : _size(0) {}
00433
00434     Driver_mem_region_t(l4_uint64_t drv_base, l4_umword_t size,
00435                         l4_addr_t offset, Ds_cap &&ds)
00436         : _drv_base(l4_trunc_page(drv_base)), _size(0), _flags(0),
00437           _ds_offset(l4_trunc_page(offset))
00438     {}

```

```

00502 {
00503     using L4Re::chksys;
00504     using L4Re::Env;
00505
00506     L4Re::Dataspace::Stats ds_info = L4Re::Dataspace::Stats();
00507     // Sometimes we work with dataspace that do not implement all dataspace
00508     // methods and return an error instead. An example of such a dataspace is
00509     // io's Vi::System_bus. We detect this case when the info method returns
00510     // -L4_ENOSYS and simply assume the dataspace is good for us.
00511     long err = ds->info(&ds_info);
00512     if (err >= 0)
00513     {
00514         l4_addr_t ds_size = l4_round_page(ds_info.size);
00515
00516         if (ds_size < L4_PAGESIZE)
00517             chksys(-L4_EINVAL, "DS too small");
00518
00519         if (_ds_offset >= ds_size)
00520             chksys(-L4_ERANGE, "offset larger than DS size");
00521
00522         size = l4_round_page(size);
00523         if (size > ds_size)
00524             chksys(-L4_EINVAL, "size larger than DS size");
00525
00526         if (_ds_offset > ds_size - size)
00527             chksys(-L4_EINVAL, "invalid offset or size");
00528
00529         // overflow check
00530         if ((ULONG_MAX - size) < _drv_base)
00531             chksys(-L4_EINVAL, "invalid size");
00532
00533         _flags.rw() = (ds_info.flags & L4Re::Dataspace::F::W).raw != 0;
00534     }
00535     else if (err == -L4_ENOSYS)
00536     {
00537         _flags.rw() = true;
00538     }
00539     else
00540     {
00541         chksys(err, "getting data-space infos");
00542     }
00543
00544     auto f = L4Re::Rm::F::Search_addr | L4Re::Rm::F::R;
00545     if (_flags.rw())
00546         f |= L4Re::Rm::F::W;
00547
00548     // use a big alignment to save PT/TLB entries and kernel memory resources!
00549     chksys(Env::env()->rm()->attach(&_local_base, size, f,
00550                                     L4::Ipc::make_cap(ds.get(), _flags.rw()
00551                                                         ? L4_CAP_FPAGE_RW
00552                                                         : L4_CAP_FPAGE_RO),
00553                                     _ds_offset, L4_SUPERPAGESHIFT));
00554
00555     _size = size;
00556     _ds = cxx::move(ds);
00557     _trans_offset = _drv_base - _local_base.get();
00558 }
00559
00561 bool is_writable() const { return _flags.rw(); }
00562
00564 Flags flags() const { return _flags; }
00565
00567 bool empty() const
00568 { return _size == 0; }
00569
00571 l4_uint64_t drv_base() const { return _drv_base; }
00572
00574 l4_addr_t local_base() const { return _local_base.get(); }
00575
00577 l4_umword_t size() const { return _size; }
00578
00580 l4_addr_t ds_offset() const { return _ds_offset; }
00581
00583 L4::Cap<L4Re::Dataspace> ds() const { return _ds.get(); }
00584
00592 bool contains(l4_uint64_t base, l4_umword_t size) const
00593 {
00594     if (base < _drv_base)
00595         return false;
00596
00597     if (base > _drv_base + _size - 1)
00598         return false;
00599
00600     if (size > _size)
00601         return false;
00602
00603     if (base - _drv_base > _size - size)

```

```

00604         return false;
00605
00606         return true;
00607     }
00608
00615     template<typename T>
00616     T *local(Ptr<T> p) const
00617     { return _local<T*>(p.get()); }
00618 };
00619
00620 typedef Driver_mem_region_t<No_custom_data> Driver_mem_region;
00621
00628 template <typename DATA>
00629 class Driver_mem_list_t
00630 {
00631 public:
00632     typedef Driver_mem_region_t<DATA> Mem_region;
00633
00634 private:
00635     cxx::unique_ptr<Mem_region[]> _l;
00636     unsigned _max;
00637     unsigned _free;
00638
00639 public:
00641     typedef L4Re::Util::Unique_cap<L4Re::Dataspace> Ds_cap;
00642
00644     Driver_mem_list_t() : _max(0), _free(0) {}
00645
00650     void init(unsigned max)
00651     {
00652         _l = cxx::make_unique<Driver_mem_region_t<DATA>[]>(max);
00653         _max = max;
00654         _free = 0;
00655     }
00656
00658     bool full() const
00659     { return _free == _max; }
00660
00669     Mem_region const *add(l4_uint64_t drv_base, l4_umword_t size,
00670                          l4_addr_t offset, Ds_cap &&ds)
00671     {
00672         if (full())
00673             L4Re::chksys(-L4_ENOMEM);
00674
00675         _l[_free++] = Mem_region(drv_base, size, offset, cxx::move(ds));
00676         return &_l[_free - 1];
00677     }
00678
00683     void remove(Mem_region const *r)
00684     {
00685         if (r < &_l[0] || r >= &_l[_free])
00686             L4Re::chksys(-L4_ERANGE);
00687
00688         unsigned idx = r - &_l[0];
00689
00690         for (unsigned i = idx + 1; i < _free - 1; ++i)
00691             _l[i] = cxx::move(_l[i + 1]);
00692
00693         _l[--_free] = Mem_region();
00694     }
00695
00703     Mem_region *find(l4_uint64_t base, l4_umword_t size) const
00704     {
00705         return _find(base, size);
00706     }
00707
00717     void load_desc(Virtqueue::Desc const &desc, Request_processor const *p,
00718                  Virtqueue::Desc const **table) const
00719     {
00720         Mem_region const *r = find(desc.addr.get(), desc.len);
00721         if (L4_UNLIKELY(!r))
00722             throw Bad_descriptor(p, Bad_descriptor::Bad_address);
00723
00724         *table = static_cast<Virtqueue::Desc const *>(r->local(desc.addr));
00725     }
00726
00737     void load_desc(Virtqueue::Desc const &desc, Request_processor const *p,
00738                  Mem_region const **data) const
00739     {
00740         Mem_region const *r = find(desc.addr.get(), desc.len);
00741         if (L4_UNLIKELY(!r))
00742             throw Bad_descriptor(p, Bad_descriptor::Bad_address);
00743
00744         *data = r;
00745     }
00746
00763     template<typename ARG>

```

```

00764 void load_desc(Virtqueue::Desc const &desc, Request_processor const *p,
00765                ARG *data) const
00766 {
00767     Mem_region *r = find(desc.addr.get(), desc.len);
00768     if (L4_UNLIKELY(!r))
00769         throw Bad_descriptor(p, Bad_descriptor::Bad_address);
00770
00771     *data = ARG(r, desc, p);
00772 }
00773
00774 private:
00775 Mem_region *_find(l4_uint64_t base, l4_umword_t size) const
00776 {
00777     for (unsigned i = 0; i < _free; ++i)
00778         if (_l[i].contains(base, size))
00779             return &_l[i];
00780     return 0;
00781 }
00782
00783 };
00784
00785 typedef Driver_mem_list_t<No_custom_data> Driver_mem_list;
00786
00787 template<typename DATA>
00795 class Device_t
00796 {
00797 public:
00798     typedef Driver_mem_list_t<DATA> Mem_list;
00799
00800 protected:
00801     Mem_list _mem_info;
00802
00803 private:
00804     Dev_config *_device_config;
00805
00806     using Ds_vector = std::vector<L4::Cap<L4Re::Dataspace>>;
00807     std::shared_ptr<Ds_vector const> _trusted_ds_caps;
00808
00809     bool _trusted_ds_validation_enabled = false;
00810
00811 public:
00812     L4_RPC_LEGACY_DISPATCH(L4virtio::Device);
00813     template<typename IOS> int virtio_dispatch(unsigned r, IOS &ios)
00814     { return dispatch(r, ios); }
00815
00816     virtual void reset() = 0;
00817
00818     virtual bool check_features()
00819     { return true; }
00820
00821     virtual bool check_queues() = 0;
00822
00823     virtual int reconfig_queue(unsigned idx) = 0;
00824
00825     virtual void cfg_changed(unsigned /* reg */) {};
00826
00827     virtual void register_single_driver_irq()
00828     { L4Re::chksys(-L4_ENOSYS, "Legacy single IRQ interface not implemented."); }
00829
00830     virtual void trigger_driver_config_irq() = 0;
00831
00832     virtual L4::Cap<L4::Irq> device_notify_irq() const
00833     {
00834         L4Re::chksys(-L4_ENOSYS, "Legacy single IRQ interface not implemented.");
00835         return L4::Cap<L4::Irq>();
00836     }
00837
00838     virtual void register_driver_irq(unsigned idx)
00839     {
00840         if (idx != 0)
00841             L4Re::chksys(-L4_ENOSYS, "Multi IRQ interface not implemented.");
00842         register_single_driver_irq();
00843     }
00844
00845     virtual L4::Cap<L4::Irq> device_notify_irq(unsigned idx)
00846     {
00847         if (idx != 0)
00848             L4Re::chksys(-L4_ENOSYS, "Multi IRQ interface not implemented.");
00849         return device_notify_irq();
00850     }
00851
00852     virtual unsigned num_events_supported() const
00853     { return 1; }
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879

```

```

00880     virtual L4::Ipc_svr::Server_iface *server_iface() const = 0;
00881
00885     Device_t(Dev_config *dev_config)
00886     : _device_config(dev_config)
00887     {}
00888
00892     Mem_list const *mem_info() const
00893     { return &_amp;_mem_info; };
00894
00895     long op_set_status(L4virtio::Device::Rights, unsigned status)
00896     { return _set_status(status); }
00897
00898     long op_config_queue(L4virtio::Device::Rights, unsigned queue)
00899     {
00900         Dev_config::Status status = _device_config->status();
00901         if (status.fail_state() || !status.acked() || !status.driver())
00902             return -L4_EIO;
00903
00904         return reconfig_queue(queue);
00905     }
00906
00907     long op_register_ds(L4virtio::Device::Rights,
00908                        L4::Ipc::Snd_fpage ds_cap_fp, l4_uint64_t ds_base,
00909                        l4_umword_t offset, l4_umword_t sz)
00910     {
00911         L4Re::Util::Dbg()
00912             .printf("Registering dataspace from 0x%llx with %lu KiB, offset 0x%lx\n",
00913                   ds_base, sz » 10, offset);
00914
00915         _check_n_init_shm(ds_cap_fp, ds_base, sz, offset);
00916
00917         return 0;
00918     }
00919
00920     long op_device_config(L4virtio::Device::Rights,
00921                          L4::Ipc::Cap<L4Re::Dataspace> &config_ds,
00922                          l4_addr_t &ds_offset)
00923     {
00924         L4Re::Util::Dbg()
00925             .printf("register client: host IRQ: %lx config DS: %lx\n",
00926                   device_notify_irq().cap(), _device_config->ds().cap());
00927
00928         config_ds = L4::Ipc::make_cap(_device_config->ds(), L4_CAP_FPAGE_RW);
00929         ds_offset = _device_config->ds_offset();
00930         return 0;
00931     }
00932
00933     long op_device_notification_irq(L4virtio::Device::Rights,
00934                                     unsigned idx,
00935                                     L4::Ipc::Cap<L4::Triggerable> &irq)
00936     {
00937         auto cap = device_notify_irq(idx);
00938
00939         if (!cap.is_valid())
00940             return -L4_EINVAL;
00941
00942         irq = L4::Ipc::make_cap(cap, L4_CAP_FPAGE_RO);
00943         return L4_EOK;
00944     }
00945
00946     int op_bind(L4::Icu::Rights, l4_umword_t idx, L4::Ipc::Snd_fpage irq_cap_fp)
00947     {
00948         if (idx >= num_events_supported())
00949             return -L4_ERANGE;
00950
00951         if (!irq_cap_fp.cap_received())
00952             return -L4_EINVAL;
00953
00954         register_driver_irq(idx);
00955
00956         return L4_EOK;
00957     }
00958
00959     int op_unbind(L4::Icu::Rights, l4_umword_t, L4::Ipc::Snd_fpage)
00960     {
00961         return -L4_ENOSYS;
00962     }
00963
00964     int op_info(L4::Icu::Rights, L4::Icu::_Info &info)
00965     {
00966         info.features = 0;
00967         info.nr_irqs = num_events_supported();
00968         info.nr_msis = 0;
00969
00970         return L4_EOK;
00971     }
00972

```



```

00973 int op_msi_info(L4::Icu::Rights, l4_umword_t, l4_uint64_t, l4_icu_msi_info_t &)
00974 { return -L4_ENOSYS; }
00975
00976 int op_mask(L4::Icu::Rights, l4_umword_t)
00977 { return -L4_ENOSYS; }
00978
00979 int op_unmask(L4::Icu::Rights, l4_umword_t)
00980 { return -L4_ENOREPLY; }
00981
00982 int op_set_mode(L4::Icu::Rights, l4_umword_t, l4_umword_t)
00983 { return -L4_ENOSYS; }
00984
00996 void reset_queue_config(unsigned idx, unsigned num_max,
00997                          bool inc_generation = false)
00998 {
00999     _device_config->reset_queue(idx, num_max, inc_generation);
01000 }
01001
01006 void init_mem_info(unsigned num)
01007 {
01008     _mem_info.init(num);
01009 }
01010
01018 void device_error()
01019 {
01020     reset();
01021     _device_config->set_device_needs_reset();
01022
01023     // the device MUST NOT notify the driver before DRIVER_OK.
01024     if (_device_config->status().driver_ok())
01025         trigger_driver_config_irq();
01026 }
01027
01041 bool setup_queue(Virtqueue *q, unsigned qn, unsigned num_max)
01042 {
01043     l4virtio_config_queue_t volatile const *qc;
01044     qc = _device_config->qconfig(qn);
01045     if (L4_UNLIKELY(qc == 0))
01046         return false;
01047
01048     if (!qc->ready)
01049     {
01050         q->disable();
01051         return true;
01052     }
01053
01054     // read to local variables before check
01055     l4_uint32_t num = qc->num;
01056     l4_uint64_t desc = qc->desc_addr;
01057     l4_uint64_t avail = qc->avail_addr;
01058     l4_uint64_t used = qc->used_addr;
01059
01060     if (0)
01061         printf("%p: setup queue: num=0x%x max_num=0x%x desc=0x%llx avail=0x%llx used=0x%llx\n",
01062               this, num, num_max, desc, avail, used);
01063
01064     if (!num || num > num_max)
01065         return false;
01066
01067     // num must be power of two
01068     if (num & (num - 1))
01069         return false;
01070
01071     if (desc & 0xf)
01072         return false;
01073
01074     if (avail & 0x1)
01075         return false;
01076
01077     if (used & 0x3)
01078         return false;
01079
01080     auto const *desc_info = _mem_info.find(desc, Virtqueue::desc_size(num));
01081     if (L4_UNLIKELY(!desc_info))
01082         return false;
01083
01084     auto const *avail_info = _mem_info.find(avail, Virtqueue::avail_size(num));
01085     if (L4_UNLIKELY(!avail_info))
01086         return false;
01087
01088     auto const *used_info = _mem_info.find(used, Virtqueue::used_size(num));
01089     if (L4_UNLIKELY(!used_info || !used_info->is_writable()))
01090         return false;
01091
01092     L4Re::Util::Dbg()
01093         .printf("shm=[%llx-%llx] local=[%llx-%llx] desc=[%llx-%llx] (%p-%p)\n",
01094               desc_info->drv_base(), desc_info->drv_base() + desc_info->size() - 1,

```

```

01095         desc_info->local_base(),
01096         desc_info->local_base() + desc_info->size() - 1,
01097         desc, desc + Virtqueue::desc_size(num),
01098         desc_info->local(Ptr<char>(desc)),
01099         desc_info->local(Ptr<char>(desc)) + Virtqueue::desc_size(num));
01100
01101     L4Re::Util::Dbg()
01102     .printf("shm=[%llx-%llx] local=[%lx-%lx] avail=[%llx-%llx] (%p-%p)\n",
01103            avail_info->drv_base(), avail_info->drv_base() + avail_info->size() - 1,
01104            avail_info->local_base(),
01105            avail_info->local_base() + avail_info->size() - 1,
01106            avail, avail + Virtqueue::avail_size(num),
01107            avail_info->local(Ptr<char>(avail)),
01108            avail_info->local(Ptr<char>(avail)) + Virtqueue::avail_size(num));
01109
01110     L4Re::Util::Dbg()
01111     .printf("shm=[%llx-%llx] local=[%lx-%lx] used=[%llx-%llx] (%p-%p)\n",
01112            used_info->drv_base(), used_info->drv_base() + used_info->size() - 1,
01113            used_info->local_base(),
01114            used_info->local_base() + used_info->size() - 1,
01115            used, used + Virtqueue::used_size(num),
01116            used_info->local(Ptr<char>(used)),
01117            used_info->local(Ptr<char>(used)) + Virtqueue::used_size(num));
01118
01119     q->setup(num, desc_info->local(Ptr<void>(desc)),
01120            avail_info->local(Ptr<void>(avail)),
01121            used_info->local(Ptr<void>(used)));
01122     return true;
01123 }
01124
01125 void check_n_init_shm(L4Re::Util::Unique_cap<L4Re::Dataspace> &&shm,
01126                    l4_uint64_t base, l4_umword_t size, l4_addr_t offset)
01127 {
01128     if (_mem_info.full())
01129         L4Re::chksys(-L4_ENOMEM);
01130
01131     auto const *i = _mem_info.add(base, size, offset, cxx::move(shm));
01132     L4Re::Util::Dbg()
01133     .printf("PORT[%p]: DMA guest [%llx-%llx] local [%lx-%lx] offset %lx\n",
01134            this, i->drv_base(), i->drv_base() + i->size() - 1,
01135            i->local_base(),
01136            i->local_base() + i->size() - 1,
01137            i->ds_offset());
01138 }
01139
01140 bool handle_mem_cmd_write()
01141 {
01142     l4_uint32_t cmd = _device_config->get_cmd();
01143     if (L4_LIKELY(!(cmd & L4VIRTIO_CMD_MASK)))
01144         return false;
01145
01146     switch (cmd & L4VIRTIO_CMD_MASK)
01147     {
01148     case L4VIRTIO_CMD_SET_STATUS:
01149         _set_status(cmd & ~L4VIRTIO_CMD_MASK);
01150         break;
01151
01152     case L4VIRTIO_CMD_CFG_QUEUE:
01153         reconfig_queue(cmd & ~L4VIRTIO_CMD_MASK);
01154         break;
01155
01156     case L4VIRTIO_CMD_CFG_CHANGED:
01157         cfg_changed(cmd & ~L4VIRTIO_CMD_MASK);
01158         break;
01159
01160     default:
01161         // unknown command
01162         break;
01163     }
01164
01165     _device_config->reset_cmd();
01166
01167     return true;
01168 }
01169
01170 void enable_trusted_ds_validation()
01171 {
01172     _trusted_ds_validation_enabled = true;
01173 }
01174
01175 void
01176 add_trusted_dataspaces(std::shared_ptr<Ds_vector const> ds)
01177 {
01178     _trusted_ds_caps = ds;
01179 }
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197

```

```

01198 private:
01210 long validate_ds(L4::Cap<L4Re::Dataspace> ds)
01211 {
01212     if (!_trusted_ds_caps)
01213         return -L4_EINVAL;
01214     if (std::any_of(_trusted_ds_caps->cbegin(), _trusted_ds_caps->cend(),
01215         [&ds](L4::Cap<L4Re::Dataspace> cap)
01216         {
01217             return L4Re::Env::env()->task()
01218                 ->cap_equal(ds, cap).label() == 1;
01219         })
01220     )
01221     {
01222         return L4_EOK;
01223     }
01224     return -L4_EINVAL;
01225 }
01226
01227 void _check_n_init_shm(L4::Ipc::Snd_fpage shm_cap_fp,
01228     l4_uint64_t base, l4_umword_t size, l4_addr_t offset)
01229 {
01230     if (!shm_cap_fp.cap_received())
01231         L4Re::chksys(-L4_EINVAL);
01232
01233     L4Re::Util::Unique_cap<L4Re::Dataspace> ds(
01234         L4Re::chkcap(server_iface()->template rcv_cap<L4Re::Dataspace>(0)));
01235     L4Re::chksys(server_iface()->realloc_rcv_cap(0));
01236
01237     if (_trusted_ds_validation_enabled)
01238         L4Re::chksys(validate_ds(ds.get()), "Validating the dataspace.");
01239
01240     check_n_init_shm(cxx::move(ds), base, size, offset);
01241 }
01242
01243 bool check_features_internal()
01244 {
01245     static_assert(sizeof(l4virtio_config_hdr_t::driver_features_map)
01246         == sizeof(l4virtio_config_hdr_t::dev_features_map),
01247         "Driver and device feature maps must be of the same size");
01248
01249     // From the Virtio 1.0 specification 6.1 Driver Requirements and 6.2 Device
01250     // Requirements: A driver MUST accept VIRTIO_F_VERSION_1 if it is offered.
01251     // A device MUST offer VIRTIO_F_VERSION_1. A device MAY fail to operate
01252     // further if VIRTIO_F_VERSION_1 is not accepted.
01253     //
01254     // The L4virtio implementation does not support legacy interfaces so we
01255     // fail here if the Virtio 1.0 feature was not accepted.
01256     if (!_device_config->get_guest_feature(L4VIRTIO_FEATURE_VERSION_1))
01257         return false;
01258
01259     for (auto i = 0u;
01260         i < sizeof(l4virtio_config_hdr_t::driver_features_map)
01261             / sizeof(l4virtio_config_hdr_t::driver_features_map[0]);
01262         i++)
01263     {
01264         // Driver must not accept features that were not offered by device
01265         if (_device_config->guest_features(i)
01266             & ~_device_config->host_features(i))
01267             return false;
01268     }
01269     return check_features();
01270 }
01271
01272 void check_and_update_status(Dev_config::Status status)
01273 {
01274     // snapshot of current status
01275     Dev_config::Status current_status = _device_config->status();
01276
01277     // handle reset
01278     if (!status.raw)
01279     {
01280         _device_config->set_status(status);
01281         return;
01282     }
01283
01284     // Do no further processing in case of driver or device failure. If FAILED
01285     // or DEVICE_NEEDS_RESET are set only these fail_state bits will be set in
01286     // addition to the current status bits already set.
01287     if (current_status.fail_state() || status.fail_state())
01288     {
01289         if (current_status.fail_state() != status.fail_state())
01290         {
01291             current_status.fail_state() =
01292                 current_status.fail_state() | status.fail_state();
01293             _device_config->set_status(current_status);
01294         }
01295     }
01296     return;
01297 }

```

```

01317     }
01318
01319     // Enforce init sequence ACKNOWLEDGE, DRIVER, FEATURES_OK, DRIVER_OK.
01320     // We do not enforce that only one additional new bit is set per call.
01321     if (!status.acked() && status.driver())
01322         || (!status.driver() && status.features_ok())
01323         || (!status.features_ok() && status.driver_ok()))
01324     {
01325         current_status.device_needs_reset() = 1;
01326         _device_config->set_status(current_status);
01327         return;
01328     }
01329
01330     // only check feature compatibility before DRIVER_OK is set
01331     if (status.features_ok() && !status.driver_ok()
01332         && !check_features_internal())
01333         status.features_ok() = 0;
01334
01335     // Note that if FEATURES_OK and DRIVER_OK are both updated to being set
01336     // at the same time the above check_features_internal() is skipped; this is
01337     // considered undefined behaviour but it is not prevented.
01338     if (status.running() && !check_queues())
01339     {
01340         current_status.device_needs_reset() = 1;
01341         _device_config->set_status(current_status);
01342         return;
01343     }
01344
01345     _device_config->set_status(status);
01346 }
01347
01360 int _set_status(unsigned new_status)
01361 {
01362     if (new_status == 0)
01363     {
01364         L4Re::Util::Dbg().printf("Resetting device\n");
01365         reset();
01366         _device_config->reset_hdr(true);
01367     }
01368
01369     Dev_config::Status status(new_status);
01370     check_and_update_status(status);
01371
01372     return 0;
01373 }
01374
01375 };
01376
01377 typedef Device_t<No_custom_data> Device;
01378
01379 } // namespace Svr
01380
01381 }

```

## 16.230 virtio

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2014-2020 Kernkonzept GmbH.
00005  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00006  *
00007  */
00008 #pragma once
00009
00010 #include <l4/sys/types.h>
00011 #include <l4/cxx/bitfield>
00012 #include <l4/cxx/minmax>
00013 #include <l4/cxx/utils>
00014
00015 #include <limits.h>
00016 #include <string.h>
00017 #include <stdio.h>
00018
00019 #include "../virtqueue"
00020
00026 namespace L4virtio {
00027 namespace Svr {
00028
00032 struct Dev_status
00033 {
00034     unsigned char raw;
00035     Dev_status() = default;

```

```

00036
00038     explicit Dev_status(l4_uint32_t v) : raw(v) {}
00039
00040     CXX_BITFIELD_MEMBER(0, 0, acked, raw);
00041     CXX_BITFIELD_MEMBER(1, 1, driver, raw);
00042     CXX_BITFIELD_MEMBER(2, 2, driver_ok, raw);
00043     CXX_BITFIELD_MEMBER(3, 3, features_ok, raw);
00044     CXX_BITFIELD_MEMBER(6, 7, fail_state, raw);
00045     CXX_BITFIELD_MEMBER(6, 6, device_needs_reset, raw);
00046     CXX_BITFIELD_MEMBER(7, 7, failed, raw);
00047
00057     bool running() const
00058     {
00059         return (raw == 0xf);
00060     }
00061 };
00062
00066 struct Dev_features
00067 {
00068     l4_uint32_t raw;
00069     Dev_features() = default;
00070
00072     explicit Dev_features(l4_uint32_t v) : raw(v) {}
00073
00074     CXX_BITFIELD_MEMBER(28, 28, ring_indirect_desc, raw);
00075     CXX_BITFIELD_MEMBER(29, 29, ring_event_idx, raw);
00076 };
00077
00078
00087 class Virtqueue : public L4virtio::Virtqueue
00088 {
00089 public:
00093     class Head_desc
00094     {
00095     friend class Virtqueue;
00096 private:
00097     Virtqueue::Desc const *_d;
00098     Head_desc(Virtqueue *r, unsigned i) : _d(r->desc(i)) {}
00099
00100     struct Null_ptr_check;
00101
00102 public:
00104     Head_desc() : _d(0) {}
00105
00107     bool valid() const { return _d; }
00108
00110     operator Null_ptr_check const * () const
00111     { return reinterpret_cast<Null_ptr_check const *>(_d); }
00112
00114     Desc const *desc() const
00115     { return _d; }
00116 };
00117
00118 struct Request : Head_desc
00119 {
00120     Virtqueue *ring = nullptr;
00121     Request() = default;
00122 private:
00123     friend class Virtqueue;
00124     Request(Virtqueue *r, unsigned i) : Head_desc(r, i), ring(r) {}
00125 };
00126
00127
00138 Request next_avail()
00139 {
00140     if (L4_LIKELY(_current_avail != _avail->idx))
00141     {
00142         rmb();
00143         unsigned head = _current_avail & _idx_mask;
00144         ++_current_avail;
00145         return Request(this, _avail->ring[head]);
00146     }
00147     return Request();
00148 }
00149
00156 bool desc_avail() const
00157 {
00158     return _current_avail != _avail->idx;
00159 }
00160
00171 void consumed(Head_desc const &r, l4_uint32_t len = 0)
00172 {
00173     l4_uint16_t i = _used->idx & _idx_mask;
00174     _used->ring[i] = Used_elem(r._d - _desc, len);
00175     wmb();
00176     ++_used->idx;
00177 }

```

```

00178
00193 template<typename ITER>
00194 void consumed(ITER const &begin, ITER const &end)
00195 {
00196     l4_uint16_t added = 0;
00197     l4_uint16_t idx = _used->idx;
00198
00199     for (auto elem = begin ; elem != end; ++elem, ++added)
00200         _used->ring[(idx + added) & _idx_mask]
00201             = Used_elem(elem->first._d - _desc, elem->second);
00202
00203     wmb();
00204     _used->idx += added;
00205 }
00206
00220 template<typename QUEUE_OBSERVER>
00221 void finish(Head_desc &d, QUEUE_OBSERVER *o, l4_uint32_t len = 0)
00222 {
00223     consumed(d, len);
00224     o->notify_queue(this);
00225     d._d = 0;
00226 }
00227
00242 template<typename ITER, typename QUEUE_OBSERVER>
00243 void finish(ITER const &begin, ITER const &end, QUEUE_OBSERVER *o)
00244 {
00245     consumed(begin, end);
00246     o->notify_queue(this);
00247 }
00248
00254 void disable_notify()
00255 {
00256     if (L4_LIKELY(ready()))
00257         _used->flags.no_notify() = 1;
00258 }
00259
00265 void enable_notify()
00266 {
00267     if (L4_LIKELY(ready()))
00268         _used->flags.no_notify() = 0;
00269 }
00270
00279 Desc const *desc(unsigned idx) const
00280 { return _desc + idx; }
00281
00282 };
00283
00287 struct Data_buffer
00288 {
00289     char *pos;
00290     l4_uint32_t left;
00291
00292     Data_buffer() = default;
00293
00303 template<typename T>
00304 explicit Data_buffer(T *p)
00305 : pos(reinterpret_cast<char *>(p)), left(sizeof(T))
00306 {}
00307
00317 template<typename T>
00318 void set(T *p)
00319 {
00320     pos = reinterpret_cast<char *>(p);
00321     left = sizeof(T);
00322 }
00323
00335 l4_uint32_t copy_to(Data_buffer *dst, l4_uint32_t max = UINT_MAX)
00336 {
00337     unsigned long bytes = cxx::min(cxx::min(left, dst->left), max);
00338     memcpy(dst->pos, pos, bytes);
00339     left -= bytes;
00340     pos += bytes;
00341     dst->left -= bytes;
00342     dst->pos += bytes;
00343     return bytes;
00344 }
00345
00356 l4_uint32_t skip(l4_uint32_t bytes)
00357 {
00358     unsigned long b = cxx::min(left, bytes);
00359     left -= b;
00360     pos += b;
00361     return b;
00362 }
00363
00369 bool done() const
00370 { return left == 0; }

```

```

00371 };
00372
00373 class Request_processor;
00374
00375 struct Bad_descriptor
00376 {
00377     enum Error
00378     {
00379         Bad_address,
00380         Bad_rights,
00381         Bad_flags,
00382         Bad_next,
00383         Bad_size
00384     };
00385 };
00386
00387 Request_processor const *proc;
00388
00389 // The error code
00390 Error error;
00391
00392 Bad_descriptor(Request_processor const *proc, Error e)
00393 : proc(proc), error(e)
00394 {}
00395
00396 char const *message() const
00397 {
00398     static char const *const err[] =
00399     {
00400         [Bad_address] = "Descriptor address cannot be translated",
00401         [Bad_rights]   = "Insufficient memory access rights",
00402         [Bad_flags]    = "Invalid descriptor flags",
00403         [Bad_next]     = "The descriptor's `next` index is invalid",
00404         [Bad_size]     = "Invalid size of the memory block"
00405     };
00406
00407     if (error >= (sizeof(err) / sizeof(err[0])) || !err[error])
00408         return "Unknown error";
00409
00410     return err[error];
00411 }
00412 };
00413
00414 class Request_processor
00415 {
00416 private:
00417     Virtqueue::Desc const *_table;
00418
00419     Virtqueue::Desc _current;
00420
00421     l4_uint16_t _num;
00422
00423 public:
00424     template<typename DESC_MAN, typename ...ARGS>
00425     void start(DESC_MAN *dm, Virtqueue *ring, Virtqueue::Head_desc const &request, ARGS... args)
00426     {
00427         _current = cxx::access_once(request.desc());
00428
00429         if (_current.flags.indirect())
00430         {
00431             dm->load_desc(_current, this, &_table);
00432             _num = _current.len / sizeof(Virtqueue::Desc);
00433             if (L4_UNLIKELY(!_num))
00434                 throw Bad_descriptor(this, Bad_descriptor::Bad_size);
00435
00436             _current = cxx::access_once(_table);
00437         }
00438         else
00439         {
00440             _table = ring->desc(0);
00441             _num = ring->num();
00442         }
00443
00444         dm->load_desc(_current, this, cxx::forward<ARGS>(args)...);
00445     }
00446
00447     template<typename DESC_MAN, typename ...ARGS>
00448     Virtqueue::Request const &start(DESC_MAN *dm, Virtqueue::Request const &request, ARGS... args)
00449     {
00450         start(dm, request.ring, request, cxx::forward<ARGS>(args)...);
00451         return request;
00452     }
00453
00454     Virtqueue::Desc::Flags current_flags() const
00455     { return _current.flags; }
00456
00457     bool has_more() const

```

```

00535     { return _current.flags.next(); }
00536
00550     template<typename DESC_MAN, typename ...ARGS>
00551     bool next(DESC_MAN *dm, ARGS... args)
00552     {
00553         if (!_current.flags.next())
00554             return false;
00555
00556         if (L4_UNLIKELY(_current.next >= _num))
00557             throw Bad_descriptor(this, Bad_descriptor::Bad_next);
00558
00559         _current = cxx::access_once(_table + _current.next);
00560
00561         if (0) // we ignore this for performance reasons
00562             if (L4_UNLIKELY(_current.flags.indirect()))
00563                 throw Bad_descriptor(this, Bad_descriptor::Bad_flags);
00564
00565         // must throw an exception in case of a bad descriptor
00566         dm->load_desc(_current, this, cxx::forward<ARGS>(args)...);
00567         return true;
00568     }
00569 };
00570
00571 }
00572 }

```

## 16.231 virtio-console

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2019-2023 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *            Phillip Raffeck <phillip.raffeck@kernkonzept.com>
00007  *            Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00008  *            Jan Klötzke <jan.kloetzke@kernkonzept.com>
00009  */
00010 #pragma once
00011
00012 #include <l4/l4virtio/server/l4virtio>
00013 #include <l4/re/error_helper>
00014
00015 namespace L4virtio { namespace Svr { namespace Console {
00016
00018 struct Features : Dev_config::Features
00019 {
00020     Features() = default;
00021     explicit Features(l4_uint32_t raw) : Dev_config::Features(raw) {}
00023     CXX_BITFIELD_MEMBER(0, 0, console_size, raw);
00025     CXX_BITFIELD_MEMBER(1, 1, console_multiport, raw);
00027     CXX_BITFIELD_MEMBER(2, 2, emerg_write, raw);
00028 };
00029
00031 struct Control_message
00032 {
00034     enum Events
00035     {
00037         Device_ready = 0,
00039         Device_add = 1,
00041         Device_remove = 2,
00043         Port_ready = 3,
00045         Console_port = 4,
00047         Resize = 5,
00049         Port_open = 6,
00051         Port_name = 7,
00052     };
00053
00054     l4_uint32_t id;
00055     l4_uint16_t event;
00056     l4_uint16_t value;
00057
00058     Control_message() {}
00059     Control_message(l4_uint32_t i, l4_uint16_t e, l4_uint16_t v)
00060         : id(i), event(e), value(v)
00061     {}
00062 };
00063
00065 struct Control_request
00066 {
00068     Control_message *msg;
00070     l4_uint32_t len;
00072     Driver_mem_region *mem;
00073 };

```



```

00074
00107 struct Port
00108 {
00112     enum Port_status
00113     {
00115         Port_disabled = 0,
00117         Port_added,
00119         Port_ready,
00121         Port_open,
00123         Port_failed
00124     };
00125
00127     enum { Control_queue_size = 0x10 };
00128
00129     Virtqueue tx;
00130     Virtqueue rx;
00131     Port_status status;
00132     unsigned vq_max;
00133
00134     Port() : status(Port_disabled), vq_max(Control_queue_size) {}
00135     Port(Port const &) = delete;
00136     Port &operator = (Port const &) = delete;
00137
00138     virtual ~Port() = default;
00139
00141     bool is_open() const
00142     { return status == Port_open; }
00143
00145     virtual void reset()
00146     {
00147         status = Port_disabled;
00148     }
00149
00151     bool queues_ready() const
00152     { return tx.ready() && rx.ready(); }
00153
00155     bool rx_ready() const
00156     { return is_open() && rx.ready(); }
00157
00159     bool tx_ready() const
00160     { return is_open() && tx.ready(); }
00161 };
00162
00185 class Virtio_con : public L4virtio::Svr::Device
00186 {
00187     enum Virtqueue_names
00188     {
00189         Ctrl_rx = 2,
00190         Ctrl_tx = 3,
00191     };
00192
00193     struct Serial_config_space
00194     {
00195         l4_uint16_t cols;
00196         l4_uint16_t rows;
00197         l4_uint32_t max_nr_ports;
00198         l4_uint32_t emerg_wr;
00199     } __attribute__((packed));
00200
00201 public:
00211     explicit Virtio_con(unsigned max_ports, bool enable_multiport)
00212     : L4virtio::Svr::Device(&dev_config),
00213       _num_ports(enable_multiport ? max_ports : 1),
00214       _dev_config(L4VIRTIO_VENDOR_KK, L4VIRTIO_ID_CONSOLE,
00215                  enable_multiport ? max_ports * 2 + 2 : 2)
00216     {
00217         if (_num_ports < 1)
00218             L4Re::chksys(-L4_EINVAL, "At least one port is required.");
00219
00220         Features hf(0);
00221
00222         hf.console_multiport() = enable_multiport;
00223
00224         _dev_config.host_features(0) = hf.raw;
00225
00226         if (enable_multiport)
00227             _dev_config.priv_config()->max_nr_ports = _num_ports;
00228         _dev_config.reset_hdr();
00229     }
00230
00231     void reset_queue_configs()
00232     {
00233         for (unsigned q = 0; q < _dev_config.num_queues(); ++q)
00234             reset_queue_config(q, max_queue_size(q));
00235     }
00236
00237     int reconfig_queue(unsigned index) override

```

```

00238 {
00239     if (index >= _dev_config.num_queues())
00240         return -L4_ERANGE;
00241
00242     if (setup_queue(get_queue(index), index, max_queue_size(index)))
00243         return 0;
00244
00245     return -L4_EINVAL;
00246 }
00247
00252 bool multiport_enabled() const
00253 {
00254     return _negotiated_features.console_multiport()
00255         && _dev_config.num_queues() > Ctrl_rx;
00256 }
00257
00258 bool ctrl_queue_ready() const
00259 { return _ctrl_port.is_open(); }
00260
00261 bool check_features(void) override
00262 {
00263     _negotiated_features = Features(_dev_config.negotiated_features(0));
00264     return true;
00265 }
00266
00267 bool check_queues() override
00268 {
00269     // NOTE
00270     // The VIRTIO specification states:
00271     // "The port 0 receive and transmit queues always exist"
00272     // The linux driver however does not setup port 0 if the multiport feature
00273     // is negotiated.
00274     // We just go along with the linux driver and do not expect port 0 to be up,
00275     // if the multiport feature is negotiated.
00276
00277     if (multiport_enabled())
00278         // If MULTIPORT was negotiated, ctrl queues should be set up.
00279         return _ctrl_port.queues_ready();
00280
00281     // If MULTIPORT was not negotiated, port 0 should be set up.
00282     port(0)->status = Port::Port_open;
00283     return port(0)->queues_ready();
00284 }
00285
00298 int port_add(unsigned idx)
00299 {
00300     Port *p = port(idx);
00301
00302     if (p->status != Port::Port_disabled)
00303         return -L4_EPERM;
00304
00305     int ret = send_control_message(idx, Control_message::Device_add);
00306
00307     if (ret == L4_EOK)
00308         p->status = Port::Port_added;
00309
00310     return ret;
00311 }
00312
00325 int port_remove(unsigned idx)
00326 {
00327     Port *p = port(idx);
00328
00329     if (p->status != Port::Port_open
00330         && p->status != Port::Port_ready
00331         && p->status != Port::Port_failed)
00332         return -L4_EPERM;
00333
00334     int ret = send_control_message(idx, Control_message::Device_remove);
00335
00336     if (ret == L4_EOK)
00337         p->reset();
00338
00339     return ret;
00340 }
00341
00355 int port_open(unsigned idx, bool open)
00356 {
00357     Port *p = port(idx);
00358
00359     if ((open && p->status != Port::Port_ready)
00360         || (!open && p->status != Port::Port_open))
00361         return -L4_EPERM;
00362
00363     int ret = send_control_message(idx, Control_message::Port_open, open);
00364
00365     if (ret == L4_EOK)

```

```

00366     p->status = open ? Port::Port_open : Port::Port_ready;
00367
00368     return ret;
00369 }
00370
00393 int send_control_message(l4_uint32_t idx, l4_uint16_t event,
00394                          l4_uint16_t value = 0, const char *name = 0)
00395 {
00396     if (!ctrl_queue_ready())
00397         return -L4_ENODEV;
00398
00399     Virtqueue *q = &_ctrl_port.rx;
00400     if (!q->ready())
00401         return -L4_ENODEV;
00402
00403     Virtqueue::Request r = q->next_avail();
00404     if (!r)
00405         return -L4_EBUSY;
00406
00407     Request_processor rp;
00408     Control_request req;
00409     rp.start(this, r, &req);
00410
00411     if (req.len < sizeof(Control_message))
00412         return -L4_ENOMEM;
00413
00414     Control_message msg(idx, event, value);
00415
00416     memcpy(req.msg, &msg, sizeof(msg));
00417
00418     if (event == Control_message::Port_name && name)
00419     {
00420         size_t name_len = cxx::min(req.len - sizeof(msg), strlen(name));
00421         memcpy(reinterpret_cast<char*>(req.msg) + sizeof(msg), name, name_len);
00422         q->finish(r, this, sizeof(msg) + name_len);
00423     }
00424     else
00425         q->finish(r, this, sizeof(msg));
00426
00427     return L4_EOK;
00428 }
00429
00443 int handle_control_message()
00444 {
00445     Virtqueue *q = &_ctrl_port.tx;
00446     if (!q->ready())
00447         return -L4_ENODEV;
00448
00449     Virtqueue::Request r;
00450     while ((r = q->next_avail()))
00451     {
00452         Request_processor rp;
00453         Control_request req;
00454
00455         rp.start(this, r, &req);
00456
00457         Control_message msg;
00458         if (req.len < sizeof(msg))
00459         {
00460             // Just ignore malformed input.
00461             q->consumed(r);
00462             return L4_EOK;
00463         }
00464
00465         memcpy(&msg, req.msg, sizeof(msg));
00466         q->consumed(r);
00467
00468         if (_ctrl_port.status == Port::Port_disabled)
00469         {
00470             // When the control queue is disabled, only device ready is accepted.
00471             if (msg.event == Control_message::Device_ready)
00472             {
00473                 if (msg.value)
00474                     _ctrl_port.status = Port::Port_open;
00475             }
00476
00477             process_device_ready(msg.value);
00478             continue;
00479         }
00480
00481         if (!ctrl_queue_ready())
00482             continue;
00483
00484         switch (msg.event)
00485         {
00486             case Control_message::Port_ready:
00487                 // Ignore invalid port ids

```

```

00488         if (msg.id >= max_ports())
00489             break;
00490
00491         // Ignore repeated PORT_READY messages.
00492         if ((port(msg.id)->status == Port::Port_disabled)
00493             || (!msg.value && port(msg.id)->status == Port::Port_added))
00494             break;
00495
00496         process_port_ready(msg.id, msg.value);
00497         break;
00498
00499     case Control_message::Port_open:
00500         // Ignore invalid port ids
00501         if (msg.id >= max_ports())
00502             break;
00503
00504         // Ignore misplaced PORT_OPEN messages.
00505         if ((msg.value && port(msg.id)->status != Port::Port_ready)
00506             || (!msg.value && port(msg.id)->status != Port::Port_open))
00507             break;
00508
00509         process_port_open(msg.id, msg.value);
00510         break;
00511     default:
00512         return -L4_EINVAL;
00513     }
00514 }
00515
00516 return L4_EOK;
00517 }
00518
00520 void load_desc(L4virtio::Virtqueue::Desc const &desc,
00521               Request_processor const *proc,
00522               L4virtio::Virtqueue::Desc const **table)
00523 {
00524     this->_mem_info.load_desc(desc, proc, table);
00525 }
00526
00528 void load_desc(L4virtio::Virtqueue::Desc const &desc,
00529               Request_processor const *proc,
00530               Control_request *data)
00531 {
00532     auto *region = this->_mem_info.find(desc.addr.get(), desc.len);
00533     if (L4_UNLIKELY(!region))
00534         throw Bad_descriptor(proc, Bad_descriptor::Bad_address);
00535
00536     data->msg = reinterpret_cast<Control_message *>(region->local(desc.addr));
00537     data->len = desc.len;
00538     data->mem = region;
00539 }
00540
00541 void reset() override
00542 {
00543     for (unsigned p = 0; p < _num_ports; ++p)
00544         port(p)->reset();
00545
00546     _ctrl_port.reset();
00547     reset_queue_configs();
00548     _dev_config.reset_hdr();
00549     _negotiated_features = Features(0);
00550
00551     reset_device();
00552 }
00553
00560 virtual void reset_device() {}
00561
00571 virtual void notify_queue(Virtqueue *queue) = 0;
00572
00580 virtual Port *port(unsigned port) = 0;
00581 virtual Port const *port(unsigned port) const = 0;
00582
00593 virtual void process_device_ready(l4_uint16_t value) = 0;
00594
00606 virtual void process_port_ready(l4_uint32_t id, l4_uint16_t value)
00607 {
00608     Port *p = port(id);
00609
00610     switch (p->status)
00611     {
00612     case Port::Port_added:
00613     case Port::Port_ready:
00614         p->status = value ? Port::Port_ready : Port::Port_failed;
00615         break;
00616     case Port::Port_open:
00617         if (!value)
00618             p->status = Port::Port_failed;
00619         break;

```

```

00620         default:
00621             // invalid state for PORT_READY message
00622             break;
00623     }
00624 }
00625
00636 virtual void process_port_open(l4_uint32_t id, l4_uint16_t value) = 0;
00637
00638 unsigned max_ports() const
00639 { return _num_ports; }
00640
00641 private:
00642 bool is_control_queue(unsigned q) const
00643 { return q == Ctrl_rx || q == Ctrl_tx; }
00644
00645 unsigned queue_to_port(unsigned q) const
00646 { return (q == 0 || q == 1) ? 0 : (q / 2) - 1; }
00647
00656 unsigned max_queue_size(unsigned q) const
00657 {
00658     if (is_control_queue(q))
00659         return _ctrl_port.vq_max;
00660     return port(queue_to_port(q))->vq_max;
00662 }
00663
00672 Virtqueue *get_queue(unsigned q)
00673 {
00674     Port *p;
00675     if (is_control_queue(q))
00676         p = &_ctrl_port;
00677     else
00678         p = port(queue_to_port(q));
00679
00680     if (q & 1)
00681         return &p->tx;
00682     else
00683         return &p->rx;
00684 }
00685
00686 unsigned _num_ports;
00687
00688 protected:
00689 Dev_config_t<Serial_config_space> _dev_config;
00690 Port _ctrl_port;
00691 Features _negotiated_features{0};
00692 };
00693
00694 }}} // name space

```

## 16.232 virtio-console-device

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2019-2023 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *            Phillip Raffeck <phillip.raffeck@kernkonzept.com>
00007  *            Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00008  *            Jan Klötzke <jan.kloetzke@kernkonzept.com>
00009  */
00010 #pragma once
00011
00012 #include <l4/cxx/bitmap>
00013 #include <l4/cxx/static_vector>
00014 #include <l4/l4virtio/server/l4virtio>
00015 #include <l4/l4virtio/server/virtio-console>
00016 #include <l4/re/error_helper>
00017
00018 namespace L4virtio { namespace Svr { namespace Console {
00019
00026 struct Device_port : public Port
00027 {
00028     struct Buffer : Data_buffer
00029     {
00030         Buffer() = default;
00031         Buffer(Driver_mem_region const *r,
00032              Virtqueue::Desc const &d,
00033              Request_processor const *)
00034         {
00035             pos = static_cast<char *>(r->local(d.addr));
00036             left = d.len;
00037         }

```

```

00038     };
00039
00040     Request_processor rp;
00041     Virtqueue::Request request;
00042     Buffer src;
00043
00044     bool poll_in_req = true;
00045     bool poll_out_req = false;
00046
00047     void reset() override
00048     {
00049         Port::reset();
00050         request = Virtqueue::Request();
00051     }
00052 };
00053
00054 class Device
00055 : public Virtio_con
00056 {
00057     class Irq_object : public L4::Irqep_t<Irq_object>
00058     {
00059     public:
00060         Irq_object(Device *parent) : _parent(parent) {}
00061
00062         void handle_irq() { _parent->kick(); }
00063
00064     private:
00065         Device *_parent;
00066     };
00067
00068 protected:
00069     L4::Epiface *irq_iface()
00070     { return &_irq_handler; }
00071
00072 public:
00073     explicit Device(unsigned vq_max)
00074     : Virtio_con(1, false),
00075       _irq_handler(this),
00076       _ports(cxx::make_unique<Device_port[]>(1)),
00077       _control_message_stalled(0)
00078     {
00079         _ports[0].vq_max = vq_max;
00080         reset_queue_configs();
00081     }
00082
00083     explicit Device(unsigned vq_max, unsigned ports)
00084     : Virtio_con(ports, true),
00085       _irq_handler(this),
00086       _ports(cxx::make_unique<Device_port[]>(ports)),
00087       _control_message_stalled(0)
00088     {
00089         if (ports > sizeof(_control_message_stalled) * 8)
00090             L4Re::chksys(-L4_ENOMEM, "Only 32 ports are supported.");
00091         for (unsigned i = 0; i < ports; ++i)
00092             _ports[i].vq_max = vq_max;
00093         reset_queue_configs();
00094     }
00095
00096     explicit Device(cxx::static_vector<unsigned> const &vq_max_nums)
00097     : Virtio_con(vq_max_nums.size(), true),
00098       _irq_handler(this),
00099       _ports(cxx::make_unique<Device_port[]>(max_ports())),
00100       _control_message_stalled(0)
00101     {
00102         if (max_ports() > sizeof(_control_message_stalled) * 8)
00103             L4Re::chksys(-L4_ENOMEM, "Only 32 ports are supported.");
00104         for (unsigned i = 0; i < vq_max_nums.size(); ++i)
00105             _ports[i].vq_max = vq_max_nums[i];
00106         reset_queue_configs();
00107     }
00108
00109     void register_single_driver_irq() override
00110     {
00111         _kick_driver_irq = L4Re::Util::Unique_cap<L4::Irq>(
00112             L4Re::chkcap(server_iface()->rcv_cap<L4::Irq>(0)));
00113         L4Re::chksys(server_iface()->realloc_rcv_cap(0));
00114     }
00115
00116     L4::Cap<L4::Irq> device_notify_irq() const override
00117     { return _irq_handler.obj_cap(); }
00118
00119     void notify_queue(Virtqueue *queue) override
00120     {
00121         if (queue->no_notify_guest())
00122             return;
00123     }

```

```

00213     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00214     _kick_driver_irq->trigger();
00215 }
00216
00220 virtual void rx_data_available(unsigned port) = 0;
00221
00225 virtual void tx_space_available(unsigned port) = 0;
00226
00230 virtual bool queues_stopped()
00231 { return false; }
00232
00233 void trigger_driver_config_irq() override
00234 {
00235     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_CONFIG);
00236     _kick_driver_irq->trigger();
00237 }
00238
00239 void kick()
00240 {
00241     if (queues_stopped())
00242         return;
00243
00244     // We're not interested in logging any errors, just ignore return value.
00245     do_control_work();
00246
00247     for (unsigned i = 0; i < max_ports(); ++i)
00248     {
00249         auto &p = _ports[i];
00250         if (p.poll_in_req && p.tx_ready() && p.tx.desc_avail())
00251         {
00252             p.poll_in_req = false;
00253             rx_data_available(i);
00254         }
00255
00256         if (p.poll_out_req && p.rx_ready() && p.rx.desc_avail())
00257         {
00258             p.poll_out_req = false;
00259             tx_space_available(i);
00260         }
00261     }
00262 }
00263
00278 unsigned port_read(char *buf, unsigned len, unsigned port = 0)
00279 {
00280     unsigned total = 0;
00281     Device_port &p = _ports[port];
00282     Virtqueue *q = &p.tx;
00283
00284     Data_buffer dst;
00285     dst.pos = buf;
00286     dst.left = len;
00287
00288     while (dst.left)
00289     {
00290         try
00291         {
00292             // Make sure we have a valid request where we can read data from
00293             if (!p.request.valid())
00294             {
00295                 p.request = p.tx_ready() ? q->next_avail()
00296                                         : Virtqueue::Request();
00297                 if (!p.request.valid())
00298                     break;
00299
00300                 p.rp.start(mem_info(), p.request, &p.src);
00301             }
00302
00303             total += p.src.copy_to(&dst);
00304
00305             // We might have eaten up the current descriptor. Move to the next
00306             // if this is the case. At the end of the descriptor chain we have
00307             // to retire the current request altogether.
00308             if (!p.src.left)
00309             {
00310                 if (!p.rp.next(mem_info(), &p.src))
00311                 {
00312                     q->finish(p.request, this);
00313                     p.request = Virtqueue::Request();
00314                 }
00315             }
00316         }
00317         catch (Bad_descriptor const &)
00318         {
00319             q->finish(p.request, this);
00320             p.request = Virtqueue::Request();
00321         }
00322     }

```

```

00323
00324     if (total < len)
00325         p.poll_in_req = true;
00326
00327     return total;
00328 }
00329
00345 unsigned port_write(char const *buf, unsigned len, unsigned port = 0)
00346 {
00347     unsigned total = 0;
00348     Device_port &p = _ports[port];
00349     Virtqueue *q = &p.rx;
00350
00351     Data_buffer src;
00352     src.pos = const_cast<char*>(buf);
00353     src.left = len;
00354
00355     Request_processor rp;
00356     while (src.left)
00357     {
00358         auto r = p.rx_ready() ? q->next_avail() : Virtqueue::Request();
00359         if (!r.valid())
00360             break;
00361
00362         l4_uint32_t chunk = 0;
00363         try
00364         {
00365             Device_port::Buffer dst;
00366             rp.start(mem_info(), r, &dst);
00367
00368             for (;;)
00369             {
00370                 chunk += src.copy_to(&dst);
00371                 if (!src.left)
00372                     break;
00373                 if (!rp.next(mem_info(), &dst))
00374                     break;
00375             }
00376         }
00377         catch (Bad_descriptor const &)
00378         { /* Ignore */ }
00379
00380         q->finish(r, this, chunk);
00381         total += chunk;
00382     }
00383
00384     if (total < len)
00385         p.poll_out_req = true;
00386
00387     return total;
00388 }
00389
00400 int do_control_work()
00401 {
00402     if (_control_message_stalled)
00403         resend_stalled_control_messages();
00404
00405     return handle_control_message();
00406 }
00407
00408 bool ctrl_work_pending() const
00409 {
00410     return (L4_LIKELY(_ctrl_port.queues_ready())
00411         && _ctrl_port.tx.desc_avail()) || _control_message_stalled;
00412 }
00413
00426 void process_device_ready(l4_uint16_t value) override
00427 {
00428     if (!value)
00429         return;
00430
00431     for (unsigned i = 0; i < max_ports(); ++i)
00432         check_stalled(i, port_add(i));
00433 }
00434
00448 void process_port_ready(l4_uint32_t id, l4_uint16_t value) override
00449 {
00450     Virtio_con::process_port_ready(id, value);
00451
00452     Port *p = port(id);
00453     if (p->status == Port::Port_failed)
00454         check_stalled(id, port_remove(id));
00455 }
00456
00470 void process_port_open(l4_uint32_t id, l4_uint16_t value) override
00471 { check_stalled(id, port_open(id, value)); }
00472

```



```

00473 protected:
00478 void resend_stalled_control_messages()
00479 {
00480     for (unsigned i = 0; i < max_ports(); ++i)
00481     {
00482         if (!(_control_message_stalled & (1U « i)))
00483             continue;
00484
00485         switch(_ports[i].status)
00486         {
00487             case Port::Port_disabled:
00488                 check_stalled(i, port_add(i));
00489                 break;
00490             case Port::Port_failed:
00491                 check_stalled(i, port_remove(i));
00492                 break;
00493             case Port::Port_ready:
00494                 check_stalled(i, port_open(i, 1));
00495                 break;
00496             case Port::Port_open:
00497                 check_stalled(i, port_open(i, 0));
00498                 break;
00499             default:
00500                 break;
00501         }
00502     }
00503 }
00504
00505 Port* port(unsigned idx) override
00506 {
00507     return &_ports[idx];
00508 }
00509
00510 Port const *port(unsigned idx) const override
00511 {
00512     return &_ports[idx];
00513 }
00514
00515 void check_stalled(unsigned port, int retval)
00516 {
00517     if (retval == -L4_ENODEV || retval == -L4_EBUSY)
00518         _control_message_stalled |= 1U « port;
00519     else
00520         _control_message_stalled &= ~1U « port;
00521 }
00522
00523 private:
00524     Irq_object _irq_handler;
00525     cxx::unique_ptr<Device_port[]> _ports;
00526     L4Re::Util::Unique_cap<L4::Irq> _kick_driver_irq;
00528     l4_uint32_t _control_message_stalled;
00529 };
00530
00531 }}} // name space

```

## 16.233 virtio-scmi-device

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2024 Kernkonzept GmbH.
00005  * Author(s): Christian Pöttsch <christian.potetzsch@kernkonzept.com>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/cxx/bitmap>
00012 #include <l4/l4virtio/l4virtio>
00013 #include <l4/l4virtio/server/l4virtio>
00014 #include <l4/l4virtio/server/virtio>
00015 #include <l4/re/util/object_registry>
00016
00017 #include <map>
00018 #include <vector>
00019
00020 namespace L4virtio { namespace Svr { namespace Scmi {
00021
00022     enum
00023     {
00024         Version = 0x20000
00025     };
00026 }
00027 }

```

```

00029 enum
00030 {
00031     Success = 0,
00032     Not_supported = -1,
00033     Invalid_parameters = -2,
00034     Denied = -3,
00035     Not_found = -4,
00036     Out_of_range = -5,
00037     Busy = -6,
00038     Comms_error = -7,
00039     Generic_error = -8,
00040     Hardware_error = -9,
00041     Protocol_error = -10
00042 };
00043
00045 struct Scmi_hdr_t
00046 {
00047     14_uint32_t hdr_raw = 0;
00048     CXX_BITFIELD_MEMBER(18, 27, token, hdr_raw);
00049     CXX_BITFIELD_MEMBER(10, 17, protocol_id, hdr_raw);
00050     CXX_BITFIELD_MEMBER( 8,  9, message_type, hdr_raw);
00051     CXX_BITFIELD_MEMBER( 0,  7, message_id, hdr_raw);
00052 };
00053
00055 enum
00056 {
00057     Base_protocol = 0x10,
00058     Power_domain_management_protocol = 0x11,
00059     System_power_management_protocol = 0x12,
00060     Performance_domain_management_protocol = 0x13,
00061     Clock_management_protocol = 0x14,
00062     Sensor_management_protocol = 0x15,
00063     Reset_domain_management_protocol = 0x16,
00064     Voltage_domain_management_protocol = 0x17
00065 };
00066
00068 enum
00069 {
00070     Protocol_version = 0x0,
00071     Protocol_attributes = 0x1,
00072     Protocol_message_attributes = 0x2,
00073 };
00074
00076 enum
00077 {
00078     Base_discover_vendor = 0x3,
00079     Base_discover_sub_vendor = 0x4,
00080     Base_discover_implementation_version = 0x5,
00081     Base_discover_list_protocols = 0x6,
00082     Base_discover_agent = 0x7,
00083     Base_notify_errors = 0x8,
00084     Base_set_device_permissions = 0x9,
00085     Base_set_protocol_permissions = 0xa,
00086     Base_reset_agent_configuration = 0xb
00087 };
00088
00090 struct Base_attr_t
00091 {
00092     14_uint32_t attr_raw = 0;
00093     CXX_BITFIELD_MEMBER(8, 15, nagents, attr_raw);
00094     CXX_BITFIELD_MEMBER(0,  7, nprotos, attr_raw);
00095 };
00096
00098 enum
00099 {
00100     Performance_domain_attributes = 0x3,
00101     Performance_describe_levels = 0x4,
00102     Performance_limits_set = 0x5,
00103     Performance_limits_get = 0x6,
00104     Performance_level_set = 0x7,
00105     Performance_level_get = 0x8,
00106     Performance_notify_limits = 0x9,
00107     Performance_notify_level = 0xa,
00108     Performance_describe_fastchannel = 0xb,
00109 };
00110
00112 struct Performance_attr_t
00113 {
00114     14_uint32_t attr_raw = 0;
00115     CXX_BITFIELD_MEMBER(16, 16, power, attr_raw);
00116     CXX_BITFIELD_MEMBER( 0, 15, domains, attr_raw);
00117
00118     14_uint32_t stat_addr_low = 0;
00119     14_uint32_t stat_addr_high = 0;
00120     14_uint32_t stat_len = 0;
00121 };
00122

```

```

00124 struct Performance_domain_attr_t
00125 {
00126     14_uint32_t attr_raw = 0;
00127     CXX_BITFIELD_MEMBER(31, 31, set_limits, attr_raw);
00128     CXX_BITFIELD_MEMBER(30, 30, set_perf_level, attr_raw);
00129     CXX_BITFIELD_MEMBER(29, 29, perf_limits_change_notify, attr_raw);
00130     CXX_BITFIELD_MEMBER(28, 28, perf_level_change_notify, attr_raw);
00131     CXX_BITFIELD_MEMBER(27, 27, fast_channel, attr_raw);
00132
00133     14_uint32_t rate_limit_raw = 0;
00134     CXX_BITFIELD_MEMBER( 0, 19, rate_limit, rate_limit_raw);
00135
00136     14_uint32_t sustained_freq = 0;
00137     14_uint32_t sustained_perf_level = 0;
00138     char name[16] = { 0 };
00139 };
00140
00142 struct Performance_describe_levels_n_t
00143 {
00144     14_uint32_t num_levels_raw = 0;
00145     CXX_BITFIELD_MEMBER(16, 31, nremain_perf_levels, num_levels_raw);
00146     CXX_BITFIELD_MEMBER( 0, 11, nperf_levels, num_levels_raw);
00147 };
00148
00150 struct Performance_describe_level_t
00151 {
00152     14_uint32_t perf_level = 0;
00153     14_uint32_t power_cost = 0;
00154     14_uint16_t trans_latency = 0;
00155     14_uint16_t res0 = 0;
00156 };
00157
00158 template<typename OBSERV>
00159 struct Queue_worker : Request_processor
00160 {
00161     Queue_worker(OBSERV *o, Virtqueue *queue)
00162     : o(o), q(queue)
00163     {}
00164
00165     bool init_queue()
00166     {
00167         auto r = q->next_avail();
00168
00169         if (L4_UNLIKELY(!r))
00170             return false;
00171
00172         head = start(o->mem_info(), r, &req);
00173
00174         return true;
00175     }
00176
00177     bool next()
00178     { return Request_processor::next(o->mem_info(), &req); }
00179
00180     void finish(14_uint32_t total)
00181     { q->finish(head, o, total); }
00182
00183     template<typename T>
00184     14_ssize_t read(Data_buffer *buf, T *data, 14_size_t s = sizeof(T))
00185     {
00186         buf->pos = reinterpret_cast<char *>(data);
00187         buf->left = s;
00188         14_size_t chunk = 0;
00189         for (;;)
00190         {
00191             chunk += req.copy_to(buf);
00192             if (req.done())
00193                 next();
00194             if (!buf->left)
00195                 break;
00196         }
00197         if (chunk != s)
00198             return -1;
00199         return chunk;
00200     }
00201
00202     template<typename T>
00203     14_ssize_t write(Data_buffer *buf, T *data, 14_size_t s = sizeof(T))
00204     {
00205         buf->pos = reinterpret_cast<char *>(data);
00206         buf->left = s;
00207         14_size_t chunk = 0;
00208         for (;;)
00209         {
00210             chunk += buf->copy_to(&req);
00211             if (req.done())
00212                 next();

```

```

00213         if (!buf->left)
00214             break;
00215     }
00216     if (chunk != s)
00217         return -1;
00218     return chunk;
00219 }
00220
00221 l4_ssize_t handle_request()
00222 {
00223     try
00224     {
00225         if (!head && L4_UNLIKELY(!init_queue()))
00226             return 0;
00227
00228         for (;;)
00229         {
00230             l4_ssize_t total = 0;
00231             l4_ssize_t res = 0;
00232             Scmi_hdr_t hdr;
00233             Data_buffer buf = Data_buffer(&hdr);
00234             if ((res = read(&buf, &hdr)) < 0)
00235                 return res;
00236
00237             // Search/execute handler for given protocol
00238             auto proto = o->proto(hdr.protocol_id());
00239             if (proto)
00240             {
00241                 if ((res = proto->handle_request(hdr, buf, this)) < 0)
00242                     return res;
00243                 total += res;
00244             }
00245             else
00246             {
00247                 if ((res = write(&buf, &hdr)) < 0)
00248                     return res;
00249                 total += res;
00250
00251                 l4_int32_t status = Not_supported;
00252                 if ((res = write(&buf, &status)) < 0)
00253                     return res;
00254                 total += res;
00255             }
00256
00257             finish(total);
00258
00259             head = Virtqueue::Head_desc();
00260             if (L4_UNLIKELY(!init_queue()))
00261                 return 0;
00262         }
00263     }
00264     catch (L4virtio::Svr::Bad_descriptor const &e)
00265     {
00266         return e.error;
00267     }
00268     return 0;
00269 }
00270
00271 private:
00272 struct Buffer : Data_buffer
00273 {
00274     Buffer() = default;
00275     Buffer(L4virtio::Svr::Driver_mem_region const *r,
00276           Virtqueue::Desc const &d, Request_processor const *)
00277     {
00278         pos = static_cast<char *>(r->local(d.addr));
00279         left = d.len;
00280     }
00281 };
00282
00283 Virtqueue::Head_desc head;
00284 Buffer req;
00285
00286 OBSERV *o;
00287 Virtqueue *q;
00288 };
00289
00290 template<typename OBSERV>
00291 struct Proto
00292 {
00293     virtual l4_ssize_t handle_request(Scmi_hdr_t &hdr,
00294                                       Data_buffer &buf,
00295                                       Queue_worker<OBSERV> *qw) = 0;
00296 };
00297
00298 class Scmi_dev : public L4virtio::Svr::Device
00299 {
00300

```

```

00338 struct Features : L4virtio::Svr::Dev_config::Features
00339 {
00340     Features() = default;
00341     Features(l4_uint32_t raw) : L4virtio::Svr::Dev_config::Features(raw) {}
00342 };
00343
00344 struct Host_irq : public L4::Irqep_t<Host_irq>
00345 {
00346     Scmi_dev *c;
00347     explicit Host_irq(Scmi_dev *c) : c(c) {}
00348     void handle_irq() { c->kick(); }
00349 };
00350
00351 enum
00352 {
00353     Queue_size = 0x10
00354 };
00355
00356 public:
00357     Scmi_dev(L4Re::Util::Object_registry *registry)
00358     : L4virtio::Svr::Device(&_dev_config),
00359       _dev_config(L4VIRTIO_VENDOR_KK, L4VIRTIO_ID_SCM, 1),
00360       _host_irq(this),
00361       _request_worker(this, &q[0])
00362     {
00363         init_mem_info(2);
00364
00365         L4Re::chkcap(registry->register_irq_obj(&_host_irq),
00366                     "Register irq object");
00367
00368         Features hf(0);
00369         hf.ring_indirect_desc() = true;
00370
00371         _dev_config.host_features(0) = hf.raw;
00372
00373         _dev_config.set_host_feature(L4VIRTIO_FEATURE_VERSION_1);
00374         _dev_config.reset_hdr();
00375
00376         reset();
00377     }
00378
00380 void add_proto(l4_uint32_t id, Proto<Scmi_dev> *proto)
00381 { _protos.insert({id, proto}); }
00382
00383 Proto<Scmi_dev> *proto(l4_uint32_t id) const
00384 {
00385     if (_protos.find(id) != _protos.end())
00386         return _protos.at(id);
00387     return nullptr;
00388 }
00389
00390 void notify_queue(L4virtio::Virtqueue *queue)
00391 {
00392     if (queue->no_notify_guest())
00393         return;
00394
00395     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00396     _kick_guest_irq->trigger();
00397 }
00398
00399 private:
00400     L4::Cap<L4::Irq> device_notify_irq() const override
00401     { return L4::cap_cast<L4::Irq>(_host_irq.obj_cap()); }
00402
00403 void register_single_driver_irq() override
00404 {
00405     _kick_guest_irq = L4Re::Util::Unique_cap<L4::Irq>{
00406         L4Re::chkcap(server_iface()->template rcv_cap<L4::Irq>(0));
00407
00408         L4Re::chksys(server_iface()->realloc_rcv_cap(0));
00409     }
00410
00411 void kick()
00412 {
00413     if (_request_worker.handle_request() < 0)
00414         device_error();
00415 }
00416
00417 void reset() override
00418 {
00419     for (Virtqueue &q : _q)
00420         q.disable();
00421
00422     for (l4_uint32_t i = 0; i < _dev_config.num_queues(); i++)
00423         reset_queue_config(i, Queue_size);
00424 }
00425

```

```

00426 bool check_queues() override
00427 {
00428     return true;
00429 }
00430
00431 int reconfig_queue(unsigned idx) override
00432 {
00433     if (idx >= sizeof(_q) / sizeof(_q[0]))
00434         return -L4_ERANGE;
00435
00436     return setup_queue(_q + idx, idx, Queue_size);
00437 }
00438
00439 void trigger_driver_config_irq() override
00440 {
00441     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_CONFIG);
00442     _kick_guest_irq->trigger();
00443 }
00444
00445 L4virtio::Svr::Dev_config_t<L4virtio::Svr::No_custom_data> _dev_config;
00446 Host_irq _host_irq;
00447 L4Re::Util::Unique_cap<L4::Irq> _kick_guest_irq;
00448 Virtqueue _q[1];
00449 Queue_worker<Scmi_dev> _request_worker;
00450 std::map<l4_uint32_t, Proto<Scmi_dev> *> _protos;
00451 };
00452
00453 class Base_proto : public Proto<Scmi_dev>
00454 {
00455     virtual l4_int32_t fill_attr(Base_attr_t *attr) const = 0;
00456
00457     virtual std::vector<l4_uint32_t> prots() const = 0;
00458
00459     l4_ssize_t handle_request(Scmi_hdr_t &hdr, Data_buffer &buf,
00460                               Queue_worker<Scmi_dev> *qw) override
00461     {
00462         l4_ssize_t total = 0;
00463         l4_ssize_t res = 0;
00464         switch (hdr.message_id())
00465         {
00466             case Protocol_version:
00467             {
00468                 if ((res = qw->write(&buf, &hdr)) < 0)
00469                     return res;
00470                 total += res;
00471
00472                 struct
00473                 {
00474                     l4_int32_t status = Success;
00475                     l4_uint32_t version = Version;
00476                 } version;
00477                 if ((res = qw->write(&buf, &version)) < 0)
00478                     return res;
00479                 total += res;
00480                 break;
00481             }
00482             case Protocol_attributes:
00483             {
00484                 if ((res = qw->write(&buf, &hdr)) < 0)
00485                     return res;
00486                 total += res;
00487
00488                 Base_attr_t ba;
00489                 l4_int32_t status = fill_attr(&ba);
00490                 if ((res = qw->write(&buf, &status)) < 0)
00491                     return res;
00492                 total += res;
00493                 if (status == Success)
00494                 {
00495                     if ((res = qw->write(&buf, &ba)) < 0)
00496                         return res;
00497                     total += res;
00498                 }
00499                 break;
00500             }
00501             case Protocol_message_attributes:
00502             {
00503                 l4_uint32_t msg_id = 0;
00504                 if ((res = qw->read(&buf, &msg_id)) < 0)
00505                     return res;
00506
00507                 if ((res = qw->write(&buf, &hdr)) < 0)
00508                     return res;
00509                 total += res;
00510
00511                 struct
00512                 {

```

```

00521         l4_int32_t status = Not_found;
00522         l4_uint32_t attr = 0;
00523     } attr;
00524     if (msg_id >= Protocol_version &&
00525         msg_id <= Base_discover_list_protocols)
00526         attr.status = Success;
00527
00528     if ((res = qw->write(&buf, &attr)) < 0)
00529         return res;
00530     total += res;
00531     break;
00532 }
00533 case Base_discover_vendor:
00534 {
00535     if ((res = qw->write(&buf, &hdr)) < 0)
00536         return res;
00537     total += res;
00538
00539     struct
00540     {
00541         l4_int32_t status = Success;
00542         l4_uint8_t vendor_identififer[16] = { "L4Re" };
00543     } vendor;
00544     if ((res = qw->write(&buf, &vendor)) < 0)
00545         return res;
00546     total += res;
00547     break;
00548 }
00549 case Base_discover_sub_vendor:
00550 {
00551     if ((res = qw->write(&buf, &hdr)) < 0)
00552         return res;
00553     total += res;
00554
00555     struct
00556     {
00557         l4_int32_t status = Success;
00558         l4_uint8_t vendor_identififer[16] = { "Scmi" };
00559     } vendor;
00560     if ((res = qw->write(&buf, &vendor)) < 0)
00561         return res;
00562     total += res;
00563     break;
00564 }
00565 case Base_discover_implementation_version:
00566 {
00567     if ((res = qw->write(&buf, &hdr)) < 0)
00568         return res;
00569     total += res;
00570
00571     struct
00572     {
00573         l4_int32_t status = Success;
00574         l4_uint32_t version = 1;
00575     } version;
00576     if ((res = qw->write(&buf, &version)) < 0)
00577         return res;
00578     total += res;
00579     break;
00580 }
00581 case Base_discover_list_protocols:
00582 {
00583     l4_uint32_t skip = 0;
00584     if ((res = qw->read(&buf, &skip)) < 0)
00585         return res;
00586
00587     if ((res = qw->write(&buf, &hdr)) < 0)
00588         return res;
00589     total += res;
00590
00591     auto p = prots();
00592     struct
00593     {
00594         l4_int32_t status = Success;
00595         l4_uint32_t num;
00596     } proto;
00597     proto.num = p.size();
00598     if ((res = qw->write(&buf, &proto)) < 0)
00599         return res;
00600     total += res;
00601
00602     // Array of uint32 where 4 protos are packed into one uint32. So
00603     // round up to 4 bytes and fill the array byte by byte.
00604     l4_uint8_t parr[(p.size() + 3) / 4 * 4] = { 0 };
00605     for (l4_size_t i = 0; i < p.size(); i++)
00606         parr[i] = p.at(i);
00607

```

```

00608         if ((res = qw->write(&buf, parr, sizeof(parr))) < 0)
00609             return res;
00610         total += res;
00611         break;
00612     }
00613     default:
00614     {
00615         if ((res = qw->write(&buf, &hdr)) < 0)
00616             return res;
00617         total += res;
00618
00619         l4_int32_t status = Not_supported;
00620         if ((res = qw->write(&buf, &status)) < 0)
00621             return res;
00622         total += res;
00623         break;
00624     }
00625 }
00626
00627 return total;
00628 }
00629 };
00630
00662 class Perf_proto : public Proto<Scmi_dev>
00663 {
00664     virtual l4_int32_t fill_attr(Performance_attr_t *attr) const = 0;
00665
00666     virtual l4_int32_t fill_domain_attr(l4_uint32_t domain_id,
00667                                         Performance_domain_attr_t *attr) const = 0;
00668
00669     virtual l4_int32_t fill_describe_levels_n(l4_uint32_t domain_id,
00670                                               l4_uint32_t level_idx,
00671                                               Performance_describe_levels_n_t *attr) const = 0;
00672
00673     virtual l4_int32_t fill_describe_levels(l4_uint32_t domain_id,
00674                                             l4_uint32_t level_idx,
00675                                             l4_uint32_t num,
00676                                             Performance_describe_level_t *attr) const = 0;
00677
00678     virtual l4_int32_t level_set(l4_uint32_t domain_id,
00679                                 l4_uint32_t perf_level) = 0;
00680
00681     virtual l4_int32_t level_get(l4_uint32_t domain_id,
00682                                 l4_uint32_t *perf_level) const = 0;
00683
00684     l4_ssize_t handle_request(Scmi_hdr_t &hdr, Data_buffer &buf,
00685                               Queue_worker<Scmi_dev> *qw) override
00686     {
00687         l4_ssize_t total = 0;
00688         l4_ssize_t res = 0;
00689         switch (hdr.message_id())
00690         {
00691             case Protocol_version:
00692             {
00693                 if ((res = qw->write(&buf, &hdr)) < 0)
00694                     return res;
00695                 total += res;
00696
00697                 struct
00698                 {
00699                     l4_int32_t status = Success;
00700                     l4_uint32_t version = Version;
00701                 } version;
00702                 if ((res = qw->write(&buf, &version)) < 0)
00703                     return res;
00704                 total += res;
00705                 break;
00706             }
00707             case Protocol_attributes:
00708             {
00709                 if ((res = qw->write(&buf, &hdr)) < 0)
00710                     return res;
00711                 total += res;
00712
00713                 Performance_attr_t pa;
00714                 l4_int32_t status = fill_attr(&pa);
00715                 if ((res = qw->write(&buf, &status)) < 0)
00716                     return res;
00717                 total += res;
00718                 if (status == Success)
00719                 {
00720                     if ((res = qw->write(&buf, &pa)) < 0)
00721                         return res;
00722                     total += res;
00723                 }
00724                 break;
00725             }
00726         }
00727     }
00728 }

```



```

00735     case Protocol_message_attributes:
00736     {
00737         l4_uint32_t msg_id = 0;
00738         if ((res = qw->read(&buf, &msg_id)) < 0)
00739             return res;
00740
00741         if ((res = qw->write(&buf, &hdr)) < 0)
00742             return res;
00743         total += res;
00744
00745         struct
00746         {
00747             l4_int32_t status = Not_found;
00748
00749             l4_uint32_t attr_raw = 0;
00750             CXX_BITFIELD_MEMBER(0, 0, fast_channel, attr_raw); // ignored
00751         } attr;
00752         if ((msg_id >= Protocol_version &&
00753             msg_id <= Performance_describe_levels) ||
00754             (msg_id >= Performance_level_set &&
00755             msg_id <= Performance_level_get))
00756             attr.status = Success;
00757
00758         if ((res = qw->write(&buf, &attr)) < 0)
00759             return res;
00760         total += res;
00761         break;
00762     }
00763     case Performance_domain_attributes:
00764     {
00765         l4_uint32_t domain_id = 0;
00766         if ((res = qw->read(&buf, &domain_id)) < 0)
00767             return res;
00768
00769         if ((res = qw->write(&buf, &hdr)) < 0)
00770             return res;
00771         total += res;
00772
00773         Performance_domain_attr_t attr;
00774         l4_int32_t status = fill_domain_attr(domain_id, &attr);
00775         if ((res = qw->write(&buf, &status)) < 0)
00776             return res;
00777         total += res;
00778         if (status == Success)
00779         {
00780             if ((res = qw->write(&buf, &attr)) < 0)
00781                 return res;
00782             total += res;
00783         }
00784         break;
00785     }
00786     case Performance_describe_levels:
00787     {
00788         struct
00789         {
00790             l4_uint32_t domain_id = 0;
00791             l4_uint32_t level_idx = 0;
00792         } param;
00793         if ((res = qw->read(&buf, &param)) < 0)
00794             return res;
00795
00796         if ((res = qw->write(&buf, &hdr)) < 0)
00797             return res;
00798         total += res;
00799
00800         // First figure out how many levels we support
00801         Performance_describe_levels_n_t attr;
00802         l4_int32_t status = fill_describe_levels_n(param.domain_id, param.level_idx,
00803                                                    &attr);
00804         if (status != Success)
00805         {
00806             // On error bail out early
00807             if ((res = qw->write(&buf, &status)) < 0)
00808                 return res;
00809             total += res;
00810             break;
00811         }
00812
00813         // Now fetch the actual levels
00814         Performance_describe_level_t attr1[attr.nperf_levels().get()];
00815         status = fill_describe_levels(param.domain_id, param.level_idx,
00816                                       attr.nperf_levels(), attr1);
00817         if ((res = qw->write(&buf, &status)) < 0)
00818             return res;
00819         total += res;
00820         if (status == Success)
00821         {

```

```

00822         // Write both answers to the client
00823         if ((res = qw->write(&buf, &attr)) < 0)
00824             return res;
00825         total += res;
00826         if ((res = qw->write(&buf, attr1, sizeof(attr1))) < 0)
00827             return res;
00828         total += res;
00829     }
00830     break;
00831 }
00832 case Performance_level_set:
00833 {
00834     struct
00835     {
00836         l4_uint32_t domain_id;
00837         l4_uint32_t perf_level;
00838     } param;
00839     if ((res = qw->read(&buf, &param)) < 0)
00840         return res;
00841
00842     if ((res = qw->write(&buf, &hdr)) < 0)
00843         return res;
00844     total += res;
00845
00846     l4_int32_t status = level_set(param.domain_id, param.perf_level);
00847     if ((res = qw->write(&buf, &status)) < 0)
00848         return res;
00849     total += res;
00850     break;
00851 }
00852 case Performance_level_get:
00853 {
00854     l4_uint32_t domain_id = 0;
00855     if ((res = qw->read(&buf, &domain_id)) < 0)
00856         return res;
00857
00858     if ((res = qw->write(&buf, &hdr)) < 0)
00859         return res;
00860     total += res;
00861
00862     l4_uint32_t perf_level;
00863     l4_int32_t status = level_get(domain_id, &perf_level);
00864     if ((res = qw->write(&buf, &status)) < 0)
00865         return res;
00866     total += res;
00867     if (status == Success)
00868     {
00869         if ((res = qw->write(&buf, &perf_level)) < 0)
00870             return res;
00871         total += res;
00872     }
00873     break;
00874 }
00875 default:
00876 {
00877     if ((res = qw->write(&buf, &hdr)) < 0)
00878         return res;
00879     total += res;
00880
00881     l4_int32_t status = Not_supported;
00882     if ((res = qw->write(&buf, &status)) < 0)
00883         return res;
00884     total += res;
00885     break;
00886 }
00887 }
00888 return total;
00889 }
00890 };
00891
00892 } /* Scmi */ } /* Svr */ } /* L4virtio */

```

## 16.234 virtio.h

```

00001 /* SPDX-License-Identifier: MIT */
00002 /*
00003  * Copyright (C) 2013-2022 Kernkonzept GmbH.
00004  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00005  *             Matthias Lange <matthias.lange@kernkonzept.com>
00006  *
00007  */
00008 #pragma once
00009

```

```

00028 #include <l4/sys/utcb.h>
00029 #include <l4/sys/ipc.h>
00030 #include <l4/sys/types.h>
00031
00033 enum L4_virtio_protocol
00034 {
00035     L4VIRTIO_PROTOCOL = 0,
00036 };
00037
00038 enum L4virtio_magic
00039 {
00040     L4VIRTIO_MAGIC = 0x74726976
00041 };
00042
00043 enum L4virtio_vendor
00044 {
00045     L4VIRTIO_VENDOR_KK = 0x44
00046 };
00047
00051 enum L4_virtio_opcodes
00052 {
00053     L4VIRTIO_OP_SET_STATUS      = 0,
00054     L4VIRTIO_OP_CONFIG_QUEUE   = 1,
00055     L4VIRTIO_OP_REGISTER_DS    = 3,
00056     L4VIRTIO_OP_DEVICE_CONFIG  = 4,
00057     L4VIRTIO_OP_GET_DEVICE_IRQ = 5,
00058 };
00059
00061 enum L4virtio_device_ids
00062 {
00063     L4VIRTIO_ID_NET            = 1,
00064     L4VIRTIO_ID_BLOCK          = 2,
00065     L4VIRTIO_ID_CONSOLE        = 3,
00066     L4VIRTIO_ID_RNG            = 4,
00067     L4VIRTIO_ID_BALLOON        = 5,
00068     L4VIRTIO_ID_RPMMSG         = 7,
00069     L4VIRTIO_ID_SCSI           = 8,
00070     L4VIRTIO_ID_9P             = 9,
00071     L4VIRTIO_ID_RPROC_SERIAL   = 11,
00072     L4VIRTIO_ID_CAIF           = 12,
00073     L4VIRTIO_ID_GPU            = 16,
00074     L4VIRTIO_ID_INPUT          = 18,
00075     L4VIRTIO_ID_VSOCK          = 19,
00076     L4VIRTIO_ID_CRYPT          = 20,
00077     L4VIRTIO_ID_FS             = 26,
00078     L4VIRTIO_ID_SCSI           = 32,
00080     L4VIRTIO_ID_SOCK           = 0x9999,
00081 };
00082
00084 enum L4virtio_device_status
00085 {
00086     L4VIRTIO_STATUS_ACKNOWLEDGE = 1,
00087     L4VIRTIO_STATUS_DRIVER      = 2,
00088     L4VIRTIO_STATUS_DRIVER_OK   = 4,
00089     L4VIRTIO_STATUS_FEATURES_OK = 8,
00090     L4VIRTIO_STATUS_DEVICE_NEEDS_RESET = 0x40,
00091     L4VIRTIO_STATUS_FAILED      = 0x80
00092 };
00093
00095 enum L4virtio_feature_bits
00096 {
00098     L4VIRTIO_FEATURE_VERSION_1 = 32,
00100     L4VIRTIO_FEATURE_CMD_CONFIG = 160
00101 };
00102
00107 enum L4_virtio_irq_status
00108 {
00109     L4VIRTIO_IRQ_STATUS_VRING = 1,
00110     L4VIRTIO_IRQ_STATUS_CONFIG = 2,
00111 };
00112
00116 enum L4_virtio_cmd
00117 {
00118     L4VIRTIO_CMD_NONE          = 0x00000000,
00119     L4VIRTIO_CMD_SET_STATUS    = 0x01000000,
00120     L4VIRTIO_CMD_CFG_QUEUE     = 0x02000000,
00121     L4VIRTIO_CMD_CFG_CHANGED   = 0x04000000,
00122     L4VIRTIO_CMD_NOTIFY_QUEUE  = 0x08000000,
00123     L4VIRTIO_CMD_MASK          = 0xff000000,
00124 };
00125
00129 typedef struct l4virtio_config_hdr_t
00130 {
00131     /* Virtio(0x00): device config */
00132     l4_uint32_t magic;
00133     l4_uint32_t version;
00134     l4_uint32_t device;

```

```

00135     l4_uint32_t vendor;
00137     /* Virtio(0x10): device features */
00138     l4_uint32_t dev_features;
00139     l4_uint32_t dev_features_sel;
00140     l4_uint32_t _res1[2];
00141
00142     /* Virtio(0x20): driver features */
00143     l4_uint32_t driver_features;
00144     l4_uint32_t driver_features_sel;
00145
00146     /* L4Virtio(0x28): L4 queue */
00147     l4_uint32_t num_queues;
00148     l4_uint32_t queues_offset;
00150     /* Virtio(0x30): queue status */
00151     l4_uint32_t queue_sel;
00152     l4_uint32_t queue_num_max;
00153     l4_uint32_t queue_num;
00154     l4_uint32_t _res3[2];
00155     l4_uint32_t queue_ready;
00156     l4_uint32_t _res4[2];
00157
00158     /* Virtio(0x50): queue notify */
00159     l4_uint32_t queue_notify;
00160     l4_uint32_t _res5[3];
00161
00162     /* Virtio(0x60): interrupt handling */
00163     l4_uint32_t irq_status;
00164     l4_uint32_t irq_ack;
00165     l4_uint32_t _res6[2];
00166
00167     /* Virtio(0x70): Device status register (read-only). The register must be
00168      * written using l4virtio_set_status(). */
00169     l4_uint32_t status;
00170
00171     /* L4Virtio(0x74): W: Event index to be used for config notifications (device to driver) */
00172     l4_uint32_t cfg_driver_notify_index;
00173     /* L4Virtio(0x78): R: Event index to be used for config notifications (driver to device) */
00174     l4_uint32_t cfg_device_notify_index;
00175
00176     /* L4Virtio(0x7c): L4 specific command register polled by the driver iff supported */
00177     l4_uint32_t cmd;
00178
00179     /* Virtio(0x80): queue descriptors */
00180     l4_uint64_t queue_desc;
00181     l4_uint32_t _res8[2];
00182     l4_uint64_t queue_avail;
00183     l4_uint32_t _res9[2];
00184     l4_uint64_t queue_used;
00185
00186     l4_uint32_t _res10[1];
00187
00188     /* Virtio(0xac): shared memory region */
00189     l4_uint32_t shm_sel;
00190     l4_uint64_t shm_len;
00191     l4_uint64_t shm_base;
00192
00193     /* L4Virtio(0xc0): use the unused space here for device and driver feature bitmaps */
00194     l4_uint32_t dev_features_map[6];
00195     l4_uint32_t _res11[2];
00196     l4_uint32_t driver_features_map[6];
00197     l4_uint32_t _res12[1];
00198
00199     /* Virtio(0xfc): config generation */
00200     l4_uint32_t generation;
00201 } l4virtio_config_hdr_t;
00202
00220 typedef struct l4virtio_config_queue_t
00221 {
00223     l4_uint16_t num_max;
00225     l4_uint16_t num;
00226
00228     l4_uint16_t ready;
00229
00231     l4_uint16_t driver_notify_index;
00232
00233     l4_uint64_t desc_addr;
00234     l4_uint64_t avail_addr;
00235     l4_uint64_t used_addr;
00238     l4_uint16_t device_notify_index;
00239 } l4virtio_config_queue_t;
00240
00241 EXTERN_C_BEGIN
00242
00248 L4_INLINE l4virtio_config_queue_t *
00249 l4virtio_config_queues(l4virtio_config_hdr_t const *cfg)
00250 {
00251     return (l4virtio_config_queue_t *)(((l4_addr_t)cfg) + cfg->queues_offset);

```

```

00252 }
00253
00259 L4_INLINE void *
00260 l4virtio_device_config(l4virtio_config_hdr_t const *cfg)
00261 {
00262     return (void *)(((l4_addr_t)cfg) + 0x100);
00263 }
00264
00268 L4_INLINE void
00269 l4virtio_set_feature(l4_uint32_t *feature_map, unsigned feat)
00270 {
00271     unsigned idx = feat / 32;
00272
00273     if (idx < 8)
00274         feature_map[idx] |= 1UL << (feat % 32);
00275 }
00276
00280 L4_INLINE void
00281 l4virtio_clear_feature(l4_uint32_t *feature_map, unsigned feat)
00282 {
00283     unsigned idx = feat / 32;
00284
00285     if (idx < 8)
00286         feature_map[idx] &= ~(1UL << (feat % 32));
00287 }
00288
00292 L4_INLINE unsigned
00293 l4virtio_get_feature(l4_uint32_t *feature_map, unsigned feat)
00294 {
00295     unsigned idx = feat / 32;
00296
00297     if (idx >= 8)
00298         return 0;
00299
00300     return feature_map[idx] & (1UL << (feat % 32));
00301 }
00302
00308 L4_CV int
00309 l4virtio_set_status(l4_cap_idx_t cap, unsigned status) L4_NOTHROW;
00310
00316 L4_CV int
00317 l4virtio_config_queue(l4_cap_idx_t cap, unsigned queue) L4_NOTHROW;
00318
00324 L4_CV int
00325 l4virtio_register_ds(l4_cap_idx_t cap, l4_cap_idx_t ds_cap,
00326                     l4_uint64_t base, l4_umword_t offset,
00327                     l4_umword_t size) L4_NOTHROW;
00328
00334 L4_CV int
00335 l4virtio_device_config_ds(l4_cap_idx_t cap, l4_cap_idx_t config_ds,
00336                          l4_addr_t *ds_offset) L4_NOTHROW;
00337
00343 L4_CV int
00344 l4virtio_device_notification_irq(l4_cap_idx_t cap, unsigned index,
00345                                  l4_cap_idx_t irq) L4_NOTHROW;
00346
00347 EXTERN_C_END
00348

```

## 16.235 virtio\_block.h

```

00001 /* SPDX-License-Identifier: MIT */
00002 /*
00003  * (c) 2014 Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  */
00005
00006 #pragma once
00007
00014 #include <l4/sys/types.h>
00015
00019 enum L4virtio_block_operations
00020 {
00021     L4VIRTIO_BLOCK_T_IN      = 0,
00022     L4VIRTIO_BLOCK_T_OUT     = 1,
00023     L4VIRTIO_BLOCK_T_FLUSH   = 4,
00024     L4VIRTIO_BLOCK_T_GET_ID   = 8,
00025     L4VIRTIO_BLOCK_T_DISCARD = 11,
00026     L4VIRTIO_BLOCK_T_WRITE_ZEROES = 13,
00027 };
00028
00032 enum L4virtio_block_status
00033 {
00034     L4VIRTIO_BLOCK_S_OK      = 0,

```

```

00035     L4VIRTIO_BLOCK_S_IOERR    = 1,
00036     L4VIRTIO_BLOCK_S_UNSUPP    = 2
00037 };
00038
00042 typedef struct l4virtio_block_header_t
00043 {
00044     l4_uint32_t type;
00045     l4_uint32_t ioprio;
00046     l4_uint64_t sector;
00047 } l4virtio_block_header_t;
00048
00049 enum L4virtio_block_discard_flags_t
00050 {
00051     L4VIRTIO_BLOCK_DISCARD_F_UNMAP    = 0x00000001UL,
00052     L4VIRTIO_BLOCK_DISCARD_F_RESERVED = 0xFFFFFFF0UL,
00053 };
00054
00058 typedef struct l4virtio_block_discard_t
00059 {
00060     l4_uint64_t sector;
00061     l4_uint32_t num_sectors;
00062     l4_uint32_t flags;
00063 } l4virtio_block_discard_t;
00064
00068 typedef struct l4virtio_block_config_t
00069 {
00070     l4_uint64_t capacity;
00071     l4_uint32_t size_max;
00072     l4_uint32_t seg_max;
00073     struct l4virtio_block_config_geometry_t
00074     {
00075         l4_uint16_t cylinders;
00076         l4_uint8_t heads;
00077         l4_uint8_t sectors;
00078     } geometry;
00079     l4_uint32_t blk_size;
00080     struct l4virtio_block_config_topology_t
00081     {
00082         l4_uint8_t physical_block_exp;
00083         l4_uint8_t alignment_offset;
00084         l4_uint16_t min_io_size;
00085         l4_uint32_t opt_io_size;
00086     } topology;
00087     l4_uint8_t writeback;
00088     l4_uint8_t unused0[3];
00089     l4_uint32_t max_discard_sectors;
00090     l4_uint32_t max_discard_seg;
00091     l4_uint32_t discard_sector_alignment;
00092     l4_uint32_t max_write_zeroes_sectors;
00093     l4_uint32_t max_write_zeroes_seg;
00094     l4_uint8_t write_zeroes_may_unmap;
00095     l4_uint8_t unused1[3];
00096 } l4virtio_block_config_t;
00097
00101

```

## 16.236 virtio\_input.h

```

00001 /* SPDX-License-Identifier: MIT */
00002 /*
00003  * Copyright (C) 2019, 2022 Kernkonzept GmbH.
00004  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00005  */
00006 #pragma once
00007
00014 #include <l4/sys/types.h>
00015
00019 enum L4virtio_input_config_select
00020 {
00021     L4VIRTIO_INPUT_CFG_UNSET = 0,
00022     L4VIRTIO_INPUT_CFG_ID_NAME = 1,
00023     L4VIRTIO_INPUT_CFG_ID_SERIAL = 2,
00024     L4VIRTIO_INPUT_CFG_ID_DEVIDS = 3,
00025     L4VIRTIO_INPUT_CFG_PROP_BITS = 0x10,
00026     L4VIRTIO_INPUT_CFG_EV_BITS = 0x11,
00027     L4VIRTIO_INPUT_CFG_ABS_INFO = 0x12
00028 };
00029
00033 typedef struct l4virtio_input_absinfo_t
00034 {
00035     l4_uint32_t min;
00036     l4_uint32_t max;
00037     l4_uint32_t fuzz;
00038     l4_uint32_t flat;

```

```

00039     14_uint32_t res;
00040 } 14virtio_absinfo_t;
00041
00045 typedef struct 14virtio_input_devids_t
00046 {
00047     14_uint16_t bustype;
00048     14_uint16_t vendor;
00049     14_uint16_t product;
00050     14_uint16_t version;
00051 } 14virtio_input_devids_t;
00052
00056 typedef struct 14virtio_input_config_t
00057 {
00058     14_uint8_t select;
00059     14_uint8_t subsl;
00060     14_uint8_t size;
00061     14_uint8_t reserved[5];
00062     union
00063     {
00064         char string[128];
00065         14_uint8_t bitmap[128];
00066         struct 14virtio_input_absinfo_t abs;
00067         struct 14virtio_input_devids_t ids;
00068     } u;
00069 } 14virtio_input_config_t;
00070
00074 typedef struct 14virtio_input_event_t
00075 {
00076     14_uint16_t type;
00077     14_uint16_t code;
00078     14_uint32_t value;
00079 } 14virtio_input_event_t;
00080

```

## 16.237 virtio\_net.h

```

00001 /* SPDX-License-Identifier: MIT */
00002 /*
00003  * Copyright (C) 2022 Kernkonzept GmbH.
00004  * Author(s): Stephan Gerhold <stephan.gerhold@kernkonzept.com>
00005  */
00006
00007 #pragma once
00008
00015 #include <14/sys/types.h>
00016
00020 typedef struct 14virtio_net_header_t
00021 {
00022     14_uint8_t flags;
00023     14_uint8_t gso_type;
00024     14_uint16_t hdr_len;
00025     14_uint16_t gso_size;
00026     14_uint16_t csum_start;
00027     14_uint16_t csum_offset;
00028     14_uint16_t num_buffers;
00029 } 14virtio_net_header_t;
00030
00034 typedef struct 14virtio_net_config_t
00035 {
00036     14_uint8_t mac[6];
00037     14_uint16_t status;
00038     14_uint16_t max_virtqueue_pairs;
00039     14_uint16_t mtu;
00040     14_uint32_t speed;
00041     14_uint8_t duplex;
00042 } 14virtio_net_config_t;
00043
00045 enum L4virtio_net_feature_bits
00046 {
00047     L4VIRTIO_NET_F_CSUM = 0,
00048     L4VIRTIO_NET_F_GUEST_CSUM = 1,
00049     L4VIRTIO_NET_F_MTU = 3,
00050     L4VIRTIO_NET_F_MAC = 5,
00051     L4VIRTIO_NET_F_GUEST_TSO4 = 7,
00052     L4VIRTIO_NET_F_GUEST_TSO6 = 8,
00053     L4VIRTIO_NET_F_GUEST_ECN = 9,
00054     L4VIRTIO_NET_F_GUEST_UFO = 10,
00055     L4VIRTIO_NET_F_HOST_TSO4 = 11,
00056     L4VIRTIO_NET_F_HOST_TSO6 = 12,
00057     L4VIRTIO_NET_F_HOST_ECN = 13,
00058     L4VIRTIO_NET_F_HOST_UFO = 14,
00059     L4VIRTIO_NET_F_MRG_RXBUF = 15,
00060     L4VIRTIO_NET_F_STATUS = 16,

```

```

00061     L4VIRTIO_NET_F_CTRL_VQ = 17,
00062     L4VIRTIO_NET_F_CTRL_RX = 18,
00063     L4VIRTIO_NET_F_CTRL_VLAN = 19,
00064     L4VIRTIO_NET_F_GUEST_ANNOUNCE = 21,
00065     L4VIRTIO_NET_F_MQ = 22,
00066     L4VIRTIO_NET_F_CTRL_MAC_ADDR = 23,
00067 };
00068

```

## 16.238 virtqueue

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * (c) 2014 Alexander Warg <warg@os.inf.tu-dresden.de>
00005  */
00006
00007 #include <l4/re/util/debug>
00008 #include <l4/sys/types.h>
00009 #include <l4/sys/err.h>
00010 #include <l4/cxx/bitfield>
00011 #include <l4/cxx/exceptions>
00012 #include <cstdint>
00013
00014 #pragma once
00015
00016 namespace L4virtio {
00017
00018 // __ARM_ARCH_7A__ not defined by Clang
00019 #if defined(__ARM_ARCH_7A__) \
00020     || ( defined(__ARM_ARCH) && __ARM_ARCH == 7 \
00021         && defined(__ARM_ARCH_PROFILE) && __ARM_ARCH_PROFILE >= 'A')
00022 static inline void wmb() { asm volatile ("dmb" : : : "memory"); }
00023 static inline void rmb() { asm volatile ("dmb" : : : "memory"); }
00024 // __ARM_ARCH_8A__ not defined by Clang
00025 #elif defined(__ARM_ARCH_8A__) \
00026     || ( defined(__ARM_ARCH) && __ARM_ARCH == 8 \
00027         && defined(__ARM_ARCH_PROFILE) && __ARM_ARCH_PROFILE >= 'A') \
00028     || (defined(__ARM_ARCH) && __ARM_ARCH > 8)
00029 static inline void wmb() { asm volatile ("dsb ishst" : : : "memory"); }
00030 static inline void rmb() { asm volatile ("dsb ishld" : : : "memory"); }
00031 #elif defined(__mips__)
00032 static inline void wmb() { asm volatile ("sync" : : : "memory"); }
00033 static inline void rmb() { asm volatile ("sync" : : : "memory"); }
00034 #elif defined(__amd64__) || defined(__i386__) || defined(__i686__)
00035 static inline void wmb() { asm volatile ("sfence" : : : "memory"); }
00036 static inline void rmb() { asm volatile ("lfence" : : : "memory"); }
00037 #elif defined(__riscv)
00038 static inline void wmb() { asm volatile ("fence ow, ow" : : : "memory"); }
00039 static inline void rmb() { asm volatile ("fence ir, ir" : : : "memory"); }
00040 #else
00041 #warning Missing proper memory write barrier
00042 static inline void wmb() { asm volatile ("": : : : "memory"); }
00043 static inline void rmb() { asm volatile ("": : : : "memory"); }
00044 #endif
00045
00046
00053 template< typename T >
00054 class Ptr
00055 {
00056 public:
00057     enum Invalid_type { Invalid };
00058
00059     Ptr() = default;
00060
00061     Ptr(Invalid_type) : _p(~0ULL) {}
00062
00063     explicit Ptr(l4_uint64_t vm_addr) : _p(vm_addr) {}
00064
00065     l4_uint64_t get() const { return _p; }
00066
00067     bool is_valid() const { return _p != ~0ULL; }
00068
00069 private:
00070     l4_uint64_t _p;
00071 };
00072
00073
00074 class Virtqueue
00075 {
00076 public:
00077     class Desc
00078     {
00079

```



```

00095 public:
00099 struct Flags
00100 {
00101     l4_uint16_t raw;
00102     Flags() = default;
00103
00105     explicit Flags(l4_uint16_t v) : raw(v) {}
00106
00108     CXX_BITFIELD_MEMBER( 0, 0, next, raw);
00110     CXX_BITFIELD_MEMBER( 1, 1, write, raw);
00112     CXX_BITFIELD_MEMBER( 2, 2, indirect, raw);
00113 };
00114
00115 Ptr<void> addr;
00116 l4_uint32_t len;
00117 Flags flags;
00118 l4_uint16_t next;
00119
00123 void dump(unsigned idx) const
00124 {
00125     L4Re::Util::Dbg().printf("D[%04x]: %08llx (%x) f=%04x n=%04x\n",
00126                               idx, addr.get(),
00127                               len, static_cast<unsigned>(flags.raw),
00128                               static_cast<unsigned>(next));
00129 }
00130 };
00131
00135 class Avail
00136 {
00137 public:
00141 struct Flags
00142 {
00143     l4_uint16_t raw;
00144     Flags() = default;
00145
00147     explicit Flags(l4_uint16_t v) : raw(v) {}
00148
00150     CXX_BITFIELD_MEMBER( 0, 0, no_irq, raw);
00151 };
00152
00153 Flags flags;
00154 l4_uint16_t idx;
00155 l4_uint16_t ring[];
00156 };
00157
00161 struct Used_elem
00162 {
00163     Used_elem() = default;
00164
00172     Used_elem(l4_uint16_t id, l4_uint32_t len) : id(id), len(len) {}
00173     l4_uint32_t id;
00174     l4_uint32_t len;
00175 };
00176
00180 class Used
00181 {
00182 public:
00186 struct Flags
00187 {
00188     l4_uint16_t raw;
00189     Flags() = default;
00190
00192     explicit Flags(l4_uint16_t v) : raw(v) {}
00193
00195     CXX_BITFIELD_MEMBER( 0, 0, no_notify, raw);
00196 };
00197
00198 Flags flags;
00199 l4_uint16_t idx;
00200 Used_elem ring[];
00201 };
00202
00203 protected:
00204 Desc *_desc = nullptr;
00205 Avail *_avail = nullptr;
00206 Used *_used = nullptr;
00207
00209 l4_uint16_t _current_avail = 0;
00210
00215 l4_uint16_t _idx_mask = 0;
00216
00220 Virtqueue() = default;
00221
00222 Virtqueue(Virtqueue const &) = delete;
00223
00224 public:
00230 void disable()

```

```

00231 { _desc = 0; }
00232
00236 enum
00237 {
00238     Desc_align = 4, ///< Alignment of the descriptor table.
00239     Avail_align = 1, ///< Alignment of the available ring.
00240     Used_align = 2, ///< Alignment of the used ring.
00241 };
00242
00251 static unsigned long total_size(unsigned num)
00252 {
00253     static_assert(Desc_align >= Avail_align,
00254         "virtqueue alignment assumptions broken");
00255     return l4_round_size(desc_size(num) + avail_size(num), Used_align)
00256         + used_size(num);
00257 }
00258
00267 static unsigned long desc_size(unsigned num)
00268 { return num * 16; }
00269
00275 static unsigned long desc_align()
00276 { return Desc_align; }
00277
00285 static unsigned long avail_size(unsigned num)
00286 { return 2 * num + 6; }
00287
00293 static unsigned long avail_align()
00294 { return Avail_align; }
00295
00304 static unsigned long used_size(unsigned num)
00305 { return 8 * num + 6; }
00306
00312 static unsigned long used_align()
00313 { return Used_align; }
00314
00320 unsigned long total_size() const
00321 {
00322     return (reinterpret_cast<char*>(_used) - reinterpret_cast<char*>(_desc))
00323         + used_size(num());
00324 }
00325
00329 unsigned long avail_offset() const
00330 { return reinterpret_cast<char*>(_avail) - reinterpret_cast<char*>(_desc); }
00331
00335 unsigned long used_offset() const
00336 { return reinterpret_cast<char*>(_used) - reinterpret_cast<char*>(_desc); }
00337
00355 void setup(unsigned num, void *desc, void *avail, void *used)
00356 {
00357     if (num > 0x10000)
00358         throw L4::Runtime_error(-L4_EINVAL, "Queue too large.");
00359
00360     _idx_mask = num - 1;
00361     _desc = static_cast<Desc*>(desc);
00362     _avail = static_cast<Avail*>(avail);
00363     _used = static_cast<Used*>(used);
00364
00365     _current_avail = 0;
00366
00367     L4Re::Util::Dbg().printf("VQ[%p]: num=%d d:%p a:%p u:%p\n",
00368         this, num, _desc, _avail, _used);
00369 }
00370
00384 void setup_simple(unsigned num, void *ring)
00385 {
00386     l4_addr_t desc = reinterpret_cast<l4_addr_t>(ring);
00387     l4_addr_t avail = l4_round_size(desc + desc_size(num), Avail_align);
00388     void *used = reinterpret_cast<void*>(
00389         l4_round_size(avail + avail_size(num), Used_align));
00390     setup(num, ring, reinterpret_cast<void*>(avail), used);
00391 }
00392
00398 void dump(Desc const *d) const
00399 { d->dump(d - _desc); }
00400
00406 bool ready() const
00407 { return L4_LIKELY(_desc != 0); }
00408
00410 unsigned num() const
00411 { return _idx_mask + 1; }
00412
00420 bool no_notify_guest() const
00421 {
00422     return _avail->flags.no_irq();
00423 }
00424
00432 bool no_notify_host() const

```

```

00433     {
00434         return _used->flags.no_notify();
00435     }
00436
00442 void no_notify_host(bool value)
00443 {
00444     _used->flags.no_notify() = value;
00445 }
00446
00455 l4_uint16_t get_avail_idx() const { return _avail->idx; }
00456
00462 l4_uint16_t get_tail_avail_idx() const { return _current_avail; }
00463 };
00464 };
00465
00466 namespace Driver {
00467
00476 class Virtqueue : public L4virtio::Virtqueue
00477 {
00478 private:
00480     l4_uint16_t _next_free;
00481
00482 public:
00483     enum End_of_queue
00484     {
00485         // Indicates the end of the queue.
00486         Eoq = 0xFFFF
00487     };
00488
00489     Virtqueue() : _next_free(Eoq) {}
00490
00500 void initialize_rings(unsigned num)
00501 {
00502     _used->idx = 0;
00503     _avail->idx = 0;
00504
00505     // setup the freelist
00506     for (l4_uint16_t d = 0; d < num - 1; ++d)
00507         _desc[d].next = d + 1;
00508     _desc[num - 1].next = Eoq;
00509     _next_free = 0;
00510 }
00511
00528 void init_queue(unsigned num, void *desc, void *avail, void *used)
00529 {
00530     setup(num, desc, avail, used);
00531     initialize_rings(num);
00532 }
00533
00543 void init_queue(unsigned num, void *base)
00544 {
00545     setup_simple(num, base);
00546     initialize_rings(num);
00547 }
00548
00549
00564 l4_uint16_t alloc_descriptor()
00565 {
00566     l4_uint16_t idx = _next_free;
00567     if (idx == Eoq)
00568         return Eoq;
00569
00570     _next_free = _desc[idx].next;
00571
00572     return idx;
00573 }
00574
00580 void enqueue_descriptor(l4_uint16_t descno)
00581 {
00582     if (descno > _idx_mask)
00583         throw L4::Bounds_error();
00584
00585     _avail->ring[_avail->idx & _idx_mask] = descno; // _avail->idx expected to wrap
00586     wmb();
00587     ++_avail->idx;
00588 }
00589
00596 Desc &desc(l4_uint16_t descno)
00597 {
00598     if (descno > _idx_mask)
00599         throw L4::Bounds_error();
00600
00601     return _desc[descno];
00602 }
00603
00615 l4_uint16_t find_next_used(l4_uint32_t *len = nullptr)
00616 {

```

```

00617     if (_current_avail == _used->idx)
00618         return Eoq;
00619
00620     auto elem = _used->ring[_current_avail++ & _idx_mask];
00621
00622     if (len)
00623         *len = elem.len;
00624
00625     return elem.id;
00626 }
00627
00637 void free_descriptor(l4_uint16_t head, l4_uint16_t tail)
00638 {
00639     if (head > _idx_mask || tail > _idx_mask)
00640         throw L4::Bounds_error();
00641
00642     _desc[tail].next = _next_free;
00643     _next_free = head;
00644 }
00645 };
00646
00647 }
00648 } // namespace L4virtio

```

## 16.239 block\_device\_mgr.h

```

00001 /*
00002  * Copyright (C) 2018-2020, 2022 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *             Manuel von Oltersdorff-Kalettkka <manuel.kalettkka@kernkonzept.com>
00005  *
00006  * This file is distributed under the terms of the GNU General Public
00007  * License, version 2. Please see the COPYING-GPL-2 file for details.
00008  */
00009 #pragma once
00010
00011 #include <cassert>
00012 #include <cstring>
00013 #include <memory>
00014 #include <string>
00015 #include <vector>
00016
00017 #include <l4/cxx/ref_ptr>
00018 #include <l4/cxx/ref_ptr_list>
00019 #include <l4/cxx/unique_ptr>
00020 #include <l4/re/error_helper>
00021 #include <l4/sys/factory>
00022 #include <l4/sys/cxx/ipc_epiface>
00023
00024 #include <l4/libblock-device/debug.h>
00025 #include <l4/libblock-device/errand.h>
00026 #include <l4/libblock-device/partition.h>
00027 #include <l4/libblock-device/part_device.h>
00028 #include <l4/libblock-device/virtio_client.h>
00029 #include <l4/libblock-device/scheduler.h>
00030
00031 namespace Block_device {
00032
00033 template <typename DEV>
00034 struct Simple_factory
00035 {
00036     using Device_type = DEV;
00037     using Client_type = Virtio_client<Device_type>;
00038
00039     static cxx::unique_ptr<Client_type>
00040     create_client(cxx::Ref_ptr<Device_type> const &dev,
00041                 unsigned numds, bool readonly)
00042     { return cxx::make_unique<Client_type>(dev, numds, readonly); }
00043 };
00044
00045 template <typename BASE_DEV>
00046 struct Partitionable_factory
00047 {
00048     using Device_type = BASE_DEV;
00049     using Client_type = Virtio_client<Device_type>;
00050
00051     static cxx::unique_ptr<Client_type>
00052     create_client(cxx::Ref_ptr<Device_type> const &dev,
00053                 unsigned numds, bool readonly)
00054     {
00055         return cxx::make_unique<Client_type>(dev, numds, readonly);
00056     }
00057 }

```

```

00058     static cxx::Ref_ptr<Device_type>
00059     create_partition(cxx::Ref_ptr<Device_type> const &dev, unsigned partition_id,
00060                     Partition_info const &pi)
00061     {
00062         return cxx::Ref_ptr<Device_type>{
00063             new Partitioned_device<Device_type>(dev, partition_id, pi)};
00064     }
00065 };
00066
00067
00078 template <typename DEV, typename FACTORY = Simple_factory<DEV>,
00079           typename SCHEDULER = Rr_scheduler<typename FACTORY::Device_type>
00080 class Device_mgr
00081 {
00082     using Device_factory_type = FACTORY;
00083     using Client_type = typename Device_factory_type::Client_type;
00084     using Device_type = typename Device_factory_type::Device_type;
00085     using Scheduler_type = SCHEDULER;
00086
00087     using Ds_vector = std::vector<L4::Cap<L4Re::Dataspace>;
00088
00089     using Pairing_callback = std::function<void(Device_type *)>;
00090
00094     struct Pending_client
00095     {
00097         std::string device_id;
00099         L4::Cap<L4::Rcv_endpoint> gate;
00101         int num_ds;
00103         bool readonly;
00104
00105         bool enable_trusted_ds_validation;
00106
00107         std::shared_ptr<Ds_vector const> trusted_dataspaces;
00108
00110         Pairing_callback pairing_cb;
00111
00112         Pending_client() = default;
00113
00114         Pending_client(L4::Cap<L4::Rcv_endpoint> g, std::string const &dev, int ds,
00115                       bool ro, bool enable_trusted_ds_validation,
00116                       std::shared_ptr<Ds_vector const> trusted_dataspaces,
00117                       Pairing_callback cb)
00118         : device_id(dev), gate(g), num_ds(ds), readonly(ro),
00119           enable_trusted_ds_validation(enable_trusted_ds_validation),
00120           trusted_dataspaces(trusted_dataspaces), pairing_cb(cb)
00121         {}
00122     };
00123
00124     class Connection : public cxx::Ref_obj_list_item<Connection>
00125     {
00126     public:
00127         explicit Connection(Device_mgr *mgr, cxx::Ref_ptr<Device_type> &dev)
00128         : _shutdown_state(Shutdown_type::Running),
00129           _device(cxx::move(dev)),
00130           _mgr(mgr)
00131         {}
00132
00133         L4::Cap<void> cap() const
00134         { return _interface ? _interface->obj_cap() : L4::Cap<void>(); }
00135
00136         void start_disk_scan(Errand::Callback const &callback)
00137         {
00138             _device->start_device_scan(
00139                 [=] ()
00140                 {
00141                     scan_disk_partitions(callback, 0);
00142                 });
00143         }
00144
00145         void unregister_interfaces(L4::Registry_iface *registry) const
00146         {
00147             if (_interface)
00148                 registry->unregister_obj(_interface.get());
00149
00150             for (auto *sub : _subs)
00151                 sub->unregister_interfaces(registry);
00152         }
00153
00154         int create_interface_for(Pending_client *c, L4::Registry_iface *registry)
00155         {
00156             if (_shutdown_state != Shutdown_type::Running)
00157                 return -L4_EIO;
00158
00159             if (_interface)
00160                 return contains_device(c->device_id) ? -L4_EBUSY : -L4_ENODEV;
00161
00162             // check for match in partitions

```

```

00163
00164     bool busy = false;
00165     for (auto *sub : _subs)
00166     {
00167         if (sub->_interface)
00168             busy = true;
00169
00170         int ret = sub->create_interface_for(c, registry);
00171
00172         if (ret != -L4_ENODEV) // includes L4_EOK
00173             return ret;
00174     }
00175
00176     if (!match_hid(c->device_id))
00177         return -L4_ENODEV;
00178
00179     if (busy)
00180         return -L4_EBUSY;
00181
00182     auto clt = Device_factory_type::create_client(_device, c->num_ds,
00183                                                  c->readonly);
00184
00185     clt->add_trusted_dataspaces(c->trusted_dataspaces);
00186     if (c->enable_trusted_ds_validation)
00187         clt->enable_trusted_ds_validation();
00188
00189     if (c->gate.is_valid())
00190     {
00191         if (!clt->register_obj(registry, c->gate).is_valid())
00192             return -L4_ENOMEM;
00193     }
00194     else
00195     {
00196         c->gate = L4::cap_reinterpret_cast<L4::Rcv_endpoint>(
00197             clt->register_obj(registry));
00198         if (!c->gate.is_valid())
00199             return -L4_ENOMEM;
00200     }
00201
00202     _mgr->_scheduler->add_client(clt.get());
00203     _interface.reset(clt.release());
00204
00205     // Let it be known that the client and the device paired
00206     if (c->pairing_cb)
00207         c->pairing_cb(_device.get());
00208     return L4_EOK;
00209 }
00210
00211 void check_clients(L4::Registry_iface *registry)
00212 {
00213     if (_interface)
00214     {
00215         if (_interface->obj_cap() && !_interface->obj_cap().validate().label())
00216             remove_client(registry);
00217
00218         return;
00219     }
00220
00221     // Sub-devices only need to be checked when the parent device was free.
00222     for (auto *sub : _subs)
00223         sub->check_clients(registry);
00224 }
00225
00226 void shutdown_event(Shutdown_type type)
00227 {
00228     // Set new shutdown state
00229     _shutdown_state = type;
00230     for (auto const &sub: _subs)
00231         sub->shutdown_event(type);
00232     if (_interface)
00233         _interface->shutdown_event(type);
00234 }
00235
00236 private:
00237 template <typename T = Device_factory_type>
00238 auto scan_disk_partitions(Errand::Callback const &callback, int)
00239     -> decltype((T::create_partition)(cxx::Ref_ptr<Device_type>(), 0, Partition_info(), void()))
00240 {
00241     auto reader = cxx::make_ref_obj<Partition_reader<Device_type>>(_device.get());
00242     // The reference to reader will be captured in the lambda passed to
00243     // reader's own read() method. At the same time, reader will store
00244     // the reference to the lambda.
00245     reader->read(
00246         [=] ()
00247         {
00248             l4_size_t sz = reader->table_size();
00249         }
00250     );
00251 }

```

```

00262         for (l4_size_t i = 1; i <= sz; ++i)
00263         {
00264             Partition_info info;
00265             if (reader->get_partition(i, &info) < 0)
00266                 continue;
00267
00268             auto conn = cxx::make_ref_obj<Connection>(
00269                 _mgr,
00270                 Device_factory_type::create_partition(_device, i, info));
00271             _subs.push_front(std::move(conn));
00272         }
00273
00274         callback();
00275
00276         // Prolong the life-span of reader until we are sure the reader is
00277         // not currently invoked (i.e. capture the last reference to it in
00278         // an independent timeout callback).
00279         Errand::schedule([reader]() {}, 0);
00280     });
00281 }
00282
00290 template <typename T = Device_factory_type>
00291 void scan_disk_partitions(Errand::Callback const &callback, long)
00292 { callback(); }
00293
00297 void remove_client(L4::Registry_iface *registry)
00298 {
00299     assert(_interface);
00300
00301     // This operation is idempotent.
00302     _interface->shutdown_event(Shutdown_type::Client_gone);
00303
00304     if (_interface->busy())
00305     {
00306         Dbg::trace().printf("Deferring dead client removal.\n");
00307
00308         // Cannot remove the client while it still has active I/O requests.
00309         // This means that the device did not abort its inflight requests in
00310         // its reset() callback. It is still desirable though to wait for
00311         // those requests to finish and defer the dead client removal until
00312         // later.
00313         Errand::schedule([this, registry]() { remove_client(registry); },
00314             10000);
00315         return;
00316     }
00317
00318     _interface->unregister_obj(registry);
00319     _mgr->_scheduler->remove_client(_interface.get());
00320     _interface.reset();
00321 }
00322
00323 bool contains_device(std::string const &name) const
00324 {
00325     if (match_hid(name))
00326         return true;
00327
00328     for (auto *sub : _subs)
00329         if (sub->contains_device(name))
00330             return true;
00331
00332     return false;
00333 }
00334
00335 bool match_hid(std::string const &name) const
00336 { return _device->match_hid(cxx::String(name.c_str(), name.length())); }
00337
00338 Shutdown_type _shutdown_state;
00342 cxx::Ref_ptr<Device_type> _device;
00344 cxx::unique_ptr<Client_type> _interface;
00346 cxx::Ref_ptr_list<Connection> _subs;
00347
00348 Device_mgr *_mgr;
00349 };
00350
00351 public:
00352 Device_mgr(L4::Registry_iface *registry)
00353 : _registry(registry)
00354 {
00355     _scheduler = cxx::make_unique<Scheduler_type>(registry);
00356 }
00357
00358 virtual ~Device_mgr()
00359 {
00360     for (auto *c : _connpts)
00361         c->unregister_interfaces(_registry);
00362 }

```

```

00363
00364 int add_static_client(L4::Cap<L4::Rcv_endpoint> client, const char *device,
00365                     int partno, int num_ds, bool readonly = false,
00366                     Pairing_callback cb = nullptr,
00367                     bool enable_trusted_ds_validation = false,
00368                     std::shared_ptr<Ds_vector const> trusted_dataspaces
00369                     = nullptr)
00370 {
00371     char _buf[30];
00372     const char *buf;
00373
00374     if (partno == 0)
00375     {
00376         Err().printf("Invalid partition number 0.\n");
00377         return -L4_ENODEV;
00378     }
00379
00380     if (partno != -1)
00381     {
00382         /* Could we avoid to make a string here and parsing this again
00383          * deeper in the stack? */
00384         snprintf(_buf, sizeof(_buf), "%s:%d", device, partno);
00385         buf = _buf;
00386     }
00387     else
00388         buf = device;
00389
00390     _pending_clients.emplace_back(client, buf, num_ds, readonly,
00391                                   enable_trusted_ds_validation,
00392                                   trusted_dataspaces, cb);
00393
00394     return L4_EOK;
00395 }
00396
00397 int create_dynamic_client(std::string const &device, int partno, int num_ds,
00398                          L4::Cap<void> *cap, bool readonly = false,
00399                          Pairing_callback cb = nullptr,
00400                          bool enable_trusted_ds_validation = false,
00401                          std::shared_ptr<Ds_vector const> trusted_dataspaces
00402                          = nullptr)
00403 {
00404     Pending_client clt;
00405
00406     // Maximum number of dataspaces that can be registered.
00407     clt.num_ds = num_ds;
00408
00409     clt.readonly = readonly;
00410
00411     clt.device_id = device;
00412
00413     clt.pairing_cb = cb;
00414
00415     clt.trusted_dataspaces = trusted_dataspaces;
00416
00417     clt.enable_trusted_ds_validation = enable_trusted_ds_validation;
00418
00419     if (partno > 0)
00420     {
00421         clt.device_id += ':';
00422         clt.device_id += std::to_string(partno);
00423     }
00424
00425     for (auto *c : _connpts)
00426     {
00427         int ret = c->create_interface_for(&clt, _registry);
00428
00429         if (ret == -L4_ENODEV)
00430             continue;
00431
00432         if (ret < 0)
00433             return ret;
00434
00435         // found the requested device
00436         *cap = clt.gate;
00437         return L4_EOK;
00438     }
00439
00440     return -L4_ENODEV;
00441 }
00442
00446 void check_clients()
00447 {
00448     for (auto *c : _connpts)
00449         c->check_clients(_registry);
00450 }
00451
00452 void add_disk(cxx::Ref_ptr<Device_type> &&device, Errand::Callback const &callback)

```



```

00453 {
00454     auto conn = cxx::make_ref_obj<Connection>(this, std::move(device));
00455
00456     conn->start_disk_scan(
00457         [=] ()
00458         {
00459             _connpts.push_front(conn);
00460             connect_static_clients(conn.get());
00461             callback();
00462         });
00463 }
00464
00466 void shutdown_event(Shutdown_type type)
00467 {
00468     l4_assert(type != Client_gone);
00469     l4_assert(type != Client_shutdown);
00470
00471     for (auto const &con : _connpts)
00472         con->shutdown_event(type);
00473 }
00474
00475 private:
00476 void connect_static_clients(Connection *con)
00477 {
00478     for (auto &c : _pending_clients)
00479     {
00480         Dbg::trace().printf("Checking existing client %s\n", c.device_id.c_str());
00481         if (!c.gate.is_valid())
00482             continue;
00483
00484         int ret = con->create_interface_for(&c, _registry);
00485
00486         if (ret == L4_EOK)
00487         {
00488             c.gate = L4::Cap<L4::Rcv_endpoint>();
00489             // There might be other clients waiting for other partitions.
00490             // Continue search.
00491             continue;
00492         }
00493
00494         if (ret != -L4_ENODEV)
00495             break;
00496     }
00497 }
00498
00499
00501 L4::Registry_iface *_registry;
00503 cxx::Ref_ptr_list<Connection> _connpts;
00505 std::vector<Pending_client> _pending_clients;
00507 cxx::unique_ptr<Scheduler_type> _scheduler;
00508 };
00509
00510 } // name space

```

## 16.240 device.h

```

00001 /*
00002  * Copyright (C) 2018-2020 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * This file is distributed under the terms of the GNU General Public
00006  * License, version 2. Please see the COPYING-GPL-2 file for details.
00007  */
00008 #pragma once
00009
00010 #include <l4/cxx/ref_ptr>
00011 #include <l4/cxx/string>
00012 #include <l4/re/dataspace>
00013 #include <l4/re/dma_space>
00014
00015 #include <l4/libblock-device/errand.h>
00016 #include <l4/libblock-device/types.h>
00017
00018 namespace Block_device {
00019
00033 struct Notification_domain
00034 {
00035 };
00036
00037 struct Device : public cxx::Ref_obj
00038 {
00039     virtual ~Device() = 0;
00040

```

```

00042     virtual Notification_domain const *notification_domain() const = 0;
00043
00044     virtual bool is_read_only() const = 0;
00045     virtual bool match_hid(cxx::String const &hid) const = 0;
00046     virtual l4_uint64_t capacity() const = 0;
00047     virtual l4_size_t sector_size() const = 0;
00048     virtual l4_size_t max_size() const = 0;
00049     virtual unsigned max_segments() const = 0;
00050
00051     virtual void reset() = 0;
00052
00053     virtual int dma_map(Block_device::Mem_region *region, l4_addr_t offset,
00054                        l4_size_t num_sectors, L4Re::Dma_space::Direction dir,
00055                        L4Re::Dma_space::Dma_addr *phys) = 0;
00056
00057     virtual int dma_unmap(L4Re::Dma_space::Dma_addr phys, l4_size_t num_sectors,
00058                          L4Re::Dma_space::Direction dir) = 0;
00059
00060     virtual int inout_data(l4_uint64_t sector,
00061                           Block_device::Inout_block const &blocks,
00062                           Block_device::Inout_callback const &cb,
00063                           L4Re::Dma_space::Direction dir) = 0;
00064
00065     virtual int flush(Block_device::Inout_callback const &cb) = 0;
00066
00067     virtual void start_device_scan(Block_device::Errand::Callback const &callback) = 0;
00068 };
00069
00070 inline Device::~Device() = default;
00071
00072 template <typename DEV>
00073 struct Device_with_notification_domain : DEV
00074 {
00075     Notification_domain dom;
00076     Notification_domain const *notification_domain() const override
00077     { return &dom; }
00078 };
00079
00080 struct Device_discard_feature
00081 {
00082     struct Discard_info
00083     {
00084         unsigned max_discard_sectors = 0;
00085         unsigned max_discard_seg = 0;
00086         unsigned discard_sector_alignment = 1;
00087         unsigned max_write_zeroes_sectors = 0;
00088         unsigned max_write_zeroes_seg = 0;
00089         bool write_zeroes_may_unmap = false;
00090     };
00091
00092     virtual Discard_info discard_info() const = 0;
00093
00094     virtual int discard(l4_uint64_t offset,
00095                       Block_device::Inout_block const &blocks,
00096                       Block_device::Inout_callback const &cb, bool discard) = 0;
00097
00098 protected:
00099     ~Device_discard_feature() = default;
00100 };
00101
00102 // name space

```

## 16.241 errand.h

```

00001 /*
00002  * Copyright (C) 2014, 2020 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * This file is distributed under the terms of the GNU General Public
00006  * License, version 2. Please see the COPYING-GPL-2 file for details.
00007  *
00008  * As a special exception, you may use this file as part of a free software
00009  * library without restriction. Specifically, if other files instantiate
00010  * templates or use macros or inline functions from this file, or you compile
00011  * this file and link it with other files to produce an executable, this
00012  * file does not by itself cause the resulting executable to be covered by
00013  * the GNU General Public License. This exception does not however
00014  * invalidate any other reasons why the executable file might be covered by
00015  * the GNU General Public License.
00016  */
00017 #pragma once
00018
00019 #include <l4/re/env.h>

```

```

00020 #include <l4/re/util/object_registry>
00021 #include <l4/re/util/br_manager>
00022 #include <l4/cxx/ipc_timeout_queue>
00023 #include <l4/cxx/ref_ptr>
00024 #include <l4/cxx/exceptions>
00025 #include <l4/libblock-device/debug.h>
00026
00027 #include <functional>
00028
00029 namespace Block_device { namespace Errand {
00030
00032 extern L4::Ipc_svr::Server_iface *_sif;
00033
00037 typedef std::function<void()> Callback;
00038
00042 class Poll_errand
00043 : public L4::Ipc_svr::Timeout_queue::Timeout,
00044   public cxx::Ref_obj
00045 {
00046 public:
00047   void expired() final
00048   {
00049     // Recapture the reference pointer from the timeout queue.
00050     cxx::Ref_ptr<Poll_errand> p(this, false);
00051
00052     try
00053     {
00054       if (_poll())
00055         _callback(true);
00056       else
00057         if (--_retries <= 0)
00058           _callback(false);
00059         else
00060           reschedule();
00061     }
00062     catch (L4::Runtime_error const &e)
00063     {
00064       Err().printf("Polling task failed: %s\n", e.str());
00065     }
00066   }
00067
00068   void reschedule()
00069   {
00070     // create a place holder reference pointer for the timeout queue
00071     cxx::Ref_ptr<Poll_errand> p(this);
00072
00073     _sif->add_timeout(p.release(), l4_kip_clock(l4re_kip()) + _interval);
00074   }
00075
00076   // Class can only be instantiated as a reference counting object.
00077   template< typename T, typename... Args >
00078   friend
00079   cxx::Ref_ptr<T> cxx::make_ref_obj(Args &&... args);
00080
00081 private:
00082   Poll_errand(int retries, int interval,
00083               std::function<bool()> const &poll_func,
00084               std::function<void(bool)> const &callback)
00085   : _retries(retries),
00086     _interval(interval),
00087     _poll(poll_func),
00088     _callback(callback)
00089   {}
00090
00091   int _retries;
00092   int _interval;
00093   std::function<bool()> _poll;
00094   std::function<void(bool)> _callback;
00095 };
00096
00097
00108 class Errand
00109 : public L4::Ipc_svr::Timeout_queue::Timeout,
00110   public cxx::Ref_obj
00111 {
00112 public:
00113   void expired() final
00114   {
00115     // Recapture the reference pointer from the timeout queue.
00116     cxx::Ref_ptr<Errand> p(this, false);
00117
00118     if (_callback)
00119     {
00120       try
00121       {
00122         _callback();
00123       }

```

```

00124         catch (L4::Runtime_error const &e)
00125         {
00126             Err().printf("Asynchronous task failed: %s\n", e.str());
00127         }
00128     }
00129 }
00130
00131 void reschedule(unsigned interval = 0)
00132 {
00133     // create a placeholder reference pointer for the timeout queue
00134     cxx::Ref_ptr<Errand> p(this);
00135
00136     _sif->add_timeout(p.release(), l4_kip_clock(l4re_kip()) + interval);
00137 }
00138
00139 // Class can only be instantiated as a reference counting object.
00140 template< typename T, typename... Args >
00141 friend
00142 cxx::Ref_ptr<T> cxx::make_ref_obj(Args &&... args);
00143
00144 private:
00145     Errand(Callback const &callback) : _callback(callback) {}
00146
00147     Callback _callback;
00148 };
00149
00150 struct Loop_hooks
00151 : L4::Ipc_svr::Timeout_queue_hooks<Loop_hooks, L4Re::Util::Br_manager>,
00152   L4::Ipc_svr::Ignore_errors
00153 {
00154     l4_kernel_clock_t now() { return l4_kip_clock(l4re_kip()); }
00155 };
00156
00157 using Errand_server = L4Re::Util::Registry_server<Loop_hooks>;
00158
00164 inline void set_server_iface(L4::Ipc_svr::Server_iface *sif) { _sif = sif; }
00165
00176 inline void schedule(Callback const &callback, int interval)
00177 {
00178     cxx::make_ref_obj<Errand>(callback)->reschedule(interval);
00179 }
00180
00201 inline void poll(int retries, int interval,
00202                 std::function<bool()> const &poll_func,
00203                 std::function<void(bool)> const &callback)
00204 {
00205     if (poll_func())
00206         callback(true);
00207     else
00208         cxx::make_ref_obj<Poll_errand>(retries, interval, poll_func,
00209                                       callback)->reschedule();
00210 }
00211
00212
00213 } // name space

```

## 16.242 gpt.h

```

00001 /*
00002  * Copyright (C) 2018 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * This file is distributed under the terms of the GNU General Public
00006  * License, version 2. Please see the COPYING-GPL-2 file for details.
00007  */
00008 #pragma once
00009
00010 #include <l4/sys/types.h>
00011
00012 namespace Block_device {
00013 namespace Gpt {
00014
00015     struct Header
00016     {
00017         char            signature[8];
00018         l4_uint32_t      version;
00019         l4_uint32_t      header_size;
00020         l4_uint32_t      crc;
00021         l4_uint32_t      _reserved;
00022         l4_uint64_t      current_lba;
00023         l4_uint64_t      backup_lba;
00024         l4_uint64_t      first_lba;
00025         l4_uint64_t      last_lba;

```

```

00026     char          disk_guid[16];
00027     l4_uint64_t    partition_array_lba;
00028     l4_uint32_t    partition_array_size;
00029     l4_uint32_t    entry_size;
00030     l4_uint32_t    crc_array;
00031 } __attribute__((packed));
00032
00033 struct Entry
00034 {
00035     unsigned char type_guid[16];
00036     unsigned char partition_guid[16];
00037     l4_uint64_t    first;
00038     l4_uint64_t    last;
00039     l4_uint64_t    flags;
00040     l4_uint16_t    name[36];
00041 };
00042
00043 } // namespace
00044
00045 namespace Pc_partition_table {
00046
00047 struct Part_table {
00048     l4_uint8_t    bootable;
00049     l4_uint8_t    first_sector_chs[3];
00050     l4_uint8_t    type;
00051     l4_uint8_t    last_sector_chs[3];
00052     l4_uint32_t    start_sector_lba;
00053     l4_uint32_t    num_sector_lba;
00054 } __attribute__((packed));
00055
00056 } // namespace
00057 } // namespace

```

## 16.243 inout\_memory.h

```

00001 /*
00002  * Copyright (C) 2014, 2019-2020 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * This file is distributed under the terms of the GNU General Public
00006  * License, version 2. Please see the COPYING-GPL-2 file for details.
00007  */
00008 #pragma once
00009
00010 #include <l4/re/error_helper>
00011 #include <l4/re/env>
00012 #include <l4/re/util/unique_cap>
00013 #include <l4/re/rm>
00014 #include <l4/re/dma_space>
00015 #include <l4/cxx/ref_ptr>
00016
00017 #include <l4/libblock-device/types.h>
00018
00019 namespace Block_device {
00020
00021 template <typename DEV>
00022 class Inout_memory : public cxx::Ref_obj
00023 {
00024 public:
00025     using Device_type = DEV;
00026
00027     Inout_memory() : _paddr(0) {}
00028     Inout_memory(l4_uint32_t num_sectors, Device_type *dev,
00029                 L4Re::Dma_space::Direction dir)
00030     : _device(dev), _paddr(0), _dir(dir), _num_sectors(num_sectors)
00031     {
00032         auto lcap = L4Re::chkcap(L4Re::Util::make_unique_cap<L4Re::Dataspace>()),
00033             "Allocate dataspace capability for IO memory.");
00034
00035         auto *e = L4Re::Env::env();
00036         long sz = num_sectors * _device->sector_size();
00037         L4Re::chksys(e->mem_alloc()->alloc(sz, lcap.get(),
00038                                           L4Re::Mem_alloc::Continuous
00039                                           | L4Re::Mem_alloc::Pinned),
00040                     "Allocate pinned memory.");
00041
00042         L4Re::chksys(e->rm()->attach(&_region, sz,
00043                                     L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00044                                     L4::Ipc::make_cap_rw(lcap.get()), 0,
00045                                     L4_PAGESHIFT),
00046                     "Attach IO memory.");
00047
00048         _mem_region =

```

```

00053         cxx::make_unique<Block_device::Mem_region>(0, sz, 0, cxx::move(lcap));
00054         L4Re::chksys(_device->dma_map(_mem_region.get(), 0, _num_sectors, dir,
00055                                     &_paddr),
00056                 "Lock memory region for DMA.");
00057     }
00058
00059     Inout_memory(Inout_memory const &) = delete;
00060     Inout_memory(Inout_memory &&) = delete;
00061
00062     Inout_memory &operator=(Inout_memory &&rhs)
00063     {
00064         if (this != &rhs)
00065         {
00066             _device = rhs._device;
00067             _mem_region = cxx::move(rhs._mem_region);
00068             _region = cxx::move(rhs._region);
00069             _paddr = rhs._paddr;
00070             _dir = rhs._dir;
00071             _num_sectors = rhs._num_sectors;
00072             rhs._paddr = 0;
00073         }
00074
00075         return *this;
00076     }
00077
00078     ~Inout_memory()
00079     {
00080         if (_paddr)
00081             unmap();
00082     }
00083
00084     void unmap()
00085     {
00086         L4Re::chksys(_device->dma_unmap(_paddr, _num_sectors, _dir));
00087         _paddr = 0;
00088     }
00089
00090     Inout_block inout_block() const
00091     {
00092         Inout_block blk;
00093
00094         blk.dma_addr = _paddr;
00095         blk.virt_addr = _region.get();
00096         blk.num_sectors = _num_sectors;
00097         blk.next.reset();
00098
00099         return blk;
00100     }
00101
00102     template <class T>
00103     T *get(unsigned offset) const
00104     { return reinterpret_cast<T *>(_region.get() + offset); }
00105
00106 private:
00107     Device_type *_device;
00108     cxx::unique_ptr<Block_device::Mem_region> _mem_region;
00109     L4Re::Rm::Unique_region<char *> _region;
00110     L4Re::Dma_space::Dma_addr _paddr;
00111     L4Re::Dma_space::Direction _dir;
00112     l4_uint32_t _num_sectors;
00113 };
00114
00115 // name space
00116 }

```

## 16.244 part\_device.h

```

00001 /*
00002  * Copyright (C) 2018-2022 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * This file is distributed under the terms of the GNU General Public
00006  * License, version 2. Please see the COPYING-GPL-2 file for details.
00007  */
00008 #pragma once
00009
00010 #include <l4/cxx/ref_ptr>
00011
00012 #include <l4/libblock-device/device.h>
00013 #include <l4/libblock-device/partition.h>
00014
00015 #include <string>
00016 #include <locale>

```

```

00017 #include <codecvt>
00018
00019 namespace Block_device {
00020
00021 namespace Impl {
00022
00027     template <typename PART_DEV, typename BASE_DEV,
00028               bool = std::is_base_of<Device_discard_feature, BASE_DEV>::value>
00029     class Partitioned_device_discard_mixin : public BASE_DEV {};
00030
00037     template <typename PART_DEV, typename BASE_DEV>
00038     class Partitioned_device_discard_mixin<PART_DEV, BASE_DEV, true>
00039     : public BASE_DEV
00040     {
00041     public:
00042         using Base = BASE_DEV;
00043         using Part_device = PART_DEV;
00044
00045         typename Base::Discard_info discard_info() const override
00046         {
00047             return dev()->parent()->discard_info();
00048         }
00049
00050         int discard(l4_uint64_t offset, Inout_block const &blocks,
00051                   Inout_callback const &cb, bool discard) override
00052         {
00053             auto sz = dev()->partition_size();
00054
00055             if (offset > sz)
00056                 return -L4_EINVAL;
00057
00058             Inout_block const *cur = &blocks;
00059             while (cur)
00060             {
00061                 if (cur->sector >= sz - offset)
00062                     return -L4_EINVAL;
00063                 if (cur->num_sectors > sz)
00064                     return -L4_EINVAL;
00065                 if (offset + cur->sector > sz - cur->num_sectors)
00066                     return -L4_EINVAL;
00067
00068                 cur = cur->next.get();
00069             }
00070
00071             auto start = offset + dev()->partition_start();
00072             Dbg::trace("partition")
00073             .printf("Starting sector on disk: 0x%llx\n", start);
00074             return dev()->parent()->discard(start, blocks, cb, discard);
00075         }
00076
00077     private:
00078         Part_device const *dev() const
00079         { return static_cast<Part_device const *>(this); }
00080     };
00081
00082 }
00083
00091 template <typename BASE_DEV = Device>
00092 class Partitioned_device
00093 : public Impl::Partitioned_device_discard_mixin<Partitioned_device<BASE_DEV>, BASE_DEV>
00094 {
00095 public:
00096     using Device_type = BASE_DEV;
00097
00098     Partitioned_device(cxx::Ref_ptr<Device_type> const &dev,
00099                       unsigned partition_id, Partition_info const &pi)
00100     : _name(pi.name),
00101       _parent(dev),
00102       _start(pi.first),
00103       _size(pi.last - pi.first + 1)
00104     {
00105         if (pi.last < pi.first)
00106             L4Re::chksys(-L4_EINVAL,
00107                         "Last sector of partition before first sector.");
00108
00109         if (partition_id > 999)
00110             L4Re::chksys(-L4_EINVAL,
00111                         "Partition ID must be smaller than 1000.");
00112
00113         snprintf(_partition_id, sizeof(_partition_id), "%d", partition_id);
00114
00115         static_assert(sizeof(_guid) == sizeof(pi.guid), "String size mismatch");
00116         memcpy(_guid, pi.guid, sizeof(_guid));
00117     }
00118
00119     Notification_domain const *notification_domain() const override
00120     { return _parent->notification_domain(); }

```

```

00121
00122 bool is_read_only() const override
00123 { return _parent->is_read_only(); }
00124
00125 bool match_hid(cxx::String const &hid) const override
00126 {
00127     if (hid == cxx::String(_guid, 36))
00128         return true;
00129
00130     std::u16string whid =
00131         std::wstring_convert<std::codecvt_utf8_utf16<char16_t>, char16_t>{}
00132             .from_bytes(std::string(hid.start(), hid.len()));
00133     if (whid == _name)
00134         return true;
00135
00136     // check for identifier of form: <device_name>:<partition id>
00137     char const *delim = ":";
00138     char const *pos = hid.rfind(delim);
00139
00140     if (pos == hid.end() || !_parent->match_hid(cxx::String(hid.start(), pos)))
00141         return false;
00142
00143     return cxx::String(pos + 1, hid.end()) == cxx::String(_partition_id);
00144 }
00145
00146 l4_uint64_t capacity() const override
00147 { return _size * _parent->sector_size(); }
00148
00149 l4_size_t sector_size() const override
00150 { return _parent->sector_size(); }
00151
00152 l4_size_t max_size() const override
00153 { return _parent->max_size(); }
00154
00155 unsigned max_segments() const override
00156 { return _parent->max_segments(); }
00157
00158 void reset() override
00159 {}
00160
00161 int dma_map(Block_device::Mem_region *region, l4_addr_t offset,
00162             l4_size_t num_sectors, L4Re::Dma_space::Direction dir,
00163             L4Re::Dma_space::Dma_addr *phys) override
00164 { return _parent->dma_map(region, offset, num_sectors, dir, phys); }
00165
00166 int dma_unmap(L4Re::Dma_space::Dma_addr phys, l4_size_t num_sectors,
00167              L4Re::Dma_space::Direction dir) override
00168 { return _parent->dma_unmap(phys, num_sectors, dir); }
00169
00170 int inout_data(l4_uint64_t sector, Inout_block const &blocks,
00171               Inout_callback const &cb,
00172               L4Re::Dma_space::Direction dir) override
00173 {
00174     if (sector >= _size)
00175         return -L4_EINVAL;
00176
00177     l4_uint64_t total = 0;
00178     Inout_block const *cur = &blocks;
00179     while (cur)
00180     {
00181         total += cur->num_sectors;
00182         cur = cur->next.get();
00183     }
00184
00185     if (total > _size - sector)
00186         return -L4_EINVAL;
00187
00188     Dbg::trace("partition").printf("Sector on disk: 0x%llx\n", sector + _start);
00189     return _parent->inout_data(sector + _start, blocks, cb, dir);
00190 }
00191
00192 int flush(Inout_callback const &cb) override
00193 {
00194     return _parent->flush(cb);
00195 }
00196
00197 void start_device_scan(Block_device::Errand::Callback const &callback) override
00198 { callback(); }
00199
00200 l4_uint64_t partition_size() const
00201 { return _size; }
00202
00203 l4_uint64_t partition_start() const
00204 { return _start; }
00205
00206 Device_type *parent() const
00207 { return _parent.get(); }

```



```

00208
00209
00210 private:
00211     char _guid[37];
00212     std::ul6string _name;
00213     char _partition_id[4];
00214     cxx::Ref_ptr<Device_type> _parent;
00215     l4_uint64_t _start;
00216     l4_uint64_t _size;
00217 };
00218
00219 } // name space

```

## 16.245 partition.h

```

00001 /*
00002  * Copyright (C) 2018, 2020-2022 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * This file is distributed under the terms of the GNU General Public
00006  * License, version 2. Please see the COPYING-GPL-2 file for details.
00007  */
00008 #pragma once
00009
00010 #include <cstring>
00011 #include <string>
00012 #include <cassert>
00013
00014 #include <l4/cxx/ref_ptr>
00015
00016 #include <l4/l4virtio/virtio_block.h>
00017
00018 #include <l4/libblock-device/debug.h>
00019 #include <l4/libblock-device/errand.h>
00020 #include <l4/libblock-device/inout_memory.h>
00021 #include <l4/libblock-device/gpt.h>
00022
00023 #include <l4/sys/cache.h>
00024
00025 namespace Block_device {
00026
00030 struct Partition_info
00031 {
00032     char            guid[37];
00033     std::ul6string  name;
00034     l4_uint64_t     first;
00035     l4_uint64_t     last;
00036     l4_uint64_t     flags;
00037 };
00038
00039
00043 template <typename DEV>
00044 class Partition_reader : public cxx::Ref_obj
00045 {
00046     enum
00047     {
00048         Max_partitions = 1024
00049     };
00050
00051 public:
00052     using Device_type = DEV;
00053
00054     Partition_reader(Device_type *dev)
00055     : _num_partitions(0),
00056       _dev(dev),
00057       _header(2, dev, L4Re::Dma_space::Direction::From_device)
00058     {}
00059
00060     void read(Errand::Callback const &callback)
00061     {
00062         _num_partitions = 0;
00063         _callback = callback;
00064
00065         // preparation: read the first two sectors
00066         _db = _header.inout_block();
00067         read_sectors(0, &Partition_reader::get_gpt);
00068     }
00069
00070     l4_size_t table_size() const
00071     { return _num_partitions; }
00072
00073     int get_partition(l4_size_t idx, Partition_info *inf) const
00074     {

```

```

00075     if (idx == 0 || idx > _num_partitions)
00076         return -L4_ERANGE;
00077
00078     unsigned secsz = _dev->sector_size();
00079     auto *header = _header.template get<Gpt::Header const>(secsz);
00080
00081     Gpt::Entry *e = _parray.template get<Gpt::Entry>((idx - 1) * header->entry_size);
00082
00083     if (*(l4_uint64_t *) &e->partition_guid) == 0ULL)
00084         return -L4_ENODEV;
00085
00086     render_guid(e->partition_guid, inf->guid);
00087
00088     auto name =
00089         std::u16string((char16_t *)e->name, sizeof(e->name) / sizeof(e->name[0]));
00090     inf->name = name.substr(0, name.find((char16_t) 0));
00091
00092     inf->first = e->first;
00093     inf->last = e->last;
00094     inf->flags = e->flags;
00095
00096     auto info = Dbg::info();
00097     if (info.is_active())
00098     {
00099         info.printf("%3zu: %10lld %10lld %5gMiB [%.37s]\n",
00100             idx, e->first, e->last,
00101             (e->last - e->first + 1.0) * secsz / (1 « 20),
00102             inf->guid);
00103
00104         char buf[37];
00105         info.printf("    : Type: %s\n", render_guid(e->type_guid, buf));
00106     }
00107
00108     auto warn = Dbg::warn();
00109     if (inf->last < inf->first)
00110     {
00111         warn.printf(
00112             "Invalid settings of %3zu. Last lba before first lba. Will ignore.\n",
00113             idx);
00114         // Errors in the GPT shall not crash any service -- just ignore the
00115         // corresponding partition.
00116         return -L4_ENODEV;
00117     }
00118
00119     return L4_EOK;
00120 }
00121
00122 private:
00123 void invoke_callback()
00124 {
00125     assert(_callback);
00126     _callback();
00127     // Reset the callback to drop potential transitive self-references
00128     _callback = nullptr;
00129 }
00130
00131 void get_gpt(int error, l4_size_t)
00132 {
00133     _header.unmap();
00134
00135     if (error < 0)
00136     {
00137         // can't read from device, we are done
00138         invoke_callback();
00139         return;
00140     }
00141
00142     // prepare reading of the table from disk
00143     unsigned secsz = _dev->sector_size();
00144     auto *header = _header.template get<Gpt::Header const>(secsz);
00145
00146     auto info = Dbg::info();
00147     auto trace = Dbg::trace();
00148
00149     if (strncmp(header->signature, "EFI PART", 8) != 0)
00150     {
00151         info.printf("No GUID partition header found.\n");
00152         invoke_callback();
00153         return;
00154     }
00155
00156     // XXX check CRC32 of header
00157
00158     info.printf("GUID partition header found with up to %d partitions.\n",
00159         header->partition_array_size);
00160     char buf[37];
00161     info.printf("GUID: %s\n", render_guid(header->disk_guid, buf));

```

```

00162     trace.printf("Header positions: %llx (Backup: %llx)\n",
00163                 header->current_lba, header->backup_lba);
00164     trace.printf("First + last: %llx and %llx\n",
00165                 header->first_lba, header->last_lba);
00166     trace.printf("Partition table at %llx\n",
00167                 header->partition_array_lba);
00168     trace.printf("Size of a partition entry: %d\n",
00169                 header->entry_size);
00170
00171     info.printf("GUID partition header found with %d partitions.\n",
00172                 header->partition_array_size);
00173
00174     _num_partitions = cxx::min<l4_uint32_t>(header->partition_array_size,
00175                                           Max_partitions);
00176
00177     l4_size_t arraysz = _num_partitions * header->entry_size;
00178     l4_size_t numsec = (arraysz - 1 + secsz) / secsz;
00179
00180     _parray = Inout_memory<Device_type>(numsec, _dev, L4Re::Dma_space::Direction::From_device);
00181     trace.printf("Reading GPT table @ 0x%p\n", _parray.template get<void>(0));
00182
00183     _db = _parray.inout_block();
00184     read_sectors(header->partition_array_lba, &Partition_reader::done_gpt);
00185 }
00186
00187
00188 void done_gpt(int error, l4_size_t)
00189 {
00190     _parray.unmap();
00191
00192     // XXX check CRC32 of table
00193
00194     if (error < 0)
00195         _num_partitions = 0;
00196
00197     invoke_callback();
00198 }
00199
00200 void read_sectors(l4_uint64_t sector,
00201                  void (Partition_reader::*func)(int, l4_size_t))
00202 {
00203     using namespace std::placeholders;
00204     auto next = std::bind(func, this, _1, _2);
00205
00206     l4_addr_t vstart = (l4_addr_t)_db.virt_addr;
00207     l4_addr_t vend = vstart + _db.num_sectors * _dev->sector_size();
00208     l4_cache_inv_data(vstart, vend);
00209
00210     Errand::poll(10, 10000,
00211                 [=]()
00212                 {
00213                     int ret = _dev->inout_data(
00214                         sector, _db,
00215                         [next, vstart, vend](int error, l4_size_t size)
00216                         {
00217                             l4_cache_inv_data(vstart, vend);
00218                             next(error, size);
00219                         },
00220                         L4Re::Dma_space::Direction::From_device);
00221                     if (ret < 0 && ret != -L4_EBUSY)
00222                         invoke_callback();
00223                     return ret != -L4_EBUSY;
00224                 },
00225                 [=](bool ret) { if (!ret) invoke_callback(); });
00226 }
00227
00228
00229 static char const *render_guid(void const *guid_p, char buf[])
00230 {
00231     auto *p = static_cast<unsigned char const *>(guid_p);
00232     snprintf(buf, 37,
00233              "%02X%02X%02X%02X-%02X%02X-%02X%02X-%02X%02X%02X%02X%02X",
00234              p[3], p[2], p[1], p[0], p[5], p[4], p[7], p[6],
00235              p[8], p[9], p[10], p[11], p[12], p[13], p[14], p[15]);
00236
00237     return buf;
00238 }
00239
00240 l4_size_t _num_partitions;
00241 Inout_block _db;
00242 Device_type *_dev;
00243 Inout_memory<Device_type> _header;
00244 Inout_memory<Device_type> _parray;
00245 Errand::Callback _callback;
00246 };
00247
00248

```

```
00249
00250 }
```

## 16.246 request.h

```
00001 /*
00002  * Copyright (C) 2019-2020 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * This file is distributed under the terms of the GNU General Public
00006  * License, version 2. Please see the COPYING-GPL-2 file for details.
00007  */
00008 #pragma once
00009
00010 namespace Block_device {
00011
00012 struct Pending_request
00013 {
00014     virtual ~Pending_request() = 0;
00015     virtual int handle_request() = 0;
00016     virtual void fail_request() = 0;
00017 };
00018 inline Pending_request::~Pending_request() = default;
00019 } // namespace
```

## 16.247 virtio\_client.h

```
00001 /*
00002  * Copyright (C) 2018-2022 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * This file is distributed under the terms of the GNU General Public
00006  * License, version 2. Please see the COPYING-GPL-2 file for details.
00007  */
00008 #pragma once
00009
00010 #include <l4/cxx/ref_ptr>
00011 #include <l4/cxx/unique_ptr_list>
00012 #include <l4/cxx/utils>
00013 #include <l4/sys/cache.h>
00014
00015 #include <l4/sys/task>
00016
00017 #include <l4/l4virtio/server/virtio-block>
00018
00019 #include <l4/libblock-device/debug.h>
00020 #include <l4/libblock-device/device.h>
00021 #include <l4/libblock-device/types.h>
00022 #include <l4/libblock-device/request.h>
00023
00024 namespace Block_device {
00025
00026 template <typename DEV>
00027 class Virtio_client
00028 : public L4virtio::Svr::Block_dev_base<Mem_region_info>,
00029   public L4::Epiface_t<Virtio_client<DEV>, L4virtio::Device>
00030 {
00031 protected:
00032     class Generic_pending_request : public Pending_request
00033     {
00034     protected:
00035         int check_error(int result)
00036         {
00037             if (result < 0 && result != -L4_EBUSY)
00038                 client->handle_request_error(result, this);
00039
00040             return result;
00041         }
00042
00043     public:
00044         explicit Generic_pending_request(Virtio_client *c, cxx::unique_ptr<Request> &&req)
00045             : request(cxx::move(req)), client(c)
00046         {}
00047
00048         void fail_request() override
```

```

00049     {
00050         client->finalize_request(cxx::move(request), 0, L4VIRTIO_BLOCK_S_IOERR);
00051     }
00052
00053     cxx::unique_ptr<Request> request;
00054     Virtio_client *client;
00055 };
00056
00057 struct Pending_inout_request : public Generic_pending_request
00058 {
00059     Inout_block blocks;
00060     L4Re::Dma_space::Direction dir;
00061
00062     explicit Pending_inout_request(Virtio_client *c,
00063                                   cxx::unique_ptr<Request> &req)
00064     : Generic_pending_request(c, cxx::move(req))
00065     {
00066         dir = this->request->header().type == L4VIRTIO_BLOCK_T_OUT
00067             ? L4Re::Dma_space::Direction::To_device
00068             : L4Re::Dma_space::Direction::From_device;
00069     }
00070
00071     ~Pending_inout_request() override
00072     {
00073         this->client->release_dma(this);
00074     }
00075
00076     int handle_request() override
00077     { return this->check_error(this->client->inout_request(this)); }
00078 };
00079
00080 struct Pending_flush_request : public Generic_pending_request
00081 {
00082     using Generic_pending_request::Generic_pending_request;
00083
00084     int handle_request() override
00085     { return this->check_error(this->client->flush_request(this)); }
00086 };
00087
00088 struct Pending_cmd_request : public Generic_pending_request
00089 {
00090     Inout_block blocks;
00091
00092     using Generic_pending_request::Generic_pending_request;
00093
00094     int handle_request() override
00095     {
00096         return this->check_error(this->client->discard_cmd_request(this, 0));
00097     }
00098 };
00099
00100 public:
00101     using Device_type = DEV;
00102
00103     Virtio_client(cxx::Ref_ptr<Device_type> const &dev, unsigned numds, bool readonly)
00104     : L4virtio::Svr::Block_dev_base<Mem_region_info>(L4VIRTIO_VENDOR_KK, 0x100,
00105                                                     dev->capacity() > 9,
00106                                                     dev->is_read_only()
00107                                                         || readonly),
00108       _client_invalidate_cb(nullptr),
00109       _client_idle_cb(nullptr),
00110       _numds(numds),
00111       _device(dev),
00112       _in_flight(0)
00113     {
00114         reset_client();
00115         init_discard_info(0);
00116     }
00117
00118     void reset_device() override
00119     {
00120         if (_client_invalidate_cb)
00121             _client_invalidate_cb(false);
00122         _device->reset();
00123         _negotiated_features.raw = 0;
00124     }
00125
00126     void reset_client()
00127     {
00128         init_mem_info(_numds);
00129         set_seg_max(_device->max_segments());
00130         set_size_max(_device->max_size());
00131         set_flush();
00132         set_config_wce(0); // starting in write-through mode
00133         _shutdown_state = Shutdown_type::Running;
00134         _negotiated_features.raw = 0;
00135     }

```

```

00150
00151 bool queue_stopped() override
00152 { return _shutdown_state == Shutdown_type::Client_gone; }
00153
00154 // make these interfaces public so that a request scheduler can invoke them
00155 using L4virtio::Svr::Block_dev_base<Mem_region_info>::check_for_new_requests;
00156 using L4virtio::Svr::Block_dev_base<Mem_region_info>::get_request;
00157
00158 // make it possible for the request scheduler to register a direct callback
00159 void set_client_invalidate_cb(std::function<void(bool)> &&cb)
00160 {
00161     _client_invalidate_cb = cb;
00162 }
00163
00164 void set_client_idle_cb(std::function<void()> &&cb)
00165 {
00166     _client_idle_cb = cb;
00167 }
00168
00169 // make it possible for the request scheduler to register a device notify IRQ
00170 void set_device_notify_irq(L4::Cap<L4::Irq> irq)
00171 {
00172     _device_notify_irq = irq;
00173 }
00174
00175 L4::Cap<L4::Irq> device_notify_irq() const override
00176 {
00177     return _device_notify_irq;
00178 }
00179
00185 cxx::unique_ptr<Pending_request> start_request(cxx::unique_ptr<Request> &&req)
00186 {
00187     auto trace = Dbg::trace("virtio");
00188
00189     cxx::unique_ptr<Pending_request> pending;
00190
00191     if (_shutdown_state != Shutdown_type::Running)
00192     {
00193         trace.printf("Failing requests as the client is shutting down\n");
00194         this->finalize_request(cxx::move(req), 0, L4VIRTIO_BLOCK_S_IOERR);
00195         return pending;
00196     }
00197
00198     trace.printf("request received: type 0x%x, sector 0x%llx\n",
00199                 req->header().type, req->header().sector);
00200     switch (req->header().type)
00201     {
00202     case L4VIRTIO_BLOCK_T_OUT:
00203     case L4VIRTIO_BLOCK_T_IN:
00204     {
00205         auto p = cxx::make_unique<Pending_inout_request>(this, cxx::move(req));
00206         int ret = build_inout_blocks(p.get());
00207         if (ret == L4_EOK)
00208             pending.reset(p.release());
00209         else
00210             handle_request_error(ret, p.get());
00211         break;
00212     }
00213     case L4VIRTIO_BLOCK_T_FLUSH:
00214     {
00215         auto p = cxx::make_unique<Pending_flush_request>(this, cxx::move(req));
00216         int ret = check_flush_request(p.get());
00217         if (ret == L4_EOK)
00218             pending.reset(p.release());
00219         else
00220             handle_request_error(ret, p.get());
00221         break;
00222     }
00223     case L4VIRTIO_BLOCK_T_WRITE_ZEROES:
00224     case L4VIRTIO_BLOCK_T_DISCARD:
00225     {
00226         auto p = cxx::make_unique<Pending_cmd_request>(this, cxx::move(req));
00227         int ret = build_discard_cmd_blocks(p.get());
00228         if (ret == L4_EOK)
00229             pending.reset(p.release());
00230         else
00231             handle_request_error(ret, p.get());
00232         break;
00233     }
00234     default:
00235         finalize_request(cxx::move(req), 0, L4VIRTIO_BLOCK_S_UNSUPP);
00236         break;
00237     }
00238
00239     return pending;
00240 }
00241

```

```

00242 void task_finished(Generic_pending_request *preg, int error, l4_size_t sz)
00243 {
00244     _in_flight--;
00245
00246     // move on to the next request
00247
00248     // Only finalize if the client is still alive
00249     if (_shutdown_state != Client_gone)
00250         finalize_request(cxx::move(preg->request), sz, error);
00251
00252     // New requests might be schedulable
00253     if (_client_idle_cb)
00254         _client_idle_cb();
00255
00256     // pending request can be dropped
00257     cxx::unique_ptr<Pending_request> ureq(preg);
00258 }
00259
00263 void shutdown_event(Shutdown_type type)
00264 {
00265     // If the client is already in the Client_gone state, it means that it was
00266     // already shutdown and this is another go at its removal. This situation
00267     // can occur because at the time of its previous removal attempt there were
00268     // still I/O requests in progress.
00269     if (_shutdown_state == Client_gone)
00270         return;
00271
00272     // Transitions from System_shutdown are also not allowed, the initiator
00273     // should take care of graceful handling of this.
00274     l4_assert(_shutdown_state != System_shutdown);
00275     // If we are transitioning from System_suspend, it must be only to Running,
00276     // the initiator should handle this gracefully.
00277     l4_assert(_shutdown_state != System_suspend
00278             || type == Shutdown_type::Running);
00279
00280     // Update shutdown state of the client
00281     _shutdown_state = type;
00282
00283     if (type == Shutdown_type::Client_shutdown)
00284     {
00285         reset();
00286         reset_client();
00287         // Client_shutdown must transit to the Running state
00288         l4_assert(_shutdown_state == Shutdown_type::Running);
00289     }
00290
00291     if (type != Shutdown_type::Running)
00292     {
00293         if (_client_invalidate_cb)
00294             _client_invalidate_cb(type != Shutdown_type::Client_gone);
00295         _device->reset();
00296     }
00297 }
00298
00311 L4::Cap<void> register_obj(L4::Registry_iface *registry,
00312                          char const *service = 0)
00313 {
00314     L4::Cap<void> ret;
00315     if (service)
00316         ret = registry->register_obj(this, service);
00317     else
00318         ret = registry->register_obj(this);
00319     L4Re::chkcap(ret);
00320
00321     return ret;
00322 }
00323
00324 L4::Cap<void> register_obj(L4::Registry_iface *registry,
00325                          L4::Cap<L4::Rcv_endpoint> ep)
00326 {
00327     return L4Re::chkcap(registry->register_obj(this, ep));
00328 }
00329
00335 void unregister_obj(L4::Registry_iface *registry)
00336 {
00337     registry->unregister_obj(this);
00338 }
00339
00340 bool busy() const
00341 {
00342     return _in_flight != 0;
00343 }
00344
00345 Notification_domain const *notification_domain() const
00346 { return _device->notification_domain(); }
00347
00348 protected:

```

```

00349 L4::Ipc_svr::Server_iface *server_iface() const override
00350 {
00351     return this->L4::Epiface::server_iface();
00352 }
00353
00354 private:
00355 void release_dma(Pending_inout_request *req)
00356 {
00357     // unmap DMA regions
00358     Inout_block *cur = &req->blocks;
00359     while (cur)
00360     {
00361         if (cur->num_sectors)
00362             _device->dma_unmap(cur->dma_addr, cur->num_sectors, req->dir);
00363         cur = cur->next.get();
00364     }
00365 }
00366
00367 int build_inout_blocks(Pending_inout_request *preq)
00368 {
00369     auto *req = preq->request.get();
00370     l4_size_t sps = _device->sector_size() >> 9;
00371     l4_uint64_t current_sector = req->header().sector / sps;
00372     l4_uint64_t sectors = _device->capacity() / _device->sector_size();
00373     auto dir = preq->dir;
00374
00375     l4_uint32_t flags = 0;
00376     if (req->header().type == L4VIRTIO_BLOCK_T_OUT)
00377     {
00378         // If RO was offered, every write must fail
00379         if (device_features().ro())
00380             return -L4_EIO;
00381
00382         // Figure out whether the write has a write-through or write-back semantics
00383         if (_negotiated_features.config_wce())
00384         {
00385             if (get_writeback() == 1)
00386                 flags = Block_device::Inout_f_wb;
00387         }
00388         else if (_negotiated_features.flush())
00389             flags = Block_device::Inout_f_wb;
00390     }
00391
00392     // Check alignment of the first sector
00393     if (current_sector * sps != req->header().sector)
00394         return -L4_EIO;
00395
00396     Inout_block *last_blk = nullptr;
00397
00398     size_t seg = 0;
00399
00400     while (req->has_more())
00401     {
00402         Request::Data_block b;
00403
00404         if (++seg > _device->max_segments())
00405             return -L4_EIO;
00406
00407         try
00408         {
00409             b = req->next_block();
00410         }
00411         catch (L4virtio::Svr::Bad_descriptor const &e)
00412         {
00413             Dbg::warn().printf("Descriptor error: %s\n", e.message());
00414             return -L4_EIO;
00415         }
00416
00417         l4_size_t off = b.mem->ds_offset() + (l4_addr_t) b.addr
00418             - (l4_addr_t) b.mem->local_base();
00419
00420         l4_size_t sz = b.len / _device->sector_size();
00421
00422         if (sz * _device->sector_size() != b.len)
00423         {
00424             Dbg::warn().printf("Bad block size 0x%x\n", b.len);
00425             return -L4_EIO;
00426         };
00427
00428         // Check bounds
00429         if (sz > sectors)
00430             return -L4_EIO;
00431         if (current_sector > sectors - sz)
00432             return -L4_EIO;
00433
00434         Inout_block *blk;
00435         if (last_blk)

```



```

00436         {
00437             last_blk->next = cxx::make_unique<Inout_block>();
00438             blk = last_blk->next.get();
00439         }
00440         else
00441             blk = &preq->blocks;
00442
00443         L4Re::Dma_space::Dma_addr phys;
00444         long ret = _device->dma_map(b.mem, off, sz, dir, &phys);
00445         if (ret < 0)
00446             return ret;
00447
00448         blk->dma_addr = phys;
00449         blk->virt_addr = (void *) ((l4_addr_t)b.mem->local_base() + off);
00450         blk->num_sectors = sz;
00451         current_sector += sz;
00452         blk->flags = flags;
00453
00454         last_blk = blk;
00455     }
00456
00457     return L4_EOK;
00458 }
00459
00460 void maintain_cache_before_req(Pending_inout_request const *preq)
00461 {
00462     if (preq->dir == L4Re::Dma_space::None)
00463         return;
00464     for (Inout_block const *cur = &preq->blocks; cur; cur = cur->next.get())
00465     {
00466         l4_addr_t vstart = (l4_addr_t)cur->virt_addr;
00467         if (vstart)
00468         {
00469             l4_size_t vsize = cur->num_sectors * _device->sector_size();
00470             if (preq->dir == L4Re::Dma_space::From_device)
00471                 l4_cache_inv_data(vstart, vstart + vsize);
00472             else if (preq->dir == L4Re::Dma_space::To_device)
00473                 l4_cache_clean_data(vstart, vstart + vsize);
00474             else // L4Re::Dma_space::Bidirectional
00475                 l4_cache_flush_data(vstart, vstart + vsize);
00476         }
00477     }
00478 }
00479
00480 void maintain_cache_after_req(Pending_inout_request const *preq)
00481 {
00482     if (preq->dir == L4Re::Dma_space::None)
00483         return;
00484     for (Inout_block const *cur = &preq->blocks; cur; cur = cur->next.get())
00485     {
00486         l4_addr_t vstart = (l4_addr_t)cur->virt_addr;
00487         if (vstart)
00488         {
00489             l4_size_t vsize = cur->num_sectors * _device->sector_size();
00490             if (preq->dir != L4Re::Dma_space::To_device)
00491                 l4_cache_inv_data(vstart, vstart + vsize);
00492         }
00493     }
00494 }
00495
00496 int inout_request(Pending_inout_request *preq)
00497 {
00498     auto *req = preq->request.get();
00499     l4_uint64_t sector = req->header().sector / (_device->sector_size() >> 9);
00500
00501     maintain_cache_before_req(preq);
00502     int res = _device->inout_data(
00503         sector, preq->blocks,
00504         [this, preq](int error, l4_size_t sz) {
00505             maintain_cache_after_req(preq);
00506             task_finished(preq, error, sz);
00507         },
00508         preq->dir);
00509
00510     // request successfully submitted to device
00511     if (res >= 0)
00512         _in_flight++;
00513
00514     return res;
00515 }
00516
00517 int check_flush_request(Pending_flush_request *preq)
00518 {
00519     if (!_negotiated_features.flush())
00520         return -L4_ENOSYS;
00521
00522     auto *req = preq->request.get();

```

```

00523
00524     // sector must be zero for FLUSH
00525     if (req->header().sector)
00526         return -L4_ENOSYS;
00527
00528     return L4_EOK;
00529 }
00530
00531 int flush_request(Pending_flush_request *preq)
00532 {
00533     int res = _device->flush([this, preq](int error, l4_size_t sz) {
00534         task_finished(preq, error, sz);
00535     });
00536
00537     // request successfully submitted to device
00538     if (res >= 0)
00539         _in_flight++;
00540
00541     return res;
00542 }
00543
00544 bool check_features(void) override
00545 {
00546     _negotiated_features = negotiated_features();
00547     return true;
00548 }
00549
00550 template <typename T = Device_type>
00551 void init_discard_info(long) {}
00552
00553 template <typename T = Device_type>
00554 auto init_discard_info(int)
00555     -> decltype(((T*)0)->discard_info(), void())
00556 {
00557     _di = _device->discard_info();
00558
00559     // Convert sector sizes to virtio 512-byte sectors.
00560     size_t sps = _device->sector_size() >> 9;
00561     if (_di.max_discard_sectors)
00562         set_discard(_di.max_discard_sectors * sps, _di.max_discard_seg,
00563                     _di.discard_sector_alignment * sps);
00564     if (_di.max_write_zeroes_sectors)
00565         set_write_zeroes(_di.max_write_zeroes_sectors * sps,
00566                           _di.max_write_zeroes_seg, _di.write_zeroes_may_unmap);
00567 }
00568
00569 int build_discard_cmd_blocks(Pending_cmd_request *preq)
00570 {
00571     auto *req = preq->request.get();
00572     bool discard = (req->header().type == L4VIRTIO_BLOCK_T_DISCARD);
00573
00574     if (this->device_features().ro())
00575         return -L4_EIO;
00576
00577     // sector is used only for inout requests, it must be zero for WzD
00578     if (req->header().sector)
00579         return -L4_ENOSYS;
00580
00581     if (discard)
00582     {
00583         if (!_negotiated_features.discard())
00584             return -L4_ENOSYS;
00585     }
00586     else
00587     {
00588         if (!_negotiated_features.write_zeroes())
00589             return -L4_ENOSYS;
00590     }
00591
00592     auto *d = _device.get();
00593
00594     size_t seg = 0;
00595     size_t max_seg = discard ? _di.max_discard_seg : _di.max_write_zeroes_seg;
00596
00597     l4_size_t sps = d->sector_size() >> 9;
00598     l4_uint64_t sectors = d->capacity() / d->sector_size();
00599
00600     Inout_block *last_blk = nullptr;
00601
00602     while (req->has_more())
00603     {
00604         Request::Data_block b;
00605
00606         try
00607         {
00608             b = req->next_block();
00609         }

```

```

00610         catch (L4virtio::Svr::Bad_descriptor const &e)
00611         {
00612             Dbg::warn().printf("Descriptor error: %s\n", e.message());
00613             return -L4_EIO;
00614         }
00615
00616         auto *payload = reinterpret_cast<l4virtio_block_discard_t *>(b.addr);
00617
00618         size_t items = b.len / sizeof(payload[0]);
00619         if (items * sizeof(payload[0]) != b.len)
00620             return -L4_EIO;
00621
00622         if (seg + items > max_seg)
00623             return -L4_EIO;
00624         seg += items;
00625
00626         for (auto i = 0u; i < items; i++)
00627         {
00628             auto p = cxx::access_once<l4virtio_block_discard_t>(&payload[i]);
00629
00630             // Check sector size alignment. Discard sector alignment is not
00631             // strictly enforced as it is merely a hint to the driver.
00632             if (p.sector % sps != 0)
00633                 return -L4_EIO;
00634             if (p.num_sectors % sps != 0)
00635                 return -L4_EIO;
00636
00637             // Convert to the device sector size
00638             p.sector /= sps;
00639             p.num_sectors /= sps;
00640
00641             // Check bounds
00642             if (p.num_sectors > sectors)
00643                 return -L4_EIO;
00644             if (p.sector > sectors - p.num_sectors)
00645                 return -L4_EIO;
00646
00647             if (p.flags & L4VIRTIO_BLOCK_DISCARD_F_RESERVED)
00648                 return -L4_ENOSYS;
00649
00650             Inout_block *blk;
00651             if (last_blk)
00652             {
00653                 last_blk->next = cxx::make_unique<Inout_block>();
00654                 blk = last_blk->next.get();
00655             }
00656             else
00657                 blk = &preq->blocks;
00658
00659             blk->sector = p.sector;
00660             blk->num_sectors = p.num_sectors;
00661
00662             if (discard)
00663             {
00664                 if (p.flags & L4VIRTIO_BLOCK_DISCARD_F_UNMAP)
00665                     return -L4_ENOSYS;
00666                 if (p.num_sectors > _di.max_discard_sectors)
00667                     return -L4_EIO;
00668             }
00669             else
00670             {
00671                 if (p.flags & L4VIRTIO_BLOCK_DISCARD_F_UNMAP
00672                     && _di.write_zeroes_may_unmap)
00673                     blk->flags = Inout_f_unmap;
00674                 if (p.num_sectors > _di.max_write_zeroes_sectors)
00675                     return -L4_EIO;
00676             }
00677
00678             last_blk = blk;
00679         }
00680     }
00681
00682     return L4_EOK;
00683 }
00684
00685 template <typename T = Device_type>
00686 int discard_cmd_request(Pending_cmd_request *, long)
00687 { return -L4_EIO; }
00688
00689 template <typename T = Device_type>
00690 auto discard_cmd_request(Pending_cmd_request *preq, int)
00691     -> decltype(((T*)0)->discard_info(), int())
00692 {
00693     auto *req = preq->request.get();
00694     bool discard = (req->header().type == L4VIRTIO_BLOCK_T_DISCARD);
00695
00696     int res = _device->discard(

```

```

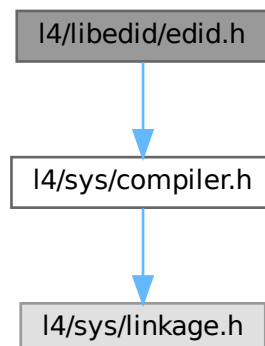
00697     0, preq->blocks,
00698     [this, preq](int error, l4_size_t sz) { task_finished(preq, error, sz); },
00699     discard);
00700
00701     // request successfully submitted to device
00702     if (res >= 0)
00703         _in_flight++;
00704
00705     return res;
00706 }
00707
00708 // only use on errors that are not busy
00709 void handle_request_error(int error, Generic_pending_request *pending)
00710 {
00711     auto trace = Dbg::trace("virtio");
00712
00713     if (error == -L4_ENOSYS)
00714     {
00715         trace.printf("Unsupported operation.\n");
00716         finalize_request(cxx::move(pending->request), 0,
00717                         L4VIRTIO_BLOCK_S_UNSUPP);
00718     }
00719     else
00720     {
00721         trace.printf("Got IO error: %d\n", error);
00722         finalize_request(cxx::move(pending->request), 0, L4VIRTIO_BLOCK_S_IOERR);
00723     }
00724 }
00725
00726 protected:
00727     L4::Cap<L4::Irq> _device_notify_irq;
00728     std::function<void(bool)> _client_invalidate_cb;
00729     std::function<void()> _client_idle_cb;
00730     unsigned _numds;
00731     Shutdown_type _shutdown_state;
00732     cxx::Ref_ptr<Device_type> _device;
00733     Device_discard_feature::Discard_info _di;
00734
00735     L4virtio::Svr::Block_features _negotiated_features;
00736
00737     unsigned _in_flight;
00738 };
00739
00740 } //name space

```

## 16.248 l4/libedid/edid.h File Reference

#include <l4/sys/compiler.h>

Include dependency graph for edid.h:



## Enumerations

- enum `Libedid_consts` { `Libedid_block_size` = 128 }  
*EDID constants.*

## Functions

- int `libedid_check_header` (const unsigned char \*edid)  
*Check for valid EDID header.*
- int `libedid_checksum` (const unsigned char \*edid)  
*Calculates the EDID checksum.*
- unsigned `libedid_version` (const unsigned char \*edid)  
*Returns the EDID version number.*
- unsigned `libedid_revision` (const unsigned char \*edid)  
*Returns the EDID revision number.*
- void `libedid_pnp_id` (const unsigned char \*edid, unsigned char \*id)  
*Extracts the display's PnP ID.*
- void `libedid_preferred_resolution` (const unsigned char \*edid, unsigned \*w, unsigned \*h)  
*Extract the display's preferred mode.*
- unsigned `libedid_num_ext_blocks` (const unsigned char \*edid)  
*Get the number of EDID extension blocks.*
- unsigned `libedid_dump_standard_timings` (const unsigned char \*edid)  
*Dump the standard timings to stdout.*
- void `libedid_dump` (const unsigned char \*edid)  
*Dump raw EDID data to stdout.*

## 16.249 edid.h

[Go to the documentation of this file.](#)

```

00001
00004 /*
00005  * (c) 2014 Matthias Lange <matthias.lange@kernkonzept.com>
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU Lesser General Public License 2.1.
00009  * Please see the COPYING-LGPL-2.1 file for details.
00010  */
00011 #pragma once
00012
00013 #include <14/sys/compiler.h>
00014
00023 enum Libedid_consts
00024 {
00025     Libedid_block_size = 128,
00026 };
00027
00028 __BEGIN_DECLS
00029
00037 int libedid_check_header(const unsigned char *edid);
00038
00046 int libedid_checksum(const unsigned char *edid);
00047
00055 unsigned libedid_version(const unsigned char *edid);
00056
00064 unsigned libedid_revision(const unsigned char *edid);
00065
00072 void libedid_pnp_id(const unsigned char *edid, unsigned char *id);
00073
00081 void libedid_preferred_resolution(const unsigned char *edid,
00082                                   unsigned *w, unsigned *h);
00083
00091 unsigned libedid_num_ext_blocks(const unsigned char *edid);
00092
00100 unsigned libedid_dump_standard_timings(const unsigned char *edid);
00101
00107 void libedid_dump(const unsigned char *edid);
00108
00111 __END_DECLS

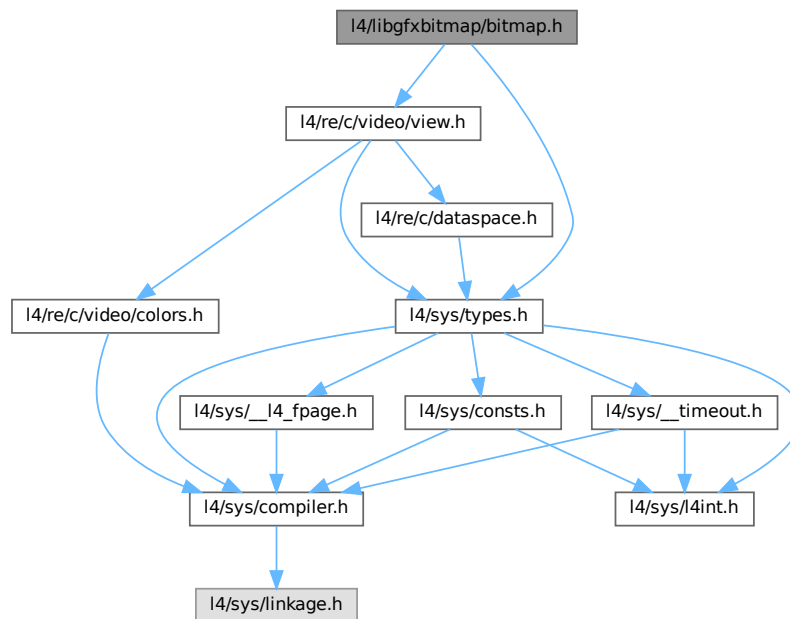
```

## 16.250 I4/libgfxbitmap/bitmap.h File Reference

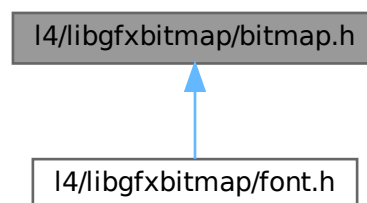
Bitmap renderer header file.

```
#include <l4/sys/types.h>
#include <l4/re/c/video/view.h>
```

Include dependency graph for bitmap.h:



This graph shows which files directly or indirectly include this file:



### Data Structures

- struct [gfxbitmap\\_offset](#)  
offsets in `pmap[]` and `bmap[]`

**Param macros for bmap\_\***

Bitmap type - start least or start most significant bit

- `#define pSLIM_BMAP_START_MSB 0x02`  
*'pbm'-style: "The bits are stored eight per byte, high bit first low bit last."*
- `#define pSLIM_BMAP_START_LSB 0x01`
- `typedef unsigned int gfxbitmap_color_t`  
*Standard color type.*
- `typedef unsigned int gfxbitmap_color_pix_t`  
*Specific color type.*
- `gfxbitmap_color_pix_t gfxbitmap_convert_color (l4re_video_view_info_t *vi, gfxbitmap_color_t rgb)`  
*Convert a color.*
- `void gfxbitmap_fill (l4_uint8_t *vfb, l4re_video_view_info_t *vi, int x, int y, int w, int h, gfxbitmap_color_pix_t color)`  
*Fill a rectangular area with a color.*
- `void gfxbitmap_bmap (l4_uint8_t *vfb, l4re_video_view_info_t *vi, l4_int16_t x, l4_int16_t y, l4_uint32_t w, l4_uint32_t h, l4_uint8_t *bmap, gfxbitmap_color_pix_t fgc, gfxbitmap_color_pix_t bgc, struct gfxbitmap_offset *offset, l4_uint8_t mode)`  
*Fill a rectangular area with a bicolor bitmap pattern.*
- `void gfxbitmap_set (l4_uint8_t *vfb, l4re_video_view_info_t *vi, l4_int16_t x, l4_int16_t y, l4_uint32_t w, l4_uint32_t h, l4_uint32_t xoffs, l4_uint32_t yoffs, l4_uint8_t *pmap, struct gfxbitmap_offset *offset, l4_uint32_t pwidth)`  
*Set area from source area.*
- `void gfxbitmap_copy (l4_uint8_t *dest, l4_uint8_t *src, l4re_video_view_info_t *vi, int x, int y, int w, int h, int dx, int dy)`  
*Copy a rectangular area.*

**16.250.1 Detailed Description**

Bitmap renderer header file.

Definition in file [bitmap.h](#).

**16.250.2 Macro Definition Documentation****16.250.2.1 pSLIM\_BMAP\_START\_LSB**

```
#define pSLIM_BMAP_START_LSB 0x01
```

the other way round

Definition at line 41 of file [bitmap.h](#).

## 16.250.3 Typedef Documentation

### 16.250.3.1 gfxbitmap\_color\_pix\_t

```
typedef unsigned int gfxbitmap_color_pix_t
```

Specific color type.

This color type is specific for a particular framebuffer, it can be use to write pixel on a framebuffer. Use `gfxbitmap_convert_color` to convert from `gfxbitmap_color_t` to `gfxbitmap_color_pix_t`.

Definition at line 64 of file [bitmap.h](#).

### 16.250.3.2 gfxbitmap\_color\_t

```
typedef unsigned int gfxbitmap_color_t
```

Standard color type.

It's a RGB type with 8bits for each channel, regardless of the framebuffer used.

Definition at line 55 of file [bitmap.h](#).

## 16.250.4 Function Documentation

### 16.250.4.1 gfxbitmap\_bmap()

```
void gfxbitmap_bmap (
    14_uint8_t * vfb,
    14re_video_view_info_t * vi,
    14_int16_t x,
    14_int16_t y,
    14_uint32_t w,
    14_uint32_t h,
    14_uint8_t * bmap,
    gfxbitmap_color_pix_t fg,
    gfxbitmap_color_pix_t bg,
    struct gfxbitmap_offset * offset,
    14_uint8_t mode )
```

Fill a rectangular area with a bicolor bitmap pattern.

#### Parameters

<i>vfb</i>	Frame buffer.
<i>vi</i>	Frame buffer information structure.
<i>x</i>	X position of area.
<i>y</i>	Y position of area.
<i>w</i>	Width of area.
<i>h</i>	Height of area.
<i>bmap</i>	Bitmap pattern.
<i>fg</i>	Foreground color.
<i>bg</i>	Background color.
<i>offset</i>	Offsets.
<i>mode</i>	Mode



See also

[pSLIM\\_BMAP\\_START\\_MSB](#) and [pSLIM\\_BMAP\\_START\\_LSB](#).

#### 16.250.4.2 gfxbitmap\_convert\_color()

```
gfxbitmap_color_pix_t gfxbitmap_convert_color (
    l4re_video_view_info_t * vi,
    gfxbitmap_color_t rgb )
```

Convert a color.

Converts a given color in standard format to the format used in the framebuffer.

#### 16.250.4.3 gfxbitmap\_copy()

```
void gfxbitmap_copy (
    l4_uint8_t * dest,
    l4_uint8_t * src,
    l4re_video_view_info_t * vi,
    int x,
    int y,
    int w,
    int h,
    int dx,
    int dy )
```

Copy a rectangular area.

##### Parameters

<i>dest</i>	Destination frame buffer.
<i>src</i>	Source frame buffer.
<i>vi</i>	Frame buffer information structure.
<i>x</i>	Source X position of area.
<i>y</i>	Source Y position of area.
<i>w</i>	Width of area.
<i>h</i>	Height of area.
<i>dx</i>	Source X position of area.
<i>dy</i>	Source Y position of area.

#### 16.250.4.4 gfxbitmap\_fill()

```
void gfxbitmap_fill (
    l4_uint8_t * vfb,
    l4re_video_view_info_t * vi,
    int x,
    int y,
    int w,
```

```
int h,
gfxbitmap_color_pix_t color )
```

Fill a rectangular area with a color.

#### Parameters

<i>vfb</i>	Frame buffer.
<i>vi</i>	Frame buffer information structure.
<i>x</i>	X position of area.
<i>y</i>	Y position of area.
<i>w</i>	Width of area.
<i>h</i>	Height of area.
<i>color</i>	Color of area.

#### 16.250.4.5 gfxbitmap\_set()

```
void gfxbitmap_set (
    l4_uint8_t * vfb,
    l4re_video_view_info_t * vi,
    l4_int16_t x,
    l4_int16_t y,
    l4_uint32_t w,
    l4_uint32_t h,
    l4_uint32_t xoffs,
    l4_uint32_t yoffs,
    l4_uint8_t * pmap,
    struct gfxbitmap_offset * offset,
    l4_uint32_t pwidth )
```

Set area from source area.

#### Parameters

<i>vfb</i>	Frame buffer.
<i>vi</i>	Frame buffer information structure.
<i>x</i>	X position of area.
<i>y</i>	Y position of area.
<i>w</i>	Width of area.
<i>h</i>	Height of area.
<i>pmap</i>	Source.
<i>xoffs</i>	X offset.
<i>yoffs</i>	Y offset.
<i>offset</i>	Offsets.
<i>pwidth</i>	Width of source in bytes.

## 16.251 bitmap.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/types.h>
00015 #include <l4/re/c/video/view.h>
00016
00032 EXTERN_C_BEGIN
00038 #define pSLIM_BMAP_START_MSB    0x02
00041 #define pSLIM_BMAP_START_LSB    0x01
00043
00048
00055 typedef unsigned int  gfxbitmap_color_t;
00056
00064 typedef unsigned int  gfxbitmap_color_pix_t;
00065
00067 struct gfxbitmap_offset
00068 {
00069     l4_uint32_t preskip_x;
00070     l4_uint32_t preskip_y;
00071     l4_uint32_t endskip_x;
00072 };
00073
00080 gfxbitmap_color_pix_t
00081 gfxbitmap_convert_color(l4re_video_view_info_t *vi, gfxbitmap_color_t rgb);
00082
00094 void
00095 gfxbitmap_fill(l4_uint8_t *vfb, l4re_video_view_info_t *vi,
00096               int x, int y, int w, int h, gfxbitmap_color_pix_t color);
00097
00115 void
00116 gfxbitmap_bmap(l4_uint8_t *vfb, l4re_video_view_info_t *vi,
00117               l4_int16_t x, l4_int16_t y, l4_uint32_t w,
00118               l4_uint32_t h, l4_uint8_t *bmap,
00119               gfxbitmap_color_pix_t fgc, gfxbitmap_color_pix_t bgc,
00120               struct gfxbitmap_offset *offset, l4_uint8_t mode);
00121
00137 void
00138 gfxbitmap_set(l4_uint8_t *vfb, l4re_video_view_info_t *vi,
00139               l4_int16_t x, l4_int16_t y, l4_uint32_t w,
00140               l4_uint32_t h, l4_uint32_t xoffs, l4_uint32_t yoffs,
00141               l4_uint8_t *pmap, struct gfxbitmap_offset *offset,
00142               l4_uint32_t pwidth);
00143
00157 void
00158 gfxbitmap_copy(l4_uint8_t *dest, l4_uint8_t *src, l4re_video_view_info_t *vi,
00159               int x, int y, int w, int h, int dx, int dy);
00162 EXTERN_C_END

```

## 16.252 l4/libgfxbitmap/font.h File Reference

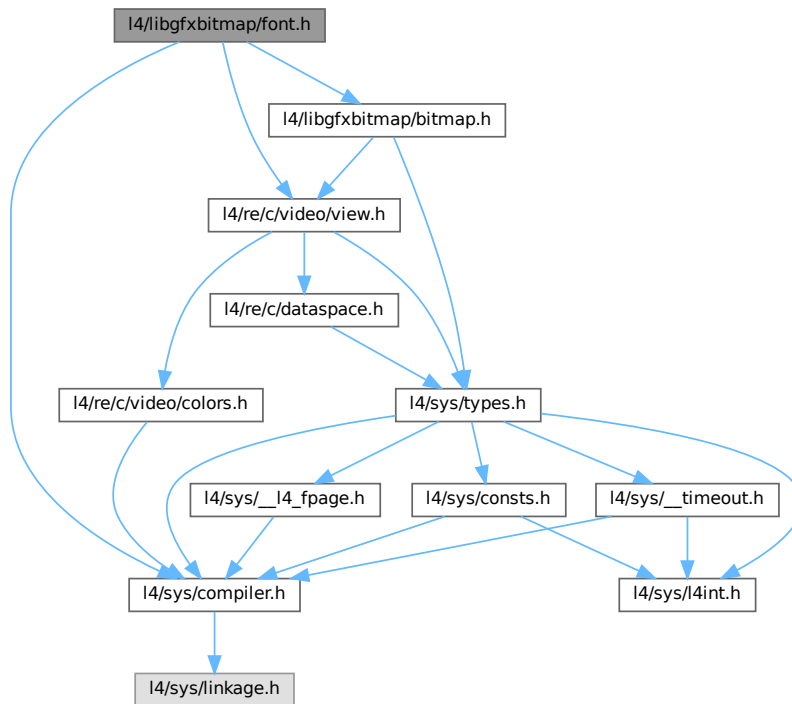
Bitmap font renderer header file.

```

#include <l4/sys/compiler.h>
#include <l4/re/c/video/view.h>
#include <l4/libgfxbitmap/bitmap.h>

```

Include dependency graph for font.h:



## Macros

- `#define GFXBITMAP_DEFAULT_FONT (void *)0`  
Constant to use for the default font.

## Typedefs

- `typedef void * gfxbitmap_font_t`  
Font.

## Enumerations

- `enum`  
Constant for length field.

## Functions

- `int gfxbitmap_font_init (void)`  
Initialize the library.
- `gfxbitmap_font_t gfxbitmap_font_get (const char *name)`  
Get a font descriptor.
- `unsigned gfxbitmap_font_width (gfxbitmap_font_t font)`

*Get the font width.*

- unsigned [gfxbitmap\\_font\\_height](#) ([gfxbitmap\\_font\\_t](#) font)

*Get the font height.*

- void \* [gfxbitmap\\_font\\_data](#) ([gfxbitmap\\_font\\_t](#) font, unsigned c)

*Get bitmap font data for a specific character.*

- void [gfxbitmap\\_font\\_text](#) (void \*fb, [l4re\\_video\\_view\\_info\\_t](#) \*vi, [gfxbitmap\\_font\\_t](#) font, const char \*text, unsigned len, unsigned x, unsigned y, [gfxbitmap\\_color\\_pix\\_t](#) fg, [gfxbitmap\\_color\\_pix\\_t](#) bg)

*Render a string to a framebuffer.*

- void [gfxbitmap\\_font\\_text\\_scale](#) (void \*fb, [l4re\\_video\\_view\\_info\\_t](#) \*vi, [gfxbitmap\\_font\\_t](#) font, const char \*text, unsigned len, unsigned x, unsigned y, [gfxbitmap\\_color\\_pix\\_t](#) fg, [gfxbitmap\\_color\\_pix\\_t](#) bg, int scale\_x, int scale\_y)

*Render a string to a framebuffer, including scaling.*

## 16.252.1 Detailed Description

Bitmap font renderer header file.

Definition in file [font.h](#).

## 16.252.2 Enumeration Type Documentation

### 16.252.2.1 anonymous enum

anonymous enum

Constant for length field.

Use this if the function should call strlen on the text argument itself.

Definition at line 38 of file [font.h](#).

## 16.252.3 Function Documentation

### 16.252.3.1 gfxbitmap\_font\_data()

```
void * gfxbitmap_font_data (  
    gfxbitmap\_font\_t font,  
    unsigned c )
```

Get bitmap font data for a specific character.

Parameters

<i>font</i>	Font.
<i>c</i>	Character.

**Returns**

Pointer to bmap data, NULL on error.

**16.252.3.2 gfxbitmap\_font\_get()**

```
gfxbitmap_font_t gfxbitmap_font_get (
    const char * name )
```

Get a font descriptor.

**Parameters**

<i>name</i>	Name of the font.
-------------	-------------------

**Returns**

A (opaque) font descriptor, or NULL if font could not be found.

**16.252.3.3 gfxbitmap\_font\_height()**

```
unsigned gfxbitmap_font_height (
    gfxbitmap_font_t font )
```

Get the font height.

**Parameters**

<i>font</i>	Font.
-------------	-------

**Returns**

Font height, 0 if font height could not be retrieved.

**16.252.3.4 gfxbitmap\_font\_init()**

```
int gfxbitmap_font_init (
    void )
```

Initialize the library.

This function must be called before any other font function of this library.

**Returns**

0 on success, other on error

### 16.252.3.5 gfxbitmap\_font\_text()

```
void gfxbitmap_font_text (
    void * fb,
    l4re_video_view_info_t * vi,
    gfxbitmap_font_t font,
    const char * text,
    unsigned len,
    unsigned x,
    unsigned y,
    gfxbitmap_color_pix_t fg,
    gfxbitmap_color_pix_t bg )
```

Render a string to a framebuffer.

#### Parameters

<i>fb</i>	Pointer to frame buffer.
<i>vi</i>	Frame buffer info structure.
<i>font</i>	Font.
<i>text</i>	Text string.
<i>len</i>	Length of the text string.
<i>x</i>	Horizontal position in the frame buffer.
<i>y</i>	Vertical position in the frame buffer.
<i>fg</i>	Foreground color.
<i>bg</i>	Background color.

### 16.252.3.6 gfxbitmap\_font\_text\_scale()

```
void gfxbitmap_font_text_scale (
    void * fb,
    l4re_video_view_info_t * vi,
    gfxbitmap_font_t font,
    const char * text,
    unsigned len,
    unsigned x,
    unsigned y,
    gfxbitmap_color_pix_t fg,
    gfxbitmap_color_pix_t bg,
    int scale_x,
    int scale_y )
```

Render a string to a framebuffer, including scaling.

#### Parameters

<i>fb</i>	Pointer to frame buffer.
<i>vi</i>	Frame buffer info structure.
<i>font</i>	Font.
<i>text</i>	Text string.
<i>len</i>	Length of the text string.
<i>x</i>	Horizontal position in the frame buffer.

## Parameters

<i>y</i>	Vertical position in the frame buffer.
<i>fg</i>	Foreground color.
<i>bg</i>	Background color.
<i>scale</i> <sub><i>_x</i></sub>	Horizonal scale factor.
<i>scale</i> <sub><i>_y</i></sub>	Vertical scale factor.

**16.252.3.7 gfxbitmap\_font\_width()**

```
unsigned gfxbitmap_font_width (
    gfxbitmap_font_t font )
```

Get the font width.

## Parameters

<i>font</i>	Font.
-------------	-------

## Returns

Font width, 0 if font width could not be retrieved.

**16.253 font.h**

[Go to the documentation of this file.](#)

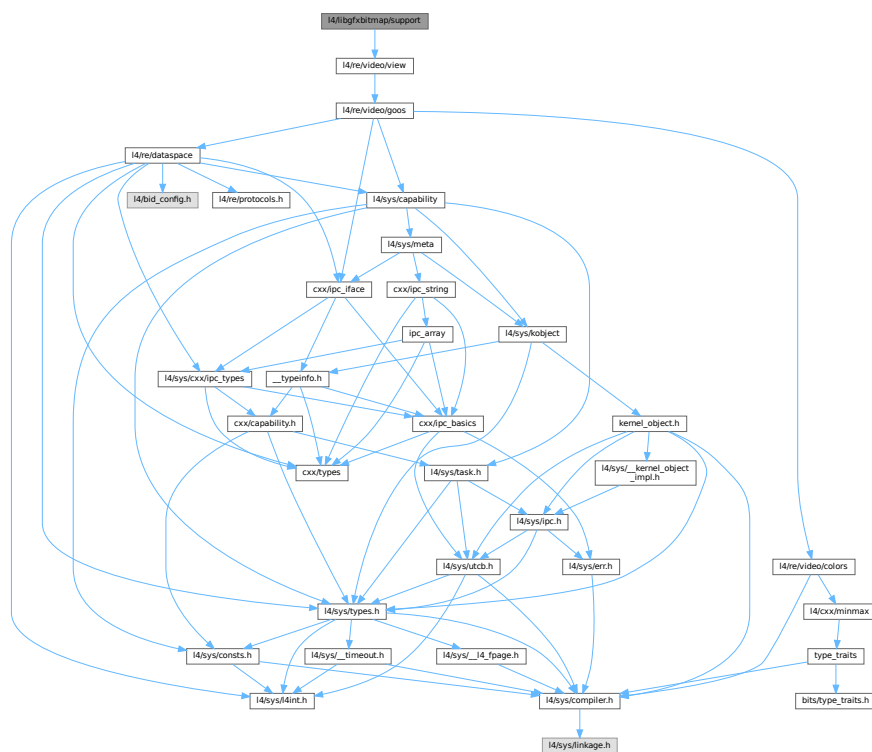
```
00001
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/compiler.h>
00015 #include <l4/re/c/video/view.h>
00016 #include <l4/libgfxbitmap/bitmap.h>
00017
00027
00031 #define GFXBITMAP_DEFAULT_FONT (void *)0
00032
00038 enum { GFXBITMAP_USE_STRLen = ~0U };
00039
00040 EXTERN_C_BEGIN
00041
00043 typedef void *gfxbitmap_font_t;
00044
00053 L4_CV int gfxbitmap_font_init(void);
00054
00062 L4_CV gfxbitmap_font_t gfxbitmap_font_get(const char *name);
00063
00070 L4_CV unsigned
00071 gfxbitmap_font_width(gfxbitmap_font_t font);
00072
00079 L4_CV unsigned
00080 gfxbitmap_font_height(gfxbitmap_font_t font);
00081
00089 L4_CV void *
```



## 16.254 I4/libgfxbitmap/support File Reference

```
#include <l4/re/video/view>
```

Include dependency graph for support:



Definition in file [support](#).

## 16.255 support

[Go to the documentation of this file.](#)

```

00001 /* vim:set ft=cpp: */
00009 /*
00010  * (c) 2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU Lesser General Public License 2.1.
00014  * Please see the COPYING-LGPL-2.1 file for details.
00015  */
00016 #ifndef __LIBTERM_SUPPORT_H__
00017 #define __LIBTERM_SUPPORT_H__
00018
00019 #include <l4/re/video/view>
00020
00021 void
00022 libterm_init_colors(L4Re::Video::View::Info *fbi);
00023
00024 int
00025 libterm_get_color(int mode, int color);
00026
00027 #endif

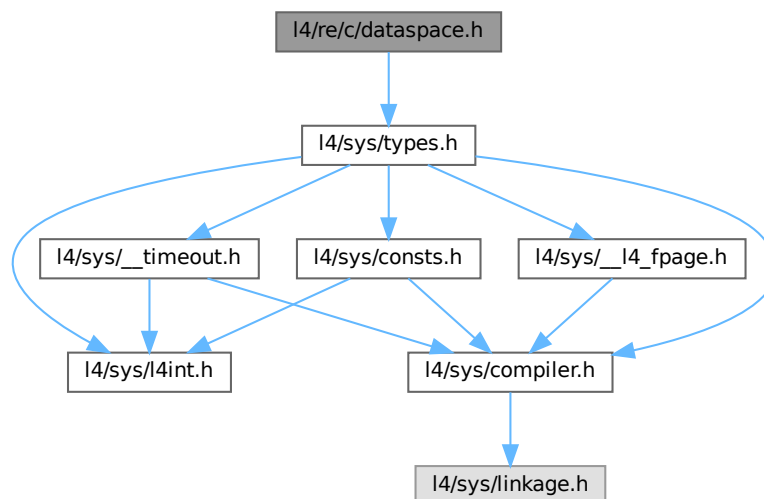
```

## 16.256 l4/re/c/dataspace.h File Reference

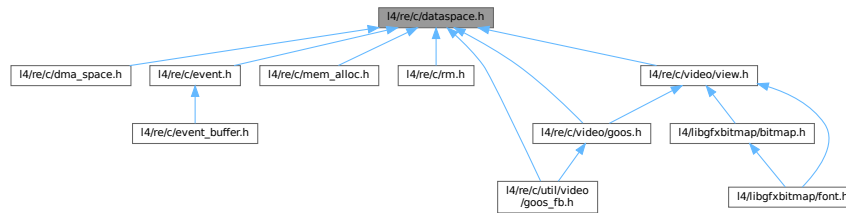
Data space C interface.

```
#include <l4/sys/types.h>
```

Include dependency graph for dataspace.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4re\\_ds\\_stats\\_t](#)  
*Information about the data space.*

## Typedefs

- typedef [l4\\_cap\\_idx\\_t](#) [l4re\\_ds\\_t](#)  
*Dataspace type.*

## Enumerations

- enum [l4re\\_ds\\_map\\_flags](#) { }  
*Flags to specify the memory mapping type of a request.*

## Functions

- long [l4re\\_ds\\_clear](#) ([l4re\\_ds\\_t](#) ds, [l4re\\_ds\\_offset\\_t](#) offset, [l4re\\_ds\\_size\\_t](#) size) [L4\\_NOTHROW](#)  
*Clear parts of a dataspace.*
- long [l4re\\_ds\\_allocate](#) ([l4re\\_ds\\_t](#) ds, [l4re\\_ds\\_offset\\_t](#) offset, [l4re\\_ds\\_size\\_t](#) size) [L4\\_NOTHROW](#)  
*Allocate a range in the dataspace.*
- int [l4re\\_ds\\_copy\\_in](#) ([l4re\\_ds\\_t](#) ds, [l4re\\_ds\\_offset\\_t](#) dst\_offs, [l4re\\_ds\\_t](#) src, [l4re\\_ds\\_offset\\_t](#) src\_offs, [l4re\\_ds\\_size\\_t](#) size) [L4\\_NOTHROW](#)  
*Copy contents from another dataspace.*
- [l4re\\_ds\\_size\\_t](#) [l4re\\_ds\\_size](#) ([l4re\\_ds\\_t](#) ds) [L4\\_NOTHROW](#)  
*Get size of a dataspace.*
- [l4re\\_ds\\_flags\\_t](#) [l4re\\_ds\\_flags](#) ([l4re\\_ds\\_t](#) ds) [L4\\_NOTHROW](#)  
*Get flags of the dataspace.*
- int [l4re\\_ds\\_info](#) ([l4re\\_ds\\_t](#) ds, [l4re\\_ds\\_stats\\_t](#) \*stats) [L4\\_NOTHROW](#)  
*Get information on the dataspace.*
- int [l4re\\_ds\\_map\\_info](#) ([l4re\\_ds\\_t](#) ds, [l4\\_addr\\_t](#) \*start\_addr, [l4\\_addr\\_t](#) \*end\_addr) [L4\\_NOTHROW](#)  
*Get mapping range of dataspace.*

### 16.256.1 Detailed Description

Data space C interface.

Definition in file [dataspace.h](#).

## 16.257 dataspace.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00031 #include <l4/sys/types.h>
00032
00033 EXTERN_C_BEGIN
00034
00039 typedef l4_cap_idx_t l4re_ds_t;
00040 typedef l4_uint64_t l4re_ds_size_t;
00041 typedef l4_uint64_t l4re_ds_offset_t;
00042 typedef l4_uint64_t l4re_ds_map_addr_t;
00043 typedef unsigned long l4re_ds_flags_t;
00044
00049 typedef struct {
00050     l4re_ds_size_t size;
00051     l4re_ds_flags_t flags;
00052 } l4re_ds_stats_t;
00053
00058 enum l4re_ds_map_flags {
00059     L4RE_DS_F_R    = L4_FPAGE_RO,
00060     L4RE_DS_F_W    = L4_FPAGE_W,
00061     L4RE_DS_F_X    = L4_FPAGE_X,
00062     L4RE_DS_F_RW   = L4_FPAGE_RW,
00063     L4RE_DS_F_RX   = L4_FPAGE_RX,
00064     L4RE_DS_F_RWX  = L4_FPAGE_RWX,
00065
00066     L4RE_DS_F_RIGHTS_MASK = 0x0f,
00067
00068     L4RE_DS_F_NORMAL      = 0x00,
00069     L4RE_DS_F_CACHEABLE  = L4RE_DS_F_NORMAL,
00070     L4RE_DS_F_BUFFERABLE = 0x10,
00071     L4RE_DS_F_UNCACHEABLE = 0x20,
00072     L4RE_DS_F_CACHING_MASK = 0x30,
00073     L4RE_DS_F_CACHING_SHIFT = 4,
00074 };
00075
00081 L4_CV int
00082 l4re_ds_map(l4re_ds_t ds,
00083             l4re_ds_offset_t offset,
00084             l4re_ds_flags_t flags,
00085             l4re_ds_map_addr_t local_addr,
00086             l4re_ds_map_addr_t min_addr,
00087             l4re_ds_map_addr_t max_addr) L4_NOTHROW;
00088
00094 L4_CV int
00095 l4re_ds_map_region(l4re_ds_t ds,
00096                   l4re_ds_offset_t offset,
00097                   l4re_ds_flags_t flags,
00098                   l4re_ds_map_addr_t min_addr,
00099                   l4re_ds_map_addr_t max_addr) L4_NOTHROW;
00100
00107 L4_CV long
00108 l4re_ds_clear(l4re_ds_t ds, l4re_ds_offset_t offset,
00109              l4re_ds_size_t size) L4_NOTHROW;
00110
00117 L4_CV long
00118 l4re_ds_allocate(l4re_ds_t ds,
00119                 l4re_ds_offset_t offset,
00120                 l4re_ds_size_t size) L4_NOTHROW;
00121
00128 L4_CV int
00129 l4re_ds_copy_in(l4re_ds_t ds, l4re_ds_offset_t dst_offs,
00130                l4re_ds_t src, l4re_ds_offset_t src_offs,
00131                l4re_ds_size_t size) L4_NOTHROW;

```

```

00132
00139 L4_CV l4re_ds_size_t
00140 l4re_ds_size(l4re_ds_t ds) L4_NOTHROW;
00141
00148 L4_CV l4re_ds_flags_t
00149 l4re_ds_flags(l4re_ds_t ds) L4_NOTHROW;
00150
00157 L4_CV int
00158 l4re_ds_info(l4re_ds_t ds, l4re_ds_stats_t *stats) L4_NOTHROW;
00159
00166 L4_CV int
00167 l4re_ds_map_info(l4re_ds_t ds,
00168                  l4_addr_t *start_addr, l4_addr_t *end_addr) L4_NOTHROW;
00169
00170 EXTERN_C_END

```

## 16.258 debug.h

```

00001 /*
00002  * Copyright (C) 2018 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * This file is distributed under the terms of the GNU General Public
00006  * License, version 2. Please see the COPYING-GPL-2 file for details.
00007  */
00008 #pragma once
00009
00010 #include <l4/re/util/debug>
00011
00012 namespace Block_device {
00013
00014 class Err : public L4Re::Util::Err
00015 {
00016 public:
00017     explicit
00018     Err(Level l = Normal) : L4Re::Util::Err(l, "") {}
00019 };
00020
00021 class Dbg : public L4Re::Util::Dbg
00022 {
00023     enum Level
00024     {
00025         Blk_warn = 1,
00026         Blk_info = 2,
00027         Blk_trace = 4,
00028         Blk_steptrace = 8
00029     };
00030
00031 public:
00032     Dbg(unsigned long l = Blk_info, char const *subsys = "")
00033         : L4Re::Util::Dbg(l, "libblock", subsys) {}
00034
00035     static Dbg warn(char const *subsys = "")
00036     { return Dbg(Blk_warn, subsys); }
00037
00038     static Dbg info(char const *subsys = "")
00039     { return Dbg(Blk_info, subsys); }
00040
00041     static Dbg trace(char const *subsys = "")
00042     { return Dbg(Blk_trace, subsys); }
00043
00044     static Dbg steptrace(char const *subsys = "")
00045     { return Dbg(Blk_steptrace, subsys); }
00046 };
00047
00048 } // name space
00049

```

## 16.259 l4/re/c/debug.h File Reference

Debug C interface.



```

00023
00029 #include <l4/sys/types.h>
00030
00031 EXTERN_C_BEGIN
00032
00040 L4_CV long
00041 l4re_debug_obj_debug(l4_cap_idx_t srv, unsigned long function) L4_NOTHROW;
00042
00043 EXTERN_C_END

```

## 16.261 l4/re/c/dma\_space.h File Reference

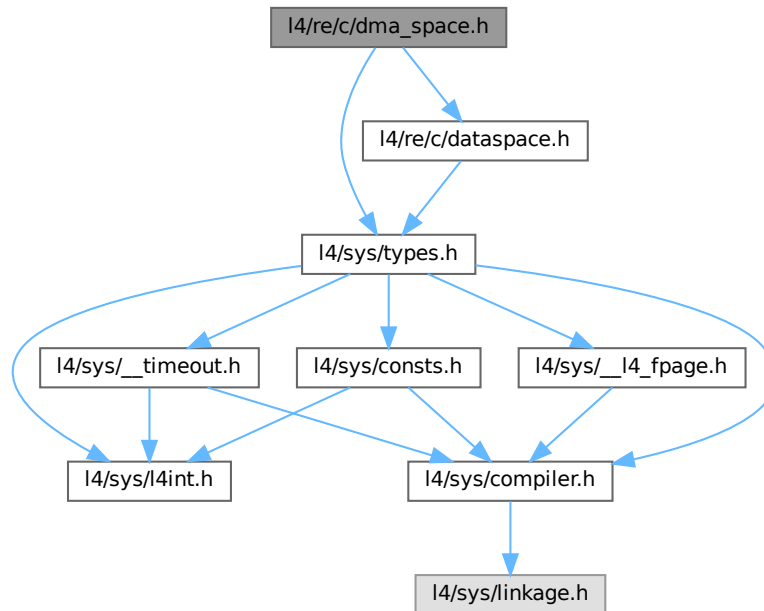
DMA space C interface.

```

#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>

```

Include dependency graph for dma\_space.h:



### Typedefs

- typedef [l4\\_cap\\_idx\\_t](#) [l4re\\_dma\\_space\\_t](#)  
DMA space capability type.
- typedef [l4\\_uint64\\_t](#) [l4re\\_dma\\_space\\_dma\\_addr\\_t](#)  
Data type for DMA addresses.

### Enumerations

- enum [l4re\\_dma\\_space\\_direction](#) { [L4RE\\_DMA\\_SPACE\\_BIDIRECTIONAL](#) , [L4RE\\_DMA\\_SPACE\\_TO\\_DEVICE](#) , [L4RE\\_DMA\\_SPACE\\_FROM\\_DEVICE](#) , [L4RE\\_DMA\\_SPACE\\_NONE](#) }  
Direction of the DMA transfers.
- enum [l4re\\_dma\\_space\\_space\\_attrbs](#) { [L4RE\\_DMA\\_SPACE\\_COHERENT](#) = 1 << 0 , [L4RE\\_DMA\\_SPACE\\_PHYS\\_SPACE](#) = 1 << 1 }  
Attributes assigned to the DMA space when associated with a specific device.

## Functions

- long [l4re\\_dma\\_space\\_map](#) ([l4re\\_dma\\_space\\_t](#) dma, [l4re\\_ds\\_t](#) src, [l4re\\_ds\\_offset\\_t](#) offset, [l4\\_size\\_t](#) \*size, unsigned long attrs, enum [l4re\\_dma\\_space\\_direction](#) dir, [l4re\\_dma\\_space\\_dma\\_addr\\_t](#) \*dma\_addr) [L4\\_NOTHROW](#)

*Map the given part of this data space into the DMA address space.*

- long [l4re\\_dma\\_space\\_unmap](#) ([l4re\\_dma\\_space\\_t](#) dma, [l4re\\_dma\\_space\\_dma\\_addr\\_t](#) dma\_addr, [l4\\_size\\_t](#) size, unsigned long attrs, enum [l4re\\_dma\\_space\\_direction](#) dir) [L4\\_NOTHROW](#)

*Unmap the given part of this data space from the DMA address space.*

- long [l4re\\_dma\\_space\\_associate](#) ([l4re\\_dma\\_space\\_t](#) dma, [l4\\_cap\\_idx\\_t](#) dma\_task, unsigned long attr) [L4\\_NOTHROW](#)

*Associate a (kernel) [DMA space](#) for a device to this [Dma\\_space](#).*

- long [l4re\\_dma\\_space\\_disassociate](#) ([l4re\\_dma\\_space\\_t](#) dma)

*Disassociate the (kernel) [DMA space](#) from this [Dma\\_space](#).*

### 16.261.1 Detailed Description

DMA space C interface.

Definition in file [dma\\_space.h](#).

### 16.261.2 Enumeration Type Documentation

#### 16.261.2.1 [l4re\\_dma\\_space\\_direction](#)

enum [l4re\\_dma\\_space\\_direction](#)

Direction of the DMA transfers.

Enumerator

<a href="#">L4RE_DMA_SPACE_BIDIRECTIONAL</a>	device reads and writes to the memory
<a href="#">L4RE_DMA_SPACE_TO_DEVICE</a>	device reads the memory
<a href="#">L4RE_DMA_SPACE_FROM_DEVICE</a>	device writes to the memory
<a href="#">L4RE_DMA_SPACE_NONE</a>	device is coherently connected

Definition at line 37 of file [dma\\_space.h](#).

#### 16.261.2.2 [l4re\\_dma\\_space\\_space\\_attrbs](#)

enum [l4re\\_dma\\_space\\_space\\_attrbs](#)

Attributes assigned to the DMA space when associated with a specific device.

See also

[Space\\_attrbs](#)



## Enumerator

L4RE_DMA_SPACE_COHERENT	The device is connected coherently with the cache. This means that the map() and unmap() do not need to sync CPU caches before and after DMA.
L4RE_DMA_SPACE_PHYS_SPACE	The DMA space has no DMA task assigned and uses the CPUs physical memory.

Definition at line 48 of file [dma\\_space.h](#).

## 16.262 dma\_space.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00007  *
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU General Public License 2.
00010  * Please see the COPYING-GPL-2 file for details.
00011  *
00012  * As a special exception, you may use this file as part of a free software
00013  * library without restriction. Specifically, if other files instantiate
00014  * templates or use macros or inline functions from this file, or you compile
00015  * this file and link it with other files to produce an executable, this
00016  * file does not by itself cause the resulting executable to be covered by
00017  * the GNU General Public License. This exception does not however
00018  * invalidate any other reasons why the executable file might be covered by
00019  * the GNU General Public License.
00020  */
00021 #pragma once
00022
00029 #include <l4/sys/types.h>
00030 #include <l4/re/c/dataspace.h>
00031
00032 EXTERN_C_BEGIN
00033
00037 enum l4re_dma_space_direction
00038 {
00039     L4RE_DMA_SPACE_BIDIRECTIONAL,
00040     L4RE_DMA_SPACE_TO_DEVICE,
00041     L4RE_DMA_SPACE_FROM_DEVICE,
00042     L4RE_DMA_SPACE_NONE
00043 };
00044
00048 enum l4re_dma_space_space_attrbs
00049 {
00050     L4RE_DMA_SPACE_COHERENT = 1 << 0,
00051     L4RE_DMA_SPACE_PHYS_SPACE = 1 << 1,
00052 };
00053
00059 typedef l4_cap_idx_t l4re_dma_space_t;
00060
00062 typedef l4_uint64_t l4re_dma_space_dma_addr_t;
00063
00070 L4_CV long
00071 l4re_dma_space_map(l4re_dma_space_t dma, l4re_ds_t src,
00072                   l4re_ds_offset_t offset,
00073                   l4_size_t * size, unsigned long attrs,
00074                   enum l4re_dma_space_direction dir,
00075                   l4re_dma_space_dma_addr_t *dma_addr) L4_NOTHROW;
00076
00077
00084 L4_CV long
00085 l4re_dma_space_unmap(l4re_dma_space_t dma, l4re_dma_space_dma_addr_t dma_addr,
00086                    l4_size_t size, unsigned long attrs,
00087                    enum l4re_dma_space_direction dir) L4_NOTHROW;
00088
00095 L4_CV long
00096 l4re_dma_space_associate(l4re_dma_space_t dma, l4_cap_idx_t dma_task,
00097                        unsigned long attr) L4_NOTHROW;
00098
00105 L4_CV long
00106 l4re_dma_space_disassociate(l4re_dma_space_t dma);
00107
00108
00109 EXTERN_C_END

```

## 16.263 event\_buffer.h

```

00001 #pragma once
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019
00020 #include <l4/sys/linkage.h>
00021 #include <l4/re/c/event.h>
00022
00023 EXTERN_C_BEGIN
00024
00025 typedef struct l4re_event_buffer_consumer_t
00026 {
00027     unsigned long _obj_buf[8];
00028 } l4re_event_buffer_consumer_t;
00029
00030 L4_CV void
00031 l4re_event_free(l4re_event_t *e) L4_NOTHROW;
00032
00033 L4_CV long
00034 l4re_event_buffer_attach(l4re_event_buffer_consumer_t *evbuf,
00035                          l4re_ds_t ds, l4_cap_idx_t rm) L4_NOTHROW;
00036
00037 L4_CV long
00038 l4re_event_buffer_detach(l4re_event_buffer_consumer_t *evbuf,
00039                          l4_cap_idx_t rm) L4_NOTHROW;
00040
00041 L4_CV l4re_event_t *
00042 l4re_event_buffer_next(l4re_event_buffer_consumer_t *evbuf) L4_NOTHROW;
00043
00044 typedef L4_CV void l4re_event_buffer_cb_t(l4re_event_t *ev, void *data);
00045
00046 L4_CV void
00047 l4re_event_buffer_consumer_foreach_available_event(l4re_event_buffer_consumer_t *evbuf,
00048                                                    void *data, l4re_event_buffer_cb_t *cb);
00049
00050
00051 L4_CV void
00052 l4re_event_buffer_consumer_process(l4re_event_buffer_consumer_t *evbuf,
00053                                   l4_cap_idx_t irq, l4_cap_idx_t thread, void *data,
00054                                   l4re_event_buffer_cb_t *cb);
00055
00056 EXTERN_C_END

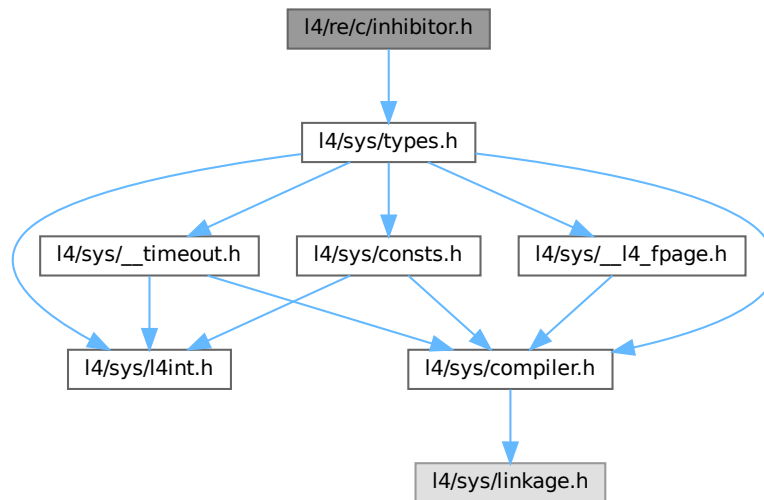
```

## 16.264 l4/re/c/inhibitor.h File Reference

Inhibitor C interface.

```
#include <l4/sys/types.h>
```

Include dependency graph for inhibitor.h:



## Functions

- long `l4re_inhibitor_acquire` (`l4_cap_idx_t` cap, `l4_umword_t` id, char const \*reason)  
*Acquire an inhibitor lock.*
- long `l4re_inhibitor_release` (`l4_cap_idx_t` cap, `l4_umword_t` id)  
*Release an inhibitor lock.*
- long `l4re_inhibitor_next_lock_info` (`l4_cap_idx_t` cap, char \*name, unsigned len, `l4_mword_t` current\_id)  
*Get information for the next available inhibitor lock.*

### 16.264.1 Detailed Description

Inhibitor C interface.

Definition in file [inhibitor.h](#).

### 16.264.2 Function Documentation

#### 16.264.2.1 l4re\_inhibitor\_acquire()

```
long l4re_inhibitor_acquire (
    l4_cap_idx_t cap,
    l4_umword_t id,
    char const * reason )
```

Acquire an inhibitor lock.

## Parameters

<i>cap</i>	Capability for the Inhibitor object (see <a href="#">L4Re::Inhibitor</a> )
<i>id</i>	ID of the inhibitor lock that shall be acquired.
<i>reason</i>	Reason why the inhibitor lock is acquired. (Used for informing the user or debugging.)

## Returns

0 for success, <0 on error.

## See also

[L4Re::Inhibitor::acquire\(\)](#).

**16.264.2.2 l4re\_inhibitor\_next\_lock\_info()**

```
long l4re_inhibitor_next_lock_info (
    l4_cap_idx_t cap,
    char * name,
    unsigned len,
    l4_mword_t current_id )
```

Get information for the next available inhibitor lock.

## Parameters

<i>cap</i>	Capability to the Inhibitor object (see <a href="#">L4Re::Inhibitor</a> )
<i>name</i>	A pointer to a buffer for the name of the lock.
<i>len</i>	The length of the available buffer (usually <a href="#">L4Re::Inhibitor::Name_max</a> is used).
<i>current_id</i>	The ID of the last available lock, use -1 to get the first lock.

## Return values

>0	The ID of the next available lock if there is one (in this case name shall contain the name of the inhibitor lock).
-L4_ENODEV	if there are no more locks.
<0	Any other negative failure value.

## See also

[L4Re::Inhibitor::next\\_lock\\_info\(\)](#).

**16.264.2.3 l4re\_inhibitor\_release()**

```
long l4re_inhibitor_release (
    l4_cap_idx_t cap,
    l4_umword_t id )
```

Release an inhibitor lock.

## Parameters

<i>cap</i>	Capability for the Inhibitor object (see <a href="#">L4Re::Inhibitor</a> ).
<i>id</i>	ID of inhibitor that shall be released.

## Returns

0 for success, <0 on error.

## See also

[L4Re::Inhibitor::release\(\)](#).

## 16.265 inhibitor.h

[Go to the documentation of this file.](#)

```

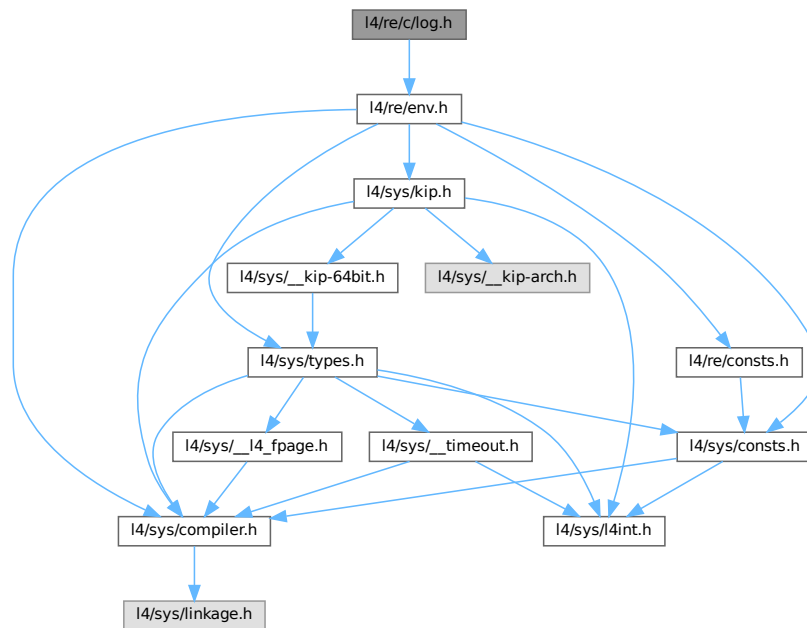
00001 /*
00002  * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00003  *
00004  * This file is licensed under the terms of the GNU Lesser General
00005  * Public License 2.1.
00006  * See the file COPYING-LGPL-2.1 for details.
00007  */
00008 #pragma once
00009
00015 #include <l4/sys/types.h>
00016
00017 EXTERN_C_BEGIN
00018
00029 L4_CV long L4_EXPORT
00030 l4re_inhibitor_acquire(l4_cap_idx_t cap, l4_umword_t id,
00031                      char const *reason);
00032
00040 L4_CV long L4_EXPORT
00041 l4re_inhibitor_release(l4_cap_idx_t cap, l4_umword_t id);
00042
00058 L4_CV long L4_EXPORT
00059 l4re_inhibitor_next_lock_info(l4_cap_idx_t cap, char *name,
00060                              unsigned len, l4_mword_t current_id);
00061
00062 EXTERN_C_END
00063

```

## 16.266 l4/re/c/log.h File Reference

Log C interface.

```
#include <l4/re/env.h>
Include dependency graph for log.h:
```



## Functions

- void [l4re\\_log\\_print](#) (char const \*string) [L4\\_NOTHROW](#)  
Write a null terminated string to the default log.
- void [l4re\\_log\\_printn](#) (char const \*string, int len) [L4\\_NOTHROW](#)  
Write a string of a given length to the default log.
- void [l4re\\_log\\_print\\_srv](#) (const [l4\\_cap\\_idx\\_t](#) logcap, char const \*string) [L4\\_NOTHROW](#)  
Write a null terminated string to a log.
- void [l4re\\_log\\_printn\\_srv](#) (const [l4\\_cap\\_idx\\_t](#) logcap, char const \*string, int len) [L4\\_NOTHROW](#)  
Write a string of a given length to a log.

### 16.266.1 Detailed Description

Log C interface.

Definition in file [log.h](#).

## 16.267 log.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *

```

```

00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #pragma once
00023
00030 #include <l4/re/env.h>
00031
00032 EXTERN_C_BEGIN
00033
00042 L4_CV L4_INLINE void
00043 l4re_log_print(char const *string) L4_NOTHROW;
00044
00054 L4_CV L4_INLINE void
00055 l4re_log_printn(char const *string, int len) L4_NOTHROW;
00056
00057
00058
00059
00069 L4_CV void
00070 l4re_log_print_srv(const l4_cap_idx_t logcap,
00071                  char const *string) L4_NOTHROW;
00072
00083 L4_CV void
00084 l4re_log_printn_srv(const l4_cap_idx_t logcap,
00085                   char const *string, int len) L4_NOTHROW;
00086
00087
00088 /***** Implementations *****/
00089
00090 L4_CV L4_INLINE void
00091 l4re_log_print(char const *string) L4_NOTHROW
00092 {
00093     l4re_log_print_srv(l4re_global_env->log, string);
00094 }
00095
00096 L4_CV L4_INLINE void
00097 l4re_log_printn(char const *string, int len) L4_NOTHROW
00098 {
00099     l4re_log_printn_srv(l4re_global_env->log, string, len);
00100 }
00101
00102 EXTERN_C_END

```

## 16.268 l4/re/c/mem\_alloc.h File Reference

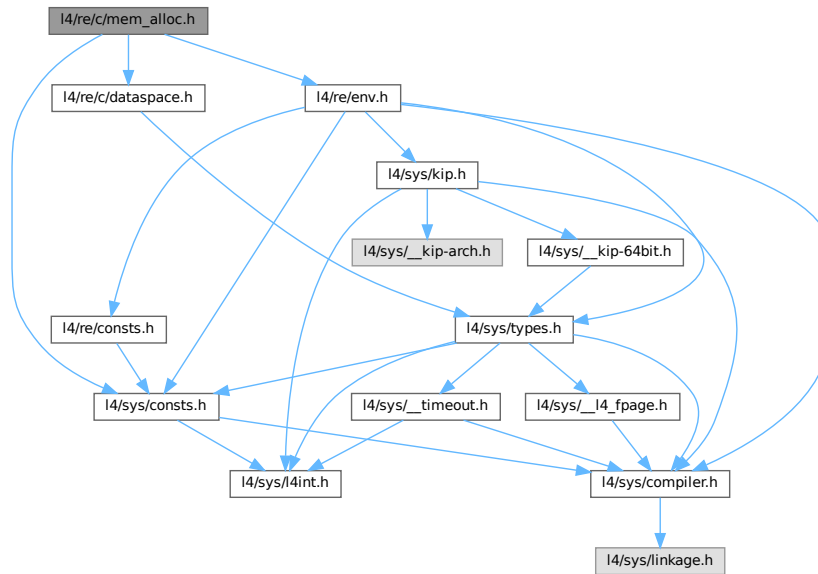
Memory allocator C interface.

```

#include <l4/re/env.h>
#include <l4/sys/consts.h>
#include <l4/re/c/dataspace.h>

```

Include dependency graph for `mem_alloc.h`:



## Enumerations

- enum [l4re\\_ma\\_flags](#)  
*Flags for requesting memory at the memory allocator.*

## Functions

- long [l4re\\_ma\\_alloc](#) (long size, [l4re\\_ds\\_t](#) const mem, unsigned long flags) [L4\\_NOTHROW](#)  
*Allocate memory.*
- long [l4re\\_ma\\_alloc\\_align](#) (long size, [l4re\\_ds\\_t](#) const mem, unsigned long flags, unsigned long align) [L4\\_NOTHROW](#)  
*Allocate memory.*
- long [l4re\\_ma\\_alloc\\_align\\_srv](#) ([l4\\_cap\\_idx\\_t](#) srv, long size, [l4re\\_ds\\_t](#) const mem, unsigned long flags, unsigned long align) [L4\\_NOTHROW](#)  
*Allocate memory.*

### 16.268.1 Detailed Description

Memory allocator C interface.

Definition in file [mem\\_alloc.h](#).



## 16.269 mem\_alloc.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #pragma once
00023
00024 #include <l4/re/env.h>
00025 #include <l4/sys/consts.h>
00026
00027 #include <l4/re/c/dataspace.h>
00028
00035 EXTERN_C_BEGIN
00036
00042 enum l4re_ma_flags {
00043     L4RE_MA_CONTINUOUS = 0x01,
00044     L4RE_MA_PINNED     = 0x02,
00045     L4RE_MA_SUPER_PAGES = 0x04,
00046 };
00047
00048
00079 L4_CV L4_INLINE long
00080 l4re_ma_alloc(long size, l4re_ds_t const mem,
00081               unsigned long flags) L4_NOTHROW;
00082
00116 L4_CV L4_INLINE long
00117 l4re_ma_alloc_align(long size, l4re_ds_t const mem,
00118                     unsigned long flags, unsigned long align) L4_NOTHROW;
00119
00138 L4_CV long
00139 l4re_ma_alloc_align_srv(l4_cap_idx_t srv, long size,
00140                         l4re_ds_t const mem, unsigned long flags,
00141                         unsigned long align) L4_NOTHROW;
00142
00143 /***** Implementation *****/
00144
00145 L4_CV L4_INLINE long
00146 l4re_ma_alloc(long size, l4re_ds_t const mem,
00147               unsigned long flags) L4_NOTHROW
00148 {
00149     return l4re_ma_alloc_align_srv(l4re_global_env->mem_alloc, size, mem,
00150                                    flags, 0);
00151 }
00152
00153 L4_CV L4_INLINE long
00154 l4re_ma_alloc_align(long size, l4re_ds_t const mem,
00155                     unsigned long flags, unsigned long align) L4_NOTHROW
00156 {
00157     return l4re_ma_alloc_align_srv(l4re_global_env->mem_alloc, size, mem,
00158                                    flags, align);
00159 }
00160
00161 EXTERN_C_END

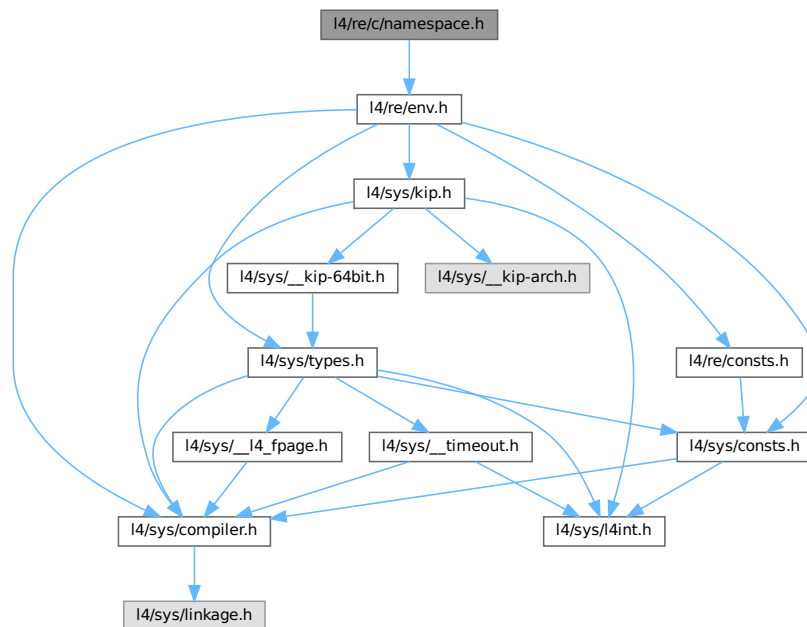
```

## 16.270 l4/re/c/namespace.h File Reference

Namespace functions, C interface.

```
#include <l4/re/env.h>
```

Include dependency graph for namespace.h:



## Typedefs

- typedef [l4\\_cap\\_idx\\_t](#) [l4re\\_namespace\\_t](#)  
*Namespace type.*

## Enumerations

- enum [l4re\\_ns\\_register\\_flags](#)  
*Namespace register flags.*

## Functions

- long [l4re\\_ns\\_query\\_to\\_srv](#) ([l4re\\_namespace\\_t](#) srv, char const \*name, [l4\\_cap\\_idx\\_t](#) const cap, int timeout) [L4\\_NOTHROW](#)  
*Query the name space for the object named by name.*
- long [l4re\\_ns\\_query\\_srv](#) ([l4re\\_namespace\\_t](#) srv, char const \*name, [l4\\_cap\\_idx\\_t](#) const cap) [L4\\_NOTHROW](#)  
*Query the name space for the object named by name.*
- long [l4re\\_ns\\_register\\_obj\\_srv](#) ([l4re\\_namespace\\_t](#) srv, char const \*name, [l4\\_cap\\_idx\\_t](#) const obj, unsigned flags) [L4\\_NOTHROW](#)  
*Register an object with a name.*

### 16.270.1 Detailed Description

Namespace functions, C interface.

Definition in file [namespace.h](#).

## 16.271 namespace.h

[Go to the documentation of this file.](#)

```

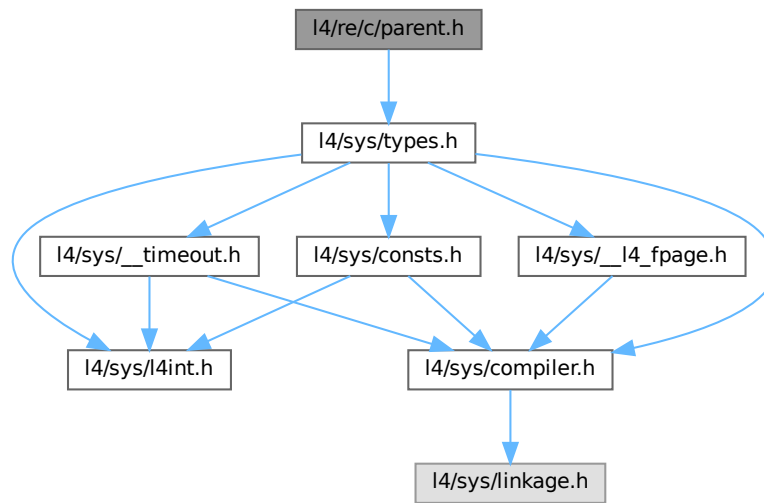
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00031 #include <l4/re/env.h>
00032
00039 enum l4re_ns_register_flags {
00040     L4RE_NS_REGISTER_RO = L4_FPAGE_RO,
00041     L4RE_NS_REGISTER_DIR = 0x10,
00042     L4RE_NS_REGISTER_RW = L4_FPAGE_RX,
00043     L4RE_NS_REGISTER_RWS = L4_FPAGE_RWX,
00044     L4RE_NS_REGISTER_S = L4_FPAGE_W,
00045 };
00046
00047 EXTERN_C_BEGIN
00048
00053 typedef l4_cap_idx_t l4re_namespace_t;
00054
00055
00056
00065 L4_CV long
00066 l4re_ns_query_to_srv(l4re_namespace_t srv, char const *name,
00067                     l4_cap_idx_t const cap, int timeout) L4_NOTHROW;
00068
00085 L4_CV L4_INLINE long
00086 l4re_ns_query_srv(l4re_namespace_t srv, char const *name,
00087                  l4_cap_idx_t const cap) L4_NOTHROW;
00088
00096 L4_CV long
00097 l4re_ns_register_obj_srv(l4re_namespace_t srv, char const *name,
00098                         l4_cap_idx_t const obj, unsigned flags) L4_NOTHROW;
00099
00100
00101
00102 /***** Implementation *****/
00103
00104 L4_CV L4_INLINE long
00105 l4re_ns_query_srv(l4re_namespace_t srv, char const *name,
00106                  l4_cap_idx_t const cap) L4_NOTHROW
00107 {
00108     return l4re_ns_query_to_srv(srv, name, cap, 40000);
00109 }
00110
00111
00112 EXTERN_C_END

```

## 16.272 l4/re/c/parent.h File Reference

Parent C interface.

```
#include <l4/sys/types.h>
Include dependency graph for parent.h:
```



## Functions

- long [l4re\\_parent\\_signal](#) ([l4\\_cap\\_idx\\_t](#) parent, unsigned long sig, unsigned long val)  
*Send a signal using the parent protocol.*

## 16.272.1 Detailed Description

Parent C interface.

Definition in file [parent.h](#).

## 16.272.2 Function Documentation

### 16.272.2.1 l4re\_parent\_signal()

```
long l4re_parent_signal (
    l4\_cap\_idx\_t parent,
    unsigned long sig,
    unsigned long val )
```

Send a signal using the parent protocol.

#### Parameters

<i>parent</i>	Capability index of parent to send signal to.
<i>sig</i>	Signal to send
<i>val</i>	Value of the signal

## Return values

0	Success
<0	IPC error

## See also

[L4Re::Parent::signal](#)

## 16.273 parent.h

[Go to the documentation of this file.](#)

```

00001  /*
00002   * Copyright (C) 2024 Kernkonzept GmbH.
00003   * Author(s): Marcus Haehnel <marcus.haehnel@kernkonzept.com>
00004   *
00005   * License: see LICENSE.spdx (in this directory or the directories above)
00006   */
00007
00013 #pragma once
00014
00020 #include <l4/sys/types.h>
00021
00022 EXTERN_C_BEGIN
00023
00036 L4_CV long L4_EXPORT
00037 l4re_parent_signal(l4_cap_idx_t parent, unsigned long sig, unsigned long val);
00038
00039 EXTERN_C_END

```

## 16.274 l4/re/c/rm.h File Reference

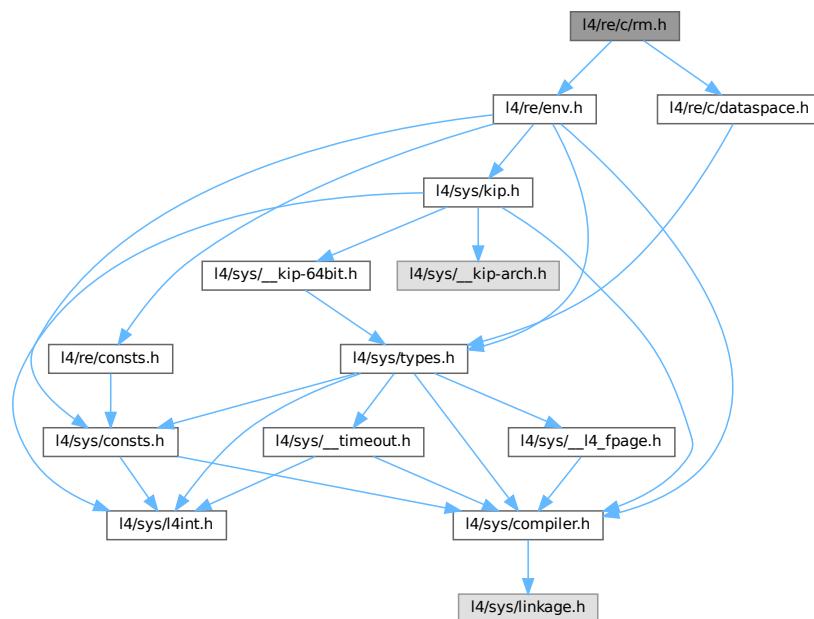
Region map interface, C interface.

```

#include <l4/re/env.h>
#include <l4/re/c/dataspace.h>

```

Include dependency graph for rm.h:



## Enumerations

- enum [l4re\\_rm\\_flags\\_values](#) {  
[L4RE\\_RM\\_F\\_R](#) = [L4RE\\_DS\\_F\\_R](#) , [L4RE\\_RM\\_F\\_W](#) = [L4RE\\_DS\\_F\\_W](#) , [L4RE\\_RM\\_F\\_X](#) = [L4RE\\_DS\\_F\\_X](#)  
, [L4RE\\_RM\\_F\\_RX](#) = [L4RE\\_DS\\_F\\_RX](#) ,  
[L4RE\\_RM\\_F\\_RW](#) = [L4RE\\_DS\\_F\\_RW](#) , [L4RE\\_RM\\_F\\_RWX](#) = [L4RE\\_DS\\_F\\_RWX](#) , [L4RE\\_RM\\_F\\_NO\\_ALIAS](#)  
= 0x200 , [L4RE\\_RM\\_F\\_PAGER](#) = 0x400 ,  
[L4RE\\_RM\\_F\\_RESERVED](#) = 0x800 , [L4RE\\_RM\\_CACHING\\_SHIFT](#) = 4 , [L4RE\\_RM\\_F\\_CACHING](#) = [L4RE\\_DS\\_F\\_CACHING\\_MASK](#) , [L4RE\\_RM\\_REGION\\_FLAGS](#) = 0xffff ,  
[L4RE\\_RM\\_F\\_CACHE\\_NORMAL](#) = [L4RE\\_DS\\_F\\_NORMAL](#) , [L4RE\\_RM\\_F\\_CACHE\\_BUFFERED](#) = [L4RE\\_DS\\_F\\_BUFFERABLE](#) , [L4RE\\_RM\\_F\\_CACHE\\_UNCACHED](#) = [L4RE\\_DS\\_F\\_UNCACHEABLE](#) ,  
[L4RE\\_RM\\_F\\_SEARCH\\_ADDR](#) = 0x020000 ,  
[L4RE\\_RM\\_F\\_IN\\_AREA](#) = 0x040000 , [L4RE\\_RM\\_F\\_EAGER\\_MAP](#) = 0x080000 , [L4RE\\_RM\\_F\\_NO\\_EAGER\\_MAP](#)  
= 0x100000 , [L4RE\\_RM\\_F\\_ATTACH\\_FLAGS](#) = 0x1f0000 }

*Flags for region operations.*

## Functions

- int [l4re\\_rm\\_reserve\\_area](#) ([l4\\_addr\\_t](#) \*start, unsigned long size, [l4re\\_rm\\_flags\\_t](#) flags, unsigned char align) [L4\\_NOTHROW](#)  
*Reserve the given area in the region map.*
- int [l4re\\_rm\\_free\\_area](#) ([l4\\_addr\\_t](#) addr) [L4\\_NOTHROW](#)  
*Free an area from the region map.*
- int [l4re\\_rm\\_attach](#) (void \*\*start, unsigned long size, [l4re\\_rm\\_flags\\_t](#) flags, [l4re\\_ds\\_t](#) mem, [l4re\\_rm\\_offset\\_t](#) offs, unsigned char align) [L4\\_NOTHROW](#)  
*Attach a data space to a region.*
- int [l4re\\_rm\\_detach](#) (void \*addr) [L4\\_NOTHROW](#)  
*Detach and unmap a region from the address space in the current task.*
- int [l4re\\_rm\\_detach\\_ds](#) (void \*addr, [l4re\\_ds\\_t](#) \*ds) [L4\\_NOTHROW](#)  
*Detach and unmap a region and return affected dataspace in the current task.*
- int [l4re\\_rm\\_detach\\_unmap](#) ([l4\\_addr\\_t](#) addr, [l4\\_cap\\_idx\\_t](#) task) [L4\\_NOTHROW](#)  
*Detach and unmap in specified task.*
- int [l4re\\_rm\\_detach\\_ds\\_unmap](#) (void \*addr, [l4re\\_ds\\_t](#) \*ds, [l4\\_cap\\_idx\\_t](#) task) [L4\\_NOTHROW](#)  
*Detach and unmap in specified task.*
- int [l4re\\_rm\\_find](#) ([l4\\_addr\\_t](#) \*addr, unsigned long \*size, [l4re\\_rm\\_offset\\_t](#) \*offset, [l4re\\_rm\\_flags\\_t](#) \*flags, [l4re\\_ds\\_t](#) \*m) [L4\\_NOTHROW](#)  
*Find a region given an address and size.*
- void [l4re\\_rm\\_show\\_lists](#) (void) [L4\\_NOTHROW](#)  
*Dump region map internal data structures.*
- int [l4re\\_rm\\_reserve\\_area\\_srv](#) ([l4\\_cap\\_idx\\_t](#) rm, [l4\\_addr\\_t](#) \*start, unsigned long size, [l4re\\_rm\\_flags\\_t](#) flags, unsigned char align) [L4\\_NOTHROW](#)
- int [l4re\\_rm\\_free\\_area\\_srv](#) ([l4\\_cap\\_idx\\_t](#) rm, [l4\\_addr\\_t](#) addr) [L4\\_NOTHROW](#)
- int [l4re\\_rm\\_attach\\_srv](#) ([l4\\_cap\\_idx\\_t](#) rm, void \*\*start, unsigned long size, [l4re\\_rm\\_flags\\_t](#) flags, [l4re\\_ds\\_t](#) mem, [l4re\\_rm\\_offset\\_t](#) offs, unsigned char align) [L4\\_NOTHROW](#)
- int [l4re\\_rm\\_detach\\_srv](#) ([l4\\_cap\\_idx\\_t](#) rm, [l4\\_addr\\_t](#) addr, [l4re\\_ds\\_t](#) \*ds, [l4\\_cap\\_idx\\_t](#) task) [L4\\_NOTHROW](#)
- int [l4re\\_rm\\_find\\_srv](#) ([l4\\_cap\\_idx\\_t](#) rm, [l4\\_addr\\_t](#) \*addr, unsigned long \*size, [l4re\\_rm\\_offset\\_t](#) \*offset, [l4re\\_rm\\_flags\\_t](#) \*flags, [l4re\\_ds\\_t](#) \*m) [L4\\_NOTHROW](#)
- void [l4re\\_rm\\_show\\_lists\\_srv](#) ([l4\\_cap\\_idx\\_t](#) rm) [L4\\_NOTHROW](#)  
*Dump region map internal data structures.*

### 16.274.1 Detailed Description

Region map interface, C interface.

Definition in file [rm.h](#).

## 16.275 rm.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00031 #include <l4/re/env.h>
00032 #include <l4/re/c/dataspace.h>
00033
00034 EXTERN_C_BEGIN
00035
00040 enum l4re_rm_flags_values {
00041     L4RE_RM_F_R      = L4RE_DS_F_R,
00042     L4RE_RM_F_W      = L4RE_DS_F_W,
00043     L4RE_RM_F_X      = L4RE_DS_F_X,
00044     L4RE_RM_F_RX     = L4RE_DS_F_RX,
00045     L4RE_RM_F_RW     = L4RE_DS_F_RW,
00046     L4RE_RM_F_RWX    = L4RE_DS_F_RWX,
00047
00048     L4RE_RM_F_NO_ALIAS    = 0x200,
00049     L4RE_RM_F_PAGER      = 0x400,
00050     L4RE_RM_F_RESERVED   = 0x800,
00052     L4RE_RM_CACHING_SHIFT = 4,
00055     L4RE_RM_F_CACHING     = L4RE_DS_F_CACHING_MASK,
00056
00057     L4RE_RM_REGION_FLAGS = 0xffff,
00060     L4RE_RM_F_CACHE_NORMAL    = L4RE_DS_F_NORMAL,
00061
00063     L4RE_RM_F_CACHE_BUFFERED = L4RE_DS_F_BUFFERABLE,
00064
00066     L4RE_RM_F_CACHE_UNCACHED = L4RE_DS_F_UNCACHEABLE,
00067
00068     L4RE_RM_F_SEARCH_ADDR    = 0x020000,
00069     L4RE_RM_F_IN_AREA       = 0x040000,
00070     L4RE_RM_F_EAGER_MAP     = 0x080000,
00071     L4RE_RM_F_NO_EAGER_MAP  = 0x100000,
00072     L4RE_RM_F_ATTACH_FLAGS  = 0x1f0000,
00073 };
00074
00075 typedef l4_uint32_t l4re_rm_flags_t;
00076 typedef l4_uint64_t l4re_rm_offset_t;
00077
00086 L4_CV L4_INLINE int
00087 l4re_rm_reserve_area(l4_addr_t *start, unsigned long size,
00088                     l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW;
00089
00098 L4_CV L4_INLINE int
00099 l4re_rm_free_area(l4_addr_t addr) L4_NOTHROW;
00100
00109 L4_CV L4_INLINE int
00110 l4re_rm_attach(void **start, unsigned long size, l4re_rm_flags_t flags,
00111               l4re_ds_t mem,
00112               l4re_rm_offset_t offs,
00113               unsigned char align) L4_NOTHROW;
00114
00115
00133 L4_CV L4_INLINE int
00134 l4re_rm_detach(void *addr) L4_NOTHROW;
00135
00155 L4_CV L4_INLINE int
00156 l4re_rm_detach_ds(void *addr, l4re_ds_t *ds) L4_NOTHROW;
00157
00169 L4_CV L4_INLINE int
00170 l4re_rm_detach_unmap(l4_addr_t addr, l4_cap_idx_t task) L4_NOTHROW;
00171
00186 L4_CV L4_INLINE int
00187 l4re_rm_detach_ds_unmap(void *addr, l4re_ds_t *ds,

```

```

00188         l4_cap_idx_t task) L4_NOTHROW;
00189
00190
00197 L4_CV L4_INLINE int
00198 l4re_rm_find(l4_addr_t *addr, unsigned long *size,
00199             l4re_rm_offset_t *offset,
00200             l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW;
00201
00208 L4_CV L4_INLINE void
00209 l4re_rm_show_lists(void) L4_NOTHROW;
00210
00211
00212 /*
00213  * Variants of functions that also take a capability of the region map
00214  * service.
00215  */
00216
00217
00222 L4_CV int
00223 l4re_rm_reserve_area_srv(l4_cap_idx_t rm, l4_addr_t *start, unsigned long size,
00224                         l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW;
00225
00230 L4_CV int
00231 l4re_rm_free_area_srv(l4_cap_idx_t rm, l4_addr_t addr) L4_NOTHROW;
00232
00237 L4_CV int
00238 l4re_rm_attach_srv(l4_cap_idx_t rm, void **start, unsigned long size,
00239                  l4re_rm_flags_t flags, l4re_ds_t mem,
00240                  l4re_rm_offset_t offs,
00241                  unsigned char align) L4_NOTHROW;
00242
00243
00248 L4_CV int
00249 l4re_rm_detach_srv(l4_cap_idx_t rm, l4_addr_t addr,
00250                  l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW;
00251
00252
00257 L4_CV int
00258 l4re_rm_find_srv(l4_cap_idx_t rm, l4_addr_t *addr,
00259                unsigned long *size, l4re_rm_offset_t *offset,
00260                l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW;
00261
00266 L4_CV void
00267 l4re_rm_show_lists_srv(l4_cap_idx_t rm) L4_NOTHROW;
00268
00269
00270 /***** Implementations *****/
00271
00272 L4_CV L4_INLINE int
00273 l4re_rm_reserve_area(l4_addr_t *start, unsigned long size,
00274                    l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW
00275 {
00276     return l4re_rm_reserve_area_srv(l4re_global_env->rm, start, size,
00277                                     flags, align);
00278 }
00279
00280 L4_CV L4_INLINE int
00281 l4re_rm_free_area(l4_addr_t addr) L4_NOTHROW
00282 {
00283     return l4re_rm_free_area_srv(l4re_global_env->rm, addr);
00284 }
00285
00286 L4_CV L4_INLINE int
00287 l4re_rm_attach(void **start, unsigned long size, l4re_rm_flags_t flags,
00288               l4re_ds_t mem, l4re_rm_offset_t offs,
00289               unsigned char align) L4_NOTHROW
00290 {
00291     return l4re_rm_attach_srv(l4re_global_env->rm, start, size,
00292                              flags, mem, offs, align);
00293 }
00294
00295
00296 L4_CV L4_INLINE int
00297 l4re_rm_detach(void *addr) L4_NOTHROW
00298 {
00299     return l4re_rm_detach_srv(l4re_global_env->rm,
00300                             (l4_addr_t)addr, 0, L4_BASE_TASK_CAP);
00301 }
00302
00303 L4_CV L4_INLINE int
00304 l4re_rm_detach_unmap(l4_addr_t addr, l4_cap_idx_t task) L4_NOTHROW
00305 {
00306     return l4re_rm_detach_srv(l4re_global_env->rm, addr, 0, task);
00307 }
00308
00309 L4_CV L4_INLINE int
00310 l4re_rm_detach_ds(void *addr, l4re_ds_t *ds) L4_NOTHROW

```



```

00311 {
00312     return l4re_rm_detach_srv(l4re_global_env->rm, (l4_addr_t)addr,
00313                             ds, L4_BASE_TASK_CAP);
00314 }
00315
00316 L4_CV L4_INLINE int
00317 l4re_rm_detach_ds_unmap(void *addr, l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW
00318 {
00319     return l4re_rm_detach_srv(l4re_global_env->rm, (l4_addr_t)addr,
00320                             ds, task);
00321 }
00322
00323 L4_CV L4_INLINE int
00324 l4re_rm_find(l4_addr_t *addr, unsigned long *size,
00325             l4re_rm_offset_t *offset,
00326             l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW
00327 {
00328     return l4re_rm_find_srv(l4re_global_env->rm, addr, size, offset, flags, m);
00329 }
00330
00331 L4_CV L4_INLINE void
00332 l4re_rm_show_lists(void) L4_NOTHROW
00333 {
00334     l4re_rm_show_lists_srv(l4re_global_env->rm);
00335 }
00336
00337 EXTERN_C_END

```

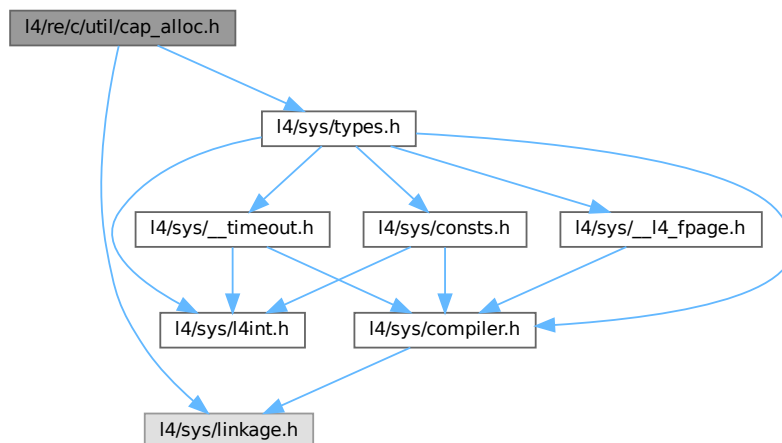
## 16.276 l4/re/c/util/cap\_alloc.h File Reference

Capability allocator C interface.

```
#include <l4/sys/types.h>
```

```
#include <l4/sys/linkage.h>
```

Include dependency graph for cap\_alloc.h:



### Functions

- `l4_cap_idx_t l4re_util_cap_alloc (void) L4_NOTHROW`  
Get free capability index at capability allocator.
- `void l4re_util_cap_free (l4_cap_idx_t cap) L4_NOTHROW`  
Return capability index to capability allocator.

- void **l4re\_util\_cap\_free\_um** (**l4\_cap\_idx\_t** cap) **L4\_NOTHROW**  
*Return capability index to capability allocator, and unmaps the object.*
- long **l4re\_util\_cap\_last** (void) **L4\_NOTHROW**  
*Return last capability index the allocator can return.*

## 16.276.1 Detailed Description

Capability allocator C interface.

Definition in file [cap\\_alloc.h](#).

## 16.277 cap\_alloc.h

[Go to the documentation of this file.](#)

```

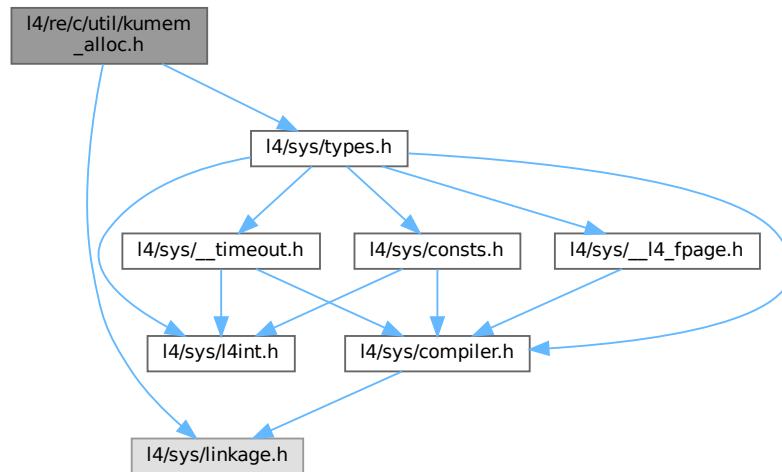
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *                      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00031 #include <l4/sys/types.h>
00032 #include <l4/sys/linkage.h>
00033
00034 EXTERN_C_BEGIN
00035
00040 L4_CV l4_cap_idx_t
00041 l4re_util_cap_alloc(void) L4_NOTHROW;
00042
00047 L4_CV void
00048 l4re_util_cap_free(l4_cap_idx_t cap) L4_NOTHROW;
00049
00055 L4_CV void
00056 l4re_util_cap_free_um(l4_cap_idx_t cap) L4_NOTHROW;
00057
00063 L4_CV long
00064 l4re_util_cap_last(void) L4_NOTHROW;
00065
00066 EXTERN_C_END

```

## 16.278 l4re/c/util/kumem\_alloc.h File Reference

Kumem allocator utility C interface.

```
#include <l4/sys/types.h>
#include <l4/sys/linkage.h>
Include dependency graph for kumem_alloc.h:
```



## Functions

- int [l4re\\_util\\_kumem\\_alloc](#) ([l4\\_addr\\_t](#) \*mem, unsigned pages\_order, [l4\\_cap\\_idx\\_t](#) task, [l4\\_cap\\_idx\\_t](#) rm)  
[L4\\_NOTHROW](#)  
*Allocate state area.*

## 16.278.1 Detailed Description

Kumem allocator utility C interface.

Definition in file [kumem\\_alloc.h](#).

## 16.278.2 Function Documentation

### 16.278.2.1 l4re\_util\_kumem\_alloc()

```
int l4re_util_kumem_alloc (
    l4\_addr\_t * mem,
    unsigned pages_order,
    l4\_cap\_idx\_t task,
    l4\_cap\_idx\_t rm )
```

Allocate state area.

#### Parameters

out	<i>mem</i>	Pointer to memory that has been allocated.
	<i>pages_order</i>	Size to allocate, in log2 pages.
Generated for L4 by Doxygen		Task to use for allocation.
	<i>rm</i>	Region manager to use for allocation.

## Return values

0	for success
<0	error code on failure

## Note

The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page. A portable implementation should not depend on allocations greater than 16KiB to succeed.

## Examples

[examples/sys/aliens/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

## 16.279 kumem\_alloc.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00031 #include <l4/sys/types.h>
00032 #include <l4/sys/linkage.h>
00033
00034 EXTERN_C_BEGIN
00035
00039 L4_CV int
00040 l4re_util_kumem_alloc(l4_addr_t *mem, unsigned pages_order,
00041                      l4_cap_idx_t task, l4_cap_idx_t rm) L4_NOTHROW;
00042
00043 EXTERN_C_END

```

## 16.280 l4/re/c/util/video/goos\_fb.h File Reference

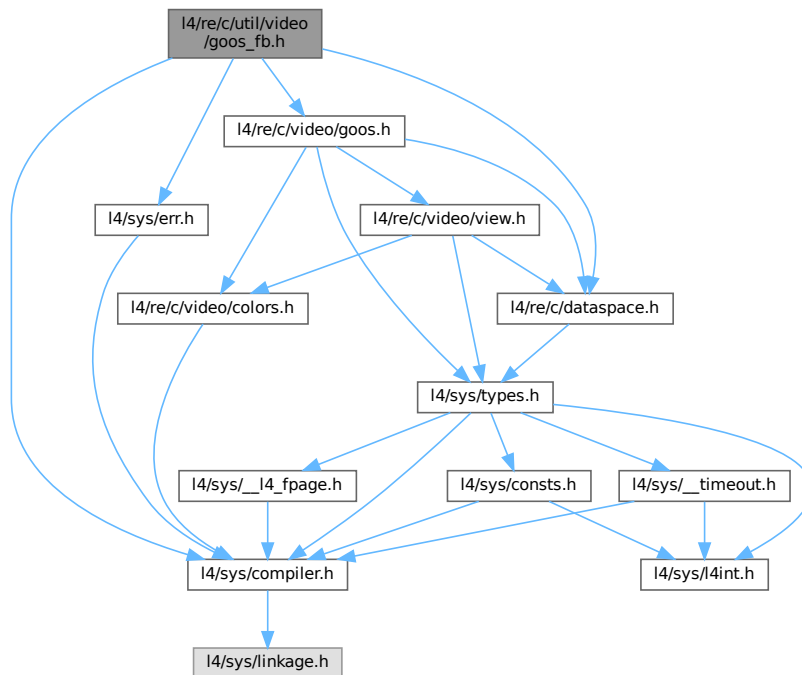
Framebuffer utility functionality.

```

#include <l4/sys/compiler.h>
#include <l4/re/c/video/goos.h>
#include <l4/sys/err.h>

```

```
#include <l4/re/c/dataspace.h>
Include dependency graph for goos_fb.h:
```



### 16.280.1 Detailed Description

Framebuffer utility functionality.

Definition in file [goos\\_fb.h](#).

## 16.281 goos\_fb.h

[Go to the documentation of this file.](#)

```

00001
00002 /*
00003  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019 #pragma once
00020
00021 #include <l4/sys/compiler.h>
00022 #include <l4/re/c/video/goos.h>

```

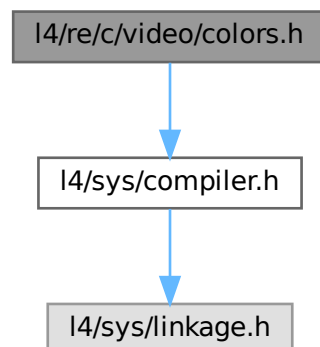
```

00026 #include <l4/sys/err.h>
00027 #include <l4/re/c/dataspace.h>
00028
00029 EXTERN_C_BEGIN
00030
00031 typedef struct
00032 {
00033     unsigned long _obj_buf[6];
00034 } l4re_util_video_goos_fb_t;
00035
00036 L4_CV int
00037 l4re_util_video_goos_fb_setup_name(l4re_util_video_goos_fb_t *goosfb,
00038                                   char const *name) L4_NOTHROW;
00039
00040 L4_CV void
00041 l4re_util_video_goos_fb_destroy(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00042
00043 L4_CV int
00044 l4re_util_video_goos_fb_view_info(l4re_util_video_goos_fb_t *goosfb,
00045                                  l4re_video_view_info_t *info) L4_NOTHROW;
00046
00047 L4_CV void *
00048 l4re_util_video_goos_fb_attach_buffer(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00049
00050 L4_CV int
00051 l4re_util_video_goos_fb_refresh(l4re_util_video_goos_fb_t *goosfb,
00052                                int x, int y, int w, int h) L4_NOTHROW;
00053
00054 L4_CV l4re_ds_t
00055 l4re_util_video_goos_fb_buffer(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00056
00057 L4_CV l4_cap_idx_t
00058 l4re_util_video_goos_fb_goos(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00059
00060 EXTERN_C_END

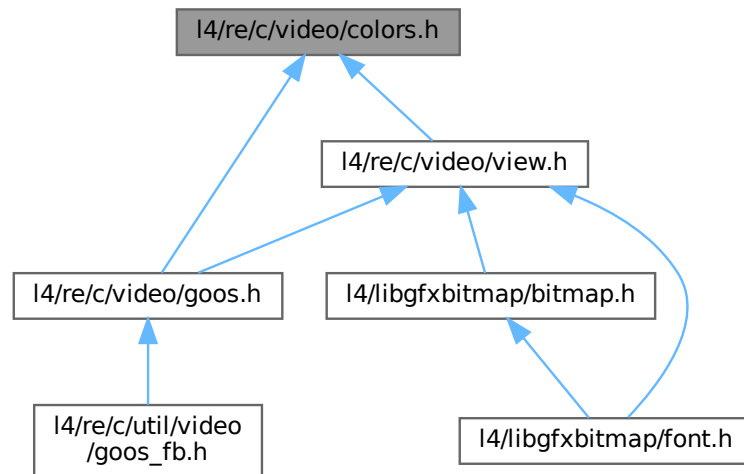
```

## 16.282 l4/re/c/video/colors.h File Reference

#include <l4/sys/compiler.h>  
 Include dependency graph for colors.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4re\\_video\\_color\\_component\\_t](#)  
*Color component structure.*
- struct [l4re\\_video\\_pixel\\_info\\_t](#)  
*Pixel\_info structure.*

## Typedefs

- typedef struct [l4re\\_video\\_color\\_component\\_t](#) [l4re\\_video\\_color\\_component\\_t](#)  
*Color component structure.*
- typedef struct [l4re\\_video\\_pixel\\_info\\_t](#) [l4re\\_video\\_pixel\\_info\\_t](#)  
*Pixel\_info structure.*

## 16.282.1 Detailed Description

### Note

The C interface of L4Re::Video does *NOT* reflect the full C++ interface on purpose. Use the C++ interface where possible.

Definition in file [colors.h](#).

## 16.283 colors.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 #include <l4/sys/compiler.h>
00026
00031 typedef struct l4re_video_color_component_t
00032 {
00033     unsigned char size;
00034     unsigned char shift;
00035 } __attribute__((packed)) l4re_video_color_component_t;
00036
00041 typedef struct l4re_video_pixel_info_t
00042 {
00043     l4re_video_color_component_t r, g, b, a;
00044     unsigned char bytes_per_pixel;
00045 } l4re_video_pixel_info_t;
00046
00047 EXTERN_C_BEGIN
00048
00049 L4_INLINE L4_CV int
00050 l4re_video_bits_per_pixel(l4re_video_pixel_info_t *p) L4_NOTHROW;
00051
00052 /* ***** */
00053 /* Implementations */
00054
00055 L4_INLINE L4_CV int
00056 l4re_video_bits_per_pixel(l4re_video_pixel_info_t *p) L4_NOTHROW
00057 {
00058     return p->r.size + p->b.size + p->g.size + p->a.size;
00059 }
00060
00061 EXTERN_C_END

```

## 16.284 l4/re/c/video/goos.h File Reference

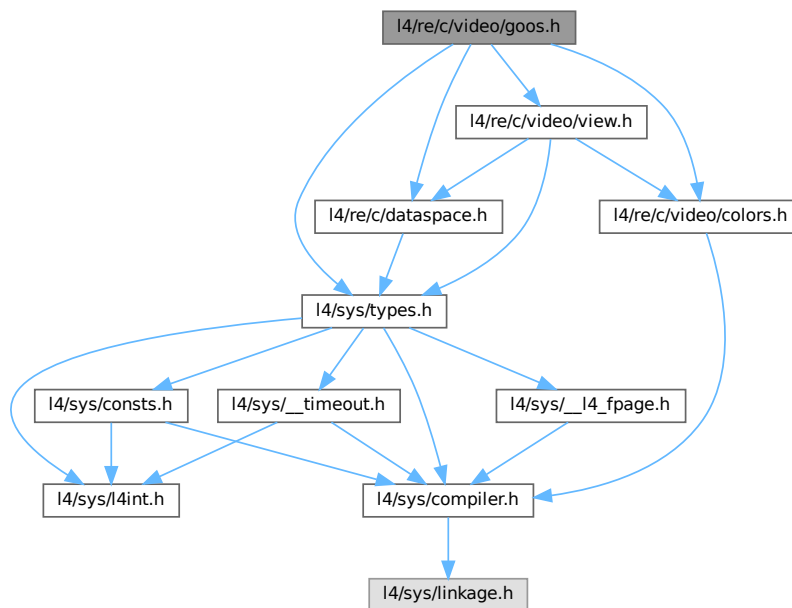
```

#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/c/video/colors.h>
#include <l4/re/c/video/view.h>

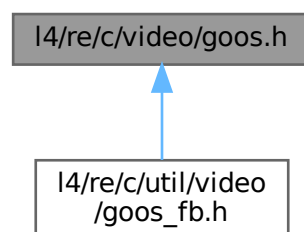
```



Include dependency graph for goos.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `l4re_video_goos_info_t`  
*Goos information structure.*

## Typedefs

- typedef `l4_cap_idx_t l4re_video_goos_t`  
*Goos object type.*

## Enumerations

- enum `l4re_video_goos_info_flags_t` { `F_l4re_video_goos_auto_refresh` = 0x01 , `F_l4re_video_goos_pointer` = 0x02 , `F_l4re_video_goos_dynamic_views` = 0x04 , `F_l4re_video_goos_dynamic_buffers` = 0x08 }

*Flags of information on the goos.*

## Functions

- int `l4re_video_goos_info` (`l4re_video_goos_t` goos, `l4re_video_goos_info_t` \*ginfo) `L4_NOTHROW`  
*Get information on a goos.*
- int `l4re_video_goos_refresh` (`l4re_video_goos_t` goos, int x, int y, int w, int h) `L4_NOTHROW`  
*Flush a rectangle of pixels of the goos screen.*
- int `l4re_video_goos_create_buffer` (`l4re_video_goos_t` goos, unsigned long size, `l4_cap_idx_t` buffer) `L4_NOTHROW`  
*Create a new buffer (memory buffer) for pixel data.*
- int `l4re_video_goos_delete_buffer` (`l4re_video_goos_t` goos, unsigned idx) `L4_NOTHROW`  
*Delete a pixel buffer.*
- int `l4re_video_goos_get_static_buffer` (`l4re_video_goos_t` goos, unsigned idx, `l4_cap_idx_t` buffer) `L4_NOTHROW`  
*Get the data-space capability of the static pixel buffer.*
- int `l4re_video_goos_create_view` (`l4re_video_goos_t` goos, `l4re_video_view_t` \*view) `L4_NOTHROW`  
*Create a new view (.*
- int `l4re_video_goos_delete_view` (`l4re_video_goos_t` goos, `l4re_video_view_t` \*view) `L4_NOTHROW`  
*Delete a view.*
- int `l4re_video_goos_get_view` (`l4re_video_goos_t` goos, unsigned idx, `l4re_video_view_t` \*view) `L4_NOTHROW`  
*Get a view for the given index.*

## 16.284.1 Detailed Description

### Note

The C interface of `L4Re::Video` does *NOT* reflect the full C++ interface on purpose. Use the C++ where possible.

Definition in file [goos.h](#).

## 16.285 goos.h

[Go to the documentation of this file.](#)

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
```

```

00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 #include <l4/sys/types.h>
00026 #include <l4/re/c/dataspace.h>
00027 #include <l4/re/c/video/colors.h>
00028 #include <l4/re/c/video/view.h>
00029
00039 enum l4re_video_goos_info_flags_t
00040 {
00041     F_l4re_video_goos_auto_refresh    = 0x01,
00042     F_l4re_video_goos_pointer        = 0x02,
00043     F_l4re_video_goos_dynamic_views  = 0x04,
00044     F_l4re_video_goos_dynamic_buffers = 0x08,
00045 };
00046
00051 typedef struct
00052 {
00053     unsigned long width;
00054     unsigned long height;
00055     unsigned flags;
00056     unsigned num_static_views;
00057     unsigned num_static_buffers;
00058     l4re_video_pixel_info_t pixel_info;
00059 } l4re_video_goos_info_t;
00060
00065 typedef l4_cap_idx_t l4re_video_goos_t;
00066
00067 EXTERN_C_BEGIN
00068
00080 L4_CV int
00081 l4re_video_goos_info(l4re_video_goos_t goos,
00082                     l4re_video_goos_info_t *ginfo) L4_NOTHROW;
00083
00093 L4_CV int
00094 l4re_video_goos_refresh(l4re_video_goos_t goos, int x, int y, int w,
00095                        int h) L4_NOTHROW;
00096
00108 L4_CV int
00109 l4re_video_goos_create_buffer(l4re_video_goos_t goos, unsigned long size,
00110                             l4_cap_idx_t buffer) L4_NOTHROW;
00111
00119 L4_CV int
00120 l4re_video_goos_delete_buffer(l4re_video_goos_t goos, unsigned idx) L4_NOTHROW;
00121
00132 L4_CV int
00133 l4re_video_goos_get_static_buffer(l4re_video_goos_t goos, unsigned idx,
00134                                 l4_cap_idx_t buffer) L4_NOTHROW;
00135
00142 L4_CV int
00143 l4re_video_goos_create_view(l4re_video_goos_t goos,
00144                            l4re_video_view_t *view) L4_NOTHROW;
00145
00153 L4_CV int
00154 l4re_video_goos_delete_view(l4re_video_goos_t goos,
00155                             l4re_video_view_t *view) L4_NOTHROW;
00156
00157
00169 L4_CV int
00170 l4re_video_goos_get_view(l4re_video_goos_t goos, unsigned idx,
00171                          l4re_video_view_t *view) L4_NOTHROW;
00172
00173 EXTERN_C_END

```

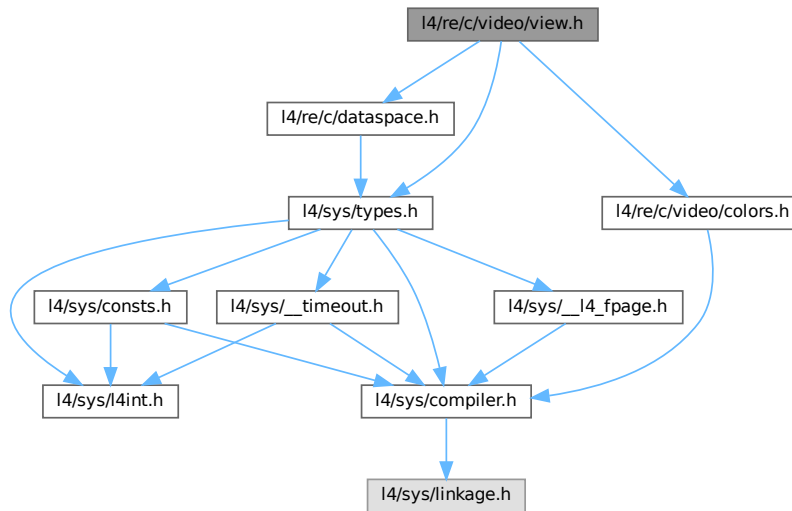
## 16.286 l4/re/c/video/view.h File Reference

```

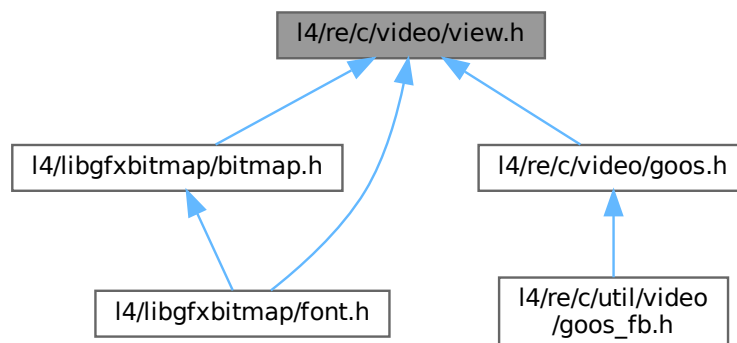
#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/c/video/colors.h>

```

Include dependency graph for view.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4re\\_video\\_view\\_info\\_t](#)  
View information structure.
- struct [l4re\\_video\\_view\\_t](#)  
C representation of a goos view.

## Typedefs

- typedef struct [l4re\\_video\\_view\\_info\\_t](#) [l4re\\_video\\_view\\_info\\_t](#)  
View information structure.
- typedef struct [l4re\\_video\\_view\\_t](#) [l4re\\_video\\_view\\_t](#)  
C representation of a goos view.

## Enumerations

- enum `l4re_video_view_info_flags_t` {  
`F_l4re_video_view_none` = 0x00 , `F_l4re_video_view_set_buffer` = 0x01 , `F_l4re_video_view_set_buffer_offset` = 0x02 , `F_l4re_video_view_set_bytes_per_line` = 0x04 ,  
`F_l4re_video_view_set_pixel` = 0x08 , `F_l4re_video_view_set_position` = 0x10 , `F_l4re_video_view_dyn_allocated` = 0x20 , `F_l4re_video_view_set_background` = 0x40 ,  
`F_l4re_video_view_set_flags` = 0x80 , **`F_l4re_video_view_fully_dynamic`** , `F_l4re_video_view_above` = 0x01000 , `F_l4re_video_view_flags_mask` = 0xff000 }

*Flags of information on a view.*

## Functions

- int `l4re_video_view_refresh` (`l4re_video_view_t` \*view, int x, int y, int w, int h) `L4_NOTHROW`  
*Flush the given rectangle of pixels of the given view.*
- int `l4re_video_view_get_info` (`l4re_video_view_t` \*view, `l4re_video_view_info_t` \*info) `L4_NOTHROW`  
*Retrieve information about the given view.*
- int `l4re_video_view_set_info` (`l4re_video_view_t` \*view, `l4re_video_view_info_t` \*info) `L4_NOTHROW`  
*Set properties of the view.*
- int `l4re_video_view_set_viewport` (`l4re_video_view_t` \*view, int x, int y, int w, int h, unsigned long bofs) `L4_NOTHROW`  
*Set the viewport parameters of a view.*
- int `l4re_video_view_stack` (`l4re_video_view_t` \*view, `l4re_video_view_t` \*pivot, int behind) `L4_NOTHROW`  
*Change the stacking order in the stack of visible views.*

## 16.286.1 Detailed Description

### Note

The C interface of `L4Re::Video` does *NOT* reflect the full C++ interface on purpose. Use the C++ where possible.

Definition in file [view.h](#).

## 16.287 view.h

[Go to the documentation of this file.](#)

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 #include <l4/sys/types.h>
```

```

00026 #include <l4/re/c/dataspace.h>
00027 #include <l4/re/c/video/colors.h>
00028
00033 enum l4re_video_view_info_flags_t
00034 {
00035     F_l4re_video_view_none           = 0x00,
00036     F_l4re_video_view_set_buffer     = 0x01,
00037     F_l4re_video_view_set_buffer_offset = 0x02,
00038     F_l4re_video_view_set_bytes_per_line = 0x04,
00039     F_l4re_video_view_set_pixel      = 0x08,
00040     F_l4re_video_view_set_position   = 0x10,
00041     F_l4re_video_view_dyn_allocated  = 0x20,
00042     F_l4re_video_view_set_background = 0x40,
00043     F_l4re_video_view_set_flags      = 0x80,
00044     F_l4re_video_view_fully_dynamic  = F_l4re_video_view_set_buffer
00045     | F_l4re_video_view_set_buffer_offset
00046     | F_l4re_video_view_set_bytes_per_line
00047     | F_l4re_video_view_set_pixel
00048     | F_l4re_video_view_set_position
00049     | F_l4re_video_view_dyn_allocated,
00050
00051     F_l4re_video_view_above          = 0x01000,
00052     F_l4re_video_view_flags_mask    = 0xff000,
00053 };
00054
00059 typedef struct l4re_video_view_info_t
00060 {
00061     unsigned flags;
00062     unsigned view_index;
00063     unsigned long xpos, ypos, width, height;
00064     unsigned long buffer_offset;
00065     unsigned long bytes_per_line;
00066     l4re_video_pixel_info_t pixel_info;
00067     unsigned buffer_index;
00068 } l4re_video_view_info_t;
00069
00070
00078 typedef struct l4re_video_view_t
00079 {
00080     l4_cap_idx_t goos;
00081     unsigned idx;
00082 } l4re_video_view_t;
00083
00084
00085 EXTERN_C_BEGIN
00086
00097 L4_CV int
00098 l4re_video_view_refresh(l4re_video_view_t *view, int x, int y, int w,
00099                        int h) L4_NOTHROW;
00100
00108 L4_CV int
00109 l4re_video_view_get_info(l4re_video_view_t *view,
00110                          l4re_video_view_info_t *info) L4_NOTHROW;
00111
00123 L4_CV int
00124 l4re_video_view_set_info(l4re_video_view_t *view,
00125                           l4re_video_view_info_t *info) L4_NOTHROW;
00126
00143 L4_CV int
00144 l4re_video_view_set_viewport(l4re_video_view_t *view, int x, int y, int w,
00145                               int h, unsigned long bofs) L4_NOTHROW;
00146
00157 L4_CV int
00158 l4re_video_view_stack(l4re_video_view_t *view, l4re_video_view_t *pivot,
00159                       int behind) L4_NOTHROW;
00160
00161 EXTERN_C_END
00162

```

## 16.288 l4/re/cap\_alloc File Reference

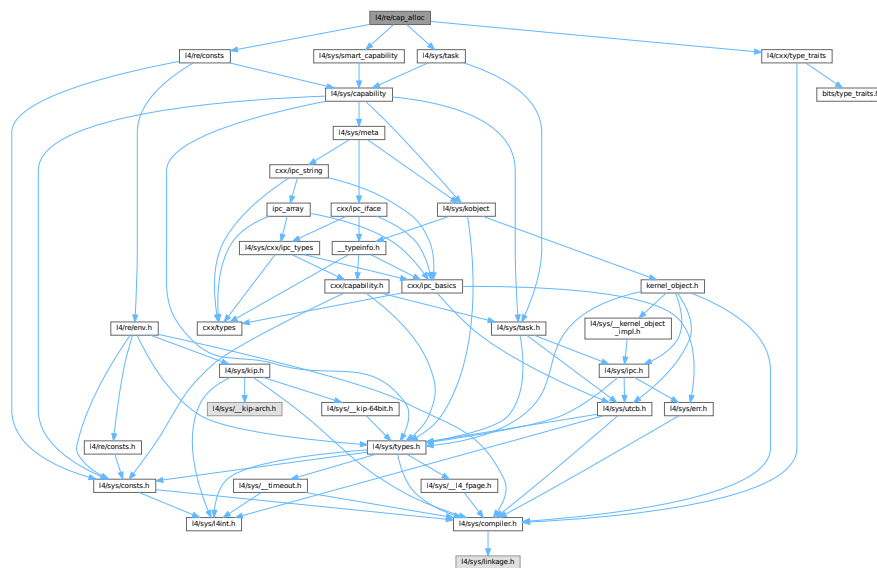
Abstract capability-allocator interface.

```

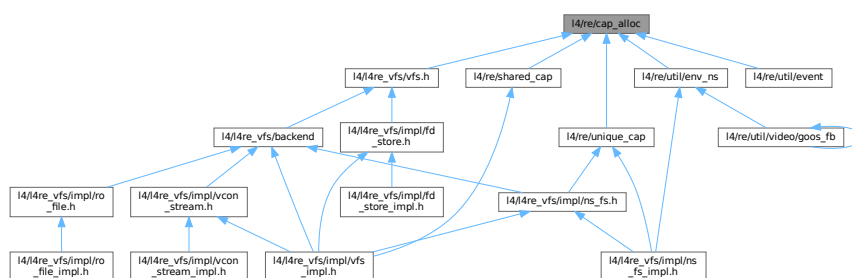
#include <l4/sys/task>
#include <l4/sys/smart_capability>
#include <l4/re/consts>

```

```
#include <l4/cxx/type_traits>
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4Re::Cap_alloc`  
*Capability allocator interface.*
- class `L4Re::Smart_cap_auto< Unmap_flags >`  
*Helper for Unique\_cap and Unique\_del\_cap.*
- class `L4Re::Smart_count_cap< Unmap_flags >`  
*Helper for Ref\_cap and Ref\_del\_cap.*

## Namespaces

- namespace **L4Re**  
*L4Re C++ Interfaces.*

## 16.288.1 Detailed Description

Abstract capability-allocator interface.

Definition in file [cap\\_alloc](#).

## 16.289 cap\_alloc

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include <l4/sys/task>
00028 #include <l4/sys/smart_capability>
00029 #include <l4/re/consts>
00030 #include <l4/cxx/type_traits>
00031
00032 namespace L4Re {
00033
00041 class Cap_alloc
00042 {
00043 private:
00044     void operator = (Cap_alloc const &);
00045
00046 protected:
00047     Cap_alloc(Cap_alloc const &) {}
00048     Cap_alloc() {}
00049
00050 public:
00051
00056     virtual L4::Cap<void> alloc() noexcept = 0;
00057     virtual void take(L4::Cap<void> cap) noexcept = 0;
00058
00063     template< typename T >
00064     L4::Cap<T> alloc() noexcept
00065     { return L4::cap_cast<T>(alloc()); }
00066
00073     virtual void free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00074                      unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept = 0;
00075     virtual bool release(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00076                         unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept = 0;
00077
00081     virtual ~Cap_alloc() = 0;
00082
00088     template< typename CAP_ALLOC >
00089     static inline L4Re::Cap_alloc *
00090     get_cap_alloc(CAP_ALLOC &ca)
00091     {
00092         struct CA : public L4Re::Cap_alloc
00093         {
00094             CAP_ALLOC &_ca;
00095             L4::Cap<void> alloc() noexcept override { return _ca.alloc(); }
00096             void take(L4::Cap<void> cap) noexcept override { _ca.take(cap); }
00097
00098             void free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00099                     unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept override
00100             { _ca.free(cap, task, unmap_flags); }
00101
00102             bool release(L4::Cap<void> cap, l4_cap_idx_t task,
```



```

00103         unsigned unmap_flags) noexcept override
00104     { return _ca.release(cap, task, unmap_flags); }
00105
00106     void operator delete(void *) {}
00107
00108     CA(CAP_ALLOC &ca) : _ca(ca) {}
00109 };
00110
00111     static CA _ca(ca);
00112     return &_ca;
00113 }
00114 };
00115
00116 template<typename ALLOC>
00117 struct Cap_alloc_t : ALLOC, L4Re::Cap_alloc
00118 {
00119     template<typename ...ARGS>
00120     Cap_alloc_t(ARGS &&...args) : ALLOC(cxx::forward<ARGS>(args)...) {}
00121
00122     L4::Cap<void> alloc() noexcept override { return ALLOC::alloc(); }
00123     void take(L4::Cap<void> cap) noexcept override { ALLOC::take(cap); }
00124
00125     void free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00126             unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept override
00127     { ALLOC::free(cap, task, unmap_flags); }
00128
00129     bool release(L4::Cap<void> cap, l4_cap_idx_t task,
00130             unsigned unmap_flags) noexcept override
00131     { return ALLOC::release(cap, task, unmap_flags); }
00132
00133     void operator delete(void *) {}
00134 };
00135
00136 inline
00137 Cap_alloc::~Cap_alloc()
00138 {}
00139
00140 extern Cap_alloc *virt_cap_alloc;
00141
00142 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00143 class Smart_cap_auto
00144 {
00145 private:
00146     Cap_alloc *_ca;
00147
00148 public:
00149     Smart_cap_auto() : _ca(0) {}
00150     Smart_cap_auto(Cap_alloc *ca) : _ca(ca) {}
00151
00152     void free(L4::Cap_base &c)
00153     {
00154         if (c.is_valid() && _ca)
00155             _ca->free(L4::Cap<void>(c.cap()), This_task, Unmap_flags);
00156         invalidate(c);
00157     }
00158
00159     static void invalidate(L4::Cap_base &c)
00160     {
00161         if (c.is_valid())
00162             c.invalidate();
00163     }
00164 };
00165
00166 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00167 class Smart_count_cap
00168 {
00169 private:
00170     Cap_alloc *_ca;
00171
00172 public:
00173     Smart_count_cap() : _ca(nullptr) {}
00174     Smart_count_cap(Cap_alloc *ca) : _ca(ca) {}
00175     void free(L4::Cap_base &c) noexcept
00176     {
00177         if (c.is_valid())
00178         {
00179             if (_ca && _ca->release(L4::Cap<void>(c.cap()), This_task, Unmap_flags))
00180                 c.invalidate();
00181         }
00182     }
00183
00184     static void invalidate(L4::Cap_base &c) noexcept
00185     {
00186         if (c.is_valid())
00187             c.invalidate();
00188     }
00189 };

```



## Data Structures

- class [L4Re::Util::Smart\\_cap\\_auto< Unmap\\_flags >](#)  
*Helper for Unique\_cap and Unique\_del\_cap.*
- class [L4Re::Util::Smart\\_count\\_cap< Unmap\\_flags >](#)  
*Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).*
- struct [L4Re::Util::Ref\\_cap< T >](#)  
*Automatic capability that implements automatic free and unmap of the capability selector.*
- struct [L4Re::Util::Ref\\_del\\_cap< T >](#)  
*Automatic capability that implements automatic free and unmap+delete of the capability selector.*

## Namespaces

- namespace [L4Re](#)  
*[L4Re](#) C++ Interfaces.*
- namespace [L4Re::Util](#)  
*Documentation of the [L4](#) Runtime Environment utility functionality in C++.*

## Functions

- template<typename T >  
[Ref\\_cap< T >::Cap](#) [L4Re::Util::make\\_ref\\_cap](#) ()  
*Allocate a capability slot and wrap it in a [Ref\\_cap](#).*
- template<typename T >  
[Ref\\_del\\_cap< T >::Cap](#) [L4Re::Util::make\\_ref\\_del\\_cap](#) ()  
*Allocate a capability slot and wrap it in a [Ref\\_del\\_cap](#).*

## Variables

- [\\_Cap\\_alloc](#) & [L4Re::Util::cap\\_alloc](#)  
*Capability allocator.*

## 16.290.1 Detailed Description

Capability allocator.

Definition in file [cap\\_alloc](#).

## 16.291 cap\_alloc

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025
00026 #pragma once
00027
00028 #include <l4/re/util/cap_alloc_impl.h>
00029 #include <l4/sys/smart_capability>
00030 #include <l4/sys/task>
00031 #include <l4/re/consts>
00032
00033 namespace L4Re { namespace Util {
00034
00050 extern _Cap_alloc &cap_alloc;
00051
00055 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00056 class Smart_cap_auto
00057 {
00058 public:
00062     static void free(L4::Cap_base &c)
00063     {
00064         if (c.is_valid())
00065         {
00066             cap_alloc.free(L4::Cap<void>(c.cap()), This_task, Unmap_flags);
00067             c.invalidate();
00068         }
00069     }
00070
00074     static void invalidate(L4::Cap_base &c)
00075     {
00076         if (c.is_valid())
00077             c.invalidate();
00078     }
00079
00080 };
00081
00082
00086 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00087 class Smart_count_cap
00088 {
00089 public:
00094     static void free(L4::Cap_base &c) noexcept
00095     {
00096         if (c.is_valid())
00097         {
00098             if (cap_alloc.release(L4::Cap<void>(c.cap()), This_task, Unmap_flags))
00099                 c.invalidate();
00100         }
00101     }
00102
00106     static void invalidate(L4::Cap_base &c) noexcept
00107     {
00108         if (c.is_valid())
00109             c.invalidate();
00110     }
00111
00115     static L4::Cap_base copy(L4::Cap_base const &src)
00116     {
00117         cap_alloc.take(L4::Cap<void>(src.cap()));
00118         return src;
00119     }
00120 };
00121
00122
00152 template< typename T >

```

```

00153 struct Ref_cap
00154 {
00155     typedef L4::Smart_cap<T, Smart_count_cap<L4_FP_ALL_SPACES> > Cap;
00156 };
00157
00193 template< typename T >
00194 struct Ref_del_cap
00195 {
00196     typedef L4::Smart_cap<T, Smart_count_cap<L4_FP_DELETE_OBJ> > Cap;
00197 };
00198
00204 template< typename T >
00205 typename Ref_cap<T>::Cap
00206 make_ref_cap() { return typename Ref_cap<T>::Cap(cap_alloc.alloc<T>()); }
00207
00213 template< typename T >
00214 typename Ref_del_cap<T>::Cap
00215 make_ref_del_cap()
00216 { return typename Ref_del_cap<T>::Cap(cap_alloc.alloc<T>()); }
00217
00220 }
00221

```

## 16.292 console

```

00001 // vi:set ft=c++: -- Mode: C++ --
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020
00021 #pragma once
00022
00023 #include <l4/re/video/goos>
00024 #include <l4/re/event>
00025
00026 namespace L4Re {
00027
00039 class L4_EXPORT Console :
00040     public L4::Kobject_2t<Console, Video::Goos, Event, L4::PROTO_EMPTY>
00041 { };
00042
00043 }
00044

```

## 16.293 l4/re/consts File Reference

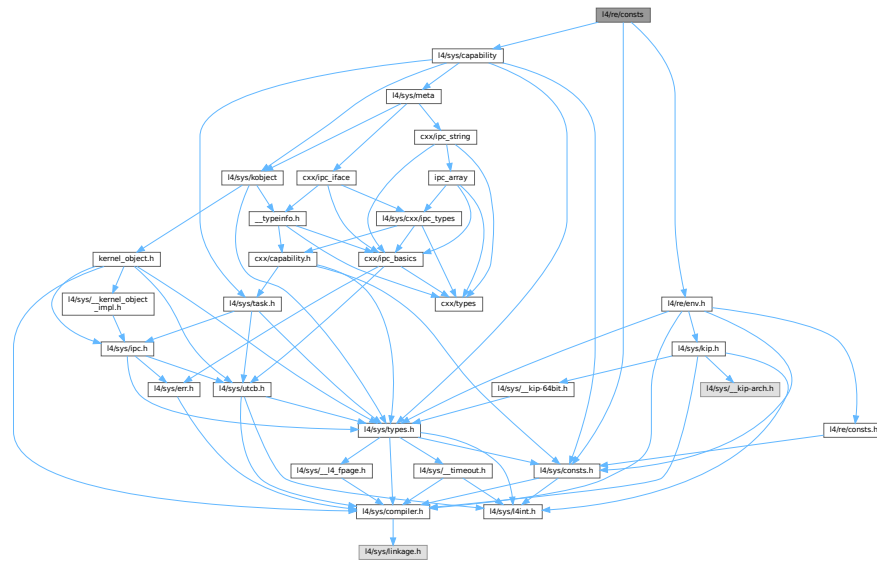
Constants.

```

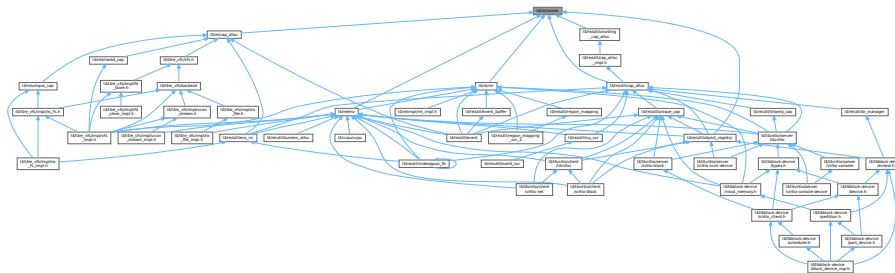
#include <l4/sys/capability>
#include <l4/sys/consts.h>
#include <l4/re/env.h>

```

Include dependency graph for consts:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace **L4Re**  
*L4Re C++ Interfaces.*

### 16.293.1 Detailed Description

Constants.

Definition in file [consts.](#)

## 16.294 consts

[Go to the documentation of this file.](#)

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/sys/capability>
00027 #include <l4/sys/consts.h>
00028 #include <l4/re/env.h>
00029
00030 namespace L4Re {
00031     static L4::Cap<L4::Task>::Cap_type const This_task
00032     = static_cast<L4::Cap<L4::Task>::Cap_type>(L4RE_THIS_TASK_CAP);
00033 }
```

## 16.295 consts

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002
00003 #pragma once
00004
00005 #include <l4/sys/consts.h>
00006
00007 namespace L4 {
00008
00017     template<typename T>
00018     constexpr T trunc_order(T val, unsigned char order)
00019     {
00020         return val & ((~T(0)) << order);
00021     }
00022
00031     template<typename T>
00032     constexpr T round_order(T val, unsigned char order)
00033     {
00034         return (val + (T(1) << order) - T(1)) & ((~T(0)) << order);
00035     }
00036
00037     template<typename T>
00038     constexpr T trunc_page(T val)
00039     {
00040         return trunc_order(val, L4_PAGESHIFT);
00041     }
00042
00043     template<typename T>
00044     constexpr T round_page(T val)
00045     {
00046         return round_order(val, L4_PAGESHIFT);
00047     }
00048
00049     template<typename T>
00050     inline unsigned char
00051     max_order(unsigned char order, T addr,
00052               T min_addr, T max_addr,
00053               T hotspot = T(0))
00054     {
00055         while (order < 30 /* limit to 1GB flexpages */)
00056         {
00057             T mask;
00058             T base = trunc_order(addr, order + 1);
00059             if (base < min_addr)
00060                 return order;
00061
00062             if (base + (T(1) << (order + 1)) - T(1) > max_addr - T(1))
```

```

00063         return order;
00064
00065         mask = ~( (~T(0)) << (order + 1));
00066         if (hotspot == ~T(0) || ((addr ^ hotspot) & mask))
00067             break;
00068
00069         ++order;
00070     }
00071
00072     return order;
00073 }
00074
00075 }
```

## 16.296 amd64/l4/sys/consts.h File Reference

Common [L4](#) constants, amd64 version.

### Macros

- `#define L4_PAGESHIFT 12`  
*Size of a page, log2-based.*
- `#define L4_SUPERPAGESHIFT 21`  
*Size of a large page, log2-based.*

### 16.296.1 Detailed Description

Common [L4](#) constants, amd64 version.

Definition in file [consts.h](#).

## 16.297 consts.h

[Go to the documentation of this file.](#)

```

00001 /*****
00007 */
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011 *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00012 *      economic rights: Technische Universität Dresden (Germany)
00013 *
00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU General Public License 2.
00016 * Please see the COPYING-GPL-2 file for details.
00017 *
00018 * As a special exception, you may use this file as part of a free software
00019 * library without restriction. Specifically, if other files instantiate
00020 * templates or use macros or inline functions from this file, or you compile
00021 * this file and link it with other files to produce an executable, this
00022 * file does not by itself cause the resulting executable to be covered by
00023 * the GNU General Public License. This exception does not however
00024 * invalidate any other reasons why the executable file might be covered by
00025 * the GNU General Public License.
00026 */
00027 /*****
00028 #ifndef __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__
00029 #define __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__
00030
00035 #define L4_PAGESHIFT      12
00036
00041 #define L4_SUPERPAGESHIFT 21
00042
00043 #include_next <l4/sys/consts.h>
00044
00045 #endif /* ! __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__ */
```

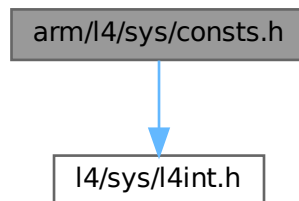


## 16.298 arm/l4/sys/consts.h File Reference

Common [L4](#) constants, arm version.

```
#include <l4/sys/l4int.h>
```

Include dependency graph for consts.h:



### Macros

- `#define L4_PAGESHIFT 12`  
*Size of a page, log2-based.*
- `#define L4_SUPERPAGESHIFT 21`  
*Size of a large page, log2-based.*

### 16.298.1 Detailed Description

Common [L4](#) constants, arm version.

Definition in file [consts.h](#).

## 16.299 consts.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
  
```

```

00025 #ifndef _L4_SYS_CONSTS_H
00026 #define _L4_SYS_CONSTS_H
00027
00028 /* L4 includes */
00029 #include <l4/sys/l4int.h>
00037 #define L4_PAGESHIFT 12
00038
00042 #define L4_SUPERPAGESHIFT 21
00043
00046 #include_next <l4/sys/consts.h>
00047
00048 #endif /* !_L4_SYS_CONSTS_H */

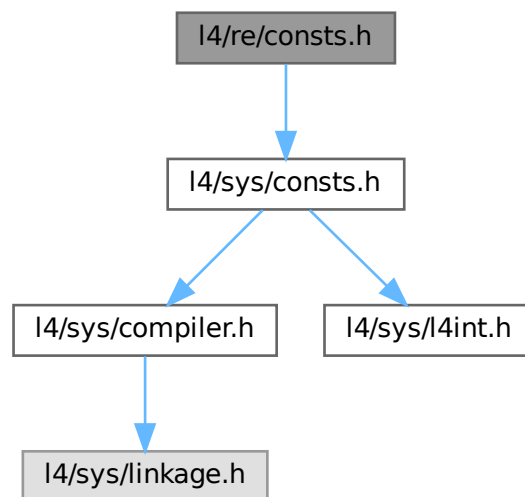
```

## 16.300 l4/re/consts.h File Reference

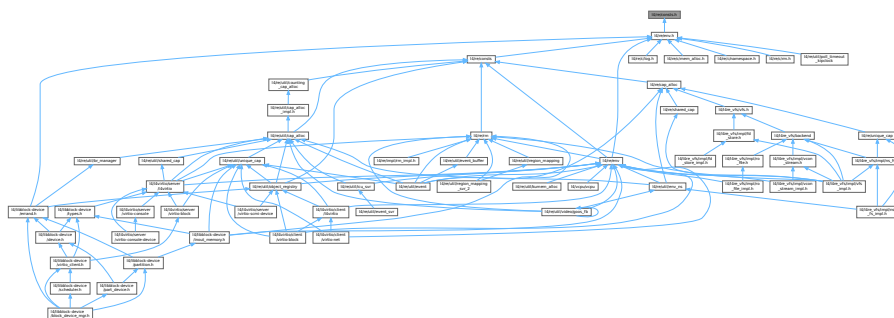
Constants.

```
#include <l4/sys/consts.h>
```

Include dependency graph for consts.h:



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum

*Defaults for local thread priorities.*

### 16.300.1 Detailed Description

Constants.

Definition in file [consts.h](#).

### 16.300.2 Enumeration Type Documentation

#### 16.300.2.1 anonymous enum

anonymous enum

Defaults for local thread priorities.

Priorities are to be seen as local. These are used by the loader and libpthread. They are to be understood as 'local', which means the actual priority of the thread (as seen by the kernel) is the base priority as defined by the scheduler plus the local priority.

Definition at line 39 of file [consts.h](#).

## 16.301 consts.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #pragma once
00023
00024 #include <14/sys/consts.h>
00025
00026 enum
00027 {
00028     L4RE_THIS_TASK_CAP = 1UL « L4_CAP_SHIFT,
00029 };
00030
00031 enum
00032 {
00033     L4RE_MAIN_THREAD_PRIO = 2, /* Priority of the main thread */
00034 };
00035

```

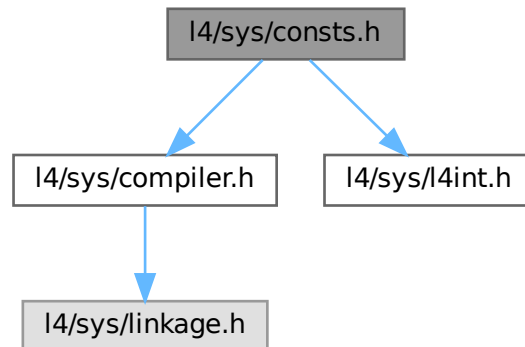
## 16.302 l4/sys/consts.h File Reference

Common constants.

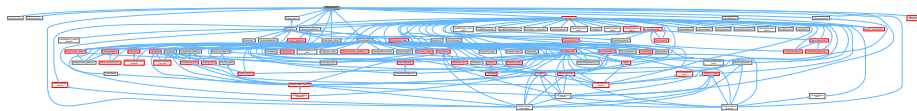
```
#include <l4/sys/compiler.h>
```

```
#include <l4/sys/l4int.h>
```

Include dependency graph for consts.h:



This graph shows which files directly or indirectly include this file:



### Macros

- **#define L4\_PAGESIZE**  
*Minimal page size (in bytes).*
- **#define L4\_PAGEMASK**  
*Mask for the page number.*
- **#define L4\_LOG2\_PAGESIZE**  
*Number of bits used for page offset.*
- **#define L4\_SUPERPAGESIZE**  
*Size of a large page.*
- **#define L4\_SUPERPAGEMASK**  
*Mask for the number of a large page.*
- **#define L4\_LOG2\_SUPERPAGESIZE**  
*Number of bits used as offset for a large page.*
- **#define L4\_INVALID\_PTR** `((void *)L4_INVALID_ADDR)`  
*Invalid address as pointer type.*

## Enumerations

- enum [l4\\_syscall\\_flags\\_t](#) {  
[L4\\_SYSF\\_NONE](#) , [L4\\_SYSF\\_SEND](#) , [L4\\_SYSF\\_RECV](#) , [L4\\_SYSF\\_OPEN\\_WAIT](#) ,  
[L4\\_SYSF\\_REPLY](#) , [L4\\_SYSF\\_CALL](#) , [L4\\_SYSF\\_WAIT](#) , [L4\\_SYSF\\_SEND\\_AND\\_WAIT](#) ,  
[L4\\_SYSF\\_REPLY\\_AND\\_WAIT](#) }  
*Capability selector flags.*
- enum [l4\\_cap\\_consts\\_t](#) {  
[L4\\_CAP\\_SHIFT](#) , [L4\\_CAP\\_SIZE](#) = 1UL << [L4\\_CAP\\_SHIFT](#) , [L4\\_CAP\\_OFFSET](#) , [L4\\_CAP\\_MASK](#) ,  
[L4\\_INVALID\\_CAP](#) , [L4\\_INVALID\\_CAP\\_BIT](#) = 1UL << ([L4\\_CAP\\_SHIFT](#) - 1) }  
*Constants related to capability selectors.*
- enum [l4\\_unmap\\_flags\\_t](#) { [L4\\_FP\\_ALL\\_SPACES](#) , [L4\\_FP\\_DELETE\\_OBJ](#) , [L4\\_FP\\_OTHER\\_SPACES](#) }  
*Flags for the unmap operation.*
- enum [l4\\_msg\\_item\\_consts\\_t](#) {  
[L4\\_ITEM\\_MAP](#) = 8 , [L4\\_ITEM\\_CONT](#) = 1 , [L4\\_MAP\\_ITEM\\_GRANT](#) = 2 , [L4\\_MAP\\_ITEM\\_MAP](#) = 0 ,  
[L4\\_RCV\\_ITEM\\_SINGLE\\_CAP](#) = [L4\\_ITEM\\_MAP](#) | 2 , [L4\\_RCV\\_ITEM\\_LOCAL\\_ID](#) = 4 }  
*Constants for message items.*
- enum [l4\\_buffer\\_desc\\_consts\\_t](#) { [L4\\_BDR\\_MEM\\_SHIFT](#) = 0 , [L4\\_BDR\\_IO\\_SHIFT](#) = 5 , [L4\\_BDR\\_OBJ\\_SHIFT](#)  
= 10 , [L4\\_BDR\\_OFFSET\\_MASK](#) = (1UL << 20) - 1 }  
*Constants for buffer descriptors.*
- enum [l4\\_default\\_caps\\_t](#) {  
[L4\\_BASE\\_TASK\\_CAP](#) , [L4\\_BASE\\_FACTORY\\_CAP](#) , [L4\\_BASE\\_THREAD\\_CAP](#) , [L4\\_BASE\\_PAGER\\_CAP](#) ,  
[L4\\_BASE\\_LOG\\_CAP](#) , [L4\\_BASE\\_ICU\\_CAP](#) , [L4\\_BASE\\_SCHEDULER\\_CAP](#) , [L4\\_BASE\\_IOMMU\\_CAP](#) ,  
[L4\\_BASE\\_DEBUGGER\\_CAP](#) , [L4\\_BASE\\_ARM\\_SMCCC\\_CAP](#) , [L4\\_BASE\\_CAPS\\_LAST\\_P1](#) , [L4\\_BASE\\_CAPS\\_LAST](#)  
= [L4\\_BASE\\_CAPS\\_LAST\\_P1](#) - 1 }  
*Default capabilities setup for the initial tasks.*
- enum [l4\\_addr\\_consts\\_t](#) { [L4\\_INVALID\\_ADDR](#) = ~0UL }  
*Address related constants.*

## Functions

- [l4\\_addr\\_t l4\\_trunc\\_page](#) ([l4\\_addr\\_t](#) address) [L4\\_NOTHROW](#)  
*Round an address down to the next lower page boundary.*
- [l4\\_addr\\_t l4\\_trunc\\_size](#) ([l4\\_addr\\_t](#) address, unsigned char bits) [L4\\_NOTHROW](#)  
*Round an address down to the next lower flex page with size bits.*
- [l4\\_addr\\_t l4\\_round\\_page](#) ([l4\\_addr\\_t](#) address) [L4\\_NOTHROW](#)  
*Round address up to the next page.*
- [l4\\_addr\\_t l4\\_round\\_size](#) ([l4\\_addr\\_t](#) value, unsigned char bits) [L4\\_NOTHROW](#)  
*Round value up to the next alignment with bits size.*
- unsigned [l4\\_bytes\\_to\\_mwords](#) (unsigned size) [L4\\_NOTHROW](#)  
*Determine how many machine words ([l4\\_umword\\_t](#)) are required to store a buffer of 'size' bytes.*

### 16.302.1 Detailed Description

Common constants.

Definition in file [consts.h](#).

## 16.303 consts.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00010  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction. Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License. This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 #ifndef __L4_SYS__INCLUDE__CONSTS_H__
00027 #define __L4_SYS__INCLUDE__CONSTS_H__
00028
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/l4int.h>
00031
00061 enum l4_syscall_flags_t
00062 {
00071     L4_SYSF_NONE          = 0x00,
00072
00084     L4_SYSF_SEND          = 0x01,
00085
00095     L4_SYSF_RECV          = 0x02,
00096
00106     L4_SYSF_OPEN_WAIT    = 0x04,
00107
00115     L4_SYSF_REPLY        = 0x08,
00116
00123     L4_SYSF_CALL          = L4_SYSF_SEND | L4_SYSF_RECV,
00124
00131     L4_SYSF_WAIT          = L4_SYSF_OPEN_WAIT | L4_SYSF_RECV,
00132
00139     L4_SYSF_SEND_AND_WAIT = L4_SYSF_OPEN_WAIT | L4_SYSF_CALL,
00140
00147     L4_SYSF_REPLY_AND_WAIT = L4_SYSF_WAIT | L4_SYSF_SEND | L4_SYSF_REPLY
00148 };
00149
00154 enum l4_cap_consts_t
00155 {
00157     L4_CAP_SHIFT          = 12UL,
00159     L4_CAP_SIZE           = 1UL « L4_CAP_SHIFT,
00161     L4_CAP_OFFSET         = 1UL « L4_CAP_SHIFT,
00166     L4_CAP_MASK           = ~0UL « (L4_CAP_SHIFT - 1),
00168     L4_INVALID_CAP        = ~0UL « (L4_CAP_SHIFT - 1),
00169
00170     L4_INVALID_CAP_BIT    = 1UL « (L4_CAP_SHIFT - 1),
00171 };
00172
00173 enum l4_sched_consts_t
00174 {
00175     L4_SCHED_MIN_PRIO    = 0,
00176     L4_SCHED_MAX_PRIO    = 255,
00177 };
00178
00184 enum l4_unmap_flags_t
00185 {
00198     L4_FP_ALL_SPACES      = 0x80000000UL,
00199
00209     L4_FP_DELETE_OBJ      = 0xc0000000UL,
00210
00217     L4_FP_OTHER_SPACES    = 0x0UL
00218 };
00219
00224 enum l4_msg_item_consts_t
00225 {
00226     L4_ITEM_MAP            = 8,
00227
00232     L4_ITEM_CONT          = 1,
00233
00234     // send
00257     L4_MAP_ITEM_GRANT      = 2,

```

```

00258
00259 L4_MAP_ITEM_MAP = 0,
00260
00261 // receive
00266 L4_RCV_ITEM_SINGLE_CAP = L4_ITEM_MAP | 2,
00267
00285 L4_RCV_ITEM_LOCAL_ID = 4,
00286 };
00287
00292 enum l4_buffer_desc_consts_t
00293 {
00294     L4_BDR_MEM_SHIFT = 0,
00295     L4_BDR_IO_SHIFT = 5,
00296     L4_BDR_OBJ_SHIFT = 10,
00297     L4_BDR_OFFSET_MASK = (1UL < 20) - 1,
00298 };
00299
00313 enum l4_default_caps_t
00314 {
00316     L4_BASE_TASK_CAP = 1UL < L4_CAP_SHIFT,
00318     L4_BASE_FACTORY_CAP = 2UL < L4_CAP_SHIFT,
00320     L4_BASE_THREAD_CAP = 3UL < L4_CAP_SHIFT,
00328     L4_BASE_PAGER_CAP = 4UL < L4_CAP_SHIFT,
00336     L4_BASE_LOG_CAP = 5UL < L4_CAP_SHIFT,
00338     L4_BASE_ICU_CAP = 6UL < L4_CAP_SHIFT,
00340     L4_BASE_SCHEDULER_CAP = 7UL < L4_CAP_SHIFT,
00347     L4_BASE_IOMMU_CAP = 8UL < L4_CAP_SHIFT,
00355     L4_BASE_DEBUGGER_CAP = 10UL < L4_CAP_SHIFT,
00362     L4_BASE_ARM_SMCCC_CAP = 11UL < L4_CAP_SHIFT,
00363
00365     L4_BASE_CAPS_LAST_P1,
00367     L4_BASE_CAPS_LAST = L4_BASE_CAPS_LAST_P1 - 1
00368 };
00369
00380 #define L4_PAGESIZE (1UL < L4_PAGESHIFT)
00381
00389 #define L4_PAGEMASK (~(L4_PAGESIZE - 1))
00390
00398 #define L4_LOG2_PAGESIZE L4_PAGESHIFT
00399
00407 #define L4_SUPERPAGESIZE (1UL < L4_SUPERPAGESHIFT)
00408
00416 #define L4_SUPERPAGEMASK (~(L4_SUPERPAGESIZE - 1))
00417
00424 #define L4_LOG2_SUPERPAGESIZE L4_SUPERPAGESHIFT
00425
00436 L4_INLINE l4_addr_t l4_trunc_page(l4_addr_t address) L4_NOTHROW;
00437 L4_INLINE l4_addr_t l4_trunc_page(l4_addr_t address) L4_NOTHROW
00438 { return address & L4_PAGEMASK; }
00439
00447 L4_INLINE l4_addr_t l4_trunc_size(l4_addr_t address, unsigned char bits) L4_NOTHROW;
00448 L4_INLINE l4_addr_t l4_trunc_size(l4_addr_t address, unsigned char bits) L4_NOTHROW
00449 { return address & (~0UL < bits); }
00450
00461 L4_INLINE l4_addr_t l4_round_page(l4_addr_t address) L4_NOTHROW;
00462 L4_INLINE l4_addr_t l4_round_page(l4_addr_t address) L4_NOTHROW
00463 { return (address + L4_PAGESIZE - 1) & L4_PAGEMASK; }
00464
00472 L4_INLINE l4_addr_t l4_round_size(l4_addr_t value, unsigned char bits) L4_NOTHROW;
00473 L4_INLINE l4_addr_t l4_round_size(l4_addr_t value, unsigned char bits) L4_NOTHROW
00474 { return (value + (1UL < bits) - 1) & (~0UL < bits); }
00475
00484 L4_INLINE unsigned l4_bytes_to_mwords(unsigned size) L4_NOTHROW;
00485 L4_INLINE unsigned l4_bytes_to_mwords(unsigned size) L4_NOTHROW
00486 { return (size + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t); }
00487
00492 enum l4_addr_consts_t {
00494     L4_INVALID_ADDR = ~0UL
00495 };
00496
00501 #define L4_INVALID_PTR ((void *)L4_INVALID_ADDR)
00502
00503 #ifndef NULL
00504 #ifdef __cplusplus
00505 # define NULL ((void *)0)
00509 #else
00510 # define NULL 0
00511 #endif
00512 #endif
00513
00514 #endif /* ! __L4_SYS__INCLUDE__CONSTS_H__ */

```

## 16.304 x86/l4/sys/consts.h File Reference

Common [L4](#) constants, x86 version.

### Macros

- `#define L4_PAGESHIFT 12`  
*Size of a page log2-based.*
- `#define L4_SUPERPAGESHIFT 22`  
*Size of a large page log2-based.*

### 16.304.1 Detailed Description

Common [L4](#) constants, x86 version.

Definition in file [consts.h](#).

## 16.305 consts.h

[Go to the documentation of this file.](#)

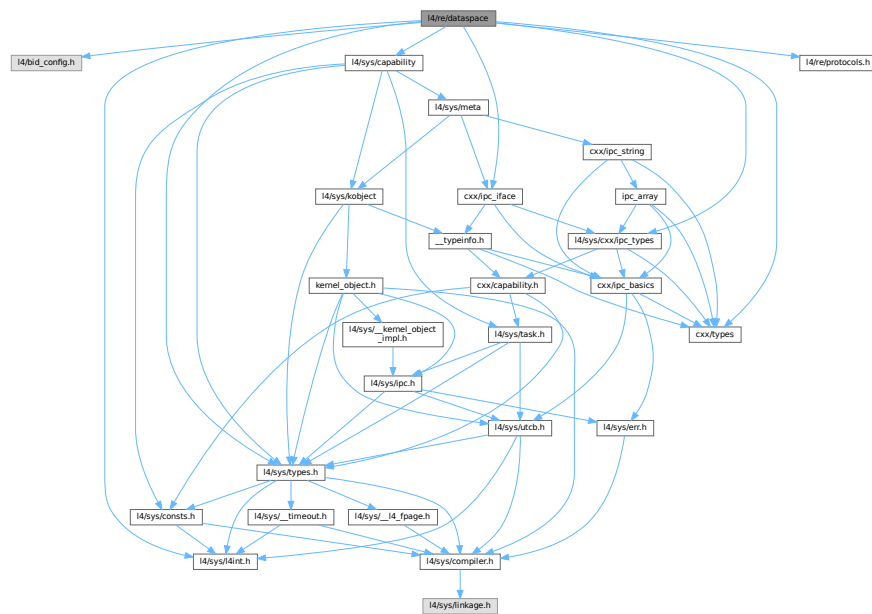
```
00001 /*****
00007 */
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011 *      Lars Reuther <reuther@os.inf.tu-dresden.de>
00012 *      economic rights: Technische Universität Dresden (Germany)
00013 *
00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU General Public License 2.
00016 * Please see the COPYING-GPL-2 file for details.
00017 *
00018 * As a special exception, you may use this file as part of a free software
00019 * library without restriction. Specifically, if other files instantiate
00020 * templates or use macros or inline functions from this file, or you compile
00021 * this file and link it with other files to produce an executable, this
00022 * file does not by itself cause the resulting executable to be covered by
00023 * the GNU General Public License. This exception does not however
00024 * invalidate any other reasons why the executable file might be covered by
00025 * the GNU General Public License.
00026 */
00027 /*****
00028 #ifndef __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__
00029 #define __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__
00030
00035 #define L4_PAGESHIFT      12
00036
00041 #define L4_SUPERPAGESHIFT 22
00042
00043 #include_next <l4/sys/consts.h>
00044
00045 #endif /* ! __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__ */
```



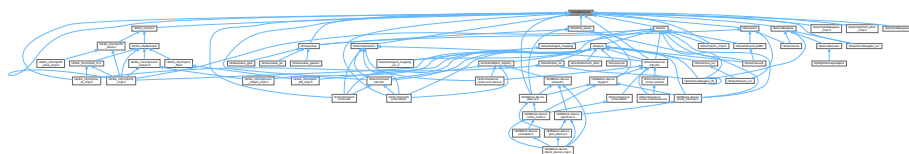
## 16.306 l4/re/dataspace File Reference

Dataspace interface.

```
#include <l4/bid_config.h>
#include <l4/sys/types.h>
#include <l4/sys/l4int.h>
#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/types>
Include dependency graph for dataspace:
```



This graph shows which files directly or indirectly include this file:



### Data Structures

- class [L4Re::Dataspace](#)  
*Interface for memory-like objects.*
- struct [L4Re::Dataspace::F](#)  
*Dataspace flags definitions.*
- struct [L4Re::Dataspace::Stats](#)  
*Information about the dataspace.*

## Namespaces

- namespace [L4Re](#)  
*[L4Re](#) C++ Interfaces.*

### 16.306.1 Detailed Description

Dataspace interface.

Definition in file [dataspace](#).

## 16.307 dataspace

[Go to the documentation of this file.](#)

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00005  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00006  *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00007  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 #include <l4/bid_config.h>
00026 #include <l4/sys/types.h>
00027 #include <l4/sys/l4int.h>
00028 #include <l4/sys/capability>
00029 #include <l4/re/protocols.h>
00030 #include <l4/sys/cxx/ipc_types>
00031 #include <l4/sys/cxx/ipc_iface>
00032 #include <l4/sys/cxx/types>
00033
00034 namespace L4Re
00035 {
00036     // MISSING:
00037     // * size support in map, mapped size in reply
00038
00039     class L4_EXPORT Dataspace :
00040     public L4::Kobject_t<Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE,
00041         L4::Type_info::Demand_t<1> >
00042     {
00043     public:
00044
00045         struct F
00046         {
00047             enum
00048             {
00049                 Caching_shift = 4,
00050             };
00051
00052             enum Flags
00053             {
00054                 R = L4_FPAGE_RO,
00055                 Ro = L4_FPAGE_RO,
00056                 RW = L4_FPAGE_RW,
00057                 W = L4_FPAGE_W,
00058                 X = L4_FPAGE_X,
```

```

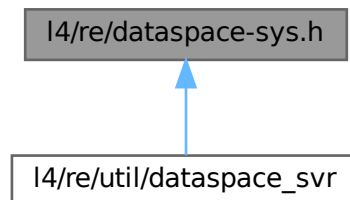
00094     RX = L4_FPAGE_RX,
00096     RWX = L4_FPAGE_RWX,
00098     Rights_mask = 0x0f,
00099
00101     Normal = 0x00,
00103     Cacheable = Normal,
00105     Bufferable = 0x10,
00107     Uncacheable = 0x20,
00109     Caching_mask = 0x30,
00110 };
00111
00112     L4_TYPES_FLAGS_OPS_DEF(Flags);
00113 };
00114
00115 struct Flags : L4::Types::Flags_ops_t<Flags>
00116 {
00117     unsigned long raw;
00118     Flags() = default;
00119     explicit constexpr Flags(unsigned long f) : raw(f) {}
00120     constexpr Flags(F::Flags f) : raw(f) {}
00121     constexpr bool r() const { return raw & L4_FPAGE_RO; }
00122     constexpr bool w() const { return raw & L4_FPAGE_W; }
00123     constexpr bool x() const { return raw & L4_FPAGE_X; }
00124
00125     constexpr unsigned long fpage_rights() const
00126     { return raw & 0xf; }
00127 };
00128
00129 typedef l4_uint64_t Size;
00130 typedef l4_uint64_t Offset;
00131 typedef l4_uint64_t Map_addr;
00132
00136 struct Stats
00137 {
00138     Size size;
00139     Flags flags;
00140 };
00141
00142
00167 long map(Offset offset, Flags flags, Map_addr local_addr,
00168         Map_addr min_addr, Map_addr max_addr,
00169         L4::Cap<L4::Task> dst = L4::Cap<L4::Task>::Invalid) const noexcept;
00170
00198 long map_region(Offset offset, Flags flags,
00199                Map_addr min_addr, Map_addr max_addr,
00200                L4::Cap<L4::Task> dst = L4::Cap<L4::Task>::Invalid) const noexcept;
00201
00219 L4_RPC(long, clear, (Offset offset, Size size));
00220
00240 L4_RPC(long, allocate, (Offset offset, Size size));
00241
00260 L4_RPC(long, copy_in, (Offset dst_offs, L4::Ipc::Cap<Dataspace> src,
00261                      Offset src_offs, Size size));
00262
00268 Size size() const noexcept;
00269
00278 Flags flags() const noexcept;
00279
00288 L4_RPC(long, info, (Stats *stats));
00289
00290 L4_RPC_NF(long, map, (Offset offset, Map_addr spot,
00291                    Flags flags, L4::Ipc::Rcv_fpage r,
00292                    L4::Ipc::Snd_fpage &fp));
00293
00313 #ifdef CONFIG_MMU
00314     L4_RPC_NF(long, map_info, (l4_addr_t *start_addr, l4_addr_t *end_addr));
00315     inline long map_info(l4_addr_t *, l4_addr_t *)
00316     { return 0; }
00317 #else
00318     L4_RPC(long, map_info, (l4_addr_t *start_addr, l4_addr_t *end_addr));
00319 #endif
00320
00321 private:
00322
00323     long __map(Offset offset, unsigned char *order, Flags flags,
00324               Map_addr local_addr, L4::Cap<L4::Task> dst) const noexcept;
00325
00326 public:
00327     typedef L4::Typeid::Rpcs<map_t, clear_t, info_t, copy_in_t,
00328                          allocate_t, map_info_t> Rpcs;
00329
00330 };
00331
00332 }
00333

```

## 16.308 I4/re/dataspace-sys.h File Reference

Dataspace protocol definition.

This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace [L4Re](#)  
*[L4Re C++ Interfaces.](#)*

### Enumerations

- enum [L4Re::Dataspace\\_::Opcodes](#)  
*Data-space communication-protocol opcodes.*

### 16.308.1 Detailed Description

Dataspace protocol definition.

Definition in file [dataspace-sys.h](#).

## 16.309 dataspace-sys.h

[Go to the documentation of this file.](#)

```

00001
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
  
```

```

00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 namespace L4Re
00026 {
00027     namespace Dataspace_
00028     {
00034         enum Opcodes { Map, Clear, Stats, Copy, Take, Release, Allocate };
00035     };
00036 };
00037

```

## 16.310 l4/re/debug File Reference

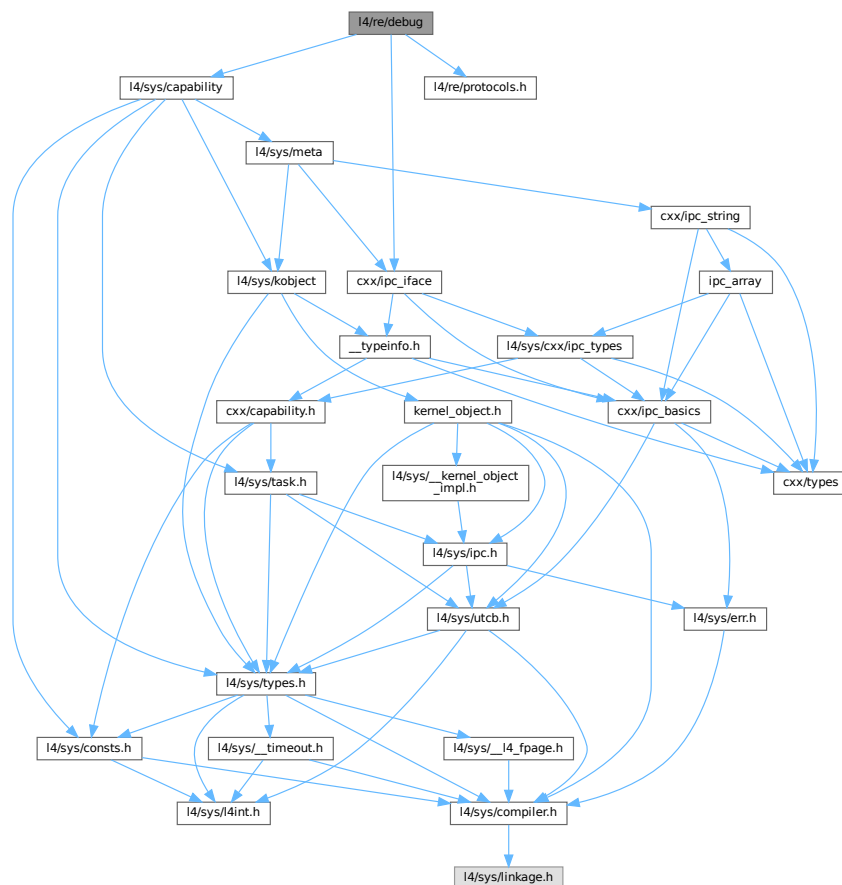
Debug interface.

```

#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/ipc_iface>

```

Include dependency graph for debug:



### Data Structures

- class [L4Re::Debug\\_obj](#)  
*Debug interface.*

## Namespaces

- namespace [L4Re](#)  
*[L4Re C++ Interfaces](#).*

### 16.310.1 Detailed Description

Debug interface.

Definition in file [debug](#).

## 16.311 debug

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/sys/capability>
00027 #include <l4/re/protocols.h>
00028 #include <l4/sys/cxx/ipc_iface>
00029
00030 namespace L4Re {
00051 class L4_EXPORT Debug_obj :
00052     public L4::Kobject_t<Debug_obj, L4::Kobject, L4RE_PROTO_DEBUG>
00053 {
00054 public:
00055
00067     L4_INLINE_RPC(long, debug, (unsigned long function));
00068     typedef L4::Typeid::Rpc_nocode<debug_t> Rpcs;
00069 };
00070
00071 template<typename BASE>
00072 class Debug_obj_t :
00073     public L4::Kobject_2t<Debug_obj_t<BASE>, BASE, Debug_obj, L4::PROTO_EMPTY>
00074 {
00075     typedef L4::Typeid::Rpcs<> Rpcs;
00076 };
00077 }
```

## 16.312 debug

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
```

```

00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/sys/types.h>
00028
00029 namespace L4Re { namespace Util {
00030 class Err
00031 {
00032 public:
00033     enum Level
00034     {
00035         Normal = 0,
00036         Fatal,
00037     };
00038
00039     static char const *const levels[];
00040
00041     void tag() const
00042     { printf("%s: %s", _component, levels[_l]); }
00043
00044     int printf(char const *fmt, ...) const
00045         __attribute__((format(printf, 2, 3)));
00046
00047     int cprintf(char const *fmt, ...) const
00048         __attribute__((format(printf, 2, 3)));
00049
00050     constexpr Err(Level l, char const *component) : _l(l), _component(component)
00051     {}
00052
00053 private:
00054     Level _l;
00055     char const *_component;
00056 };
00057
00058
00059 class Dbg
00060 {
00061 private:
00062     void tag() const;
00063
00064 #ifndef NDEBUG
00065
00066     unsigned long _m;
00067     char const *const _component;
00068     char const *const _subsys;
00069
00070 # if __clang__
00071
00072     int printf_impl(char const *fmt, ...) const
00073         __attribute__((format(printf, 2, 3)));
00074
00075     int cprintf_impl(char const *fmt, ...) const
00076         __attribute__((format(printf, 2, 3)));
00077
00078 # endif
00079
00080 public:
00081     static unsigned long level;
00082
00083     static void set_level(unsigned long l) { level = l; }
00084
00085     bool is_active() const { return _m & level; }
00086
00087 # if __clang__
00088
00089     int printf(char const *fmt, ...) const
00090         __attribute__((format(printf, 2, 3)));
00091
00092     int cprintf(char const *fmt, ...) const
00093         __attribute__((format(printf, 2, 3)));
00094
00095 # else
00096
00097     int __attribute__((always_inline, format(printf, 2, 3)))
00098     printf(char const *fmt, ...) const
00099     {
00100         if (!(level & _m))
00101             return 0;
00102     }

```

```

00103     return printf_impl(fmt, __builtin_va_arg_pack());
00104 }
00105
00106 int __attribute__((always_inline, format(printf, 2, 3)))
00107 cprintf(char const *fmt, ...) const
00108 {
00109     if (!(level & _m))
00110         return 0;
00111
00112     return cprintf_impl(fmt, __builtin_va_arg_pack());
00113 }
00114
00115 # endif
00116
00117 explicit constexpr
00118 Dbg() : _m(1), _component(0), _subsys(0) { };
00119
00120 explicit constexpr
00121 Dbg(unsigned long mask, char const *comp, char const *subs)
00122 : _m(mask), _component(comp), _subsys(subs)
00123 {}
00124
00125 #else
00126
00127 public:
00128     static void set_level(unsigned long) {}
00129     bool is_active() const { return false; }
00130
00131     int printf(char const * /*fmt*/, ...) const
00132         __attribute__((format(printf, 2, 3)))
00133     { return 0; }
00134
00135     int cprintf(char const * /*fmt*/, ...) const
00136         __attribute__((format(printf, 2, 3)))
00137     { return 0; }
00138
00139     explicit constexpr
00140     Dbg() {}
00141
00142     explicit constexpr
00143     Dbg(unsigned long, char const *, char const *) {}
00144
00145 #endif
00146
00147 };
00148
00149 }}
00150

```

## 16.313 l4/re/dma\_space File Reference

```

#include <l4/sys/types.h>
#include <l4/sys/l4int.h>
#include <l4/sys/capability>
#include <l4/re/dataspace>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/types>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>

```



[illegible]

```

graph TD
    I4libblockdeviceblock_device_mgrh["I4/libblock-device/block_device_mgr.h"] --> I4libblockdevicepart_deviceh["I4/libblock-device/part_device.h"]
    I4libblockdeviceblock_device_mgrh --> I4libblockdevicevirtio_clienth["I4/libblock-device/virtio_client.h"]
    I4libblockdeviceblock_device_mgrh --> I4libblockdeviceschedulerh["I4/libblock-device/scheduler.h"]
    I4libblockdevicepart_deviceh --> I4libblockdevicepartitionh["I4/libblock-device/partition.h"]
    I4libblockdevicepart_deviceh --> I4libblockdeviceinout_memoryh["I4/libblock-device/inout_memory.h"]
    I4libblockdevicevirtio_clienth --> I4libblockdevicedeviceh["I4/libblock-device/device.h"]
    I4libblockdevicevirtio_clienth --> I4re_dma_space["I4/re/dma_space"]
    I4libblockdeviceschedulerh --> I4libblockdevicedeviceh
    I4libblockdevicedeviceh --> I4libblockdevicetypesh["I4/libblock-device/types.h"]
    I4libblockdevicedeviceh --> I4re_dma_space
    I4re_dma_space --> I4vbus_vbus["I4/vbus/vbus"]
    I4vbus_vbus --> I4vbus_vbus_generic["I4/vbus/vbus_generic"]
    I4vbus_vbus --> I4vbus_vbus_gpio["I4/vbus/vbus_gpio"]
    I4vbus_vbus --> I4vbus_vbus_pci["I4/vbus/vbus_pci"]
  
```

- class `L4Re::Dma_space`  
*Managed DMA Address Space.*

- namespace **L4Re**  
*L4Re C++ Interfaces.*

## 16.314 dma\_space

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00006 /*
00007  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00025 #include <l4/sys/types.h>
00026 #include <l4/sys/l4int.h>
00027 #include <l4/sys/capability>
00028 #include <l4/re/dataspace>
00029 #include <l4/re/protocols.h>
00030 #include <l4/sys/cxx/types>
00031 #include <l4/sys/cxx/ipc_types>
00032 #include <l4/sys/cxx/ipc_iface>
00033
00034 namespace L4Re
00035 {
00036
00063 class Dma_space :
00064     public L4::Kobject_0t< Dma_space,
00065                          L4RE_PROTO_DMA_SPACE,
00066                          L4::Type_info::Demand_t<1> >
00067 {
00068 public:
00070     typedef l4_uint64_t Dma_addr;
00071
00075     enum Direction
00076     {
00077         Bidirectional,
00078         To_device,
00079         From_device,
00080         None
00081     };
00082
00087     enum Attribute
00088     {
00100         No_sync
00101     };
00102
00108     typedef L4::Types::Flags<Attribute> Attributes;
00109
00115     enum Space_attrib
00116     {
00123         Coherent,
00124
00129         Phys_space
00130     };
00131
00133     typedef L4::Types::Flags<Space_attrib> Space_attribs;
00134
00170     L4_INLINE_RPC(
00171         long, map, (L4::Ipc::Cap<L4Re::Dataspace> src,
00172                   L4Re::Dataspace::Offset offset,
00173                   L4::Ipc::In_out<l4_size_t *> size,
00174                   Attributes attrs, Direction dir,
00175                   Dma_addr *dma_addr));
00176
00189     L4_INLINE_RPC(
00190         long, unmap, (Dma_addr dma_addr,
00191                      l4_size_t size, Attributes attrs, Direction dir));
00192
00214     L4_INLINE_RPC(
00215         long, associate, (L4::Ipc::Opt<L4::Ipc::Cap<L4::Task> > dma_task,
00216                          Space_attribs attr),
00217         L4::Ipc::Call_t<L4_CAP_FPAGE_RW>);
00218
00229     L4_INLINE_RPC(

```

```

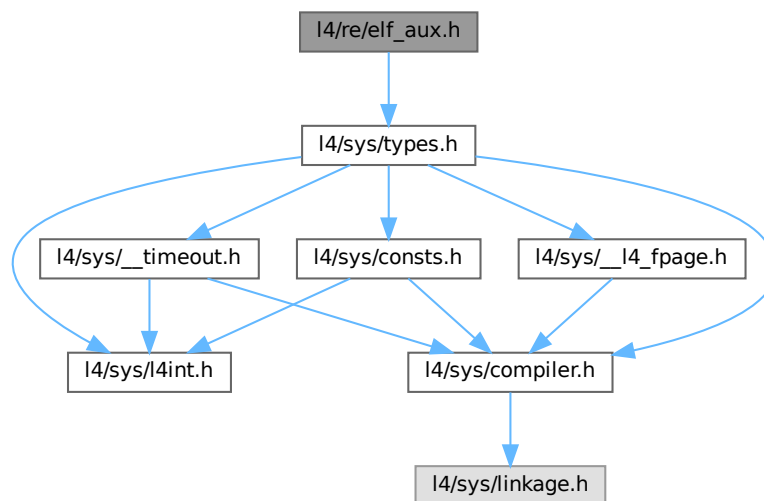
00230     long, disassociate, (),
00231     L4::Ipc::Call_t<L4_CAP_FPAGE_RW>);
00232
00233     typedef L4::Typeid::Rpc<map_t, unmap_t, associate_t, disassociate_t> Rpc;
00234 };
00235
00236 }

```

## 16.315 l4/re/elf\_aux.h File Reference

Auxiliary information for binaries.

```
#include <l4/sys/types.h>
Include dependency graph for elf_aux.h:
```



### Data Structures

- struct `l4re_elf_aux_t`  
*Generic header for each auxiliary vector element.*
- struct `l4re_elf_aux_vma_t`  
*Auxiliary vector element for a reserved virtual memory area.*
- struct `l4re_elf_aux_mword_t`  
*Auxiliary vector element for a single unsigned data word.*

### Macros

- `#define L4RE_ELF_AUX_ELEM` `const __attribute__((used, section(".ro.l4re_elf_aux"), aligned(sizeof(l4_umword_t))))`  
*Define an auxiliary vector element.*
- `#define L4RE_ELF_AUX_ELEM_T`(type, id, tag, val...) `static L4RE_ELF_AUX_ELEM type id = {tag, sizeof(type), val}`  
*Define an auxiliary vector element.*

## Typedefs

- typedef struct [l4re\\_elf\\_aux\\_t](#) [l4re\\_elf\\_aux\\_t](#)  
*Generic header for each auxiliary vector element.*
- typedef struct [l4re\\_elf\\_aux\\_vma\\_t](#) [l4re\\_elf\\_aux\\_vma\\_t](#)  
*Auxiliary vector element for a reserved virtual memory area.*
- typedef struct [l4re\\_elf\\_aux\\_mword\\_t](#) [l4re\\_elf\\_aux\\_mword\\_t](#)  
*Auxiliary vector element for a single unsigned data word.*

## Enumerations

- enum {  
[L4RE\\_ELF\\_AUX\\_T\\_NONE](#) = 0 , [L4RE\\_ELF\\_AUX\\_T\\_VMA](#) , [L4RE\\_ELF\\_AUX\\_T\\_STACK\\_SIZE](#) ,  
[L4RE\\_ELF\\_AUX\\_T\\_STACK\\_ADDR](#) ,  
[L4RE\\_ELF\\_AUX\\_T\\_KIP\\_ADDR](#) , [L4RE\\_ELF\\_AUX\\_T\\_EX\\_REGS\\_FLAGS](#) }

### 16.315.1 Detailed Description

Auxiliary information for binaries.

Definition in file [elf\\_aux.h](#).

## 16.316 elf\_aux.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #pragma once
00023
00024 #include <l4/sys/types.h>
00025
00026
00052 #define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".ro.l4re_elf_aux"),
    aligned(sizeof(l4_umword_t))))
00053
00067 #define L4RE_ELF_AUX_ELEM_T(type, id, tag, val...) \
00068     static L4RE_ELF_AUX_ELEM type id = {tag, sizeof(type), val}
00069
00070 enum
00071 {
00075     L4RE\_ELF\_AUX\_T\_NONE = 0,
00076
00080     L4RE\_ELF\_AUX\_T\_VMA,
00081
00086     L4RE\_ELF\_AUX\_T\_STACK\_SIZE,
00087
00092     L4RE\_ELF\_AUX\_T\_STACK\_ADDR,
00093
00098     L4RE\_ELF\_AUX\_T\_KIP\_ADDR,

```

```

00099
00103     L4RE_ELF_AUX_T_EX_REGS_FLAGS,
00104 };
00105
00109 typedef struct l4re_elf_aux_t
00110 {
00111     l4_umword_t type;
00112     l4_umword_t length;
00113 } l4re_elf_aux_t;
00114
00118 typedef struct l4re_elf_aux_vma_t
00119 {
00120     l4_umword_t type;
00121     l4_umword_t length;
00122     l4_umword_t start;
00123     l4_umword_t end;
00124 } l4re_elf_aux_vma_t;
00125
00129 typedef struct l4re_elf_aux_mword_t
00130 {
00131     l4_umword_t type;
00132     l4_umword_t length;
00133     l4_umword_t value;
00134 } l4re_elf_aux_mword_t;
00135

```

## 16.317 l4/re/env File Reference

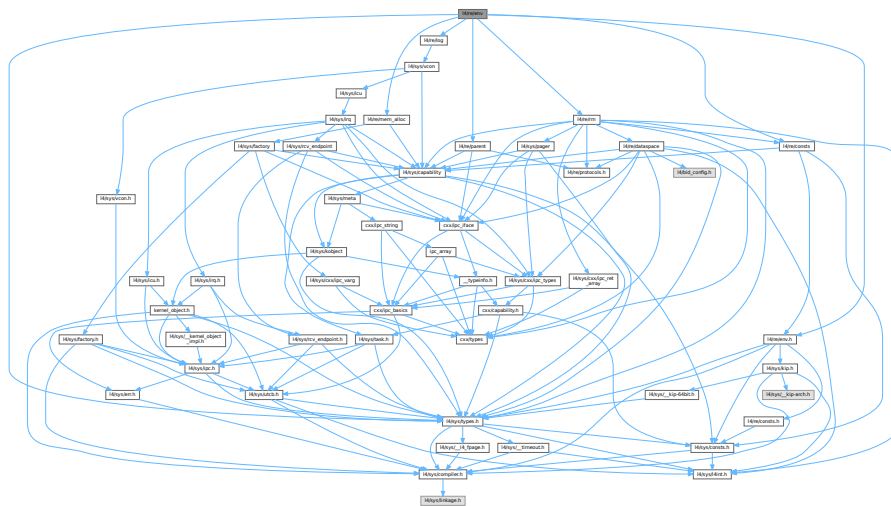
Environment interface.

```

#include <l4/sys/types.h>
#include <l4/re/rm>
#include <l4/re/parent>
#include <l4/re/mem_alloc>
#include <l4/re/log>
#include <l4/re/consts>
#include <l4/re/env.h>

```

Include dependency graph for env:





```

00029
00030 #include <l4/re/rm>
00031 #include <l4/re/parent>
00032 #include <l4/re/mem_alloc>
00033 #include <l4/re/log>
00034 #include <l4/re/consts>
00035
00036 #include <l4/re/env.h>
00037
00038 namespace L4 {
00039 class Scheduler;
00040 }
00041
00042 namespace L4Re
00043 {
00044     class L4_EXPORT Env
00045     {
00046     private:
00047         l4re_env_t _env;
00048     public:
00049
00050         typedef l4re_env_cap_entry_t Cap_entry;
00051
00052         static Env const *env() noexcept
00053         { return reinterpret_cast<Env*>(l4re_global_env); }
00054
00055         L4::Cap<Parent> parent() const noexcept
00056         { return L4::Cap<Parent>(_env.parent); }
00057         L4::Cap<Mem_alloc> mem_alloc() const noexcept
00058         { return L4::Cap<Mem_alloc>(_env.mem_alloc); }
00059         L4::Cap<L4::Factory> user_factory() const noexcept
00060         { return L4::Cap<L4::Factory>(_env.mem_alloc); }
00061         L4::Cap<Rm> rm() const noexcept
00062         { return L4::Cap<Rm>(_env.rm); }
00063         L4::Cap<Log> log() const noexcept
00064         { return L4::Cap<Log>(_env.log); }
00065         L4::Cap<L4::Thread> main_thread() const noexcept
00066         { return L4::Cap<L4::Thread>(_env.main_thread); }
00067         L4::Cap<L4::Task> task() const noexcept
00068         { return L4::Cap<L4::Task>(L4RE_THIS_TASK_CAP); }
00069         L4::Cap<L4::Factory> factory() const noexcept
00070         { return L4::Cap<L4::Factory>(_env.factory); }
00071         l4_cap_idx_t first_free_cap() const noexcept
00072         { return _env.first_free_cap; }
00073         l4_fpage_t utcb_area() const noexcept
00074         { return _env.utcb_area; }
00075         l4_addr_t first_free_utcb() const noexcept
00076         { return _env.first_free_utcb; }
00077
00078         Cap_entry const *initial_caps() const noexcept
00079         { return _env.caps; }
00080
00081         Cap_entry const *get(char const *name, unsigned l) const noexcept
00082         { return l4re_env_get_cap_l(name, l, &_env); }
00083
00084         template< typename T >
00085         L4::Cap<T> get_cap(char const *name, unsigned l) const noexcept
00086         {
00087             if (Cap_entry const *e = get(name, l))
00088                 return L4::Cap<T>(e->cap);
00089             return L4::Cap<T>(-L4_ENOENT);
00090         }
00091
00092         template< typename T >
00093         L4::Cap<T> get_cap(char const *name) const noexcept
00094         { return get_cap<T>(name, __builtin_strlen(name)); }
00095
00096         void parent(L4::Cap<Parent> const &c) noexcept
00097         { _env.parent = c.cap(); }
00098         void mem_alloc(L4::Cap<Mem_alloc> const &c) noexcept
00099         { _env.mem_alloc = c.cap(); }
00100         void rm(L4::Cap<Rm> const &c) noexcept
00101         { _env.rm = c.cap(); }
00102         void log(L4::Cap<Log> const &c) noexcept
00103         { _env.log = c.cap(); }
00104         void main_thread(L4::Cap<L4::Thread> const &c) noexcept
00105         { _env.main_thread = c.cap(); }
00106         void factory(L4::Cap<L4::Factory> const &c) noexcept
00107         { _env.factory = c.cap(); }
00108         void first_free_cap(l4_cap_idx_t c) noexcept
00109         { _env.first_free_cap = c; }
00110         void utcb_area(l4_fpage_t utcb) noexcept
00111         { _env.utcb_area = utcb; }
00112         void first_free_utcb(l4_addr_t u) noexcept
00113         { _env.first_free_utcb = u; }
00114     };
00115 }

```

```

00282     L4::Cap<L4::Scheduler> scheduler() const noexcept
00283     { return L4::Cap<L4::Scheduler>(_env.scheduler); }
00284
00289     void scheduler(L4::Cap<L4::Scheduler> const &c) noexcept
00290     { _env.scheduler = c.cap(); }
00291
00296     void initial_caps(Cap_entry *first) noexcept
00297     { _env.caps = first; }
00298 };
00299 };

```

## 16.319 I4/re/env.h File Reference

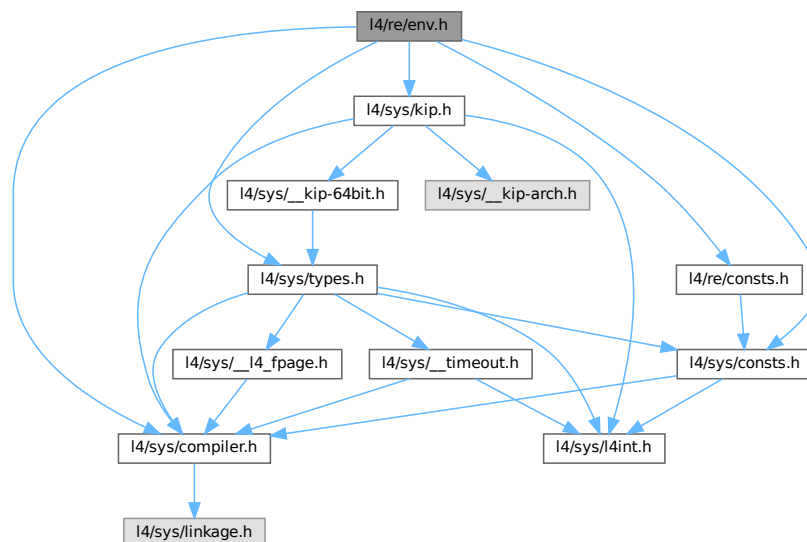
Environment interface.

```

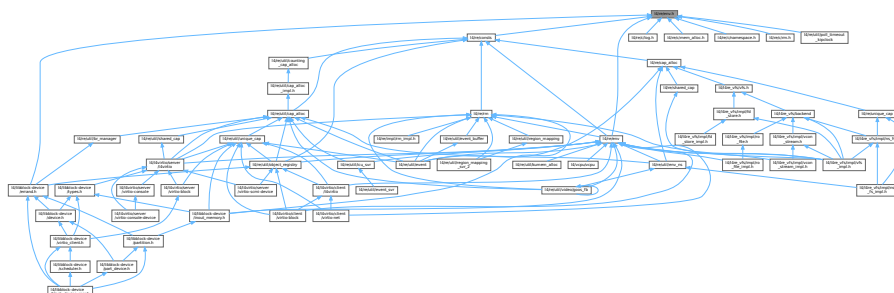
#include <l4/sys/consts.h>
#include <l4/sys/types.h>
#include <l4/sys/kip.h>
#include <l4/sys/compiler.h>
#include <l4/re/consts.h>

```

Include dependency graph for env.h:



This graph shows which files directly or indirectly include this file:





## Data Structures

- struct [l4re\\_env\\_cap\\_entry\\_t](#)  
*Entry in the [L4Re](#) environment array for the named initial objects.*
- struct [l4re\\_env\\_t](#)  
*Initial environment data structure.*

## Typedefs

- typedef struct [l4re\\_env\\_cap\\_entry\\_t](#) [l4re\\_env\\_cap\\_entry\\_t](#)  
*Entry in the [L4Re](#) environment array for the named initial objects.*
- typedef struct [l4re\\_env\\_t](#) [l4re\\_env\\_t](#)  
*Initial environment data structure.*

## Functions

- [l4re\\_env\\_t](#) \* [l4re\\_env](#) (void) [L4\\_NOTHROW](#)  
*Get [L4Re](#) initial environment.*
- [l4\\_kernel\\_info\\_t](#) const \* [l4re\\_kip](#) (void) [L4\\_NOTHROW](#)  
*Get Kernel Info Page.*
- [l4\\_cap\\_idx\\_t](#) [l4re\\_env\\_get\\_cap](#) (char const \*name) [L4\\_NOTHROW](#)  
*Get the capability selector for the object named name.*
- [l4\\_cap\\_idx\\_t](#) [l4re\\_env\\_get\\_cap\\_e](#) (char const \*name, [l4re\\_env\\_t](#) const \*e) [L4\\_NOTHROW](#)  
*Get the capability selector for the object named name.*
- [l4re\\_env\\_cap\\_entry\\_t](#) const \* [l4re\\_env\\_get\\_cap\\_l](#) (char const \*name, unsigned l, [l4re\\_env\\_t](#) const \*e) [L4\\_NOTHROW](#)  
*Get the full [l4re\\_env\\_cap\\_entry\\_t](#) for the object named name.*

### 16.319.1 Detailed Description

Environment interface.

Definition in file [env.h](#).

### 16.319.2 Typedef Documentation

#### 16.319.2.1 [l4re\\_env\\_t](#)

```
typedef struct l4re\_env\_t l4re\_env\_t
```

Initial environment data structure.

See also

[Initial environment](#)

## 16.320 env.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 #include <l4/sys/consts.h>
00026 #include <l4/sys/types.h>
00027 #include <l4/sys/kip.h>
00028 #include <l4/sys/compiler.h>
00029
00030 #include <l4/re/consts.h>
00031
00050 typedef struct l4re_env_cap_entry_t
00051 {
00055     l4_cap_idx_t cap;
00056
00061     l4_umword_t flags;
00062
00066     char name[16];
00067 #ifdef __cplusplus
00068
00072     l4re_env_cap_entry_t() L4_NOTHROW : cap(L4_INVALID_CAP), flags(~0UL) {}
00073
00081     l4re_env_cap_entry_t(char const *n, l4_cap_idx_t c, l4_umword_t f = 0) L4_NOTHROW
00082     : cap(c), flags(f)
00083     {
00084         for (unsigned i = 0; n && i < sizeof(name); ++i, ++n)
00085         {
00086             name[i] = *n;
00087             if (!*n)
00088                 break;
00089         }
00090     }
00091
00092     static bool is_valid_name(char const *n) L4_NOTHROW
00093     {
00094         for (unsigned i = 0; *n; ++i, ++n)
00095             if (i > sizeof(name))
00096                 return false;
00097         return true;
00098     }
00099 } #endif
00100 } l4re_env_cap_entry_t;
00101
00102
00103
00109 typedef struct l4re_env_t
00110 {
00111     l4_cap_idx_t parent;
00112     l4_cap_idx_t rm;
00113     l4_cap_idx_t mem_alloc;
00114     l4_cap_idx_t log;
00115     l4_cap_idx_t main_thread;
00116     l4_cap_idx_t factory;
00117     l4_cap_idx_t scheduler;
00118     l4_cap_idx_t first_free_cap;
00119     l4_fpage_t utcb_area;
00120     l4_addr_t first_free_utcb;
00126     l4re_env_cap_entry_t *caps;
00127 } l4re_env_t;
00128
00134 extern l4re_env_t *l4re_global_env;
00135
00136
00142 L4_INLINE l4re_env_t *l4re_env(void) L4_NOTHROW;
00143

```

```

00144  /*
00145   * FIXME: this seems to be at the wrong place here
00146   */
00152  L4_INLINE l4_kernel_info_t const *l4re_kip(void) L4_NOTHROW;
00153
00154
00162  L4_INLINE l4_cap_idx_t
00163  l4re_env_get_cap(char const *name) L4_NOTHROW;
00164
00173  L4_INLINE l4_cap_idx_t
00174  l4re_env_get_cap_e(char const *name, l4re_env_t const *e) L4_NOTHROW;
00175
00186  L4_INLINE l4re_env_cap_entry_t const *
00187  l4re_env_get_cap_l(char const *name, unsigned l, l4re_env_t const *e) L4_NOTHROW;
00188
00189  L4_INLINE
00190  l4re_env_t *l4re_env(void) L4_NOTHROW
00191  { return l4re_global_env; }
00192
00193  L4_INLINE
00194  l4_kernel_info_t const *l4re_kip(void) L4_NOTHROW
00195  { return l4_kip(); }
00196
00197  L4_INLINE l4re_env_cap_entry_t const *
00198  l4re_env_get_cap_l(char const *name, unsigned l, l4re_env_t const *e) L4_NOTHROW
00199  {
00200      l4re_env_cap_entry_t const *c = e->caps;
00201      for (; c && c->flags != ~0UL; ++c)
00202      {
00203          unsigned i;
00204          for (i = 0;
00205               i < sizeof(c->name) && i < l && c->name[i] && name[i] && name[i] == c->name[i];
00206               ++i)
00207              ;
00208
00209          if (i == l && (i == sizeof(c->name) || !c->name[i]))
00210              return c;
00211      }
00212      return NULL;
00213  }
00214
00215  L4_INLINE l4_cap_idx_t
00216  l4re_env_get_cap_e(char const *name, l4re_env_t const *e) L4_NOTHROW
00217  {
00218      unsigned l;
00219      l4re_env_cap_entry_t const *r;
00220      for (l = 0; name[l]; ++l) ;
00221      r = l4re_env_get_cap_l(name, l, e);
00222      if (r)
00223          return r->cap;
00224
00225      return L4_INVALID_CAP;
00226  }
00227
00228  L4_INLINE l4_cap_idx_t
00229  l4re_env_get_cap(char const *name) L4_NOTHROW
00230  { return l4re_env_get_cap_e(name, l4re_env()); }
00231

```

## 16.321 l4/re/error\_helper File Reference

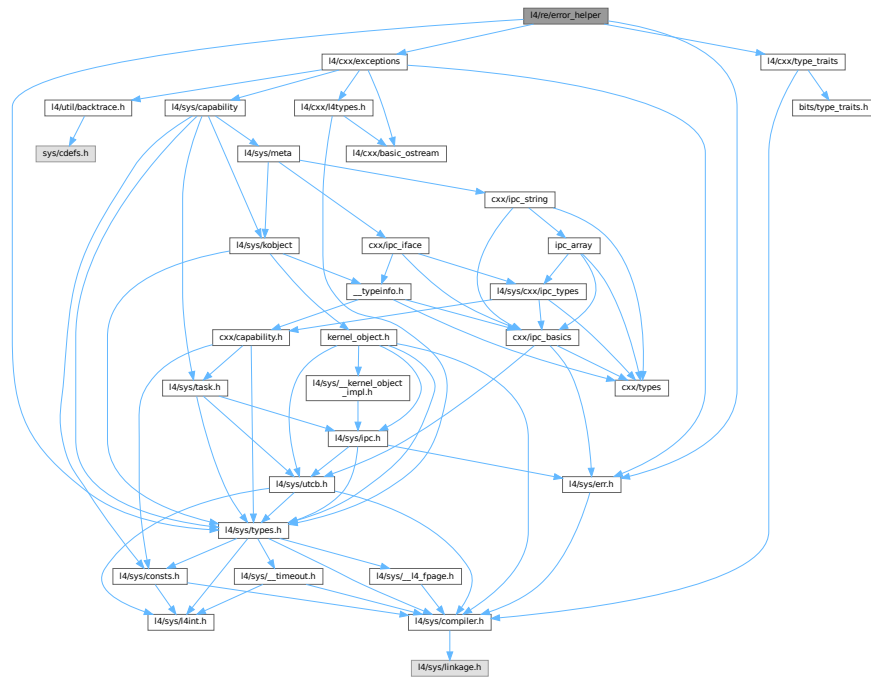
Error helper.

```

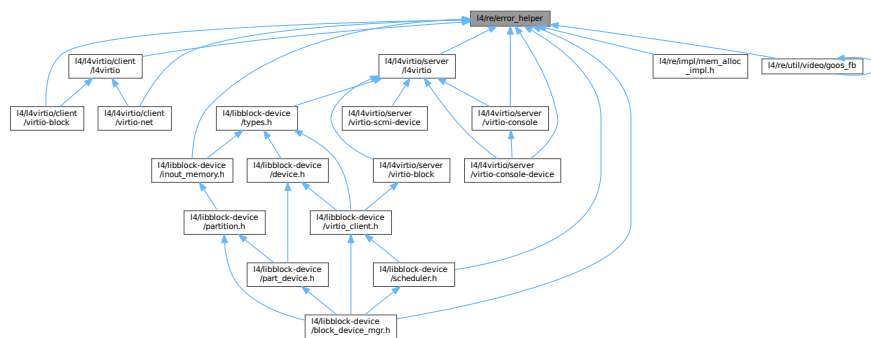
#include <l4/sys/types.h>
#include <l4/cxx/exceptions>
#include <l4/cxx/type_traits>
#include <l4/sys/err.h>

```

Include dependency graph for error\_helper:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace **L4Re**  
*L4Re C++ Interfaces.*

## Functions

- void `L4Re::throw_error` (long err, char const \*extra="")  
*Generate C++ exception.*
- long `L4Re::chksys` (long err, char const \*extra="", long ret=0)  
*Generate C++ exception on error.*

- `long L4Re::chksys (l4_msgtag_t const &t, char const *extra="", l4_utcb_t *utcb=l4_utcb(), long ret=0)`  
Generate C++ exception on error.
- `long L4Re::chksys (l4_msgtag_t const &t, l4_utcb_t *utcb, char const *extra="")`  
Generate C++ exception on error.
- `template<typename T>`  
`T L4Re::chkcap (T &&cap, char const *extra="", long err=-L4_ENOMEM)`  
Check for valid capability or raise C++ exception.
- `l4_msgtag_t L4Re::chkipc (l4_msgtag_t tag, char const *extra="", l4_utcb_t *utcb=l4_utcb())`  
Test a message tag for IPC errors.

### 16.321.1 Detailed Description

Error helper.

Definition in file [error\\_helper](#).

## 16.322 error\_helper

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *               economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/sys/types.h>
00028 #include <l4/cxx/exceptions>
00029 #include <l4/cxx/type_traits>
00030 #include <l4/sys/err.h>
00031
00032 namespace L4Re {
00033
00034 #ifdef __EXCEPTIONS
00035
00045 [[noreturn]] inline void throw_error(long err, char const *extra = "")
00046 {
00047     switch (err)
00048     {
00049     case -L4_ENOENT: throw (L4::Element_not_found(extra));
00050     case -L4_ENOMEM: throw (L4::Out_of_memory(extra));
00051     case -L4_EEXIST: throw (L4::Element_already_exists(extra));
00052     case -L4_ERANGE: throw (L4::Bounds_error(extra));
00053     default: throw (L4::Runtime_error(err, extra));
00054     }
00055 }
00056
00067 inline
00068 long chksys(long err, char const *extra = "", long ret = 0)
00069 {
00070     if (L4_UNLIKELY(err < 0))
00071         throw_error(ret ? ret : err, extra);
00072     return err;
00073 }
00074 }
```

```

00075
00088 inline
00089 long chksys(l4_msgtag_t const &t, char const *extra = "",
00090             l4_utcb_t *utcb = l4_utcb(), long ret = 0)
00091 {
00092     if (L4_UNLIKELY(t.has_error()))
00093         throw_error(ret ? ret : l4_error_u(t, utcb), extra);
00094     else if (L4_UNLIKELY(t.label() < 0))
00095         throw_error(ret ? ret : t.label(), extra);
00096
00097     return t.label();
00098 }
00099
00111 inline
00112 long chksys(l4_msgtag_t const &t, l4_utcb_t *utcb, char const *extra = "")
00113 { return chksys(t, extra, utcb); }
00114
00115 #if 0
00116 inline
00117 long chksys(long ret, long err, char const *extra = "")
00118 {
00119     if (L4_UNLIKELY(ret < 0))
00120         throw_error(err, extra);
00121
00122     return ret;
00123 }
00124 #endif
00125
00142 template<typename T>
00143 inline
00144 #if __cplusplus >= 201103L
00145 T chkcap(T &&cap, char const *extra = "", long err = -L4_ENOMEM)
00146 #else
00147 T chkcap(T cap, char const *extra = "", long err = -L4_ENOMEM)
00148 #endif
00149 {
00150     if (L4_UNLIKELY(!cap.is_valid()))
00151         throw_error(err ? err : cap.cap(), extra);
00152
00153 #if __cplusplus >= 201103L
00154     return cxx::forward<T>(cap);
00155 #else
00156     return cap;
00157 #endif
00158 }
00159
00174 inline
00175 l4_msgtag_t
00176 chkipc(l4_msgtag_t tag, char const *extra = "",
00177         l4_utcb_t *utcb = l4_utcb())
00178 {
00179     if (L4_UNLIKELY(tag.has_error()))
00180         chksys(l4_error_u(tag, utcb), extra);
00181
00182     return tag;
00183 }
00184 #endif
00185
00186 }

```

## 16.323 event

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020
00021 #pragma once

```

```

00022
00023 #include <l4/sys/capability>
00024 #include <l4/sys/irq>
00025 #include <l4/sys/cxx/ipc_iface>
00026 #include <l4/sys/cxx/ipc_array>
00027 #include <l4/re/dataspace>
00028 #include <l4/re/event.h>
00029
00030 namespace L4Re {
00031
00051 typedef l4re_event_stream_id_t Event_stream_id;
00052 typedef l4re_event_absinfo_t Event_absinfo;
00053
00054 class L4_EXPORT Event_stream_bitmap_h
00055 {
00056 protected:
00057     static unsigned __get_idx(unsigned idx)
00058     { return idx / (sizeof(unsigned long)*8); }
00059
00060     static unsigned long __get_mask(unsigned idx)
00061     { return 1ul < (idx % (sizeof(unsigned long)*8)); }
00062
00063     static bool __get_bit(unsigned long const *bm, unsigned max, unsigned idx)
00064     {
00065         if (idx <= max)
00066             return bm[__get_idx(idx)] & __get_mask(idx);
00067         return false;
00068     }
00069
00070     static void __set_bit(unsigned long *bm, unsigned max, unsigned idx, bool v)
00071     {
00072         if (idx > max)
00073             return;
00074
00075         if (v)
00076             bm[__get_idx(idx)] |= __get_mask(idx);
00077         else
00078             bm[__get_idx(idx)] &= ~__get_mask(idx);
00079     }
00080 };
00081
00082 class L4_EXPORT Event_stream_info
00083 : public l4re_event_stream_info_t,
00084   private Event_stream_bitmap_h
00085 {
00086 public:
00087     bool get_propbit(unsigned idx) const
00088     { return __get_bit(propbits, L4RE_EVENT_PROP_MAX, idx); }
00089
00090     void set_propbit(unsigned idx, bool v)
00091     { __set_bit(propbits, L4RE_EVENT_PROP_MAX, idx, v); }
00092
00093     bool get_evbit(unsigned idx) const
00094     { return __get_bit(evbits, L4RE_EVENT_EV_MAX, idx); }
00095
00096     void set_evbit(unsigned idx, bool v)
00097     { __set_bit(evbits, L4RE_EVENT_EV_MAX, idx, v); }
00098
00099     bool get_keybit(unsigned idx) const
00100     { return __get_bit(keybits, L4RE_EVENT_KEY_MAX, idx); }
00101
00102     void set_keybit(unsigned idx, bool v)
00103     { __set_bit(keybits, L4RE_EVENT_KEY_MAX, idx, v); }
00104
00105     bool get_relbit(unsigned idx) const
00106     { return __get_bit(relbits, L4RE_EVENT_REL_MAX, idx); }
00107
00108     void set_relbit(unsigned idx, bool v)
00109     { __set_bit(relbits, L4RE_EVENT_REL_MAX, idx, v); }
00110
00111     bool get_absbit(unsigned idx) const
00112     { return __get_bit(absbits, L4RE_EVENT_ABS_MAX, idx); }
00113
00114     void set_absbit(unsigned idx, bool v)
00115     { __set_bit(absbits, L4RE_EVENT_ABS_MAX, idx, v); }
00116
00117     bool get_swbit(unsigned idx) const
00118     { return __get_bit(swbits, L4RE_EVENT_SW_MAX, idx); }
00119
00120     void set_swbit(unsigned idx, bool v)
00121     { __set_bit(swbits, L4RE_EVENT_SW_MAX, idx, v); }
00122 };
00123
00124 class L4_EXPORT Event_stream_state
00125 : public l4re_event_stream_state_t,
00126   private Event_stream_bitmap_h
00127 {

```

```

00128 public:
00129     bool get_keybit(unsigned idx) const
00130     { return __get_bit(keybits, L4RE_EVENT_KEY_MAX, idx); }
00131
00132     void set_keybit(unsigned idx, bool v)
00133     { __set_bit(keybits, L4RE_EVENT_KEY_MAX, idx, v); }
00134
00135     bool get_swbit(unsigned idx) const
00136     { return __get_bit(swbits, L4RE_EVENT_SW_MAX, idx); }
00137
00138     void set_swbit(unsigned idx, bool v)
00139     { __set_bit(swbits, L4RE_EVENT_SW_MAX, idx, v); }
00140 };
00141
00142 class L4_EXPORT Event :
00143     public L4::Kobject_t<Event, L4::Icu, L4RE_PROTO_EVENT>
00144 {
00151 public:
00160     L4_RPC(long, get_buffer, (L4::Ipc::Out<L4::Cap<Dataspace> > ds));
00161
00168     L4_RPC(long, get_num_streams, ());
00169
00181     L4_RPC(long, get_stream_info, (int idx, Event_stream_info *info));
00182
00192     L4_RPC(long, get_stream_info_for_id, (l4_umword_t stream_id, Event_stream_info *info));
00193
00204     L4_RPC_NF(long, get_axis_info, (l4_umword_t stream_id,
00205                                     L4::Ipc::Array<unsigned const, unsigned long> axes,
00206                                     L4::Ipc::Array<Event_absinfo, unsigned long> &info));
00207
00208     long get_axis_info(l4_umword_t stream_id, unsigned naxes,
00209                        unsigned const *axis, Event_absinfo *info) const noexcept
00210     {
00211         L4::Ipc::Array<Event_absinfo, unsigned long> i(naxes, info);
00212         return get_axis_info_t::call(c(), stream_id,
00213                                     L4::Ipc::Array<unsigned const, unsigned long>(naxes, axis), i);
00214     }
00215
00225     L4_RPC(long, get_stream_state_for_id, (l4_umword_t stream_id,
00226                                           Event_stream_state *state));
00227
00228     typedef L4::Typeid::Rpc<
00229         get_buffer_t,
00230         get_num_streams_t,
00231         get_stream_info_t,
00232         get_stream_info_for_id_t,
00233         get_axis_info_t,
00234         get_stream_state_for_id_t
00235     > Rpc<
00236     >;
00237
00242 struct L4_EXPORT Default_event_payload
00243 {
00244     unsigned short type;
00245     unsigned short code;
00246     int value;
00247     l4_umword_t stream_id;
00248 };
00249
00250
00255 template< typename PAYLOAD = Default_event_payload >
00256 class L4_EXPORT Event_buffer_t
00257 {
00258 public:
00259
00263     struct Event
00264     {
00265         long long time;
00266         PAYLOAD payload;
00267
00271         void free() noexcept { l4_mb(); time = 0; }
00272     };
00273
00274 private:
00275     Event *_current;
00276     Event *_begin;
00277     Event const *_end;
00278
00279     void inc() noexcept
00280     {
00281         ++_current;
00282         if (_current == _end)
00283             _current = _begin;
00284     }
00285
00286 public:
00287

```



```

00288 Event_buffer_t() : _current(0), _begin(0), _end(0) {}
00289
00290 void reset()
00291 {
00292     for (Event *i = _begin; i != _end; ++i)
00293         i->time = 0;
00294     _current = _begin;
00295 }
00296
00303 Event_buffer_t(void *buffer, l4_addr_t size)
00304 : _current(static_cast<Event*>(buffer)), _begin(_current),
00305   _end(_begin + size / sizeof(Event))
00306 { reset(); }
00307
00313 Event *next() noexcept
00314 {
00315     Event *c = _current;
00316     if (c->time)
00317     {
00318         inc();
00319         return c;
00320     }
00321     return 0;
00322 }
00323
00330 bool put(Event const &ev) noexcept
00331 {
00332     Event *c = _current;
00333     if (c->time)
00334         return false;
00335
00336     inc();
00337     c->payload = ev.payload;
00338     l4_wmb();
00339     c->time = ev.time;
00340     return true;
00341 }
00342 };
00343
00344 typedef Event_buffer_t<Default_event_payload> Event_buffer;
00345
00346 }

```

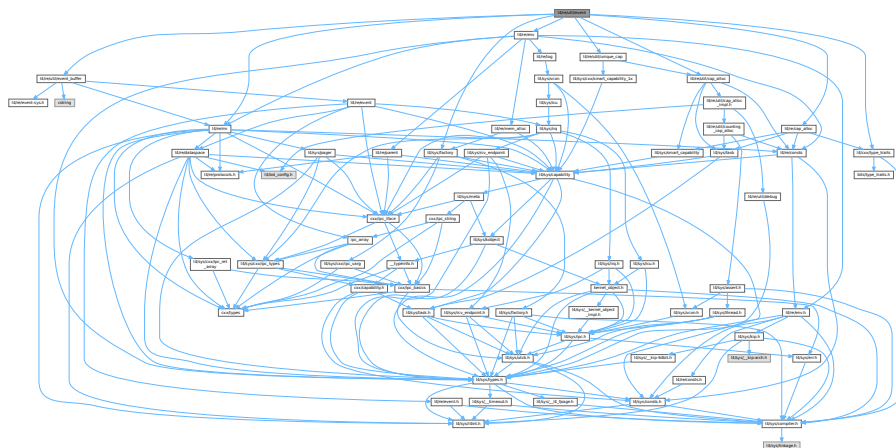
## 16.324 l4/re/util/event File Reference

```

#include <l4/re/cap_alloc>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/unique_cap>
#include <l4/re/env>
#include <l4/re/rm>
#include <l4/re/util/event_buffer>
#include <l4/sys/factory>
#include <l4/cxx/type_traits>

```

Include dependency graph for event:



## Data Structures

- class [L4Re::Util::Event\\_t< PAYLOAD >](#)  
*Convenience wrapper for getting access to an event object.*

## Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*
- namespace [L4Re::Util](#)  
*Documentation of the [L4](#) Runtime Environment utility functionality in C++.*

## 16.325 event

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 #include <l4/re/cap_alloc>
00026 #include <l4/re/util/cap_alloc>
00027 #include <l4/re/util/unique_cap>
00028 #include <l4/re/env>
00029 #include <l4/re/rm>
00030 #include <l4/re/util/event_buffer>
00031 #include <l4/sys/factory>
00032 #include <l4/cxx/type_traits>
00033
00034 namespace L4Re { namespace Util {
00035
00042 template< typename PAYLOAD >
00043 class Event_t
00044 {
00045 public:
00049     enum Mode
00050     {
00051         Mode_irq,
00052         Mode_polling,
00053     };
00054
00069     template<typename IRQ_TYPE>
00070     int init(L4::Cap<L4Re::Event> event,
00071             L4Re::Env const *env = L4Re::Env::env(),
00072             L4Re::Cap_alloc *ca = L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc))
00073     {
00074         Unique_cap<L4Re::Dataspace> ev_ds(ca->alloc<L4Re::Dataspace>());
00075         if (!ev_ds.is_valid())
00076             return -L4_ENOMEM;
00077
00078         int r;
00079
00080         Unique_del_cap<IRQ_TYPE> ev_irq(ca->alloc<IRQ_TYPE>());
00081         if (!ev_irq.is_valid())
00082             return -L4_ENOMEM;
00083
00084         if ((r = l4_error(env->factory()->create(ev_irq.get()))))
```

```

00085         return r;
00086
00087     if ((r = L4_error(event->bind(0, ev_irq.get()))))
00088         return r;
00089
00090     if ((r = event->get_buffer(ev_ds.get()))
00091         return r;
00092
00093     long sz = ev_ds->size();
00094     if (sz < 0)
00095         return sz;
00096
00097     Rm::Unique_region<void*> buf;
00098
00099     if ((r = env->rm()->attach(&buf, sz,
00100                             L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00101                             L4::Ipc::make_cap_rw(ev_ds.get()))))
00102         return r;
00103
00104     _ev_buffer = L4Re::Event_buffer_t<PAYLOAD>(buf.get(), sz);
00105     _ev_ds      = cxx::move(ev_ds);
00106     _ev_irq     = cxx::move(ev_irq);
00107     _buf        = cxx::move(buf);
00108
00109     return 0;
00110 }
00111
00123 int init_poll(L4::Cap<L4Re::Event> event,
00124              L4Re::Env const *env = L4Re::Env::env(),
00125              L4Re::Cap_alloc *ca = L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc))
00126 {
00127     Unique_cap<L4Re::Dataspace> ev_ds(ca->alloc<L4Re::Dataspace>());
00128     if (!ev_ds.is_valid())
00129         return -L4_ENOMEM;
00130
00131     int r;
00132
00133     if ((r = event->get_buffer(ev_ds.get()))
00134         return r;
00135
00136     long sz = ev_ds->size();
00137     if (sz < 0)
00138         return sz;
00139
00140     Rm::Unique_region<void*> buf;
00141
00142     if ((r = env->rm()->attach(&buf, sz,
00143                             L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00144                             L4::Ipc::make_cap_rw(ev_ds.get()))))
00145         return r;
00146
00147     _ev_buffer = L4Re::Event_buffer_t<PAYLOAD>(buf.get(), sz);
00148     _ev_ds      = cxx::move(ev_ds);
00149     _buf        = cxx::move(buf);
00150
00151     return 0;
00152 }
00153
00159 L4Re::Event_buffer_t<PAYLOAD> &buffer() { return _ev_buffer; }
00160
00166 L4::Cap<L4::Triggerable> irq() const { return _ev_irq.get(); }
00167
00168 private:
00169     Unique_cap<L4Re::Dataspace> _ev_ds;
00170     Unique_del_cap<L4::Triggerable> _ev_irq;
00171     L4Re::Event_buffer_t<PAYLOAD> _ev_buffer;
00172     Rm::Unique_region<void*> _buf;
00173 };
00174
00175 typedef Event_t<Default_event_payload> Event;
00176
00177 }

```

## 16.326 event-sys.h

```

00001 /*
00002  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.

```

```

00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019 #pragma once
00020 namespace L4Re
00021 {
00022     namespace Event_
00023     {
00029         enum Opcodes
00030         {
00031             Get, Get_num_streams, Get_stream_info, Get_stream_info_for_id,
00032             Get_axis_info, Get_stream_state_for_id
00033         };
00034     };
00035 };

```

## 16.327 I4/re/c/event.h File Reference

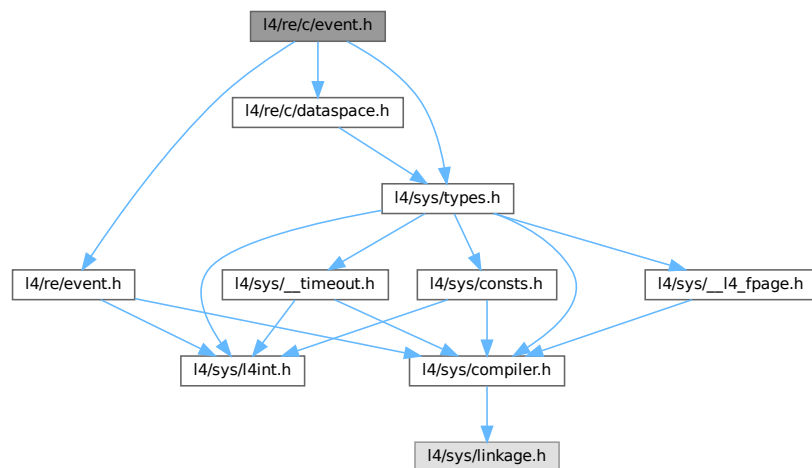
Event C interface.

```

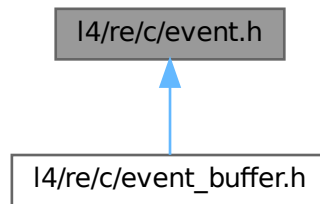
#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/event.h>

```

Include dependency graph for event.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4re\\_event\\_t](#)  
*Event structure used in buffer.*

## Functions

- long [l4re\\_event\\_get\\_buffer](#) (const [l4\\_cap\\_idx\\_t](#) server, const [l4re\\_ds\\_t](#) ds) [L4\\_NOTHROW](#)  
*Get an event signal buffer.*
- long [l4re\\_event\\_get\\_num\\_streams](#) (const [l4\\_cap\\_idx\\_t](#) server) [L4\\_NOTHROW](#)  
*Get number of streams.*
- long [l4re\\_event\\_get\\_stream\\_info](#) (const [l4\\_cap\\_idx\\_t](#) server, int idx, [l4re\\_event\\_stream\\_info\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Get information on a stream.*
- long [l4re\\_event\\_get\\_stream\\_info\\_for\\_id](#) (const [l4\\_cap\\_idx\\_t](#) server, [l4\\_umword\\_t](#) stream\_id, [l4re\\_event\\_stream\\_info\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Get info for a stream given a stream id.*
- long [l4re\\_event\\_get\\_axis\\_info](#) (const [l4\\_cap\\_idx\\_t](#) server, [l4\\_umword\\_t](#) id, unsigned naxes, unsigned const \*axis, [l4re\\_event\\_absinfo\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Get Axis information for a stream.*

### 16.327.1 Detailed Description

Event C interface.

Definition in file [event.h](#).

## 16.328 event.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00031 #include <l4/sys/types.h>
00032 #include <l4/re/c/dataspace.h>
00033 #include <l4/re/event.h>
00034
00035 EXTERN_C_BEGIN
00036
00040 typedef struct
00041 {
00042     long long time;
00043     unsigned short type;
00044     unsigned short code;
00045     int value;
00046     l4_umword_t stream_id;
00047 } l4re_event_t;
00048
00060 L4_CV long
00061 l4re_event_get_buffer(const l4_cap_idx_t server,
00062                      const l4re_ds_t ds) L4_NOTHROW;
00063
00074 L4_CV long
00075 l4re_event_get_num_streams(const l4_cap_idx_t server) L4_NOTHROW;
00076
00089 L4_CV long
00090 l4re_event_get_stream_info(const l4_cap_idx_t server,
00091                            int idx, l4re_event_stream_info_t *info) L4_NOTHROW;
00092
00105 L4_CV long
00106 l4re_event_get_stream_info_for_id(const l4_cap_idx_t server,
00107                                   l4_umword_t stream_id,
00108                                   l4re_event_stream_info_t *info) L4_NOTHROW;
00109
00125 L4_CV long
00126 l4re_event_get_axis_info(const l4_cap_idx_t server, l4_umword_t id,
00127                          unsigned naxes, unsigned const *axis,
00128                          l4re_event_absinfo_t *info) L4_NOTHROW;
00129
00130 EXTERN_C_END

```

## 16.329 l4/re/event.h File Reference

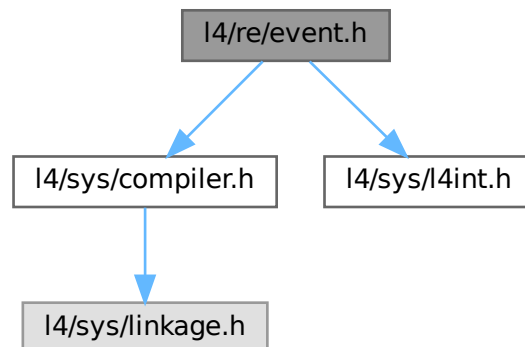
Events.

```

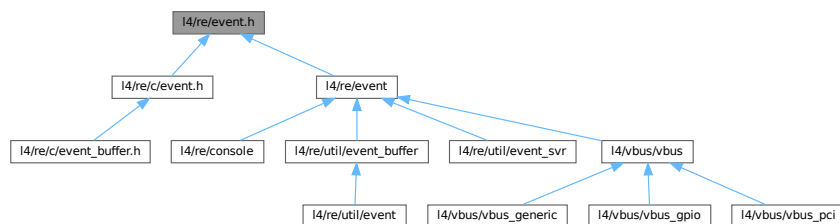
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>

```

Include dependency graph for event.h:



This graph shows which files directly or indirectly include this file:



### 16.329.1 Detailed Description

Events.

Definition in file [event.h](#).

## 16.330 event.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
  
```

```

00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022  #pragma once
00023
00024  #include <l4/sys/compiler.h>
00025  #include <l4/sys/l4int.h>
00026
00027  typedef struct L4_EXPORT_TYPE l4re_event_stream_id_t
00028  {
00029      l4_uint16_t bustype;
00030      l4_uint16_t vendor;
00031      l4_uint16_t product;
00032      l4_uint16_t version;
00033  } l4re_event_stream_id_t;
00034
00035  typedef struct L4_EXPORT_TYPE l4re_event_absinfo_t
00036  {
00037      l4_int32_t value;
00038      l4_int32_t min;
00039      l4_int32_t max;
00040      l4_int32_t fuzz;
00041      l4_int32_t flat;
00042      l4_int32_t resolution;
00043  } l4re_event_absinfo_t;
00044
00045  enum l4re_event_stream_max_values_t
00046  {
00047      L4RE_EVENT_EV_MAX = 0x1f,
00048      L4RE_EVENT_KEY_MAX = 0x1ff,
00049      L4RE_EVENT_REL_MAX = 0xf,
00050      L4RE_EVENT_ABS_MAX = 0x3f,
00051      L4RE_EVENT_PROP_MAX = 0x1f,
00052      L4RE_EVENT_SW_MAX = 0xf, // should be >= L4RE_SW_MAX
00053  };
00054
00055  enum l4re_event_stream_props_t
00056  {
00057      L4RE_EVENT_STREAM_CALIBRATE = 0x001,
00058  };
00059
00060  #define __UNUM_B(x) ((x+1) + sizeof(unsigned long)*8 - 1) / (sizeof(unsigned long)*8)
00061
00062  typedef struct L4_EXPORT_TYPE l4re_event_stream_info_t
00063  {
00064      l4_umword_t stream_id;
00065      char name[32];
00066      char phys[32];
00067      l4re_event_stream_id_t id;
00068
00069      unsigned long propbits[__UNUM_B(L4RE_EVENT_PROP_MAX)];
00070
00071      unsigned long evbits[__UNUM_B(L4RE_EVENT_EV_MAX)];
00072      unsigned long keybits[__UNUM_B(L4RE_EVENT_KEY_MAX)];
00073      unsigned long relbits[__UNUM_B(L4RE_EVENT_REL_MAX)];
00074      unsigned long absbits[__UNUM_B(L4RE_EVENT_ABS_MAX)];
00075      unsigned long swbits[__UNUM_B(L4RE_EVENT_SW_MAX)];
00076  } l4re_event_stream_info_t;
00077
00078  typedef struct L4_EXPORT_TYPE l4re_event_stream_state_t
00079  {
00080      unsigned long keybits[__UNUM_B(L4RE_EVENT_KEY_MAX)];
00081      unsigned long swbits[__UNUM_B(L4RE_EVENT_SW_MAX)];
00082  } l4re_event_stream_state_t;
00083
00084  #undef __UNUM_B
00085
00086
00087

```

## 16.331 event\_enums.h

```

00001  #pragma once
00002
00003  /*
00004   *
00005   *
00006   * Constants for L4Re events ...
00007   */
00008
00009

```



```
00010 enum L4Re_events_key
00011 {
00012     L4RE_KEY_RESERVED           = 0,
00013     L4RE_KEY_ESC                 = 1,
00014     L4RE_KEY_1                   = 2,
00015     L4RE_KEY_2                   = 3,
00016     L4RE_KEY_3                   = 4,
00017     L4RE_KEY_4                   = 5,
00018     L4RE_KEY_5                   = 6,
00019     L4RE_KEY_6                   = 7,
00020     L4RE_KEY_7                   = 8,
00021     L4RE_KEY_8                   = 9,
00022     L4RE_KEY_9                   = 10,
00023     L4RE_KEY_0                   = 11,
00024     L4RE_KEY_MINUS               = 12,
00025     L4RE_KEY_EQUAL               = 13,
00026     L4RE_KEY_BACKSPACE          = 14,
00027     L4RE_KEY_TAB                 = 15,
00028     L4RE_KEY_Q                   = 16,
00029     L4RE_KEY_W                   = 17,
00030     L4RE_KEY_E                   = 18,
00031     L4RE_KEY_R                   = 19,
00032     L4RE_KEY_T                   = 20,
00033     L4RE_KEY_Y                   = 21,
00034     L4RE_KEY_U                   = 22,
00035     L4RE_KEY_I                   = 23,
00036     L4RE_KEY_O                   = 24,
00037     L4RE_KEY_P                   = 25,
00038     L4RE_KEY_LEFTBRACE          = 26,
00039     L4RE_KEY_RIGHTBRACE         = 27,
00040     L4RE_KEY_ENTER               = 28,
00041     L4RE_KEY_LEFTCTRL           = 29,
00042     L4RE_KEY_A                   = 30,
00043     L4RE_KEY_S                   = 31,
00044     L4RE_KEY_D                   = 32,
00045     L4RE_KEY_F                   = 33,
00046     L4RE_KEY_G                   = 34,
00047     L4RE_KEY_H                   = 35,
00048     L4RE_KEY_J                   = 36,
00049     L4RE_KEY_K                   = 37,
00050     L4RE_KEY_L                   = 38,
00051     L4RE_KEY_SEMICOLON          = 39,
00052     L4RE_KEY_APOSTROPHE         = 40,
00053     L4RE_KEY_GRAVE              = 41,
00054     L4RE_KEY_LEFTSHIFT          = 42,
00055     L4RE_KEY_BACKSLASH          = 43,
00056     L4RE_KEY_Z                   = 44,
00057     L4RE_KEY_X                   = 45,
00058     L4RE_KEY_C                   = 46,
00059     L4RE_KEY_V                   = 47,
00060     L4RE_KEY_B                   = 48,
00061     L4RE_KEY_N                   = 49,
00062     L4RE_KEY_M                   = 50,
00063     L4RE_KEY_COMMA              = 51,
00064     L4RE_KEY_DOT                = 52,
00065     L4RE_KEY_SLASH              = 53,
00066     L4RE_KEY_RIGHTSHIFT         = 54,
00067     L4RE_KEY_KPASTERISK         = 55,
00068     L4RE_KEY_LEFTALT            = 56,
00069     L4RE_KEY_SPACE              = 57,
00070     L4RE_KEY_CAPSLOCK           = 58,
00071     L4RE_KEY_F1                 = 59,
00072     L4RE_KEY_F2                 = 60,
00073     L4RE_KEY_F3                 = 61,
00074     L4RE_KEY_F4                 = 62,
00075     L4RE_KEY_F5                 = 63,
00076     L4RE_KEY_F6                 = 64,
00077     L4RE_KEY_F7                 = 65,
00078     L4RE_KEY_F8                 = 66,
00079     L4RE_KEY_F9                 = 67,
00080     L4RE_KEY_F10                = 68,
00081     L4RE_KEY_NUMLOCK            = 69,
00082     L4RE_KEY_SCROLLLOCK         = 70,
00083     L4RE_KEY_KP7                = 71,
00084     L4RE_KEY_KP8                = 72,
00085     L4RE_KEY_KP9                = 73,
00086     L4RE_KEY_KPMINUS            = 74,
00087     L4RE_KEY_KP4                = 75,
00088     L4RE_KEY_KP5                = 76,
00089     L4RE_KEY_KP6                = 77,
00090     L4RE_KEY_KPPLUS             = 78,
00091     L4RE_KEY_KP1                = 79,
00092     L4RE_KEY_KP2                = 80,
00093     L4RE_KEY_KP3                = 81,
00094     L4RE_KEY_KP0                = 82,
00095     L4RE_KEY_KPDOT              = 83,
00096     L4RE_KEY_ZENKAKUHANKAKU    = 85,
```

00097	L4RE_KEY_102ND	= 86,
00098	L4RE_KEY_F11	= 87,
00099	L4RE_KEY_F12	= 88,
00100	L4RE_KEY_RO	= 89,
00101	L4RE_KEY_KATAKANA	= 90,
00102	L4RE_KEY_HIRAGANA	= 91,
00103	L4RE_KEY_HENKAN	= 92,
00104	L4RE_KEY_KATAKANAHIRAGANA	= 93,
00105	L4RE_KEY_MUHENKAN	= 94,
00106	L4RE_KEY_KPJPCOMMA	= 95,
00107	L4RE_KEY_KPENTER	= 96,
00108	L4RE_KEY_RIGHTCTRL	= 97,
00109	L4RE_KEY_KPSLASH	= 98,
00110	L4RE_KEY_SYSRQ	= 99,
00111	L4RE_KEY_RIGHTALT	= 100,
00112	L4RE_KEY_LINEFEED	= 101,
00113	L4RE_KEY_HOME	= 102,
00114	L4RE_KEY_UP	= 103,
00115	L4RE_KEY_PAGEUP	= 104,
00116	L4RE_KEY_LEFT	= 105,
00117	L4RE_KEY_RIGHT	= 106,
00118	L4RE_KEY_END	= 107,
00119	L4RE_KEY_DOWN	= 108,
00120	L4RE_KEY_PAGEDOWN	= 109,
00121	L4RE_KEY_INSERT	= 110,
00122	L4RE_KEY_DELETE	= 111,
00123	L4RE_KEY_MACRO	= 112,
00124	L4RE_KEY_MUTE	= 113,
00125	L4RE_KEY_VOLUMEDOWN	= 114,
00126	L4RE_KEY_VOLUMEUP	= 115,
00127	L4RE_KEY_POWER	= 116,
00128	L4RE_KEY_KPEQUAL	= 117,
00129	L4RE_KEY_KPPLUSMINUS	= 118,
00130	L4RE_KEY_PAUSE	= 119,
00131	L4RE_KEY_KPCOMMA	= 121,
00132	L4RE_KEY_HANGEUL	= 122,
00133	L4RE_KEY_HANGUEL	= L4RE_KEY_HANGEUL,
00134	L4RE_KEY_HANJA	= 123,
00135	L4RE_KEY_YEN	= 124,
00136	L4RE_KEY_LEFTMETA	= 125,
00137	L4RE_KEY_RIGHTMETA	= 126,
00138	L4RE_KEY_COMPOSE	= 127,
00139	L4RE_KEY_STOP	= 128,
00140	L4RE_KEY_AGAIN	= 129,
00141	L4RE_KEY_PROPS	= 130,
00142	L4RE_KEY_UNDO	= 131,
00143	L4RE_KEY_FRONT	= 132,
00144	L4RE_KEY_COPY	= 133,
00145	L4RE_KEY_OPEN	= 134,
00146	L4RE_KEY_PASTE	= 135,
00147	L4RE_KEY_FIND	= 136,
00148	L4RE_KEY_CUT	= 137,
00149	L4RE_KEY_HELP	= 138,
00150	L4RE_KEY_MENU	= 139,
00151	L4RE_KEY_CALC	= 140,
00152	L4RE_KEY_SETUP	= 141,
00153	L4RE_KEY_SLEEP	= 142,
00154	L4RE_KEY_WAKEUP	= 143,
00155	L4RE_KEY_FILE	= 144,
00156	L4RE_KEY_SENDFILE	= 145,
00157	L4RE_KEY_DELETEFILE	= 146,
00158	L4RE_KEY_XFER	= 147,
00159	L4RE_KEY_PROG1	= 148,
00160	L4RE_KEY_PROG2	= 149,
00161	L4RE_KEY_WWW	= 150,
00162	L4RE_KEY_MSDOS	= 151,
00163	L4RE_KEY_COFFEE	= 152,
00164	L4RE_KEY_DIRECTION	= 153,
00165	L4RE_KEY_CYCLEWINDOWS	= 154,
00166	L4RE_KEY_MAIL	= 155,
00167	L4RE_KEY_BOOKMARKS	= 156,
00168	L4RE_KEY_COMPUTER	= 157,
00169	L4RE_KEY_BACK	= 158,
00170	L4RE_KEY_FORWARD	= 159,
00171	L4RE_KEY_CLOSECD	= 160,
00172	L4RE_KEY_EJECTCD	= 161,
00173	L4RE_KEY_EJECTCLOSECD	= 162,
00174	L4RE_KEY_NEXTSONG	= 163,
00175	L4RE_KEY_PLAYPAUSE	= 164,
00176	L4RE_KEY_PREVIOUSSONG	= 165,
00177	L4RE_KEY_STOPCD	= 166,
00178	L4RE_KEY_RECORD	= 167,
00179	L4RE_KEY_REWIND	= 168,
00180	L4RE_KEY_PHONE	= 169,
00181	L4RE_KEY_ISO	= 170,
00182	L4RE_KEY_CONFIG	= 171,
00183	L4RE_KEY_HOMEPAGE	= 172,

```
00184 L4RE_KEY_REFRESH = 173,
00185 L4RE_KEY_EXIT = 174,
00186 L4RE_KEY_MOVE = 175,
00187 L4RE_KEY_EDIT = 176,
00188 L4RE_KEY_SCROLLUP = 177,
00189 L4RE_KEY_SCROLLDOWN = 178,
00190 L4RE_KEY_KPLEFTPAREN = 179,
00191 L4RE_KEY_KPRIGHTPAREN = 180,
00192 L4RE_KEY_NEW = 181,
00193 L4RE_KEY_REDO = 182,
00194 L4RE_KEY_F13 = 183,
00195 L4RE_KEY_F14 = 184,
00196 L4RE_KEY_F15 = 185,
00197 L4RE_KEY_F16 = 186,
00198 L4RE_KEY_F17 = 187,
00199 L4RE_KEY_F18 = 188,
00200 L4RE_KEY_F19 = 189,
00201 L4RE_KEY_F20 = 190,
00202 L4RE_KEY_F21 = 191,
00203 L4RE_KEY_F22 = 192,
00204 L4RE_KEY_F23 = 193,
00205 L4RE_KEY_F24 = 194,
00206 L4RE_KEY_PLAYCD = 200,
00207 L4RE_KEY_PAUSECD = 201,
00208 L4RE_KEY_PROG3 = 202,
00209 L4RE_KEY_PROG4 = 203,
00210 L4RE_KEY_SUSPEND = 205,
00211 L4RE_KEY_CLOSE = 206,
00212 L4RE_KEY_PLAY = 207,
00213 L4RE_KEY_FASTFORWARD = 208,
00214 L4RE_KEY_BASSBOOST = 209,
00215 L4RE_KEY_PRINT = 210,
00216 L4RE_KEY_HP = 211,
00217 L4RE_KEY_CAMERA = 212,
00218 L4RE_KEY_SOUND = 213,
00219 L4RE_KEY_QUESTION = 214,
00220 L4RE_KEY_EMAIL = 215,
00221 L4RE_KEY_CHAT = 216,
00222 L4RE_KEY_SEARCH = 217,
00223 L4RE_KEY_CONNECT = 218,
00224 L4RE_KEY_FINANCE = 219,
00225 L4RE_KEY_SPORT = 220,
00226 L4RE_KEY_SHOP = 221,
00227 L4RE_KEY_ALTERASE = 222,
00228 L4RE_KEY_CANCEL = 223,
00229 L4RE_KEY_BRIGHTNESSDOWN = 224,
00230 L4RE_KEY_BRIGHTNESSUP = 225,
00231 L4RE_KEY_MEDIA = 226,
00232 L4RE_KEY_SWITCHVIDEOMODE = 227,
00233 L4RE_KEY_KBDILLUMTOGGLE = 228,
00234 L4RE_KEY_KBDILLUMDOWN = 229,
00235 L4RE_KEY_KBDILLUMUP = 230,
00236 L4RE_KEY_SEND = 231,
00237 L4RE_KEY_REPLY = 232,
00238 L4RE_KEY_FORWARDMAIL = 233,
00239 L4RE_KEY_SAVE = 234,
00240 L4RE_KEY_DOCUMENTS = 235,
00241 L4RE_KEY_UNKNOWN = 240,
00242 L4RE_KEY_OK = 0x160,
00243 L4RE_KEY_SELECT = 0x161,
00244 L4RE_KEY_GOTO = 0x162,
00245 L4RE_KEY_CLEAR = 0x163,
00246 L4RE_KEY_POWER2 = 0x164,
00247 L4RE_KEY_OPTION = 0x165,
00248 L4RE_KEY_INFO = 0x166,
00249 L4RE_KEY_TIME = 0x167,
00250 L4RE_KEY_VENDOR = 0x168,
00251 L4RE_KEY_ARCHIVE = 0x169,
00252 L4RE_KEY_PROGRAM = 0x16a,
00253 L4RE_KEY_CHANNEL = 0x16b,
00254 L4RE_KEY_FAVORITES = 0x16c,
00255 L4RE_KEY_EPG = 0x16d,
00256 L4RE_KEY_PVR = 0x16e,
00257 L4RE_KEY_MHP = 0x16f,
00258 L4RE_KEY_LANGUAGE = 0x170,
00259 L4RE_KEY_TITLE = 0x171,
00260 L4RE_KEY_SUBTITLE = 0x172,
00261 L4RE_KEY_ANGLE = 0x173,
00262 L4RE_KEY_ZOOM = 0x174,
00263 L4RE_KEY_MODE = 0x175,
00264 L4RE_KEY_KEYBOARD = 0x176,
00265 L4RE_KEY_SCREEN = 0x177,
00266 L4RE_KEY_PC = 0x178,
00267 L4RE_KEY_TV = 0x179,
00268 L4RE_KEY_TV2 = 0x17a,
00269 L4RE_KEY_VCR = 0x17b,
00270 L4RE_KEY_VCR2 = 0x17c,
```

```

00271 L4RE_KEY_SAT           = 0x17d,
00272 L4RE_KEY_SAT2          = 0x17e,
00273 L4RE_KEY_CD             = 0x17f,
00274 L4RE_KEY_TAPE          = 0x180,
00275 L4RE_KEY_RADIO          = 0x181,
00276 L4RE_KEY_TUNER        = 0x182,
00277 L4RE_KEY_PLAYER        = 0x183,
00278 L4RE_KEY_TEXT           = 0x184,
00279 L4RE_KEY_DVD           = 0x185,
00280 L4RE_KEY_AUX           = 0x186,
00281 L4RE_KEY_MP3           = 0x187,
00282 L4RE_KEY_AUDIO         = 0x188,
00283 L4RE_KEY_VIDEO         = 0x189,
00284 L4RE_KEY_DIRECTORY     = 0x18a,
00285 L4RE_KEY_LIST          = 0x18b,
00286 L4RE_KEY_MEMO          = 0x18c,
00287 L4RE_KEY_CALENDAR      = 0x18d,
00288 L4RE_KEY_RED           = 0x18e,
00289 L4RE_KEY_GREEN        = 0x18f,
00290 L4RE_KEY_YELLOW        = 0x190,
00291 L4RE_KEY_BLUE         = 0x191,
00292 L4RE_KEY_CHANNELUP     = 0x192,
00293 L4RE_KEY_CHANNELDOWN   = 0x193,
00294 L4RE_KEY_FIRST        = 0x194,
00295 L4RE_KEY_LAST          = 0x195,
00296 L4RE_KEY_AB            = 0x196,
00297 L4RE_KEY_NEXT         = 0x197,
00298 L4RE_KEY_RESTART      = 0x198,
00299 L4RE_KEY_SLOW          = 0x199,
00300 L4RE_KEY_SHUFFLE       = 0x19a,
00301 L4RE_KEY_BREAK        = 0x19b,
00302 L4RE_KEY_PREVIOUS      = 0x19c,
00303 L4RE_KEY_DIGITS       = 0x19d,
00304 L4RE_KEY_TEEN         = 0x19e,
00305 L4RE_KEY_TWEN         = 0x19f,
00306 L4RE_KEY_DEL_EOL      = 0x1c0,
00307 L4RE_KEY_DEL_EOS      = 0x1c1,
00308 L4RE_KEY_INS_LINE      = 0x1c2,
00309 L4RE_KEY_DEL_LINE     = 0x1c3,
00310 L4RE_KEY_FN            = 0x1d0,
00311 L4RE_KEY_FN_ESC        = 0x1d1,
00312 L4RE_KEY_FN_F1         = 0x1d2,
00313 L4RE_KEY_FN_F2         = 0x1d3,
00314 L4RE_KEY_FN_F3         = 0x1d4,
00315 L4RE_KEY_FN_F4         = 0x1d5,
00316 L4RE_KEY_FN_F5         = 0x1d6,
00317 L4RE_KEY_FN_F6         = 0x1d7,
00318 L4RE_KEY_FN_F7         = 0x1d8,
00319 L4RE_KEY_FN_F8         = 0x1d9,
00320 L4RE_KEY_FN_F9         = 0x1da,
00321 L4RE_KEY_FN_F10        = 0x1db,
00322 L4RE_KEY_FN_F11       = 0x1dc,
00323 L4RE_KEY_FN_F12       = 0x1dd,
00324 L4RE_KEY_FN_1         = 0x1de,
00325 L4RE_KEY_FN_2         = 0x1df,
00326 L4RE_KEY_FN_D         = 0x1e0,
00327 L4RE_KEY_FN_E         = 0x1e1,
00328 L4RE_KEY_FN_F         = 0x1e2,
00329 L4RE_KEY_FN_S         = 0x1e3,
00330 L4RE_KEY_FN_B         = 0x1e4,
00331 L4RE_KEY_MAX          = 0x1ff,
00332 };
00333
00334 enum L4Re_events_rel
00335 {
00336     L4RE_REL_X           = 0x00,
00337     L4RE_REL_Y           = 0x01,
00338     L4RE_REL_Z           = 0x02,
00339     L4RE_REL_RX          = 0x03,
00340     L4RE_REL_RY          = 0x04,
00341     L4RE_REL_RZ          = 0x05,
00342     L4RE_REL_HWHEEL      = 0x06,
00343     L4RE_REL_DIAL        = 0x07,
00344     L4RE_REL_WHEEL       = 0x08,
00345     L4RE_REL_MISC        = 0x09,
00346     L4RE_REL_MAX         = 0x0f,
00347 };
00348
00349 enum L4Re_events_snd
00350 {
00351     L4RE_SND_CLICK       = 0x00,
00352     L4RE_SND_BELL        = 0x01,
00353     L4RE_SND_TONE        = 0x02,
00354     L4RE_SND_MAX         = 0x07,
00355 };
00356
00357 enum L4Re_events_rep

```

```
00358 {
00359     L4RE_REP_DELAY = 0x00,
00360     L4RE_REP_PERIOD = 0x01,
00361     L4RE_REP_MAX = 0x01,
00362 };
00363
00364 enum L4Re_events_led
00365 {
00366     L4RE_LED_NUML = 0x00,
00367     L4RE_LED_CAPSL = 0x01,
00368     L4RE_LED_SCROLLL = 0x02,
00369     L4RE_LED_COMPOSE = 0x03,
00370     L4RE_LED_KANA = 0x04,
00371     L4RE_LED_SLEEP = 0x05,
00372     L4RE_LED_SUSPEND = 0x06,
00373     L4RE_LED_MUTE = 0x07,
00374     L4RE_LED_MISC = 0x08,
00375     L4RE_LED_MAIL = 0x09,
00376     L4RE_LED_CHARGING = 0x0a,
00377     L4RE_LED_MAX = 0x0f,
00378 };
00379
00380 enum L4Re_events_btn
00381 {
00382     L4RE_BTN_MISC = 0x100,
00383     L4RE_BTN_0 = 0x100,
00384     L4RE_BTN_1 = 0x101,
00385     L4RE_BTN_2 = 0x102,
00386     L4RE_BTN_3 = 0x103,
00387     L4RE_BTN_4 = 0x104,
00388     L4RE_BTN_5 = 0x105,
00389     L4RE_BTN_6 = 0x106,
00390     L4RE_BTN_7 = 0x107,
00391     L4RE_BTN_8 = 0x108,
00392     L4RE_BTN_9 = 0x109,
00393     L4RE_BTN_MOUSE = 0x110,
00394     L4RE_BTN_LEFT = 0x110,
00395     L4RE_BTN_RIGHT = 0x111,
00396     L4RE_BTN_MIDDLE = 0x112,
00397     L4RE_BTN_SIDE = 0x113,
00398     L4RE_BTN_EXTRA = 0x114,
00399     L4RE_BTN_FORWARD = 0x115,
00400     L4RE_BTN_BACK = 0x116,
00401     L4RE_BTN_TASK = 0x117,
00402     L4RE_BTN_JOYSTICK = 0x120,
00403     L4RE_BTN_TRIGGER = 0x120,
00404     L4RE_BTN_THUMB = 0x121,
00405     L4RE_BTN_THUMB2 = 0x122,
00406     L4RE_BTN_TOP = 0x123,
00407     L4RE_BTN_TOP2 = 0x124,
00408     L4RE_BTN_PINKIE = 0x125,
00409     L4RE_BTN_BASE = 0x126,
00410     L4RE_BTN_BASE2 = 0x127,
00411     L4RE_BTN_BASE3 = 0x128,
00412     L4RE_BTN_BASE4 = 0x129,
00413     L4RE_BTN_BASE5 = 0x12a,
00414     L4RE_BTN_BASE6 = 0x12b,
00415     L4RE_BTN_DEAD = 0x12f,
00416     L4RE_BTN_GAMEPAD = 0x130,
00417     L4RE_BTN_A = 0x130,
00418     L4RE_BTN_B = 0x131,
00419     L4RE_BTN_C = 0x132,
00420     L4RE_BTN_X = 0x133,
00421     L4RE_BTN_Y = 0x134,
00422     L4RE_BTN_Z = 0x135,
00423     L4RE_BTN_TL = 0x136,
00424     L4RE_BTN_TR = 0x137,
00425     L4RE_BTN_TL2 = 0x138,
00426     L4RE_BTN_TR2 = 0x139,
00427     L4RE_BTN_SELECT = 0x13a,
00428     L4RE_BTN_START = 0x13b,
00429     L4RE_BTN_MODE = 0x13c,
00430     L4RE_BTN_THUMBL = 0x13d,
00431     L4RE_BTN_THUMBR = 0x13e,
00432     L4RE_BTN_DIGI = 0x140,
00433     L4RE_BTN_TOOL_PEN = 0x140,
00434     L4RE_BTN_TOOL_RUBBER = 0x141,
00435     L4RE_BTN_TOOL_BRUSH = 0x142,
00436     L4RE_BTN_TOOL_PENCIL = 0x143,
00437     L4RE_BTN_TOOL_AIRBRUSH = 0x144,
00438     L4RE_BTN_TOOL_FINGER = 0x145,
00439     L4RE_BTN_TOOL_MOUSE = 0x146,
00440     L4RE_BTN_TOOL_LENS = 0x147,
00441     L4RE_BTN_TOUCH = 0x14a,
00442     L4RE_BTN_STYLUS = 0x14b,
00443     L4RE_BTN_STYLUS2 = 0x14c,
00444     L4RE_BTN_TOOL_DOUBLETAP = 0x14d,
```

```
00445     L4RE_BTN_TOOL_TRIPLETAP = 0x14e,
00446     L4RE_BTN_WHEEL           = 0x150,
00447     L4RE_BTN_GEAR_DOWN       = 0x150,
00448     L4RE_BTN_GEAR_UP         = 0x151,
00449 };
00450
00451 enum L4Re_events_sw
00452 {
00453     L4RE_SW_0   = 0x00,
00454     L4RE_SW_1   = 0x01,
00455     L4RE_SW_2   = 0x02,
00456     L4RE_SW_3   = 0x03,
00457     L4RE_SW_4   = 0x04,
00458     L4RE_SW_5   = 0x05,
00459     L4RE_SW_6   = 0x06,
00460     L4RE_SW_7   = 0x07,
00461     L4RE_SW_MAX = 0x0f,
00462 };
00463
00464 enum L4Re_events_ev
00465 {
00466     L4RE_EV_SYN      = 0x00,
00467     L4RE_EV_KEY      = 0x01,
00468     L4RE_EV_REL      = 0x02,
00469     L4RE_EV_ABS      = 0x03,
00470     L4RE_EV_MSC      = 0x04,
00471     L4RE_EV_SW       = 0x05,
00472     L4RE_EV_LED      = 0x11,
00473     L4RE_EV_SND      = 0x12,
00474     L4RE_EV_REP      = 0x14,
00475     L4RE_EV_FF       = 0x15,
00476     L4RE_EV_PWR      = 0x16,
00477     L4RE_EV_FF_STATUS = 0x17,
00478     L4RE_EV_WINDOW   = 0x18,
00479     L4RE_EV_PM        = 0x1e, // power management signals
00480     L4RE_EV_MAX       = 0x1f,
00481 };
00482
00483 enum L4Re_events_syn
00484 {
00485     L4RE_SYN_REPORT      = 0,
00486     L4RE_SYN_CONFIG      = 1,
00487     L4RE_SYN_MT_REPORT   = 2,
00488
00489     L4RE_SYN_STREAM_CFG  = 0x80,
00490 };
00491
00492 enum L4Re_stream_cfg
00493 {
00494     L4RE_SYN_STREAM_NEW   = 0,
00495     L4RE_SYN_STREAM_CLOSE = 1,
00496 };
00497
00498 enum L4Re_events_abs
00499 {
00500     L4RE_ABS_X           = 0x00,
00501     L4RE_ABS_Y           = 0x01,
00502     L4RE_ABS_Z           = 0x02,
00503     L4RE_ABS_RX          = 0x03,
00504     L4RE_ABS_RY          = 0x04,
00505     L4RE_ABS_RZ          = 0x05,
00506     L4RE_ABS_THROTTLE    = 0x06,
00507     L4RE_ABS_RUDDER      = 0x07,
00508     L4RE_ABS_WHEEL       = 0x08,
00509     L4RE_ABS_GAS         = 0x09,
00510     L4RE_ABS_BRAKE       = 0x0a,
00511     L4RE_ABS_HAT0X       = 0x10,
00512     L4RE_ABS_HAT0Y       = 0x11,
00513     L4RE_ABS_HAT1X       = 0x12,
00514     L4RE_ABS_HAT1Y       = 0x13,
00515     L4RE_ABS_HAT2X       = 0x14,
00516     L4RE_ABS_HAT2Y       = 0x15,
00517     L4RE_ABS_HAT3X       = 0x16,
00518     L4RE_ABS_HAT3Y       = 0x17,
00519     L4RE_ABS_PRESSURE     = 0x18,
00520     L4RE_ABS_DISTANCE     = 0x19,
00521     L4RE_ABS_TILT_X       = 0x1a,
00522     L4RE_ABS_TILT_Y       = 0x1b,
00523     L4RE_ABS_TOOL_WIDTH   = 0x1c,
00524     L4RE_ABS_VOLUME       = 0x20,
00525     L4RE_ABS_MISC         = 0x28,
00526     L4RE_ABS_MT_TOUCH_MAJOR = 0x30,
00527     L4RE_ABS_MT_TOUCH_MINOR = 0x31,
00528     L4RE_ABS_MT_WIDTH_MAJOR = 0x32,
00529     L4RE_ABS_MT_WIDTH_MINOR = 0x33,
00530     L4RE_ABS_MT_ORIENTATION = 0x34,
00531     L4RE_ABS_MT_POSITION_X = 0x35,
```

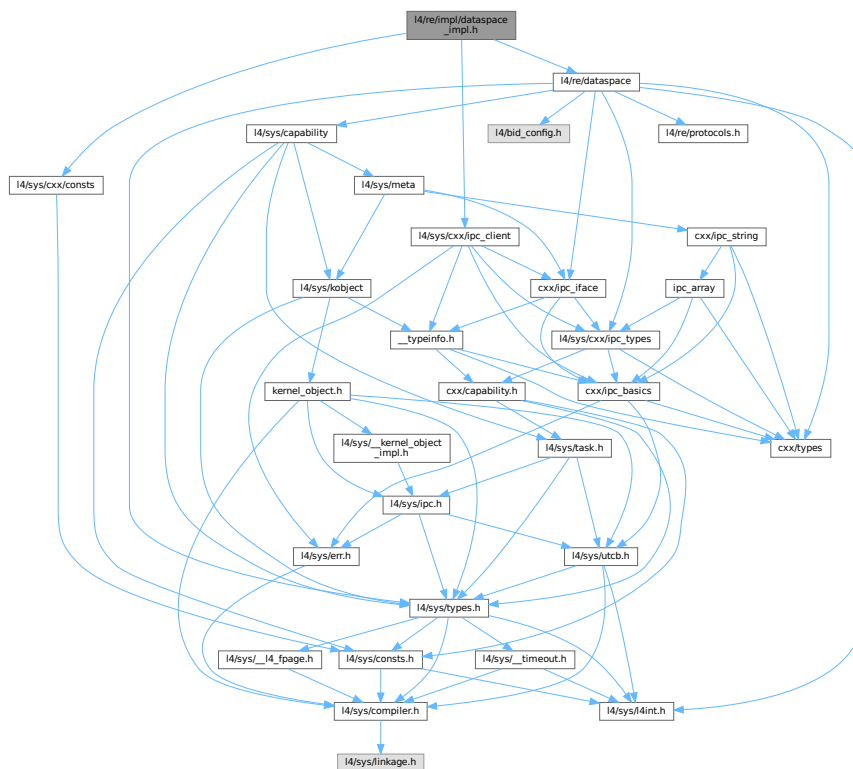
```

00532     L4RE_ABS_MT_POSITION_Y    = 0x36,
00533     L4RE_ABS_MT_TOOL_TYPE     = 0x37,
00534     L4RE_ABS_MT_BLOB_ID       = 0x38,
00535     L4RE_ABS_MT_TRACKING_ID   = 0x39,
00536     L4RE_ABS_MT_PRESSURE      = 0x3a,
00537     L4RE_ABS_MT_DISTANCE      = 0x3b,
00538
00539     L4RE_ABS_MAX               = 0x3f,
00540 };
00541
00542 enum L4Re_events_msc
00543 {
00544     L4RE_MSC_SERIAL           = 0x00,
00545     L4RE_MSC_PULSELED         = 0x01,
00546     L4RE_MSC_GESTURE          = 0x02,
00547     L4RE_MSC_RAW               = 0x03,
00548     L4RE_MSC_SCAN             = 0x04,
00549     L4RE_MSC_MAX               = 0x07,
00550 };
00551
00552 enum L4Re_events_properties
00553 {
00554     L4RE_EVENT_PROP_POINTER    = 0x00,
00555     L4RE_EVENT_PROP_DIRECT     = 0x01,
00556     L4RE_EVENT_PROP_BUTTONPAD  = 0x02,
00557     L4RE_EVENT_PROP_SEMI_MT    = 0x03,
00558     //L4RE_EVENT_PROP_MAX      = 0x1f
00559 };

```

### Dataspace client stub implementation.

```
#include <l4/re/dataspace>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/cxx/consts>
Include dependency graph for dataspace_impl.h:
```



## Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*

### 16.332.1 Detailed Description

Dataspace client stub implementation.

Definition in file [dataspace\\_impl.h](#).

## 16.333 dataspace\_impl.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *           Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *           economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #include <l4/re/dataspace>
00024 #include <l4/sys/cxx/ipc_client>
00025 #include <l4/sys/cxx/consts>
00026
00027 L4_RPC_DEF(L4Re::Dataspace::clear);
00028 L4_RPC_DEF(L4Re::Dataspace::allocate);
00029 L4_RPC_DEF(L4Re::Dataspace::copy_in);
00030 L4_RPC_DEF(L4Re::Dataspace::info);
00031 L4_RPC_DEF(L4Re::Dataspace::map_info);
00032
00033 namespace L4Re {
00034
00035 long
00036 Dataspace::__map(Dataspace::Offset offset, unsigned char *size,
00037                 Dataspace::Flags flags,
00038                 Dataspace::Map_addr local_addr,
00039                 L4::Cap<L4::Task> dst) const noexcept
00040 {
00041     Map_addr spot = local_addr & ~(~0ULL « l4_umword_t(*size));
00042     Map_addr base = local_addr & (~0ULL « l4_umword_t(*size));
00043     L4::Ipc::Rcv_fpage r = L4::Ipc::Rcv_fpage::mem(base, *size, 0, dst);
00044
00045     L4::Ipc::Snd_fpage fp;
00046     long err = map_t::call(c(), offset, spot, flags, r, fp, l4_utcb());
00047     if (L4_UNLIKELY(err < 0))
00048         return err;
00049
00050     *size = fp.rcv_order();
00051     return err;
00052 }
00053
00054 long
00055 Dataspace::map_region(Dataspace::Offset offset, Dataspace::Flags flags,
00056                      Dataspace::Map_addr min_addr,
00057                      Dataspace::Map_addr max_addr,
00058                      L4::Cap<L4::Task> dst) const noexcept
00059 {
00060     min_addr = L4::trunc_page(min_addr);
00061     max_addr = L4::round_page(max_addr);
00062     unsigned char order = L4_LOG2_PAGESIZE;

```



```

00064
00065     long err = 0;
00066
00067     while (min_addr < max_addr)
00068     {
00069         unsigned char order_mapped;
00070         order_mapped = order
00071             = L4::max_order(order, min_addr, min_addr, max_addr, min_addr);
00072
00073         err = __map(offset, &order_mapped, flags, min_addr, dst);
00074         if (L4_UNLIKELY(err < 0))
00075             return err;
00076
00077         if (order > order_mapped)
00078             order = order_mapped;
00079
00080         min_addr += Map_addr(1) « order;
00081         offset   += Map_addr(1) « order;
00082
00083         if (min_addr >= max_addr)
00084             return 0;
00085
00086         while (min_addr != L4::trunc_order(min_addr, order)
00087             || max_addr < L4::round_order(min_addr + 1, order))
00088             --order;
00089     }
00090
00091     return 0;
00092 }
00093
00094
00095 long
00096 Dataspace::map(Dataspace::Offset offset, Dataspace::Flags flags,
00097               Dataspace::Map_addr local_addr,
00098               Dataspace::Map_addr min_addr,
00099               Dataspace::Map_addr max_addr,
00100               L4::Cap<L4::Task> dst) const noexcept
00101 {
00102     min_addr = L4::trunc_page(min_addr);
00103     max_addr = L4::round_page(max_addr);
00104     local_addr = L4::trunc_page(local_addr);
00105     unsigned char order
00106         = L4::max_order(L4_LOG2_PAGESIZE, local_addr, min_addr, max_addr, local_addr);
00107
00108     return __map(offset, &order, flags, local_addr, dst);
00109 }
00110
00111 Dataspace::Size
00112 Dataspace::size() const noexcept
00113 {
00114     Stats stats = Stats();
00115     int err = info(&stats);
00116     if (err < 0)
00117         return 0;
00118     return stats.size;
00119 }
00120
00121 Dataspace::Flags
00122 Dataspace::flags() const noexcept
00123 {
00124     Stats stats = Stats();
00125     int err = info(&stats);
00126     if (err < 0)
00127         return Flags(0);
00128     return stats.flags;
00129 }
00130
00131 };

```

## 16.334 l4/re/impl/mem\_alloc\_impl.h File Reference

Memory allocator client stub implementation.

```

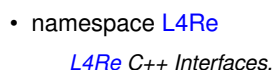
#include <l4/re/mem_alloc>
#include <l4/re/mem_alloc-sys.h>
#include <l4/re/dataspace>
#include <l4/re/error_helper>

```



## 16.336 l4/re/impl/namespace\_impl.h File Reference

```
#include <l4/re/namespace>
#include <l4/util/util.h>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/assert.h>
#include <cstring>
Include dependency graph for namespace_impl.h:
```



## 16.336.1 Detailed Description

Namespace client stub implementation.

Definition in file [namespace\\_impl.h](#).

## 16.337 namespace\_impl.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #include <l4/re/namespace>
00024
00025 #include <l4/util/util.h>
00026 #include <l4/sys/cxx/ipc_client>
00027 #include <l4/sys/assert.h>
00028
00029 #include <cstring>
00030
00031 L4_RPC_DEF(L4Re::Namespace::query);
00032 L4_RPC_DEF(L4Re::Namespace::register_obj);
00033 L4_RPC_DEF(L4Re::Namespace::unlink);
00034
00035 namespace L4Re {
00036
00037     long
00038     Namespace::_query(char const *name, unsigned len,
00039                      L4::Cap<void> const &target,
00040                      l4_umword_t *local_id, bool iterate) const noexcept
00041     {
00042         l4_assert(target.is_valid());
00043
00044         L4::Cap<Namespace> ns = c();
00045         L4::Ipc::Array<char const, unsigned long> _name(len, name);
00046
00047         while (_name.length > 0)
00048         {
00049             L4::Ipc::Snd_fpage cap;
00050             L4::Opcode dummy;
00051             int err = query_t::call(ns, _name,
00052                                   L4::Ipc::Small_buf(target.cap(),
00053                                                         local_id
00054                                                         ? L4_RCV_ITEM_LOCAL_ID
00055                                                         : 0),
00056                                   cap, dummy, _name);
00057             if (err < 0)
00058                 return err;
00059
00060             bool const partly = err & Partly_resolved;
00061             if (cap.id_received())
00062             {
00063                 *local_id = cap.data();
00064                 return _name.length;
00065             }
00066
00067             if (partly && iterate)
00068                 ns = L4::cap_cast<Namespace>(target);
00069             else
00070                 return err;
00071         }
00072     }

```

```

00073     return _name.length;
00074 }
00075
00076 long
00077 Namespace::query(char const *name, unsigned len, L4::Cap<void> const &target,
00078                 int timeout, l4_umword_t *local_id, bool iterate) const noexcept
00079 {
00080     if (L4_UNLIKELY(len == 0))
00081         return -L4_EINVAL;
00082     if (L4_UNLIKELY(timeout < 0))
00083         return -L4_EINVAL;
00084     long ret;
00085     long rem = timeout;
00086     long to = 0;
00087     if (rem)
00088         to = 10;
00089     do
00090     {
00091         ret = _query(name, len, target, local_id, iterate);
00092         if (ret >= 0)
00093             return ret;
00094         if (L4_UNLIKELY(ret != -L4_EAGAIN))
00095             return ret;
00096         if (rem == to)
00097             return ret;
00098         l4_sleep(to);
00099         if (rem > 0)
00100         {
00101             rem -= to;
00102             if (rem < 0)
00103             {
00104                 to = rem = 0;
00105                 continue; // one final try
00106             }
00107         }
00108         if (to < 100)
00109             to += to;
00110     }
00111     while (486);
00112 }
00113
00114 long
00115 Namespace::query(char const *name, L4::Cap<void> const &target,
00116                 int timeout, l4_umword_t *local_id,
00117                 bool iterate) const noexcept
00118 {
00119     return query(name, __builtin_strlen(name), target,
00120                 timeout, local_id, iterate);
00121 }
00122
00123 }
00124
00125 long
00126 Namespace::query(char const *name, L4::Cap<void> const &target,
00127                 int timeout, l4_umword_t *local_id,
00128                 bool iterate) const noexcept
00129 {
00130     return query(name, __builtin_strlen(name), target,
00131                 timeout, local_id, iterate);
00132 }
00133 }

```

## 16.338 l4/re/impl/rm\_impl.h File Reference

Region map client stub implementation.

```

#include <l4/bid_config.h>
#include <l4/re/rm>
#include <l4/re/dataspace>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/task>
#include <l4/sys/err.h>

```



```

00031
00032 L4_RPC_DEF(L4Re::Rm::reserve_area);
00033 L4_RPC_DEF(L4Re::Rm::free_area);
00034 L4_RPC_DEF(L4Re::Rm::attach);
00035 L4_RPC_DEF(L4Re::Rm::detach);
00036 L4_RPC_DEF(L4Re::Rm::get_regions);
00037 L4_RPC_DEF(L4Re::Rm::get_areas);
00038 L4_RPC_DEF(L4Re::Rm::find);
00039
00040 namespace L4Re
00041 {
00042
00043 long
00044 Rm::attach(l4_addr_t *start, unsigned long size, Rm::Flags flags,
00045           L4::Ipc::Cap<Dataspace> mem, Rm::Offset offs,
00046           unsigned char align, L4::Cap<L4::Task> const task) const noexcept
00047 {
00048     if ((flags & F::Rights_mask) == Flags(0) || (flags & F::Reserved))
00049         mem = L4::Ipc::Cap<L4Re::Dataspace>();
00050
00051     long e = attach_t::call(c(), start, size, flags, mem, offs, align, mem.cap().cap());
00052     if (e < 0)
00053         return e;
00054
00055 #ifdef CONFIG_MMU
00056     if ((flags & (F::Eager_map | F::No_eager_map)) == F::Eager_map)
00057     #else
00058     if (!(flags & F::No_eager_map) && mem.is_valid())
00059     #endif
00060         e = mem.cap()->map_region(offs, map_flags(flags), *start, *start + size,
00061                                   task);
00062
00063     return e;
00064 }
00065
00066 int
00067 Rm::detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
00068           L4::Cap<L4::Task> task, unsigned flags) const noexcept
00069 {
00070     l4_addr_t rstart = 0, rsize = 0;
00071     l4_cap_idx_t mem_cap = L4_INVALID_CAP;
00072     long e = detach_t::call(c(), start, size, flags, rstart, rsize, mem_cap);
00073     if (L4_UNLIKELY(e < 0))
00074         return e;
00075
00076     if (mem)
00077         *mem = L4::Cap<L4Re::Dataspace>(mem_cap);
00078
00079     if (!task.is_valid())
00080         return e;
00081
00082     rsize = l4_round_page(rsize);
00083     unsigned order = L4_LOG2_PAGESIZE;
00084     unsigned long sz = (1UL << order);
00085     for (unsigned long p = rstart; rsize; p += sz, rsize -= sz)
00086     {
00087         while (sz > rsize)
00088         {
00089             --order;
00090             sz >>= 1;
00091         }
00092
00093         for (;;)
00094         {
00095             unsigned long m = sz << 1;
00096             if (m > rsize)
00097                 break;
00098
00099             if (p & (m - 1))
00100                 break;
00101
00102             ++order;
00103             sz <<= 1;
00104         }
00105
00106         task->unmap(l4_fpage(p, order, L4_FPAGE_RWX),
00107                   L4_FP_ALL_SPACES);
00108     }
00109
00110     return e;
00111 }
00112 }

```

## 16.340 inhibitor

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00004  *
00005  * This file is licensed under the terms of the GNU Lesser General
00006  * Public License 2.1.
00007  * See the file COPYING-LGPL-2.1 for details.
00008  */
00009 #pragma once
00010
00011 #include <l4/sys/capability>
00012 #include <l4/sys/cxx/ipc_iface>
00013 #include <l4/sys/cxx/ipc_string>
00014 #include <l4/re/protocols.h>
00015
00016 namespace L4Re {
00017
00040 class Inhibitor :
00041     public L4::Kobject_t<Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR>
00042 {
00043 public:
00044     enum
00045     {
00046         Name_max = 20
00047     };
00048
00059 L4_INLINE_RPC(long, acquire, (l4_umword_t id, L4::Ipcc::String<> reason));
00060
00069 L4_INLINE_RPC(long, release, (l4_umword_t id));
00070
00086 long next_lock_info(char *name, unsigned len, l4_mword_t current_id = -1,
00087                     l4_utcb_t *utcb = l4_utcb())
00088 {
00089     L4::Ipcc::String<char> name_buf(len, name);
00090     long r = next_lock_info_t::call(c(), &current_id, name_buf, utcb);
00091     if (r < 0)
00092         return r;
00093
00094     return current_id;
00095 }
00096
00097 L4_INLINE_RPC_NF(long, next_lock_info, (L4::Ipcc::In_out<l4_mword_t *> current_id,
00098                                         L4::Ipcc::String<char> &name));
00099
00100 typedef L4::Typeid::Rpccs<acquire_t, release_t, next_lock_info_t> Rpccs;
00101 };
00102
00103 }

```

## 16.341 inhibitor-sys.h

```

00001 /*
00002  * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00003  *
00004  * This file is licensed under the terms of the GNU Lesser General Public
00005  * License 2.1.
00006  * See the file COPYING-LGPL-2.1 for details.
00007  */
00008 #pragma once
00009
00010 namespace L4Re {
00011     namespace Inhibitor_ {
00017         enum Opcodes { Acquire, Release, Next_lock_info };
00018     }
00019 }

```

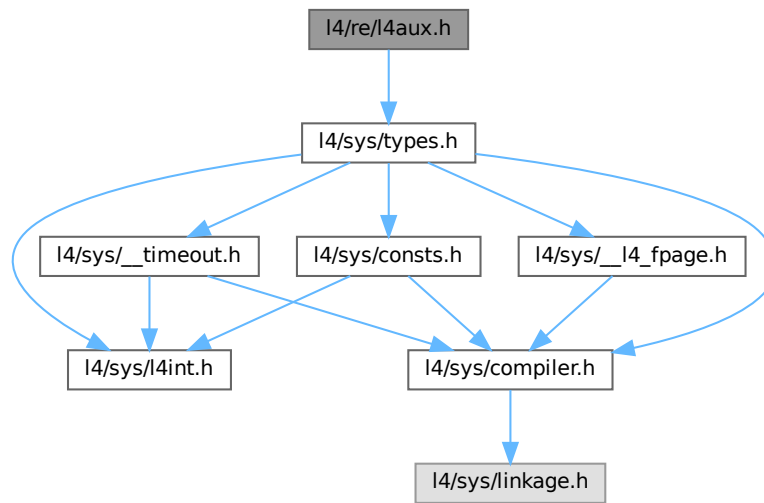
## 16.342 l4/re/l4aux.h File Reference

Auxiliary definitions.



```
#include <l4/sys/types.h>
```

Include dependency graph for l4aux.h:



## Data Structures

- struct `l4re_aux_t`  
*Auxiliary descriptor.*

## Typedefs

- typedef struct `l4re_aux_t` `l4re_aux_t`  
*Auxiliary descriptor.*

## Enumerations

- enum `l4re_aux_ldr_flags_t`  
*Flags for program loading.*

### 16.342.1 Detailed Description

Auxiliary definitions.

Definition in file `l4aux.h`.

## 16.343 l4aux.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025
00026 #include <l4/sys/types.h>
00027
00039 enum l4re_aux_ldr_flags_t
00040 {
00041     L4RE_AUX_LDR_FLAG_EAGER_MAP    = 0x1,
00042     L4RE_AUX_LDR_FLAG_ALL_SEGS_COW = 0x2,
00043     L4RE_AUX_LDR_FLAG_PINNED_SEGS  = 0x4,
00044 };
00045
00051 typedef struct l4re_aux_t
00052 {
00053     char const *    binary;
00054     l4_cap_idx_t    kip_ds;
00055     l4_umword_t     dbg_lvl;
00056     l4_umword_t     ldr_flags;
00057     l4_addr_t        ldr_base;
00058 } l4re_aux_t;
00059
```

## 16.344 l4/re/log File Reference

Log interface.



## 16.344.1 Detailed Description

Log interface.

Definition in file [log](#).

## 16.345 log

[Go to the documentation of this file.](#)

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/sys/vcon>
00028
00029 namespace L4Re {
00030
00044 class L4_EXPORT Log : public L4::Kobject_t<Log, L4::Vcon, L4::PROTO_EMPTY>
00045 {
00046 public:
00047
00054     void println(char const *string, int len) const noexcept;
00055
00061     void print(char const *string) const noexcept;
00062 };
00063 }
```

## 16.346 l4/re/log-sys.h File Reference

Log protocol definition.

### Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*

### Enumerations

- enum [L4Re::Log::Opcodes](#)  
*Logging-service communication-protocol opcodes.*

## 16.346.1 Detailed Description

Log protocol definition.

Definition in file [log-sys.h](#).

## 16.347 log-sys.h

[Go to the documentation of this file.](#)

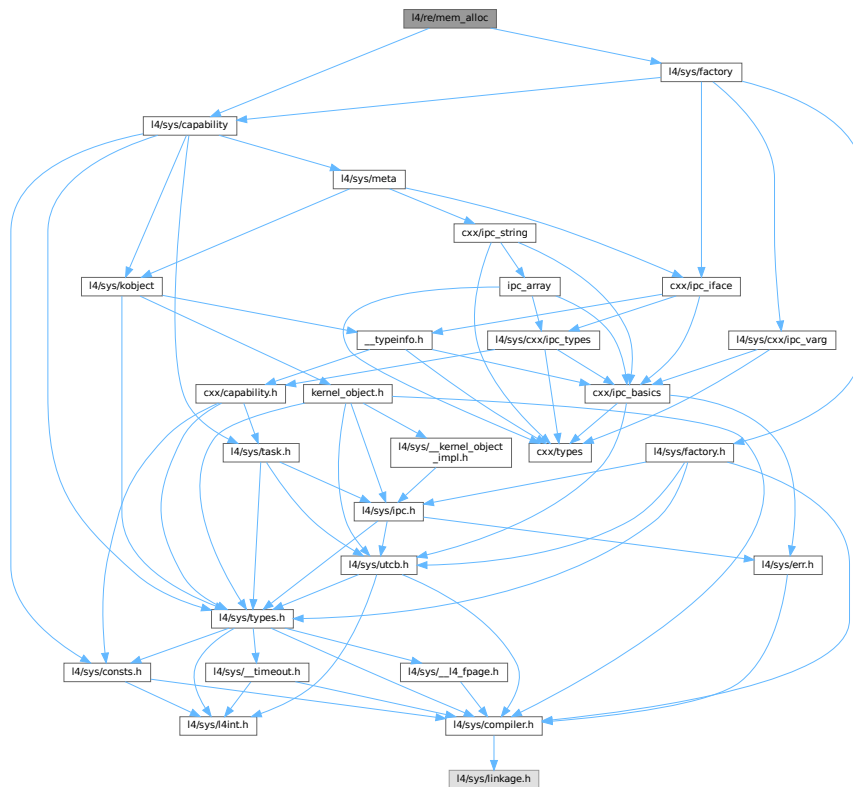
```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 namespace L4Re
00026 {
00027     namespace Log_
00028     {
00034         enum Opcodes { Print };
00035     };
00036 };
```

## 16.348 l4/re/mem\_alloc File Reference

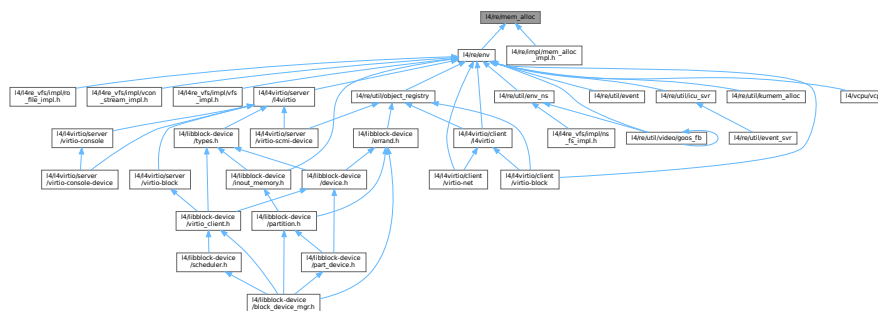
Memory allocator interface.

```
#include <l4/sys/capability>
#include <l4/sys/factory>
```

Include dependency graph for `mem_alloc`:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4Re::Mem_alloc`  
*Memory allocation interface.*

## Namespaces

- namespace `L4Re`  
*L4Re C++ Interfaces.*

## 16.348.1 Detailed Description

Memory allocator interface.

Definition in file [mem\\_alloc](#).

## 16.349 mem\_alloc

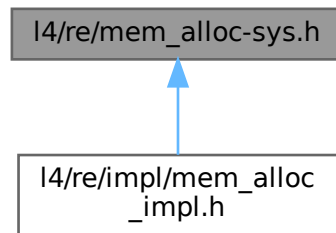
[Go to the documentation of this file.](#)

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * Copyright (C) 2014-2016, 2019, 2021 Kernkonzept GmbH.
00009  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00010  */
00011 /*
00012  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00013  *           Alexander Warg <warg@os.inf.tu-dresden.de>,
00014  *           Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00015  *           economic rights: Technische Universität Dresden (Germany)
00016  *
00017  * This file is part of TUD:OS and distributed under the terms of the
00018  * GNU General Public License 2.
00019  * Please see the COPYING-GPL-2 file for details.
00020  *
00021  * As a special exception, you may use this file as part of a free software
00022  * library without restriction. Specifically, if other files instantiate
00023  * templates or use macros or inline functions from this file, or you compile
00024  * this file and link it with other files to produce an executable, this
00025  * file does not by itself cause the resulting executable to be covered by
00026  * the GNU General Public License. This exception does not however
00027  * invalidate any other reasons why the executable file might be covered by
00028  * the GNU General Public License.
00029  */
00030 #pragma once
00031
00032 #include <l4/sys/capability>
00033 #include <l4/sys/factory>
00034
00035 namespace L4Re {
00036 class Dataspace;
00037
00038 // MISSING:
00039 // * alignment constraints
00040 // * shall we support superpages in noncont memory?
00041
00061 class L4_EXPORT Mem_alloc :
00062     public L4::Kobject_t<Mem_alloc, L4::Factory, L4::PROTO_EMPTY>
00063 {
00064 public:
00071     enum Mem_alloc_flags
00072     {
00073         Continuous    = 0x01,
00074         Pinned        = 0x02,
00075         Super_pages   = 0x04,
00076         Fixed_paddr   = 0x08,
00077     };
00078
00079     long alloc(long size, L4::Cap<Dataspace> mem,
00109                unsigned long flags = 0, unsigned long align = 0,
00110                l4_addr_t paddr = 0) const noexcept;
00111 };
00112
00113 ;;
```

## 16.350 l4/re/mem\_alloc-sys.h File Reference

Memory allocator protocol definitions.

This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*

## Enumerations

- enum [L4Re::Mem\\_alloc\\_::Opcodes](#)  
*Memory-allocator communication-protocol opcodes.*

### 16.350.1 Detailed Description

Memory allocator protocol definitions.

Definition in file [mem\\_alloc-sys.h](#).

## 16.351 mem\_alloc-sys.h

[Go to the documentation of this file.](#)

```

00001
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020 #pragma once
00021
00022 namespace L4Re
00023 {
00024     namespace Mem_alloc_
00025     {
00026         enum Opcodes { Alloc, Free };
00027     };
00028 };

```



Interface definition to emit MMIO-like accesses via IPC.

```

graph TD
    I4_re_mmio_space["I4/re/mmio_space"]
    I4_re_protocols_h["I4/re/protocols.h"]
    I4_sys_capability["I4/sys/capability"]
    I4_sys_meta["I4/sys/meta"]
    I4_sys_kobject["I4/sys/kobject"]
    I4_sys_types_h["I4/sys/types.h"]
    I4_sys_constants_h["I4/sys/constants.h"]
    I4_sys_timeout_h["I4/sys/_timeout.h"]
    I4_sys_compiler_h["I4/sys/compiler.h"]
    I4_sys_linkage_h["I4/sys/linkage.h"]
    I4_sys_err_h["I4/sys/err.h"]
    I4_sys_ipc_h["I4/sys/ipc.h"]
    I4_sys_ipc_tcb_h["I4/sys/utcb.h"]
    I4_sys_kernel_object_impl_h["I4/sys/_kernel_object_impl.h"]
    I4_sys_task_h["I4/sys/task.h"]
    I4_sys_ipc_array["ipc_array"]
    I4_sys_ipc_iface["cxx/ipc_iface"]
    I4_sys_ipc_types_h["I4/sys/cxx/ipc_types"]
    I4_sys_ipc_types_info_h["_typeinfo.h"]
    I4_sys_ipc_basics_h["cxx/ipc_basics"]
    I4_sys_ipc_types_cxx_h["cxx/types"]
    I4_sys_capability_cxx_h["cxx/capability.h"]
    I4_sys_kernel_object_h["kernel_object.h"]
    I4_sys_ipc_string["cxx/ipc_string"]

    I4_re_mmio_space --> I4_re_protocols_h
    I4_re_mmio_space --> I4_sys_capability
    I4_re_mmio_space --> I4_sys_meta
    I4_re_mmio_space --> I4_sys_kobject
    I4_re_mmio_space --> I4_sys_types_h
    I4_re_mmio_space --> I4_sys_constants_h
    I4_re_mmio_space --> I4_sys_timeout_h
    I4_re_mmio_space --> I4_sys_compiler_h
    I4_re_mmio_space --> I4_sys_err_h
    I4_re_mmio_space --> I4_sys_ipc_h
    I4_re_mmio_space --> I4_sys_ipc_tcb_h
    I4_re_mmio_space --> I4_sys_kernel_object_impl_h
    I4_re_mmio_space --> I4_sys_task_h
    I4_re_mmio_space --> I4_sys_ipc_array
    I4_re_mmio_space --> I4_sys_ipc_iface
    I4_re_mmio_space --> I4_sys_ipc_types_h
    I4_re_mmio_space --> I4_sys_ipc_types_info_h
    I4_re_mmio_space --> I4_sys_ipc_basics_h
    I4_re_mmio_space --> I4_sys_ipc_types_cxx_h
    I4_re_mmio_space --> I4_sys_capability_cxx_h
    I4_re_mmio_space --> I4_sys_kernel_object_h
    I4_re_mmio_space --> I4_sys_ipc_string

    I4_sys_capability --> I4_sys_kobject
    I4_sys_capability --> I4_sys_types_h
    I4_sys_capability --> I4_sys_constants_h
    I4_sys_capability --> I4_sys_timeout_h
    I4_sys_capability --> I4_sys_compiler_h
    I4_sys_capability --> I4_sys_err_h
    I4_sys_capability --> I4_sys_ipc_h
    I4_sys_capability --> I4_sys_ipc_tcb_h
    I4_sys_capability --> I4_sys_kernel_object_impl_h
    I4_sys_capability --> I4_sys_task_h
    I4_sys_capability --> I4_sys_ipc_array
    I4_sys_capability --> I4_sys_ipc_iface
    I4_sys_capability --> I4_sys_ipc_types_h
    I4_sys_capability --> I4_sys_ipc_types_info_h
    I4_sys_capability --> I4_sys_ipc_basics_h
    I4_sys_capability --> I4_sys_ipc_types_cxx_h
    I4_sys_capability --> I4_sys_capability_cxx_h
    I4_sys_capability --> I4_sys_kernel_object_h
    I4_sys_capability --> I4_sys_ipc_string

    I4_sys_meta --> I4_sys_kobject
    I4_sys_meta --> I4_sys_types_h
    I4_sys_meta --> I4_sys_constants_h
    I4_sys_meta --> I4_sys_timeout_h
    I4_sys_meta --> I4_sys_compiler_h
    I4_sys_meta --> I4_sys_err_h
    I4_sys_meta --> I4_sys_ipc_h
    I4_sys_meta --> I4_sys_ipc_tcb_h
    I4_sys_meta --> I4_sys_kernel_object_impl_h
    I4_sys_meta --> I4_sys_task_h
    I4_sys_meta --> I4_sys_ipc_array
    I4_sys_meta --> I4_sys_ipc_iface
    I4_sys_meta --> I4_sys_ipc_types_h
    I4_sys_meta --> I4_sys_ipc_types_info_h
    I4_sys_meta --> I4_sys_ipc_basics_h
    I4_sys_meta --> I4_sys_ipc_types_cxx_h
    I4_sys_meta --> I4_sys_capability_cxx_h
    I4_sys_meta --> I4_sys_kernel_object_h
    I4_sys_meta --> I4_sys_ipc_string

    I4_sys_kobject --> I4_sys_types_h
    I4_sys_kobject --> I4_sys_constants_h
    I4_sys_kobject --> I4_sys_timeout_h
    I4_sys_kobject --> I4_sys_compiler_h
    I4_sys_kobject --> I4_sys_err_h
    I4_sys_kobject --> I4_sys_ipc_h
    I4_sys_kobject --> I4_sys_ipc_tcb_h
    I4_sys_kobject --> I4_sys_kernel_object_impl_h
    I4_sys_kobject --> I4_sys_task_h
    I4_sys_kobject --> I4_sys_ipc_array
    I4_sys_kobject --> I4_sys_ipc_iface
    I4_sys_kobject --> I4_sys_ipc_types_h
    I4_sys_kobject --> I4_sys_ipc_types_info_h
    I4_sys_kobject --> I4_sys_ipc_basics_h
    I4_sys_kobject --> I4_sys_ipc_types_cxx_h
    I4_sys_kobject --> I4_sys_capability_cxx_h
    I4_sys_kobject --> I4_sys_kernel_object_h
    I4_sys_kobject --> I4_sys_ipc_string

    I4_sys_types_h --> I4_sys_constants_h
    I4_sys_types_h --> I4_sys_timeout_h
    I4_sys_types_h --> I4_sys_compiler_h
    I4_sys_types_h --> I4_sys_err_h
    I4_sys_types_h --> I4_sys_ipc_h
    I4_sys_types_h --> I4_sys_ipc_tcb_h
    I4_sys_types_h --> I4_sys_kernel_object_impl_h
    I4_sys_types_h --> I4_sys_task_h
    I4_sys_types_h --> I4_sys_ipc_array
    I4_sys_types_h --> I4_sys_ipc_iface
    I4_sys_types_h --> I4_sys_ipc_types_h
    I4_sys_types_h --> I4_sys_ipc_types_info_h
    I4_sys_types_h --> I4_sys_ipc_basics_h
    I4_sys_types_h --> I4_sys_ipc_types_cxx_h
    I4_sys_types_h --> I4_sys_capability_cxx_h
    I4_sys_types_h --> I4_sys_kernel_object_h
    I4_sys_types_h --> I4_sys_ipc_string

    I4_sys_constants_h --> I4_sys_compiler_h
    I4_sys_constants_h --> I4_sys_err_h
    I4_sys_constants_h --> I4_sys_ipc_h
    I4_sys_constants_h --> I4_sys_ipc_tcb_h
    I4_sys_constants_h --> I4_sys_kernel_object_impl_h
    I4_sys_constants_h --> I4_sys_task_h
    I4_sys_constants_h --> I4_sys_ipc_array
    I4_sys_constants_h --> I4_sys_ipc_iface
    I4_sys_constants_h --> I4_sys_ipc_types_h
    I4_sys_constants_h --> I4_sys_ipc_types_info_h
    I4_sys_constants_h --> I4_sys_ipc_basics_h
    I4_sys_constants_h --> I4_sys_ipc_types_cxx_h
    I4_sys_constants_h --> I4_sys_capability_cxx_h
    I4_sys_constants_h --> I4_sys_kernel_object_h
    I4_sys_constants_h --> I4_sys_ipc_string

    I4_sys_timeout_h --> I4_sys_compiler_h
    I4_sys_timeout_h --> I4_sys_err_h
    I4_sys_timeout_h --> I4_sys_ipc_h
    I4_sys_timeout_h --> I4_sys_ipc_tcb_h
    I4_sys_timeout_h --> I4_sys_kernel_object_impl_h
    I4_sys_timeout_h --> I4_sys_task_h
    I4_sys_timeout_h --> I4_sys_ipc_array
    I4_sys_timeout_h --> I4_sys_ipc_iface
    I4_sys_timeout_h --> I4_sys_ipc_types_h
    I4_sys_timeout_h --> I4_sys_ipc_types_info_h
    I4_sys_timeout_h --> I4_sys_ipc_basics_h
    I4_sys_timeout_h --> I4_sys_ipc_types_cxx_h
    I4_sys_timeout_h --> I4_sys_capability_cxx_h
    I4_sys_timeout_h --> I4_sys_kernel_object_h
    I4_sys_timeout_h --> I4_sys_ipc_string

    I4_sys_compiler_h --> I4_sys_err_h
    I4_sys_compiler_h --> I4_sys_ipc_h
    I4_sys_compiler_h --> I4_sys_ipc_tcb_h
    I4_sys_compiler_h --> I4_sys_kernel_object_impl_h
    I4_sys_compiler_h --> I4_sys_task_h
    I4_sys_compiler_h --> I4_sys_ipc_array
    I4_sys_compiler_h --> I4_sys_ipc_iface
    I4_sys_compiler_h --> I4_sys_ipc_types_h
    I4_sys_compiler_h --> I4_sys_ipc_types_info_h
    I4_sys_compiler_h --> I4_sys_ipc_basics_h
    I4_sys_compiler_h --> I4_sys_ipc_types_cxx_h
    I4_sys_compiler_h --> I4_sys_capability_cxx_h
    I4_sys_compiler_h --> I4_sys_kernel_object_h
    I4_sys_compiler_h --> I4_sys_ipc_string

    I4_sys_err_h --> I4_sys_ipc_h
    I4_sys_err_h --> I4_sys_ipc_tcb_h
    I4_sys_err_h --> I4_sys_kernel_object_impl_h
    I4_sys_err_h --> I4_sys_task_h
    I4_sys_err_h --> I4_sys_ipc_array
    I4_sys_err_h --> I4_sys_ipc_iface
    I4_sys_err_h --> I4_sys_ipc_types_h
    I4_sys_err_h --> I4_sys_ipc_types_info_h
    I4_sys_err_h --> I4_sys_ipc_basics_h
    I4_sys_err_h --> I4_sys_ipc_types_cxx_h
    I4_sys_err_h --> I4_sys_capability_cxx_h
    I4_sys_err_h --> I4_sys_kernel_object_h
    I4_sys_err_h --> I4_sys_ipc_string

    I4_sys_ipc_h --> I4_sys_ipc_tcb_h
    I4_sys_ipc_h --> I4_sys_kernel_object_impl_h
    I4_sys_ipc_h --> I4_sys_task_h
    I4_sys_ipc_h --> I4_sys_ipc_array
    I4_sys_ipc_h --> I4_sys_ipc_iface
    I4_sys_ipc_h --> I4_sys_ipc_types_h
    I4_sys_ipc_h --> I4_sys_ipc_types_info_h
    I4_sys_ipc_h --> I4_sys_ipc_basics_h
    I4_sys_ipc_h --> I4_sys_ipc_types_cxx_h
    I4_sys_ipc_h --> I4_sys_capability_cxx_h
    I4_sys_ipc_h --> I4_sys_kernel_object_h
    I4_sys_ipc_h --> I4_sys_ipc_string

    I4_sys_ipc_tcb_h --> I4_sys_kernel_object_impl_h
    I4_sys_ipc_tcb_h --> I4_sys_task_h
    I4_sys_ipc_tcb_h --> I4_sys_ipc_array
    I4_sys_ipc_tcb_h --> I4_sys_ipc_iface
    I4_sys_ipc_tcb_h --> I4_sys_ipc_types_h
    I4_sys_ipc_tcb_h --> I4_sys_ipc_types_info_h
    I4_sys_ipc_tcb_h --> I4_sys_ipc_basics_h
    I4_sys_ipc_tcb_h --> I4_sys_ipc_types_cxx_h
    I4_sys_ipc_tcb_h --> I4_sys_capability_cxx_h
    I4_sys_ipc_tcb_h --> I4_sys_kernel_object_h
    I4_sys_ipc_tcb_h --> I4_sys_ipc_string

    I4_sys_kernel_object_impl_h --> I4_sys_task_h
    I4_sys_kernel_object_impl_h --> I4_sys_ipc_array
    I4_sys_kernel_object_impl_h --> I4_sys_ipc_iface
    I4_sys_kernel_object_impl_h --> I4_sys_ipc_types_h
    I4_sys_kernel_object_impl_h --> I4_sys_ipc_types_info_h
    I4_sys_kernel_object_impl_h --> I4_sys_ipc_basics_h
    I4_sys_kernel_object_impl_h --> I4_sys_ipc_types_cxx_h
    I4_sys_kernel_object_impl_h --> I4_sys_capability_cxx_h
    I4_sys_kernel_object_impl_h --> I4_sys_kernel_object_h
    I4_sys_kernel_object_impl_h --> I4_sys_ipc_string

    I4_sys_task_h --> I4_sys_ipc_array
    I4_sys_task_h --> I4_sys_ipc_iface
    I4_sys_task_h --> I4_sys_ipc_types_h
    I4_sys_task_h --> I4_sys_ipc_types_info_h
    I4_sys_task_h --> I4_sys_ipc_basics_h
    I4_sys_task_h --> I4_sys_ipc_types_cxx_h
    I4_sys_task_h --> I4_sys_capability_cxx_h
    I4_sys_task_h --> I4_sys_kernel_object_h
    I4_sys_task_h --> I4_sys_ipc_string

    I4_sys_ipc_array --> I4_sys_ipc_iface
    I4_sys_ipc_array --> I4_sys_ipc_types_h
    I4_sys_ipc_array --> I4_sys_ipc_types_info_h
    I4_sys_ipc_array --> I4_sys_ipc_basics_h
    I4_sys_ipc_array --> I4_sys_ipc_types_cxx_h
    I4_sys_ipc_array --> I4_sys_capability_cxx_h
    I4_sys_ipc_array --> I4_sys_kernel_object_h
    I4_sys_ipc_array --> I4_sys_ipc_string

    I4_sys_ipc_iface --> I4_sys_ipc_types_h
    I4_sys_ipc_iface --> I4_sys_ipc_types_info_h
    I4_sys_ipc_iface --> I4_sys_ipc_basics_h
    I4_sys_ipc_iface --> I4_sys_ipc_types_cxx_h
    I4_sys_ipc_iface --> I4_sys_capability_cxx_h
    I4_sys_ipc_iface --> I4_sys_kernel_object_h
    I4_sys_ipc_iface --> I4_sys_ipc_string

    I4_sys_ipc_types_h --> I4_sys_ipc_types_info_h
    I4_sys_ipc_types_h --> I4_sys_ipc_basics_h
    I4_sys_ipc
```

- struct `L4Re::Mmio_space`  
*Interface for memory-like address space accessible via IPC.*

- namespace **L4Re**  
*L4Re C++ Interfaces.*

Interface definition to emit MMIO-like accesses via IPC.

Generated for L4Re by Doxygen

## 16.353 mmio\_space

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * Copyright (C) 2017-2018, 2022 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *
00007  * This file is distributed under the terms of the GNU General Public
00008  * License, version 2. Please see the COPYING-GPL-2 file for details.
00009  */
00014 #pragma once
00015
00016 #include <l4/re/protocols.h>
00017 #include <l4/sys/capability>
00018 #include <l4/sys/cxx/ipc_types>
00019 #include <l4/sys/cxx/ipc_iface>
00020
00021 namespace L4Re
00022 {
00023
00046 struct L4_EXPORT Mmio_space
00047 : public L4::Kobject_t<Mmio_space, L4::Kobject, L4RE_PROTO_MMIO_SPACE>
00048 {
00050     enum Access_width
00051     {
00052         Wd_8bit = 0,
00053         Wd_16bit = 1,
00054         Wd_32bit = 2,
00055         Wd_64bit = 3
00056     };
00057
00059     typedef l4_uint64_t Addr;
00060
00075     L4_INLINE_RPC(long, mmio_read, (Addr addr, char width, l4_uint64_t *value));
00076
00091     L4_INLINE_RPC(long, mmio_write, (Addr addr, char width, l4_uint64_t value));
00092
00093     typedef L4::Typeid::Rpc<mmio_read_t, mmio_write_t> Rpcs;
00094 };
00095
00096 }
```

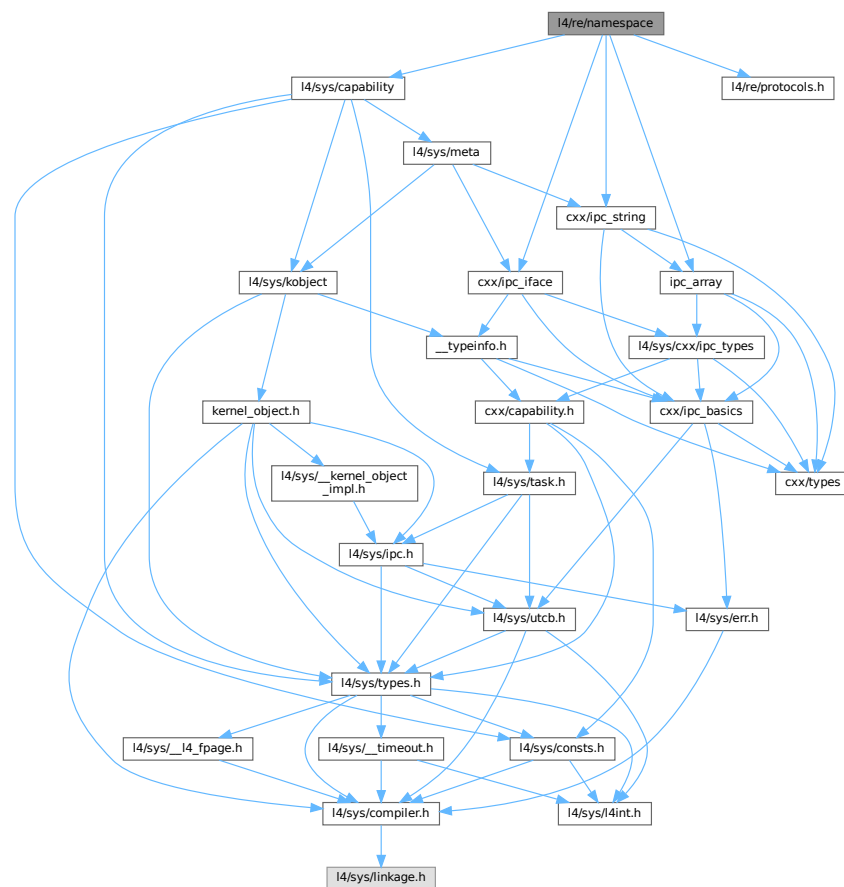
## 16.354 l4/re/namespace File Reference

Namespace interface.

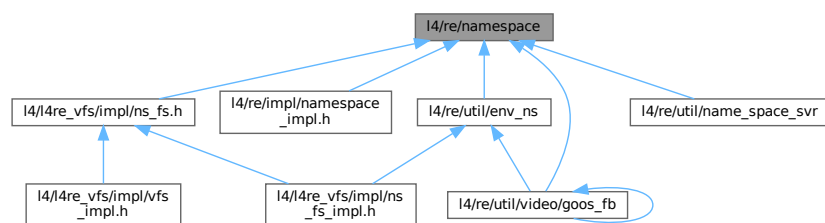
```

#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_array>
#include <l4/sys/cxx/ipc_string>
```

Include dependency graph for namespace:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4Re::Namespace](#)  
*Name-space interface.*

## Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*

## 16.354.1 Detailed Description

Namespace interface.

Definition in file [namespace](#).

## 16.355 namespace

[Go to the documentation of this file.](#)

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  * Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  * Björn Döbel <doebel@os.inf.tu-dresden.de>
00011  * economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction. Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License. This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 #pragma once
00027
00028 #include <l4/sys/capability>
00029 #include <l4/re/protocols.h>
00030 #include <l4/sys/cxx/ipc_iface>
00031 #include <l4/sys/cxx/ipc_array>
00032 #include <l4/sys/cxx/ipc_string>
00033
00034 namespace L4Re {
00035
00060 class L4_EXPORT Namespace :
00061     public L4::Kobject_t<Namespace, L4::Kobject, L4RE_PROTO_NAMESPACE,
00062         L4::Type_info::Demand_t<1> >
00063 {
00064 public:
00068     enum Register_flags
00069     {
00070         Ro      = L4_CAP_FPAGE_RO,
00071         Rw      = L4_CAP_FPAGE_RW,
00072         Rs      = L4_CAP_FPAGE_RS,
00073         Rws     = L4_CAP_FPAGE_RWS,
00074         Strong  = L4_CAP_FPAGE_S,
00075         Trusted = 0x008,
00076
00077         Cap_flags = Ro | Rw | Strong | Trusted,
00078
00079         Link      = 0x100,
00080         Overwrite = 0x200,
00081     };
00082
00088     enum Query_result_flags
00089     {
00090         Partly_resolved = 0x020,
00091     };
00092
00094     enum Query_timeout
00095     {
00096         To_default      = 3600000,
00097         To_non_blocking = 0,
00098     };
00099
00100     L4_RPC_NF(
00101         long, query, (L4::Ipc::Array_ref<char const, unsigned long> name,
00102                     L4::Ipc::Small_buf cap,
00103                     L4::Ipc::Snd_fpage &snd_cap, L4::Ipc::Opt<L4::Opcode &> dummy,
00104                     L4::Ipc::Opt<L4::Ipc::Array_ref<char const, unsigned long> &> out_name));
00105
00131     long query(char const *name, L4::Cap<void> const &cap,
```

```

00132         int timeout = To_default,
00133         l4_umword_t *local_id = 0, bool iterate = true) const noexcept;
00134
00144     long query(char const *name, unsigned len, L4::Cap<void> const &cap,
00145               int timeout = To_default,
00146               l4_umword_t *local_id = 0, bool iterate = true) const noexcept;
00147
00148     L4_RPC_NF(long, register_obj, (unsigned flags,
00149                                   L4::Ipc::Array<char const, unsigned long> name,
00150                                   L4::Ipc::Opt< L4::Ipc::Cap<void> > obj),
00151               L4::Ipc::Call_t<L4_CAP_FPAGE_W>);
00152
00176     long register_obj(char const *name, L4::Ipc::Cap<void> obj,
00177                       unsigned flags = Rw) const noexcept
00178     {
00179         return register_obj_t::call(c(), flags,
00180                                     L4::Ipc::Array<char const, unsigned long>(
00181                                         __builtin_strlen(name), name),
00182                                     obj);
00183     }
00184
00185     L4_RPC_NF_OP(3, // backward compatibility opcode
00186                 long, unlink, (L4::Ipc::Array<char const, unsigned long> name),
00187                 L4::Ipc::Call_t<L4_CAP_FPAGE_W>);
00188
00203     long unlink(char const* name)
00204     {
00205         return unlink_t::call(c(), L4::Ipc::Array<char const, unsigned long>(
00206                                         __builtin_strlen(name), name));
00207     }
00208
00209     typedef L4::Typeid::Rpcs<query_t, register_obj_t, unlink_t> Rpcs;
00210
00211 private:
00212     long _query(char const *name, unsigned len,
00213                L4::Cap<void> const &target, l4_umword_t *local_id,
00214                bool iterate) const noexcept;
00215
00216 };
00217
00218 };

```

## 16.356 l4/re/namespace-sys.h File Reference

Namespace protocol definitions.

### Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*

### Enumerations

- enum [L4Re::Namespace\\_::Opcodes](#)  
*Name-space communication-protocol opcodes.*

## 16.356.1 Detailed Description

Namespace protocol definitions.

Definition in file [namespace-sys.h](#).

## 16.357 namespace-sys.h

[Go to the documentation of this file.](#)

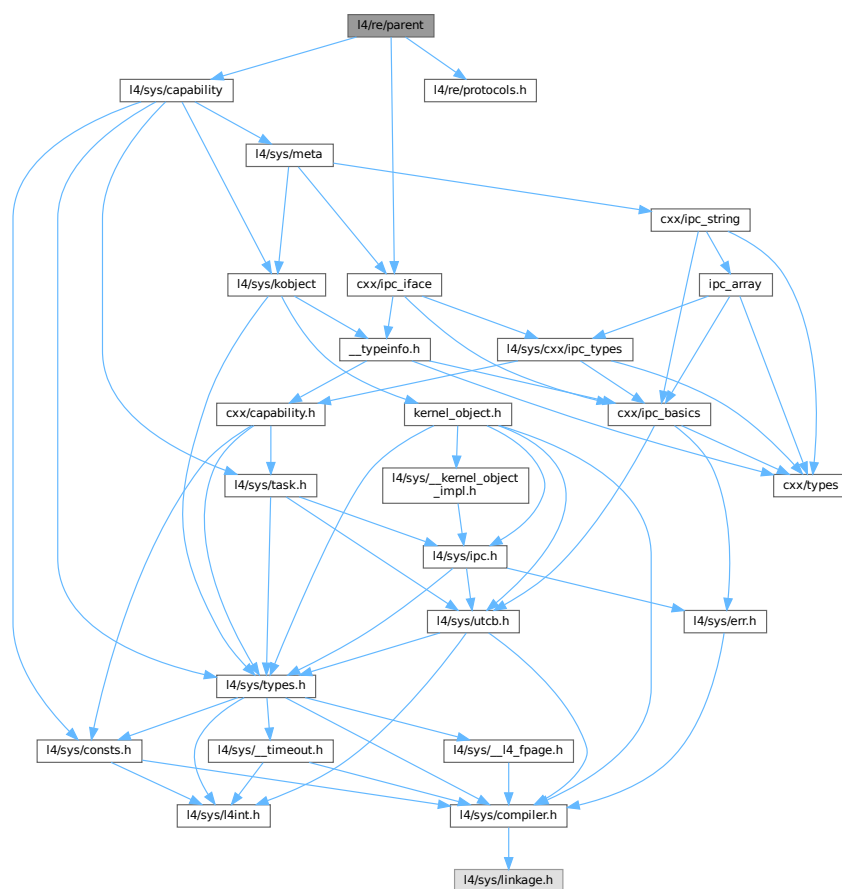
```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 namespace L4Re {
00026     namespace Namespace_
00027     {
00033         enum Opcodes { Query, Register, Link, Unlink };
00034     };
00035 };
```

## 16.358 l4/re/parent File Reference

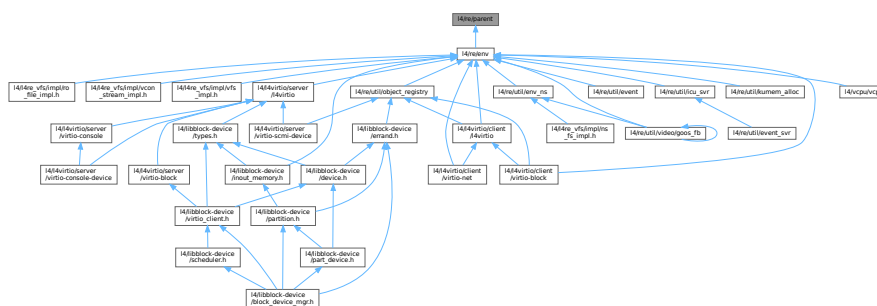
Parent interface.

```
#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/ipc_iface>
```

Include dependency graph for parent:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4Re::Parent](#)  
*Parent* interface.

## Namespaces

- namespace [L4Re](#)  
*[L4Re](#) C++ Interfaces.*

### 16.358.1 Detailed Description

Parent interface.

Definition in file [parent](#).

## 16.359 parent

[Go to the documentation of this file.](#)

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/sys/capability>
00028 #include <l4/re/protocols.h>
00029 #include <l4/sys/cxx/ipc_iface>
00030
00031 namespace L4Re {
00032
00053 class L4_EXPORT Parent :
00054     public L4::Kobject_t<Parent, L4::Kobject, L4RE_PROTO_PARENT>
00055 {
00056 public:
00072     L4_INLINE_RPC(long, signal, (unsigned long sig, unsigned long val));
00073     typedef L4::Typeid::Rpc<signal_t> Rpc;
00074 };
00075 };
00076
```

## 16.360 l4/re/parent-sys.h File Reference

Parent protocol definition.

## Namespaces

- namespace [L4Re](#)  
*[L4Re](#) C++ Interfaces.*



## Enumerations

- enum [L4Re::Parent\\_::Opcodes](#)  
*Parent communication-protocol opcodes.*

### 16.360.1 Detailed Description

Parent protocol definition.

Definition in file [parent-sys.h](#).

## 16.361 parent-sys.h

[Go to the documentation of this file.](#)

```

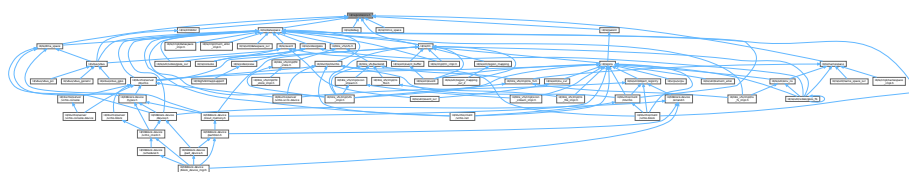
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 namespace L4Re
00026 {
00027     namespace Parent_
00028     {
00034         enum Opcodes { Signal };
00035     };
00036 };

```

## 16.362 I4/re/protocols.h File Reference

[L4Re](#) Protocol Constants (C version)

This graph shows which files directly or indirectly include this file:



## Enumerations

- enum [L4re\\_protocols](#) {  
[L4RE\\_PROTO\\_DATASPACE](#) = 0x4000 , [L4RE\\_PROTO\\_NAMESPACE](#) , [L4RE\\_PROTO\\_PARENT](#) ,  
[L4RE\\_PROTO\\_GOOS](#) ,  
[L4RE\\_PROTO\\_RSVD\\_1](#) , [L4RE\\_PROTO\\_RM](#) , [L4RE\\_PROTO\\_EVENT](#) , [L4RE\\_PROTO\\_INHIBITOR](#) ,  
[L4RE\\_PROTO\\_DMA\\_SPACE](#) , [L4RE\\_PROTO\\_MMIO\\_SPACE](#) , [L4RE\\_PROTO\\_DEBUG](#) = ~0x7fffL }

### 16.362.1 Detailed Description

[L4Re](#) Protocol Constants (C version)

Definition in file [protocols.h](#).

### 16.362.2 Enumeration Type Documentation

#### 16.362.2.1 [L4re\\_protocols](#)

enum [L4re\\_protocols](#)

##### Enumerator

<a href="#">L4RE_PROTO_DATASPACE</a>	ID for <a href="#">L4Re::Dataspace</a> RPCs
<a href="#">L4RE_PROTO_NAMESPACE</a>	ID for <a href="#">L4Re::Namespace</a> RPCs
<a href="#">L4RE_PROTO_PARENT</a>	ID for <a href="#">L4Re::Parent</a> RPCs
<a href="#">L4RE_PROTO_GOOS</a>	ID for <a href="#">L4Re::Video::Goos</a> RPCs
<a href="#">L4RE_PROTO_RSVD_1</a>	Reserved ID
<a href="#">L4RE_PROTO_RM</a>	ID for <a href="#">L4Re::Rm</a> RPCs
<a href="#">L4RE_PROTO_EVENT</a>	ID for <a href="#">L4Re::Event</a> RPCs
<a href="#">L4RE_PROTO_INHIBITOR</a>	ID for <a href="#">L4Re::Inhibitor</a> RPCs
<a href="#">L4RE_PROTO_DMA_SPACE</a>	ID for <a href="#">L4Re::Dma_space</a> RPCs
<a href="#">L4RE_PROTO_MMIO_SPACE</a>	ID for <a href="#">L4Re::Mmio_space</a>
<a href="#">L4RE_PROTO_DEBUG</a>	ID for debugging RPCs

Definition at line 30 of file [protocols.h](#).

## 16.363 [protocols.h](#)

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2015 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00030 enum L4re_protocols
00031 {
00032     L4RE_PROTO_DATASPACE = 0x4000,
00033     L4RE_PROTO_NAMESPACE,
00034     L4RE_PROTO_PARENT,
00035     L4RE_PROTO_GOOS,
00036     L4RE_PROTO_RSVD_1,
00037     L4RE_PROTO_RM,
00038     L4RE_PROTO_EVENT,
00039     L4RE_PROTO_INHIBITOR,
00040     L4RE_PROTO_DMA_SPACE,
00041     L4RE_PROTO_MMIO_SPACE,
00043     L4RE_PROTO_DEBUG = ~0x7ffffL
00044 };
00045

```

## 16.364 l4/re/random File Reference

Random number generator interface definition.

```

#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/icu>

```

- struct `L4Re::Random`  
*Low-bandwidth interface for random number generators.*

- namespace **L4Re**  
*L4Re C++ Interfaces.*

Definition in file [random](#).

```
00001 // -*- Mode: C++ -*-
00002 /* SPDX-License-Identifier: ((GPL-2.0-only WITH mif-exception) OR LicenseRef-kk-custom) */
00003 /*
00004  * Copyright (C) 2019-2020, 2022 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *
```

## 16.366 I4/re/rm File Reference

```
#include <l4/sys/types.h>
#include <l4/sys/l4int.h>
#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/pager>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_ret_array>
#include <l4/sys/cxx/types>
#include <l4/re/consts>
#include <l4/re/dataspace>
```



```

00025  * the GNU General Public License.
00026  */
00027  #pragma once
00028
00029  #include <l4/sys/types.h>
00030  #include <l4/sys/l4int.h>
00031  #include <l4/sys/capability>
00032  #include <l4/re/protocols.h>
00033  #include <l4/sys/pager>
00034  #include <l4/sys/cxx/ipc_iface>
00035  #include <l4/sys/cxx/ipc_ret_array>
00036  #include <l4/sys/cxx/types>
00037  #include <l4/re/consts>
00038  #include <l4/re/dataspace>
00039
00040  namespace L4Re {
00041
00092  class L4_EXPORT Rm :
00093  public L4::Kobject_t<Rm, L4::Pager, L4RE_PROTO_RM,
00094  L4::Type_info::Demand_t<1> >
00095  {
00096  public:
00097  typedef L4Re::Dataspace::Offset Offset;
00098
00100  enum Detach_result
00101  {
00102  Detached_ds = 0,
00103  Kept_ds     = 1,
00104  Split_ds   = 2,
00105  Detach_result_mask = 3,
00106
00107  Detach_again = 4,
00108  };
00109
00112  enum Region_flag_shifts
00113  {
00115  Caching_shift = Dataspace::F::Caching_shift,
00116  };
00117
00119  struct F
00120  {
00122  enum Attach_flags : l4_uint32_t
00123  {
00125  Search_addr = 0x20000,
00127  In_area     = 0x40000,
00129  Eager_map   = 0x80000,
00131  No_eager_map = 0x100000,
00133  Attach_mask = 0x1f0000,
00134  };
00135
00136  L4_TYPES_FLAGS_OPS_DEF(Attach_flags);
00137
00139  enum Region_flags : l4_uint16_t
00140  {
00142  Rights_mask = 0x0f,
00144  R           = Dataspace::F::R,
00146  W           = Dataspace::F::W,
00148  X           = Dataspace::F::X,
00150  RW          = Dataspace::F::RW,
00152  RX          = Dataspace::F::RX,
00154  RWX         = Dataspace::F::RWX,
00155
00157  Detach_free = 0x200,
00159  Pager       = 0x400,
00161  Reserved    = 0x800,
00162
00163
00165  Caching_mask = Dataspace::F::Caching_mask,
00167  Cache_normal = Dataspace::F::Normal,
00169  Cache_buffered = Dataspace::F::Bufferable,
00171  Cache_uncached = Dataspace::F::Uncacheable,
00172
00174  Ds_map_mask = 0xff,
00175
00177  Region_flags_mask = 0xffff,
00178  };
00179
00180  L4_TYPES_FLAGS_OPS_DEF(Region_flags);
00181
00182  friend constexpr Dataspace::Flags map_flags(Region_flags rf)
00183  {
00184  return Dataspace::Flags(static_cast<l4_uint16_t>(rf) & Ds_map_mask);
00185  }
00186
00187  struct Flags : L4::Types::Flags_ops_t<Flags>
00188  {

```

```

00189     l4_uint32_t raw;
00190     Flags() = default;
00191     explicit constexpr Flags(l4_uint32_t f) : raw(f) {}
00192     constexpr Flags(Attach_flags rf) : raw(static_cast<l4_uint32_t>(rf)) {}
00193     constexpr Flags(Region_flags rf) : raw(static_cast<l4_uint32_t>(rf)) {}
00194
00195     friend constexpr Dataspace::Flags map_flags(Flags f)
00196     {
00197         return Dataspace::Flags(f.raw & Ds_map_mask);
00198     }
00199
00200     constexpr Region_flags region_flags() const
00201     {
00202         return Region_flags(raw & Region_flags_mask);
00203     }
00204
00205     constexpr Attach_flags attach_flags() const
00206     {
00207         return Attach_flags(raw & Attach_mask);
00208     }
00209
00210     constexpr bool r() const { return raw & L4_FPAGE_RO; }
00211     constexpr bool w() const { return raw & L4_FPAGE_W; }
00212     constexpr bool x() const { return raw & L4_FPAGE_X; }
00213     constexpr unsigned cap_rights() const
00214     { return w() ? L4_CAP_FPAGE_RW : L4_CAP_FPAGE_RO; }
00215 };
00216
00217     friend constexpr Flags operator | (Region_flags l, Attach_flags r)
00218     { return Flags(l) | Flags(r); }
00219
00220     friend constexpr Flags operator | (Attach_flags l, Region_flags r)
00221     { return Flags(l) | Flags(r); }
00222 };
00223
00224     using Attach_flags = F::Attach_flags;
00225     using Region_flags = F::Region_flags;
00226     using Flags = F::Flags;
00227
00229     enum Detach_flags
00230     {
00240         Detach_exact = 1,
00250         Detach_overlap = 2,
00259
00260         Detach_keep = 4,
00261     };
00262
00288     long reserve_area(l4_addr_t *start, unsigned long size,
00289                      Flags flags = Flags(0),
00290                      unsigned char align = L4_PAGESHIFT) const noexcept
00291     { return reserve_area_t::call(c(), start, size, flags, align); }
00292
00293     L4_RPC_NF(long, reserve_area, (L4::Ipc::In_out<l4_addr_t *> start,
00294                                   unsigned long size,
00295                                   Flags flags,
00296                                   unsigned char align));
00297
00313     template< typename T >
00314     long reserve_area(T **start, unsigned long size,
00315                      Flags flags = Flags(0),
00316                      unsigned char align = L4_PAGESHIFT) const noexcept
00317     {
00318         return reserve_area_t::call(c(), reinterpret_cast<l4_addr_t*>(start), size,
00319                                     flags, align);
00320     }
00321
00334     L4_RPC(long, free_area, (l4_addr_t addr));
00335
00336     L4_RPC_NF(long, attach, (L4::Ipc::In_out<l4_addr_t *> start,
00337                              unsigned long size, Flags flags,
00338                              L4::Ipc::Opt<L4::Ipc::Cap<Dataspace> > mem,
00339                              Offset offs, unsigned char align,
00340                              L4::Ipc::Opt<l4_cap_idx_t> client_cap));
00341
00342     L4_RPC_NF(long, detach, (l4_addr_t addr, unsigned long size, unsigned flags,
00343                              l4_addr_t &start, l4_addr_t &rsz,
00344                              l4_cap_idx_t &mem_cap));
00345
00401     long attach(l4_addr_t *start, unsigned long size, Flags flags,
00402                L4::Ipc::Cap<Dataspace> mem, Offset offs = 0,
00403                unsigned char align = L4_PAGESHIFT,
00404                L4::Cap<L4::Task> const task
00405                = L4::Cap<L4::Task>::Invalid) const noexcept;
00406
00410     template< typename T >
00411     long attach(T **start, unsigned long size, Flags flags,
00412                L4::Ipc::Cap<Dataspace> mem, Offset offs = 0,

```



```

00413         unsigned char align = L4_PAGESHIFT,
00414         L4::Cap<L4::Task> const task
00415         = L4::Cap<L4::Task>::Invalid) const noexcept
00416     {
00417         union X { L4_addr_t a; T* t; };
00418         X *x = reinterpret_cast<X*>(start);
00419         return attach(&x->a, size, flags, mem, offs, align, task);
00420     }
00421
00422 #if __cplusplus >= 201103L
00433     template< typename T >
00434     class Unique_region
00435     {
00436     private:
00437         T _addr;
00438         L4::Cap<Rm> _rm;
00439
00440     public:
00441         Unique_region(Unique_region const &) = delete;
00442         Unique_region &operator = (Unique_region const &) = delete;
00443
00444         Unique_region() noexcept
00445         : _addr(0), _rm(L4::Cap<Rm>::Invalid) {}
00446
00447         explicit Unique_region(T addr) noexcept
00448         : _addr(addr), _rm(L4::Cap<Rm>::Invalid) {}
00449
00450         Unique_region(T addr, L4::Cap<Rm> const &rm) noexcept
00451         : _addr(addr), _rm(rm) {}
00452
00453         Unique_region(Unique_region &&o) noexcept : _addr(o.get()), _rm(o._rm)
00454         { o.release(); }
00455
00456         Unique_region &operator = (Unique_region &&o) noexcept
00457         {
00458             if (&o != this)
00459             {
00460                 if (_rm.is_valid())
00461                     _rm->detach(reinterpret_cast<L4_addr_t>(_addr), 0);
00462                 _rm = o._rm;
00463                 _addr = o.release();
00464             }
00465             return *this;
00466         }
00467
00468         ~Unique_region() noexcept
00469         {
00470             if (_rm.is_valid())
00471                 _rm->detach(reinterpret_cast<L4_addr_t>(_addr), 0);
00472         }
00473
00474         T get() const noexcept
00475         { return _addr; }
00476
00477         T release() noexcept
00478         {
00479             _rm = L4::Cap<Rm>::Invalid;
00480             return _addr;
00481         }
00482
00483         void reset(T addr, L4::Cap<Rm> const &rm) noexcept
00484         {
00485             if (_rm.is_valid())
00486                 _rm->detach(L4_addr_t(_addr), 0);
00487
00488             _rm = rm;
00489             _addr = addr;
00490         }
00491
00492         void reset() noexcept
00493         { reset(0, L4::Cap<Rm>::Invalid); }
00494
00495         bool is_valid() const noexcept
00496         { return _rm.is_valid(); }
00497
00498         T operator * () const noexcept { return _addr; }
00499
00500         T operator -> () const noexcept { return _addr; }
00501     };
00502
00503     template< typename T >
00504     long attach(Unique_region<T> *start, unsigned long size, Flags flags,
00505                L4::Ipc::Cap<Dataspace> mem, Offset offs = 0,
00506                unsigned char align = L4_PAGESHIFT,
00507                L4::Cap<L4::Task> const task
00508                = L4::Cap<L4::Task>::Invalid) const noexcept
00509     {

```

```

00565     l4_addr_t addr = reinterpret_cast<l4_addr_t>(start->get());
00566
00567     long res = attach(&addr, size, flags, mem, offs, align, task);
00568     if (res < 0)
00569         return res;
00570
00571     start->reset(reinterpret_cast<T>(addr), L4::Cap<Rm>(cap()));
00572     return res;
00573 }
00574 #endif
00575
00593 int detach(l4_addr_t addr, L4::Cap<Dataspace> *mem,
00594           L4::Cap<L4::Task> const &task = This_task) const noexcept;
00595
00599 int detach(void *addr, L4::Cap<Dataspace> *mem,
00600           L4::Cap<L4::Task> const &task = This_task) const noexcept;
00601
00621 int detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
00622           L4::Cap<L4::Task> const &task) const noexcept;
00623
00668 long find(l4_addr_t *addr, unsigned long *size, Offset *offset,
00669          L4Re::Rm::Flags *flags, L4::Cap<Dataspace> *m) noexcept
00670 { return find_t::call(c(), addr, size, flags, offset, m); }
00671
00672 L4_RPC_NF(long, find, (L4::Ipc::In_out<l4_addr_t *> addr,
00673                      L4::Ipc::In_out<unsigned long *> size,
00674                      L4Re::Rm::Flags *flags, Offset *offset,
00675                      L4::Ipc::As_value<L4::Cap<Dataspace> > *m));
00676
00680 struct Range
00681 {
00683     l4_addr_t start;
00685     l4_addr_t end;
00686 };
00687
00693 using Region = Range;
00694
00701 using Area = Range;
00702
00717 L4_RPC(long, get_regions, (l4_addr_t start, L4::Ipc::Ret_array<Range> regions));
00718
00734 L4_RPC(long, get_areas, (l4_addr_t start, L4::Ipc::Ret_array<Range> areas));
00735
00736 int detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
00737           L4::Cap<L4::Task> task, unsigned flags) const noexcept;
00738
00739 typedef L4::Typeid::Rpc<attach_t, detach_t, find_t,
00740                      reserve_area_t, free_area_t,
00741                      get_regions_t, get_areas_t> Rpc< >;
00742 };
00743
00744
00745 inline int
00746 Rm::detach(l4_addr_t addr, L4::Cap<Dataspace> *mem,
00747           L4::Cap<L4::Task> const &task) const noexcept
00748 { return detach(addr, 1, mem, task, Detach_overlap); }
00749
00750 inline int
00751 Rm::detach(void *addr, L4::Cap<Dataspace> *mem,
00752           L4::Cap<L4::Task> const &task) const noexcept
00753 {
00754     return detach(reinterpret_cast<l4_addr_t>(addr), 1, mem, task,
00755                  Detach_overlap);
00756 }
00757
00758 inline int
00759 Rm::detach(l4_addr_t addr, unsigned long size, L4::Cap<Dataspace> *mem,
00760           L4::Cap<L4::Task> const &task) const noexcept
00761 { return detach(addr, size, mem, task, Detach_exact); }
00762
00763 };

```

## 16.368 l4/re/rm-sys.h File Reference

Region mapper protocol definitions.

### Namespaces

- namespace [L4Re](#)  
[L4Re](#) C++ Interfaces.

## Enumerations

- enum [L4Re::Rm\\_::Opcodes](#)

*Region-map communication-protocol opcodes.*

### 16.368.1 Detailed Description

Region mapper protocol definitions.

Definition in file [rm-sys.h](#).

## 16.369 rm-sys.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  * Alexander Warg <warg@os.inf.tu-dresden.de>
00008  * economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 namespace L4Re
00026 {
00027     namespace Rm_
00028     {
00034         enum Opcodes
00035         {
00036             Attach, Detach, Find, Attach_area, Detach_area, Get_regions, Get_areas
00037         };
00038     };
00039 };

```

### 16.370 l4/re/shared\_cap File Reference

Shared\_cap / Shared\_del\_cap.

```

#include <l4/re/cap_alloc>
#include <l4/sys/cxx/smart_capability_lx>

```

```
graph BT; A[l4/l4re_vfs/impl/vfs_impl.h] --> B[l4/re/shared_cap]
```

Diagram illustrating the relationship between two components:

- Top component: `l4/re/shared_cap`
- Bottom component: `l4/l4re_vfs/impl/vfs_impl.h`

A blue arrow points from the bottom component to the top component, indicating a dependency or relationship.

- namespace **L4Re**  
*L4Re C++ Interfaces.*

- `template<typename T >`  
`using L4Re::Shared_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_ALL_SPACES > >`  
*Shared capability that implements automatic free and unmap of the capability selector.*
- `template<typename T >`  
`using L4Re::shared_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_ALL_SPACES > >`

*Shared capability that implements automatic free and unmap of the capability selector.*

- `template<typename T >`  
`using L4Re::Shared_del_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_DELETE_OBJ`  
`> >`

*Shared capability that implements automatic free and unmap+delete of the capability selector.*

- `template<typename T >`  
`using L4Re::shared_del_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_DELETE_OBJ`  
`> >`

*Shared capability that implements automatic free and unmap+delete of the capability selector.*

## Functions

- `template<typename T >`  
`Shared_cap< T > L4Re::make_shared_cap (L4Re::Cap_alloc *ca)`  
*Allocate a capability slot and wrap it in a Shared\_cap.*
- `template<typename T >`  
`Shared_del_cap< T > L4Re::make_shared_del_cap (L4Re::Cap_alloc *ca)`  
*Allocate a capability slot and wrap it in a Shared\_del\_cap.*

## 16.370.1 Detailed Description

Shared\_cap / Shared\_del\_cap.

Definition in file [shared\\_cap](#).

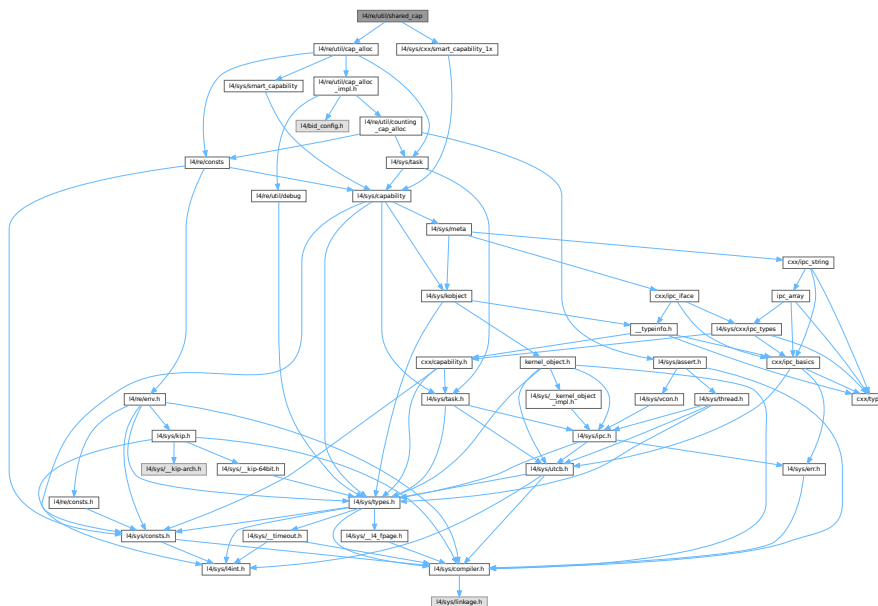
## 16.371 shared\_cap

[Go to the documentation of this file.](#)

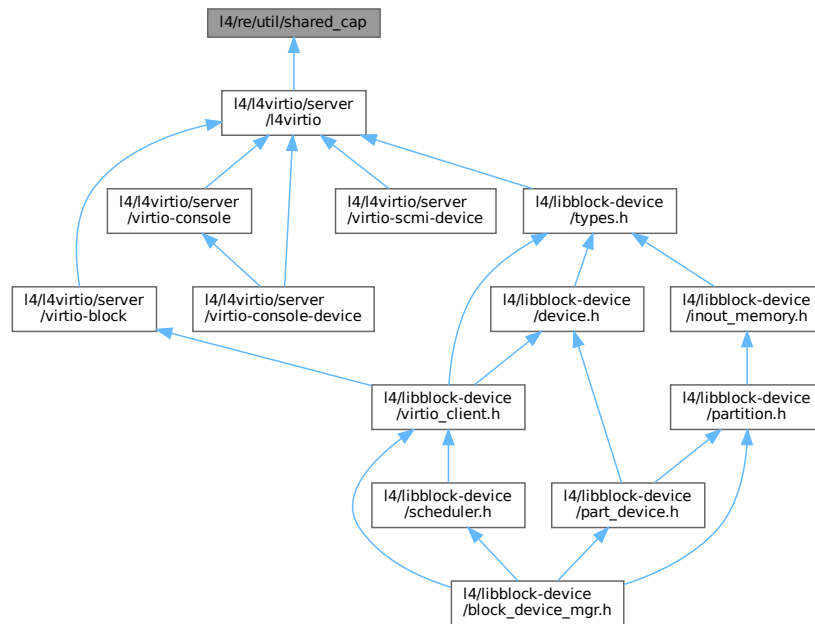
```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2018 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00025 #include <l4/re/cap_alloc>
00026 #include <l4/sys/cxx/smart_capability_1x>
00027
00028 namespace L4Re {
00029
00043 template< typename T >
00044 using Shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>;
00046 template< typename T >
00047 using shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>;
00048
00058 template< typename T >
00059 Shared_cap<T>
00060 make_shared_cap(L4Re::Cap_alloc *ca)
00061 { return Shared_cap<T>(ca->alloc<T>(), ca); }
00062
```

### 16.372 l4/re/util/shared\_cap File Reference

```
#include <l4/re/util/cap_alloc>
#include <l4/sys/cxx/smart_capability_1x>
Include dependency graph for shared_cap:
```



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [L4Re](#)  
*[L4Re](#) C++ Interfaces.*
- namespace [L4Re::Util](#)  
*Documentation of the [L4](#) Runtime Environment utility functionality in C++.*

## Typedefs

- template<typename T >  
using [L4Re::Util::Shared\\_cap](#) = L4::Detail::Shared\_cap\_impl< T, Smart\_count\_cap< [L4\\_FP\\_ALL\\_SPACES](#) > >  
*Shared capability that implements automatic free and unmap of the capability selector.*
- template<typename T >  
using [L4Re::Util::shared\\_cap](#) = L4::Detail::Shared\_cap\_impl< T, Smart\_count\_cap< [L4\\_FP\\_ALL\\_SPACES](#) > >  
*Shared capability that implements automatic free and unmap of the capability selector.*
- template<typename T >  
using [L4Re::Util::Shared\\_del\\_cap](#) = L4::Detail::Shared\_cap\_impl< T, Smart\_count\_cap< [L4\\_FP\\_DELETE\\_OBJ](#) > >  
*Shared capability that implements automatic free and unmap+delete of the capability selector.*
- template<typename T >  
using [L4Re::Util::shared\\_del\\_cap](#) = L4::Detail::Shared\_cap\_impl< T, Smart\_count\_cap< [L4\\_FP\\_DELETE\\_OBJ](#) > >  
*Shared capability that implements automatic free and unmap+delete of the capability selector.*

## Functions

- `template<typename T>`  
[Shared\\_cap< T > L4Re::Util::make\\_shared\\_cap \(\)](#)  
*Allocate a capability slot and wrap it in a Shared\_cap.*
- `template<typename T>`  
[Shared\\_del\\_cap< T > L4Re::Util::make\\_shared\\_del\\_cap \(\)](#)  
*Allocate a capability slot and wrap it in a Shared\_del\_cap.*

### 16.372.1 Detailed Description

Shared\_cap / Shared\_del\_cap.

Definition in file [shared\\_cap](#).

## 16.373 shared\_cap

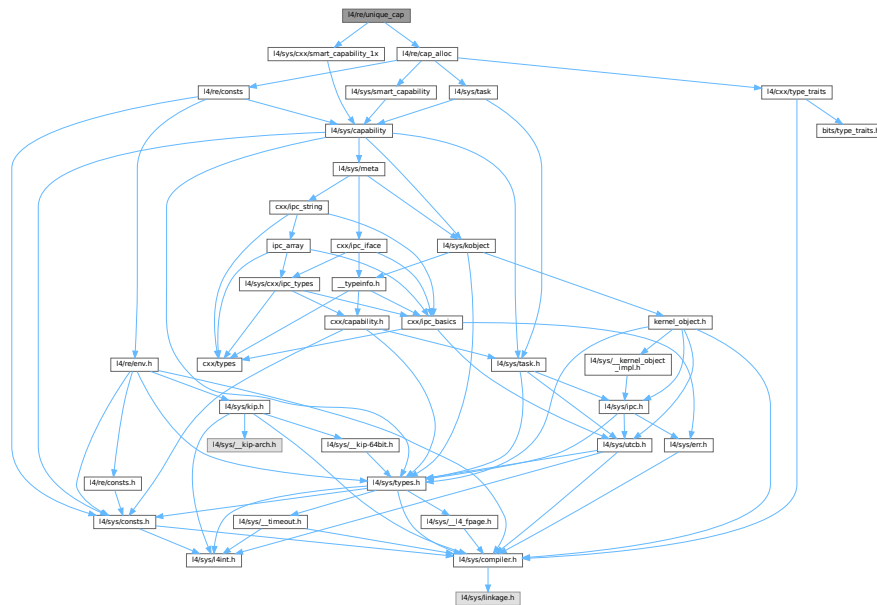
[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00025 #include <l4/re/util/cap_alloc>
00026 #include <l4/sys/cxx/smart_capability_lx>
00027
00028 namespace L4Re { namespace Util {
00029
00058 template< typename T >
00059 using Shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>;
00061 template< typename T >
00062 using shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>;
00063
00069 template< typename T >
00070 Shared_cap<T>
00071 make_shared_cap()
00072 { return Shared_cap<T>(cap_alloc.alloc<T>()); }
00073
00108 template< typename T >
00109 using Shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>;
00111 template< typename T >
00112 using shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>;
00113
00119 template< typename T >
00120 Shared_del_cap<T>
00121 make_shared_del_cap()
00122 { return Shared_del_cap<T>(cap_alloc.alloc<T>()); }
00123
00124 }} // namespace L4Re::Util
00125
```

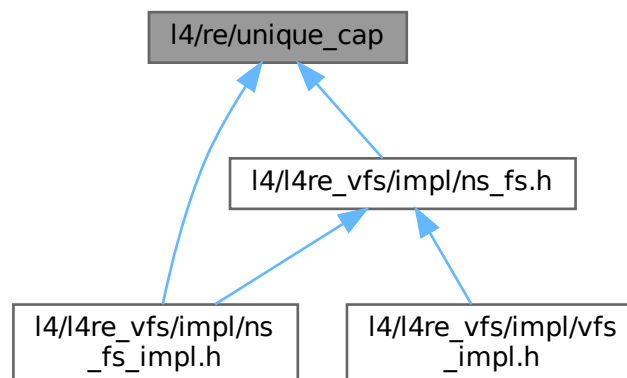


Unique\_cap / Unique\_del\_cap.

```
#include <l4/re/cap_alloc>
#include <l4/sys/cxx/smart_capability_1x>
Include dependency graph for unique_cap:
```



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace **L4Re**  
*L4Re C++ Interfaces.*

## Typedefs

- `template<typename T>`  
`using L4Re::Unique_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_ALL_SPACES > >`  
*Unique capability that implements automatic free and unmap of the capability selector.*
- `template<typename T>`  
`using L4Re::unique_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_ALL_SPACES > >`  
*Unique capability that implements automatic free and unmap of the capability selector.*
- `template<typename T>`  
`using L4Re::Unique_del_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_DELETE_OBJ > >`  
`> >`  
*Unique capability that implements automatic free and unmap+delete of the capability selector.*
- `template<typename T>`  
`using L4Re::unique_del_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_DELETE_OBJ >`  
`>`  
*Unique capability that implements automatic free and unmap+delete of the capability selector.*

## Functions

- `template<typename T>`  
`Unique_cap< T > L4Re::make_unique_cap (L4Re::Cap_alloc *ca)`  
*Allocate a capability slot and wrap it in an Unique\_cap.*
- `template<typename T>`  
`Unique_del_cap< T > L4Re::make_unique_del_cap (L4Re::Cap_alloc *ca)`  
*Allocate a capability slot and wrap it in an Unique\_del\_cap.*

### 16.374.1 Detailed Description

Unique\_cap / Unique\_del\_cap.

Definition in file [unique\\_cap](#).

## 16.375 unique\_cap

[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019 #include <l4/re/cap_alloc>
00020 #include <l4/sys/cxx/smart_capability_lx>
00021 namespace L4Re {
```

```

00029
00041 template< typename T >
00042 using Unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>;
00044 template< typename T >
00045 using unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>;
00046
00056 template< typename T >
00057 Unique_cap<T>
00058 make_unique_cap(L4Re::Cap_alloc *ca)
00059 { return Unique_cap<T>(ca->alloc<T>(), ca); }
00060
00074 template< typename T >
00075 using Unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>;
00077 template<typename T>
00078 using unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>;
00079
00089 template< typename T >
00090 Unique_del_cap<T>
00091 make_unique_del_cap(L4Re::Cap_alloc *ca)
00092 { return Unique_del_cap<T>(ca->alloc<T>(), ca); }
00093
00094 }

```

## 16.376 l4/re/util/unique\_cap File Reference

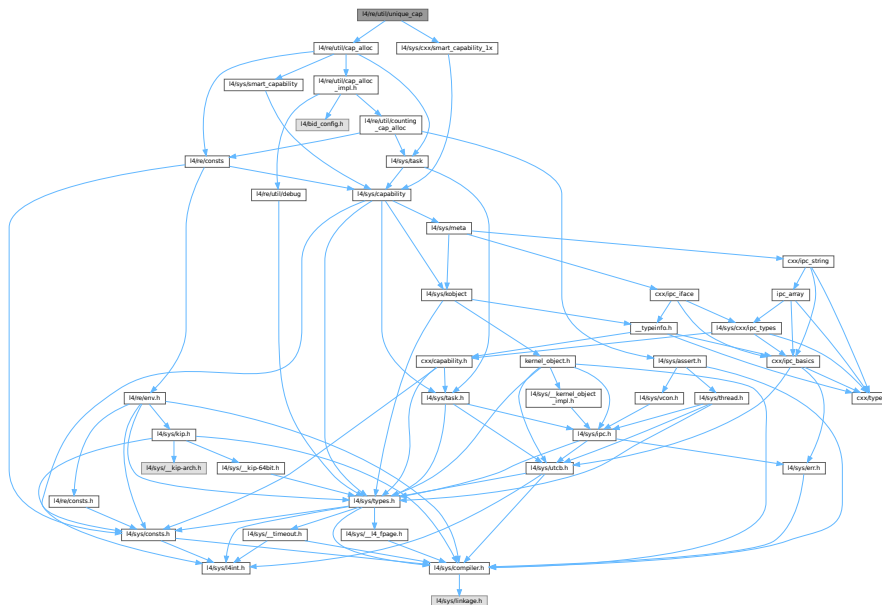
Unique\_cap / Unique\_del\_cap.

```

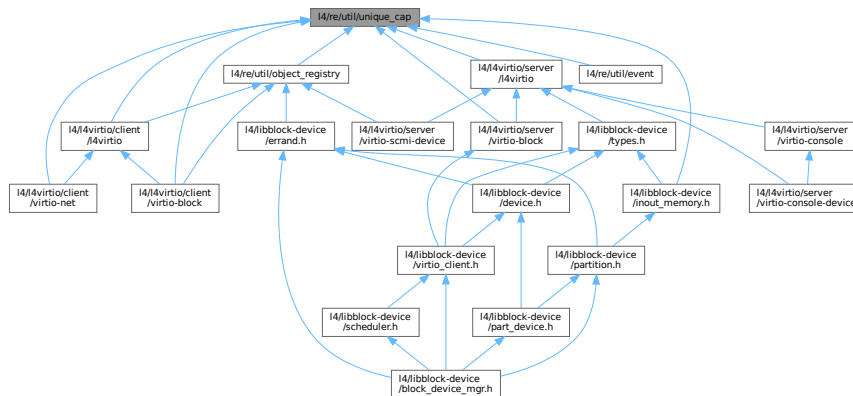
#include <l4/re/util/cap_alloc>
#include <l4/sys/cxx/smart_capability_1x>

```

Include dependency graph for unique\_cap:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*
- namespace [L4Re::Util](#)  
*Documentation of the L4 Runtime Environment utility functionality in C++.*

## Typedefs

- template<typename T >  
using [L4Re::Util::Unique\\_cap](#) = L4::Detail::Unique\_cap\_impl< T, [Smart\\_cap\\_auto](#)< [L4\\_FP\\_ALL\\_SPACES](#) > >  
*Unique capability that implements automatic free and unmap of the capability selector.*
- template<typename T >  
using [L4Re::Util::unique\\_cap](#) = L4::Detail::Unique\_cap\_impl< T, [Smart\\_cap\\_auto](#)< [L4\\_FP\\_ALL\\_SPACES](#) > >  
*Unique capability that implements automatic free and unmap of the capability selector.*
- template<typename T >  
using [L4Re::Util::Unique\\_del\\_cap](#) = L4::Detail::Unique\_cap\_impl< T, [Smart\\_cap\\_auto](#)< [L4\\_FP\\_DELETE\\_OBJ](#) > >  
*Unique capability that implements automatic free and unmap+delete of the capability selector.*
- template<typename T >  
using [L4Re::Util::unique\\_del\\_cap](#) = L4::Detail::Unique\_cap\_impl< T, [Smart\\_cap\\_auto](#)< [L4\\_FP\\_DELETE\\_OBJ](#) > >  
*Unique capability that implements automatic free and unmap+delete of the capability selector.*

## Functions

- template<typename T >  
[Unique\\_cap](#)< T > [L4Re::Util::make\\_unique\\_cap](#) ()  
*Allocate a capability slot and wrap it in an Unique\_cap.*
- template<typename T >  
[Unique\\_del\\_cap](#)< T > [L4Re::Util::make\\_unique\\_del\\_cap](#) ()  
*Allocate a capability slot and wrap it in an Unique\_del\_cap.*

## 16.376.1 Detailed Description

Unique\_cap / Unique\_del\_cap.

Definition in file [unique\\_cap](#).

## 16.377 unique\_cap

[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00025 #include <l4/re/util/cap_alloc>
00026 #include <l4/sys/cxx/smart_capability_lx>
00027
00028 namespace L4Re { namespace Util {
00029
00053 template< typename T >
00054 using Unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>;
00056 template< typename T >
00057 using unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>;
00058
00064 template< typename T >
00065 Unique_cap<T>
00066 make_unique_cap()
00067 { return Unique_cap<T>(cap_alloc.alloc<T>()); }
00068
00096 template< typename T >
00097 using Unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>;
00098 template< typename T >
00100 using unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>;
00101
00107 template< typename T >
00108 Unique_del_cap<T>
00109 make_unique_del_cap()
00110 { return Unique_del_cap<T>(cap_alloc.alloc<T>()); }
00111
00112 }}
00113
```

## 16.378 l4/re/util/bitmap\_cap\_alloc File Reference

Bitmap capability allocator.

```
#include <l4/re/util/item_alloc>
#include <l4/sys/assert.h>
#include <l4/sys/capability>
```



## 16.379 bitmap\_cap\_alloc

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00005  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00006  *      economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU General Public License 2.
00010  * Please see the COPYING-GPL-2 file for details.
00011  *
00012  * As a special exception, you may use this file as part of a free software
00013  * library without restriction. Specifically, if other files instantiate
00014  * templates or use macros or inline functions from this file, or you compile
00015  * this file and link it with other files to produce an executable, this
00016  * file does not by itself cause the resulting executable to be covered by
00017  * the GNU General Public License. This exception does not however
00018  * invalidate any other reasons why the executable file might be covered by
00019  * the GNU General Public License.
00020  */
00021
00022 #pragma once
00023
00024 #include <l4/re/util/item_alloc>
00025 #include <l4/sys/assert.h>
00026 #include <l4/sys/capability>
00027 #include <l4/sys/task.h>
00028
00029 namespace L4Re { namespace Util {
00030
00031 class Cap_alloc_base
00032 {
00033 private:
00034     long _bias;
00035     Item_alloc_base _items;
00036 public:
00037     template <unsigned COUNT>
00038     struct Storage
00039     {
00040         typename Bitmap_base::Word<COUNT>::Type _bits[Bitmap_base::Word<COUNT>::Size];
00041     };
00042
00043     enum State { Free = 0, Allocated, Unknown };
00044     Cap_alloc_base(long max, void *mem, long bias = 0, void * = 0)
00045         noexcept : _bias(bias), _items(max, mem) {}
00046
00047     L4::Cap<void> alloc() noexcept
00048     {
00049         long cap = _items.alloc();
00050         if (cap < 0)
00051             return L4::Cap<void>::Invalid;
00052
00053         return L4::Cap<void>((cap + _bias) << L4_CAP_SHIFT);
00054     }
00055
00056     long hint() const { return _items.hint(); }
00057
00058     template< typename T >
00059     L4::Cap<T> alloc() noexcept
00060     { return L4::Cap<T>(alloc().cap()); }
00061
00062     State is_allocated(L4::Cap<void> c) const noexcept
00063     {
00064         long idx = (c.cap() >> L4_CAP_SHIFT);
00065
00066         if (idx < _bias)
00067             return Unknown;
00068
00069         idx -= _bias;
00070         if (idx >= _items.size())
00071             return Unknown;
00072
00073         return _items.is_allocated(idx) ? Allocated : Free;
00074     }
00075
00076     template< typename T >
00077     void free(L4::Cap<T> const &cap, l4_cap_idx_t task = L4_INVALID_CAP,
00078              l4_umword_t unmap_flags = L4_FP_ALL_SPACES) noexcept
00079     {
00080         long idx = (cap.cap() >> L4_CAP_SHIFT);
00081         if (idx < _bias)

```

```

00097         return;
00098
00099         idx -= _bias;
00100         if (idx >= _items.size())
00101             return;
00102
00103         l4_assert(_items.is_allocated(idx));
00104
00105         if (l4_is_valid_cap(task))
00106             l4_task_unmap(task, cap.fpage(), unmap_flags | 2);
00107
00108         _items.free(idx);
00109     }
00110
00111     // since we have no counters assume counter always > 0
00112     void take(L4::Cap<void>) noexcept {}
00113     bool release(L4::Cap<void>, l4_cap_idx_t /*task*/ = L4_INVALID_CAP,
00114                 unsigned /*unmap_flags*/ = L4_FP_ALL_SPACES) noexcept
00115     { return false; }
00116
00117     long last() noexcept
00118     {
00119         return _items.size() + _bias - 1;
00120     }
00121 };
00122
00123 template< long Size >
00124 class Cap_alloc : public Cap_alloc_base
00125 {
00126 private:
00127     typename Bitmap_base::Word<Size>::Type _bits[Bitmap_base::Word<Size>::Size];
00128
00129 public:
00130     explicit Cap_alloc(long bias = 0) noexcept
00131         : Cap_alloc_base(Size, _bits, bias) {}
00132 };
00133 };
00134
00135 }
00136 }

```

## 16.380 br\_manager

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018
00019 #pragma once
00020
00021 #include <l4/re/util/cap_alloc>
00022 #include <l4/sys/cxx/ipc_server_loop>
00023 #include <l4/cxx/ipc_timeout_queue>
00024 #include <l4/sys/assert.h>
00025
00026 namespace L4Re { namespace Util {
00027
00036 class Br_manager : public L4::Ipc_svr::Server_iface
00037 {
00038 private:
00039     enum { _mem = 0, _ports = 0 };
00040     enum { Brs_per_timeout = sizeof(l4_kernel_clock_t) / sizeof(l4_umword_t) };
00041
00042 public:
00044     Br_manager() : _caps(0), _cap_flags(L4_RCV_ITEM_LOCAL_ID) {}
00045
00046     Br_manager(Br_manager const &) = delete;
00047     Br_manager &operator = (Br_manager const &) = delete;
00048
00049     Br_manager(Br_manager &&) = delete;

```



```

00050   Br_manager &operator = (Br_manager &&) = delete;
00051
00052   ~Br_manager()
00053   {
00054       // Slots for received capabilities are placed at the beginning of the
00055       // (shadowed) buffer registers. Free those.
00056       for (unsigned i = 0; i < _caps; ++i)
00057           cap_alloc.free(L4::Cap<void>(_brs[i] & L4_CAP_MASK));
00058   }
00059
00060   /*
00061    * This implementation dynamically manages assignment of buffer registers for
00062    * the necessary amount of receive buffers allocated by all calls to this
00063    * function.
00064    */
00065   int alloc_buffer_demand(Demand const &d) override
00066   {
00067       using L4::Ipc::Small_buf;
00068
00069       // memory and IO port receive windows currently not supported
00070       if (d.mem || d.ports)
00071           return -L4_EINVAL;
00072
00073       // take extra buffers for a possible timeout and for a zero terminator
00074       if (d.caps + d.mem * 2 + d.ports * 2 + Brs_per_timeout + 1
00075           > L4_UTCB_GENERIC_BUFFERS_SIZE)
00076           return -L4_ERANGE;
00077
00078       if (d.caps > _caps)
00079       {
00080           while (_caps < d.caps)
00081           {
00082               L4::Cap<void> cap = cap_alloc.alloc();
00083               if (!cap)
00084                   return -L4_ENOMEM;
00085
00086               reinterpret_cast<Small_buf&>(_brs[_caps])
00087                   = Small_buf(cap.cap(), _cap_flags);
00088               ++_caps;
00089           }
00090           _brs[_caps] = 0;
00091       }
00092
00093       return L4_EOK;
00094   }
00095
00096
00097   L4::Cap<void> get_rcv_cap(int i) const override
00098   {
00099       if (i < 0 || i >= _caps)
00100           return L4::Cap<void>::Invalid;
00101
00102       return L4::Cap<void>(_brs[i] & L4_CAP_MASK);
00103   }
00104
00105   int realloc_rcv_cap(int i) override
00106   {
00107       using L4::Ipc::Small_buf;
00108
00109       if (i < 0 || i >= _caps)
00110           return -L4_EINVAL;
00111
00112       L4::Cap<void> cap = cap_alloc.alloc();
00113       if (!cap)
00114           return -L4_ENOMEM;
00115
00116       reinterpret_cast<Small_buf&>(_brs[i])
00117           = Small_buf(cap.cap(), _cap_flags);
00118
00119       return L4_EOK;
00120   }
00121
00122
00129   void set_rcv_cap_flags(unsigned long flags)
00130   {
00131       l4_assert(_caps == 0);
00132
00133       _cap_flags = flags;
00134   }
00135
00137   int add_timeout(L4::Ipc_svr::Timeout *, l4_kernel_clock_t) override
00138   { return -L4_ENOSYS; }
00139
00141   int remove_timeout(L4::Ipc_svr::Timeout *) override
00142   { return -L4_ENOSYS; }
00143
00145   void setup_wait(l4_utcb_t *utcb, L4::Ipc_svr::Reply_mode)
00146   {

```

```

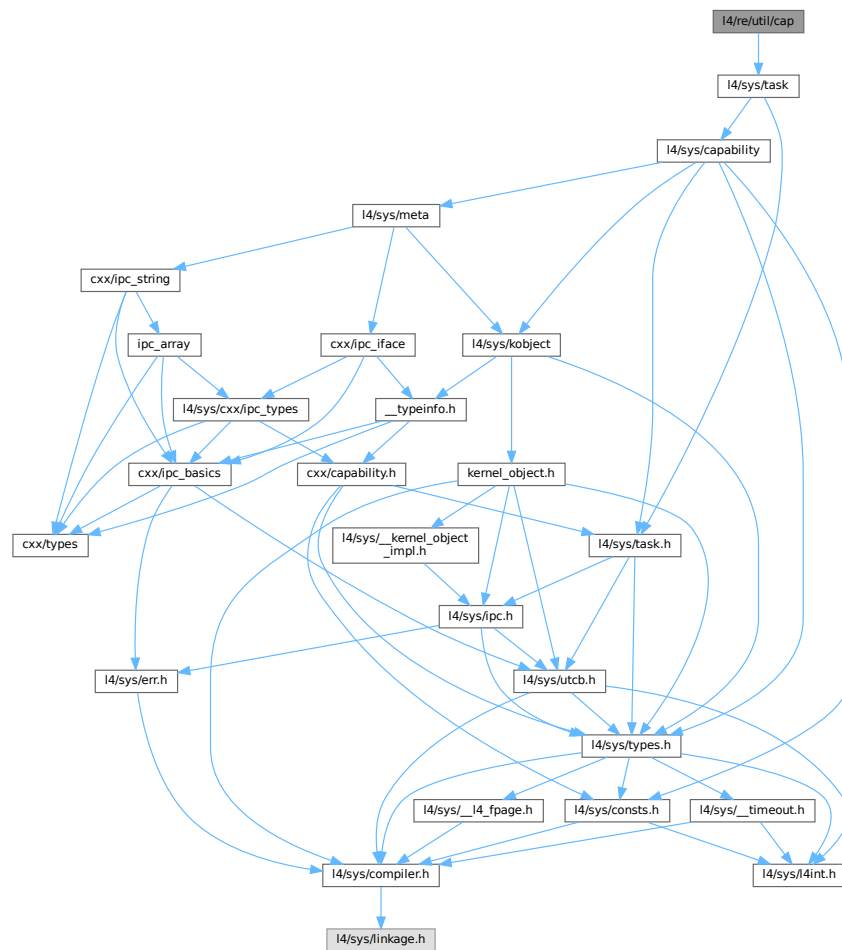
00147     l4_buf_regs_t *br = l4_utcb_br_u(utcb);
00148     br->bdr = 0;
00149     for (unsigned i = 0; i <= _caps; ++i)
00150         br->br[i] = _brs[i];
00151     }
00152
00153 protected:
00154     unsigned first_free_br() const
00155     {
00156         // The last BR (64-bit) or the last two BRs (32-bit); this is constant.
00157         return L4_UTCB_GENERIC_BUFFERS_SIZE - Brs_per_timeout;
00158         // We could also do the following dynamic approach:
00159         // return _caps + _mem + _ports + 1
00160     }
00161
00162 private:
00163     unsigned short _caps;
00164     unsigned long _cap_flags;
00165
00166     l4_umword_t _brs[L4_UTCB_GENERIC_BUFFERS_SIZE];
00167 };
00168
00169 struct Br_manager_hooks
00170 : L4::Ipc_svr::Ignore_errors,
00171   L4::Ipc_svr::Default_timeout,
00172   L4::Ipc_svr::Compound_reply,
00173   Br_manager
00174 {};
00175
00176 struct Br_manager_timeout_hooks :
00177     public L4::Ipc_svr::Timeout_queue_hooks<Br_manager_timeout_hooks, Br_manager>,
00178     public L4::Ipc_svr::Ignore_errors
00179 {
00180 public:
00181     static l4_kernel_clock_t now()
00182     { return l4_kip_clock(l4re_kip()); }
00183 };
00184
00185 {}
00186
00187 {}
00188
00189 {}
00190
00191 {}
00192
00193 {}
00194
00195 {}
00196
00197 {}
00198
00199 {}
00200

```

## 16.381 I4/re/util/cap File Reference

Capability utility functions.

```
#include <l4/sys/task>
Include dependency graph for cap:
```



## Namespaces

- namespace **L4Re**  
*L4Re C++ Interfaces.*
- namespace **L4Re::Util**

Documentation of the [L4 Runtime Environment](#) utility functionality in C++.

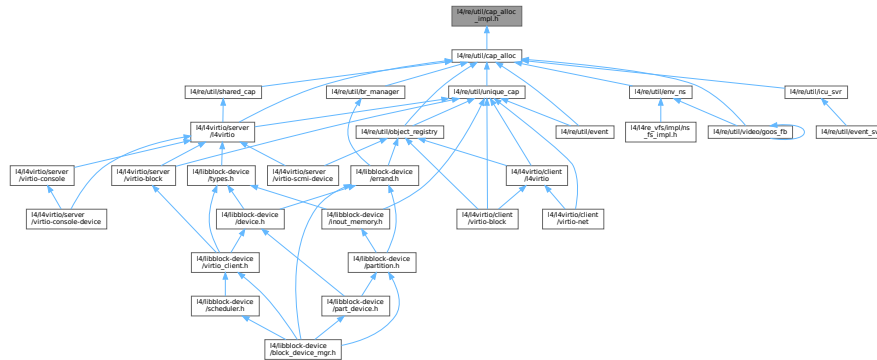
### 16.381.1 Detailed Description

Capability utility functions.

Definition in file [cap.](#)



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*
- namespace [L4Re::Util](#)  
*Documentation of the L4 Runtime Environment utility functionality in C++.*

## 16.383.1 Detailed Description

Capability allocator implementation.

Definition in file [cap\\_alloc\\_impl.h](#).

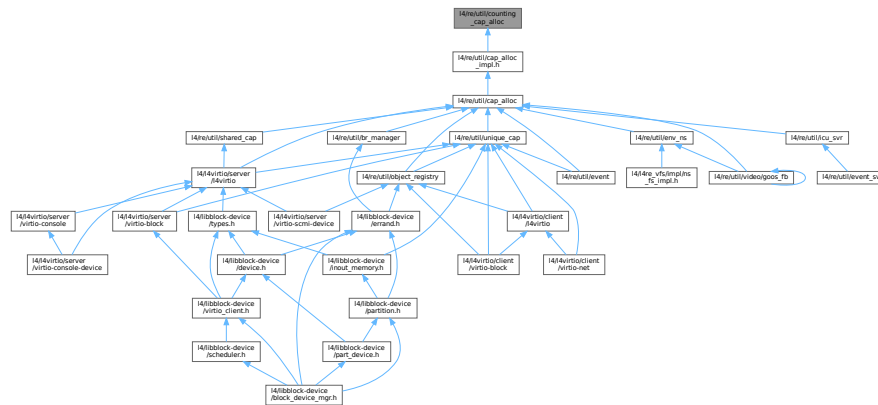
## 16.384 cap\_alloc\_impl.h

[Go to the documentation of this file.](#)

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020 #pragma once
00021 #include <l4/bid_config.h>
00022 #if defined(CONFIG_L4RE_BITMAP_CAP_ALLOC)
00023 #include <l4/re/util/bitmap_cap_alloc>
00024 namespace L4Re { namespace Util {
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `L4Re::Util::Counter< COUNTER >`  
*Counter for `Counting_cap_alloc` with variable data width.*
- struct `L4Re::Util::Counter_atomic< COUNTER >`  
*Thread safe version of counter for `Counting_cap_alloc`.*
- class `L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >`  
*Internal reference-counting cap allocator.*

## Namespaces

- namespace [L4Re](#)  
*[L4Re C++ Interfaces](#).*
- namespace [L4Re::Util](#)  
*Documentation of the [L4 Runtime Environment](#) utility functionality in C++.*

### 16.385.1 Detailed Description

Reference-counting capability allocator.

Definition in file [counting\\_cap\\_alloc](#).

## 16.386 counting\_cap\_alloc

[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
```

```

00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023
00024 #pragma once
00025
00026 #include <l4/sys/task>
00027 #include <l4/sys/assert.h>
00028 #include <l4/re/consts>
00029
00030 namespace L4Re { namespace Util {
00031
00037 template< typename COUNTER = unsigned char >
00038 struct Counter
00039 {
00040     typedef COUNTER Type;
00041     Type _cnt;
00042
00043     static Type nil() { return 0; }
00044     static Type unused() { return 0; }
00045
00046     void free() { _cnt = 0; }
00047     bool is_free() const { return _cnt == 0; }
00048     bool is_saturated() const { return static_cast<Type>(_cnt + 1) == 0; }
00049
00061     bool inc()
00062     {
00063         if (is_saturated())
00064             return true; // no change and no warning
00065         ++_cnt;
00066         if (is_saturated())
00067             return false; // warn caller that counter is now saturated
00068         else
00069             return true; // success
00070     }
00071
00078     Type dec()
00079     {
00080         if (is_saturated())
00081             return _cnt; // no change
00082         else
00083             return --_cnt; // success
00084     }
00085
00086     bool try_alloc()
00087     {
00088         if (_cnt == 0)
00089         {
00090             _cnt = 1;
00091             return true;
00092         }
00093         return false;
00094     }
00095 };
00096
00108 template< typename COUNTER = unsigned char >
00109 struct Counter_atomic
00110 {
00111     typedef COUNTER Type;
00112     Type _cnt;
00113
00114     static Type nil() { return 0; }
00115     static Type unused() { return 1; }
00116
00117     bool is_free() const { return __atomic_load_n(&_cnt, __ATOMIC_RELAXED) == 0; }
00118     static bool is_saturated(Type cnt) { return static_cast<Type>(cnt + 1) == 0; }
00119
00120     bool try_alloc()
00121     {
00122         Type expected = nil();
00123         // Use "acquire" memory ordering. Any operations tied to the capability slot
00124         // must only be observable after the slot has been occupied.
00125         return __atomic_compare_exchange_n(&_cnt, &expected, 2, false,
00126                                           __ATOMIC_ACQUIRE, __ATOMIC_RELAXED);
00127     }
00128
00132     bool inc()
00133     {
00134         Type old_cnt = __atomic_load_n(&_cnt, __ATOMIC_RELAXED);
00135         Type new_cnt;
00136         do
00137         {

```



```

00138         if (is_saturated(old_cnt))
00139             return true; // no change and no warning
00140         new_cnt = old_cnt + 1;
00141     }
00142     while (!__atomic_compare_exchange_n(&cnt, &old_cnt, new_cnt, false,
00143                                         __ATOMIC_RELAXED, __ATOMIC_RELAXED));
00144     if (is_saturated(new_cnt))
00145         return false; // warn caller that counter is now saturated
00146     else
00147         return true; // success
00148 }
00149
00153 Type dec()
00154 {
00155     Type old_cnt = __atomic_load_n(&cnt, __ATOMIC_RELAXED);
00156     Type new_cnt;
00157     do
00158     {
00159         if (is_saturated(old_cnt))
00160             return old_cnt; // no change
00161         new_cnt = old_cnt - 1;
00162     }
00163     while (!__atomic_compare_exchange_n(&cnt, &old_cnt, new_cnt, false,
00164                                         __ATOMIC_RELAXED, __ATOMIC_RELAXED));
00165     return new_cnt; // success
00166 }
00167
00168 void free()
00169 {
00170     // Use "release" memory ordering to make sure that any operations tied to
00171     // the capability slot are observable by other threads before the slot can
00172     // be reused.
00173     __atomic_store_n(&cnt, 0, __ATOMIC_RELEASE);
00174 }
00175 };
00176
00201 template <typename COUNTERTYPE, typename Dbg>
00202 class Counting_cap_alloc
00203 {
00204 private:
00205     void operator = (Counting_cap_alloc const &) { }
00206     typedef COUNTERTYPE Counter;
00207
00208     COUNTERTYPE *_items;
00209     long _free_hint;
00210     long _bias;
00211     long _capacity;
00212     Dbg *_dbg;
00213
00214 public:
00215
00216     template <unsigned COUNT>
00217     struct Storage
00218     {
00219         COUNTERTYPE _buf[COUNT];
00220         typedef COUNTERTYPE Buf_type[COUNT];
00221         enum { Size = COUNT };
00222     };
00223
00224 protected:
00225
00231 Counting_cap_alloc() noexcept
00232 : _items(0), _free_hint(0), _bias(0), _capacity(0)
00233 {}
00234
00235 Counting_cap_alloc(long capacity, void *m, long bias, Dbg *dbg) noexcept
00236 : _items((Counter*)m), _free_hint(0), _bias(bias), _capacity(capacity),
00237   _dbg(dbg)
00238 {}
00239
00254 void setup(void *m, long capacity, long bias, Dbg *dbg) noexcept
00255 {
00256     _items = static_cast<Counter*>(m);
00257     _capacity = capacity;
00258     _bias = bias;
00259     _dbg = dbg;
00260 }
00261
00262 public:
00269 L4::Cap<void> alloc() noexcept
00270 {
00271     long free_hint = __atomic_load_n(&_free_hint, __ATOMIC_RELAXED);
00272
00273     for (long i = free_hint; i < _capacity; ++i)
00274         if (_items[i].try_alloc())
00275         {
00276             _free_hint = i + 1;

```

```

00277         return L4::Cap<void>((i + _bias) « L4_CAP_SHIFT);
00278     }
00279
00280     // _free_hint is not necessarily correct in case of multi-threading! Make
00281     // sure we don't miss any potentially free slots.
00282     for (long i = 0; i < free_hint && i < _capacity; ++i)
00283     {
00284         if (_items[i].try_alloc())
00285         {
00286             _free_hint = i + 1;
00287             return L4::Cap<void>((i + _bias) « L4_CAP_SHIFT);
00288         }
00289     }
00290     return L4::Cap<void>::Invalid;
00291 }
00292
00293 template <typename T>
00294 L4::Cap<T> alloc() noexcept
00295 {
00296     return L4::cap_cast<T>(alloc());
00297 }
00298
00299 void take(L4::Cap<void> cap) noexcept
00300 {
00301     long c;
00302     if (!range_check_and_get_idx(cap, &c))
00303         return;
00304
00305     if (!L4_UNLIKELY(_items[c].inc()))
00306         _dbg->printf("Warning: Reference counter of cap 0x%lx now saturated!\n",
00307                     cap.cap() » L4_CAP_SHIFT);
00308 }
00309
00310 bool free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00311           unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept
00312 {
00313     long c;
00314     if (!range_check_and_get_idx(cap, &c))
00315         return false;
00316
00317     l4_assert(!_items[c].is_free());
00318
00319     if (l4_is_valid_cap(task))
00320         l4_task_unmap(task, cap.fpage(), unmap_flags);
00321
00322     if (c < _free_hint)
00323         _free_hint = c;
00324
00325     _items[c].free();
00326
00327     return true;
00328 }
00329
00330 bool release(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00331             unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept
00332 {
00333     long c;
00334     if (!range_check_and_get_idx(cap, &c))
00335         return false;
00336
00337     l4_assert(!_items[c].is_free());
00338
00339     if (_items[c].dec() == Counter::unused())
00340     {
00341         if (task != L4_INVALID_CAP)
00342             l4_task_unmap(task, cap.fpage(), unmap_flags);
00343
00344         if (c < _free_hint)
00345             _free_hint = c;
00346
00347         // Let others allocate this slot only after the l4_task_unmap() has
00348         // finished.
00349         _items[c].free();
00350
00351         return true;
00352     }
00353     return false;
00354 }
00355
00356 long last() noexcept
00357 {
00358     return _capacity + _bias - 1;
00359 }
00360
00361 private:
00362 bool range_check_and_get_idx(L4::Cap<void> cap, long *c)

```

```

00407 {
00408     *c = cap.cap() » L4_CAP_SHIFT;
00409     if (*c < _bias)
00410         return false;
00411
00412     *c -= _bias;
00413
00414     return *c < _capacity;
00415 }
00416 };
00417
00418 }}

```

## 16.387 dataspace\_svr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00005  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00006  *      economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU General Public License 2.
00010  * Please see the COPYING-GPL-2 file for details.
00011  *
00012  * As a special exception, you may use this file as part of a free software
00013  * library without restriction. Specifically, if other files instantiate
00014  * templates or use macros or inline functions from this file, or you compile
00015  * this file and link it with other files to produce an executable, this
00016  * file does not by itself cause the resulting executable to be covered by
00017  * the GNU General Public License. This exception does not however
00018  * invalidate any other reasons why the executable file might be covered by
00019  * the GNU General Public License.
00020  */
00021 #pragma once
00022
00023 #include <cstring>
00024 #include <cstdint>
00025 #include <l4/bid_config.h>
00026 #include <l4/sys/types.h>
00027 #include <l4/cxx/minmax>
00028 #include <l4/re/dataspace>
00029 #include <l4/re/dataspace-sys.h>
00030 #include <l4/sys/cxx/ipc_legacy>
00031
00032 namespace L4Re { namespace Util {
00033
00040 class Dataspace_svr
00041 {
00042 public:
00043     L4_RPC_LEGACY_DISPATCH(L4Re::Dataspace);
00044
00045     typedef L4::Ipc::Snd_fpage::Map_type Map_type;
00046     typedef L4::Ipc::Snd_fpage::Cacheopt Cache_type;
00047
00048     Dataspace_svr() noexcept
00049     : _ds_start(0), _ds_size(0), _map_flags(L4::Ipc::Snd_fpage::Map),
00050       _cache_flags(L4::Ipc::Snd_fpage::Cached)
00051     {}
00052
00053     virtual ~Dataspace_svr() noexcept {}
00054
00068     int map(Dataspace::Offset offset,
00069            Dataspace::Map_addr local_addr,
00070            Dataspace::Flags flags,
00071            Dataspace::Map_addr min_addr,
00072            Dataspace::Map_addr max_addr,
00073            L4::Ipc::Snd_fpage &memory)
00074     {
00075         memory = L4::Ipc::Snd_fpage();
00076
00077         offset = l4_trunc_page(offset);
00078         local_addr = l4_trunc_page(local_addr);
00079
00080         if (!check_limit(offset))
00081         {
00082             #if 0
00083                 printf("limit failed: off=%lx sz=%lx\n", offset, size());
00084             #endif
00085             return -L4_ERANGE;
00086         }
00087

```

```

00088     min_addr = l4_trunc_page(min_addr);
00089     max_addr = l4_round_page(max_addr);
00090
00091     l4_addr_t addr = _ds_start + offset;
00092     unsigned char order = L4_PAGESHIFT;
00093
00094     while (order < 30 /* limit to 1GB flexpage */)
00095     {
00096         l4_addr_t map_base = l4_trunc_size(addr, order + 1);
00097         if (map_base < _ds_start)
00098             break;
00099
00100         if (map_base + (1UL << (order + 1)) - 1 > (_ds_start + round_size() - 1))
00101             break;
00102
00103         map_base = l4_trunc_size(local_addr, order + 1);
00104         if (map_base < min_addr)
00105             break;
00106
00107         if (map_base + (1UL << (order + 1)) - 1 > max_addr - 1)
00108             break;
00109
00110         l4_addr_t mask = ~(~0UL << (order + 1));
00111         if (local_addr == ~0UL || ((addr ^ local_addr) & mask))
00112             break;
00113
00114         ++order;
00115     }
00116
00117     l4_addr_t map_base = l4_trunc_size(addr, order);
00118
00119     Dataspace::Map_addr b = map_base;
00120     unsigned send_order = order;
00121     int err = map_hook(offset /*map_base - _ds_start*/, order, flags,
00122                        &b, &send_order);
00123     if (err < 0)
00124         return err;
00125
00126     l4_fpage_t fpage = l4_fpage(b, send_order, flags.fpage_rights());
00127
00128     memory = L4::Ipc::Snd_fpage(fpage, local_addr, _map_flags, _cache_flags);
00129
00130     return L4_EOK;
00131 }
00132
00147 virtual int map_hook([[maybe_unused]] Dataspace::Offset offs,
00148                     [[maybe_unused]] unsigned order,
00149                     [[maybe_unused]] Dataspace::Flags flags,
00150                     [[maybe_unused]] Dataspace::Map_addr *base,
00151                     [[maybe_unused]] unsigned *send_order)
00152 {
00153     return 0;
00154 }
00155
00161 virtual void take() noexcept
00162 {}
00163
00171 virtual unsigned long release() noexcept
00172 { return 0; }
00173
00186 virtual long copy([[maybe_unused]] l4_addr_t dst_offs,
00187                  [[maybe_unused]] l4_umword_t src_id,
00188                  [[maybe_unused]] l4_addr_t src_offs,
00189                  [[maybe_unused]] unsigned long size) noexcept
00190 {
00191     return -L4_ENODEV;
00192 }
00193
00203 virtual long clear(unsigned long offs, unsigned long size) const noexcept
00204 {
00205     if (!check_limit(offs))
00206         return -L4_ERANGE;
00207
00208     unsigned long sz = size = cxx::min(size, round_size() - offs);
00209
00210     while (sz)
00211     {
00212         unsigned long b_addr = _ds_start + offs;
00213         unsigned long b_sz = cxx::min(size - offs, sz);
00214
00215         memset(reinterpret_cast<void *>(b_addr), 0, b_sz);
00216
00217         offs += b_sz;
00218         sz -= b_sz;
00219     }
00220
00221     return 0;

```

```

00222     }
00223
00235     virtual long allocate([[maybe_unused]] l4_addr_t offset,
00236                          [[maybe_unused]] l4_size_t size,
00237                          [[maybe_unused]] unsigned access) noexcept
00238     {
00239         return -L4_ENODEV;
00240     }
00241
00247     virtual unsigned long page_shift() const noexcept
00248     { return L4_LOG2_PAGESIZE; }
00249
00255     virtual bool is_static() const noexcept
00256     { return true; }
00257
00270     virtual long map_info([[maybe_unused]] l4_addr_t &start_addr,
00271                          [[maybe_unused]] l4_addr_t &end_addr) noexcept
00272     { return -L4_EPERM; }
00273
00274
00275     long op_map(L4Re::Dataspace::Rights rights,
00276                L4Re::Dataspace::Offset offset,
00277                L4Re::Dataspace::Map_addr spot,
00278                L4Re::Dataspace::Flags flags,
00279                L4::Ipc::Snd_fpage &fp)
00280     {
00281         auto rf = map_flags(rights);
00282
00283         if (!rf.w() && flags.w())
00284             return -L4_EPERM;
00285
00286         return map(offset, spot, flags & rf, 0, ~0, fp);
00287     }
00288
00289     long op_allocate(L4Re::Dataspace::Rights rights,
00290                     L4Re::Dataspace::Offset offset,
00291                     L4Re::Dataspace::Size size)
00292     { return allocate(offset, size, rights & 3); }
00293
00294     long op_copy_in(L4Re::Dataspace::Rights rights,
00295                    L4Re::Dataspace::Offset dst_offs,
00296                    L4::Ipc::Snd_fpage const &src_cap,
00297                    L4Re::Dataspace::Offset src_offs,
00298                    L4Re::Dataspace::Size sz)
00299     {
00300         if (!src_cap.id_received())
00301             return -L4_EINVAL;
00302
00303         if (!(rights & L4_CAP_FPAGE_W))
00304             return -L4_EACCESS;
00305
00306         if (sz == 0)
00307             return L4_EOK;
00308
00309         return copy(dst_offs, src_cap.data(), src_offs, sz);
00310     }
00311
00312     long op_info(L4Re::Dataspace::Rights rights, L4Re::Dataspace::Stats &s)
00313     {
00314         s.size = size();
00315         // only return writable if really writable
00316         s.flags = Dataspace::Flags(0);
00317         if (map_flags(rights).w())
00318             s.flags |= Dataspace::F::W;
00319         return L4_EOK;
00320     }
00321
00322     long op_clear(L4Re::Dataspace::Rights rights,
00323                  L4Re::Dataspace::Offset offset,
00324                  L4Re::Dataspace::Size size)
00325     {
00326         if (!map_flags(rights).w())
00327             return -L4_EACCESS;
00328
00329         return clear(offset, size);
00330     }
00331
00332     long op_map_info(L4Re::Dataspace::Rights,
00333                     [[maybe_unused]] l4_addr_t &start_addr,
00334                     [[maybe_unused]] l4_addr_t &end_addr)
00335     {
00336 #ifdef CONFIG_MMU
00337         return 0;
00338 #else
00339         return map_info(start_addr, end_addr);
00340 #endif
00341     }

```

```

00342
00343 protected:
00344     unsigned long size() const noexcept
00345     { return _ds_size; }
00346     unsigned long map_flags() const noexcept
00347     { return _map_flags; }
00348     unsigned long page_size() const noexcept
00349     { return 1UL < page_shift(); }
00350     unsigned long round_size() const noexcept
00351     { return l4_round_size(size(), page_shift()); }
00352     bool check_limit(l4_addr_t offset) const noexcept
00353     { return offset < round_size(); }
00354
00355     L4Re::Dataspace::Flags
00356     map_flags(L4Re::Dataspace::Rights rights = L4_CAP_FPAGE_W) const noexcept
00357     {
00358         auto f = (_rw_flags & L4Re::Dataspace::Flags(0x0f)) | L4Re::Dataspace::F::Caching_mask;
00359         if (!(rights & L4_CAP_FPAGE_W))
00360             f &= ~L4Re::Dataspace::F::W;
00361
00362         return f;
00363     }
00364
00365 protected:
00366     void size(unsigned long size) noexcept { _ds_size = size; }
00367
00368     l4_addr_t _ds_start;
00369     l4_size_t _ds_size;
00370     Map_type _map_flags;
00371     Cache_type _cache_flags;
00372     L4Re::Dataspace::Flags _rw_flags;
00373 };
00374
00375 }}

```

## 16.388 env\_ns

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002
00003 #pragma once
00004
00005 #include <l4/re/cap_alloc>
00006 #include <l4/re/util/cap_alloc>
00007 #include <l4/re/namespace>
00008 #include <l4/re/env>
00009 #include <cstring>
00010
00011 namespace L4Re { namespace Util {
00012
00013     class Env_ns
00014     {
00015     private:
00016         L4Re::Cap_alloc *_ca;
00017         Env const *_env;
00018
00019     public:
00020         explicit Env_ns(Env const *env = Env::env(),
00021                        L4Re::Cap_alloc *ca = L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc))
00022             : _ca(ca), _env(env) {}
00023
00024         L4::Cap<void>
00025         query(char const *name, unsigned len, int timeout = Namespace::To_default,
00026              l4_umword_t *local_id = 0, bool iterate = true) const noexcept
00027         {
00028             typedef Env::Cap_entry Cap_entry;
00029
00030             // Skip possible first slash
00031             if (len && name[0] == '/')
00032             {
00033                 ++name;
00034                 --len;
00035             }
00036
00037             char const *n = name;
00038             for (; len && *n != '/'; ++n, --len) // Count first path element
00039                 ;
00040
00041             Cap_entry const *e = _env->get(name, n - name);
00042             if (!e)
00043                 return L4::Cap<void>(-L4_ENOENT);
00044
00045             if (len > 0 && *n == '/')
00046             {

```

```

00047     L4::Cap<L4Re::Namespace> ns(e->cap);
00048     L4::Cap<void> cap = _ca->alloc<void>();
00049
00050     if (!cap.is_valid())
00051         return L4::Cap<void>(-L4_ENOMEM);
00052
00053     long r = ns->query(n + 1, len - 1, cap, timeout, local_id, iterate);
00054     if (r >= 0)
00055         return cap;
00056
00057     _ca->free(cap);
00058
00059     return L4::Cap<void>(r);
00060 }
00061
00062     return L4::Cap<void>(e->cap);
00063 }
00064
00065 L4::Cap<void>
00066 query(char const *name, int timeout = Namespace::To_default,
00067       l4_umword_t *local_id = 0, bool iterate = true) const noexcept
00068 { return query(name, __builtin_strlen(name), timeout, local_id, iterate); }
00069
00070 template<typename T >
00071 L4::Cap<T>
00072 query(char const *name, int timeout = Namespace::To_default,
00073       l4_umword_t *local_id = 0, bool iterate = true) const noexcept
00074 {
00075     return L4::cap_cast<T>(query(name, __builtin_strlen(name),
00076                                timeout, local_id, iterate));
00077 }
00078 };
00079
00080 }}

```

## 16.389 event\_buffer

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020
00021 #pragma once
00022
00023 #include <l4/re/event>
00024 #include <l4/re/event-sys.h>
00025 #include <l4/re/rm>
00026
00027 #include <cstring>
00028
00029 namespace L4Re { namespace Util {
00030
00031     template< typename PAYLOAD >
00032     class Event_buffer_t : public L4Re::Event_buffer_t<PAYLOAD>
00033     {
00034     private:
00035         void *_buf;
00036     public:
00037         void *_buf() const noexcept { return _buf; }
00038
00039         long attach(L4::Cap<L4Re::Dataspac> ds, L4::Cap<L4Re::Rm> rm) noexcept
00040         {
00041             l4_addr_t sz = ds->size();
00042             _buf = 0;
00043
00044             long r = rm->attach(&_buf, sz,
00045                               L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00046                               L4::Ipc::make_cap_rw(ds));
00047         }
00048     };
00049 }
00050 }

```

```

00064     if (r < 0)
00065         return r;
00066
00067     *static_cast<L4Re::Event_buffer_t<PAYLOAD>*>(this)
00068     = L4Re::Event_buffer_t<PAYLOAD>(_buf, sz);
00069     return 0;
00070 }
00071
00079 long detach(L4::Cap<L4Re::Rm> rm) noexcept
00080 {
00081     L4::Cap<L4Re::Dataspace> ds;
00082     if (_buf)
00083         return rm->detach(_buf, &ds);
00084     return 0;
00085 }
00086
00087 };
00088
00093 template< typename PAYLOAD >
00094 class Event_buffer_consumer_t : public Event_buffer_t<PAYLOAD>
00095 {
00096 public:
00097
00104     template< typename CB, typename D >
00105     void foreach_available_event(CB const &cb, D data = D())
00106     {
00107         typename Event_buffer_t<PAYLOAD>::Event *e;
00108         while ((e = Event_buffer_t<PAYLOAD>::next()))
00109             {
00110                 cb(e, data);
00111                 e->free();
00112             }
00113     }
00114
00125     template< typename CB, typename D >
00126     void process(L4::Cap<L4::Irq> irq,
00127                 L4::Cap<L4::Thread> thread,
00128                 CB const &cb, D data = D())
00129     {
00130
00131         if (l4_error(irq->bind_thread(thread, 0)))
00132             return;
00133
00134         while (1)
00135             {
00136                 long r;
00137                 r = l4_ipc_error(l4_irq_receive(irq.cap(), L4_IPC_NEVER),
00138                                 l4_utcb());
00139                 if (r)
00140                     continue;
00141
00142                 foreach_available_event(cb, data);
00143             }
00144     }
00145 };
00146
00147 typedef Event_buffer_t<Default_event_payload> Event_buffer;
00148 typedef Event_buffer_consumer_t<Default_event_payload> Event_buffer_consumer;
00149
00150 }

```

## 16.390 event\_svr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020

```



```

00021 #pragma once
00022
00023 #include <l4/re/event_enums.h>
00024 #include <l4/re/event>
00025 #include <l4/re/event-sys.h>
00026 #include <l4/re/util/icu_svr>
00027 #include <l4/cxx/minmax>
00028
00029 #include <l4/sys/cxx/ipc_legacy>
00030
00031 namespace L4Re { namespace Util {
00032
00033 template< typename SVR >
00034 class Event_svr : public Icu_cap_array_svr<SVR>
00035 {
00036 private:
00037     typedef Icu_cap_array_svr<SVR> Icu_svr;
00038
00039 protected:
00040     L4::Cap<L4Re::Dataspace> _ds;
00041     typename Icu_svr::Irq_irq;
00042
00043 public:
00044     Event_svr() : Icu_svr(1, &_irq) {}
00045
00046     L4_RPC_LEGACY_DISPATCH(L4Re::Event);
00047     L4_RPC_LEGACY_USING(Icu_svr);
00048
00049     long op_get_buffer(L4Re::Event::Rights, L4::Ipc::Cap<L4Re::Dataspace> &ds)
00050     {
00051         static_cast<SVR*>(this)->reset_event_buffer();
00052         ds = L4::Ipc::Cap<L4Re::Dataspace>(_ds, L4_CAP_FPAGE_RW);
00053         return 0;
00054     }
00055
00056     long op_get_num_streams(L4Re::Event::Rights)
00057     { return static_cast<SVR*>(this)->get_num_streams(); }
00058
00059     long op_get_stream_info(L4Re::Event::Rights, int idx, Event_stream_info &info)
00060     { return static_cast<SVR*>(this)->get_stream_info(idx, &info); }
00061
00062     long op_get_stream_info_for_id(L4Re::Event::Rights, l4_umword_t id,
00063                                     Event_stream_info &info)
00064     { return static_cast<SVR*>(this)->get_stream_info_for_id(id, &info); }
00065
00066     long op_get_axis_info(L4Re::Event::Rights, l4_umword_t id,
00067                           L4::Ipc::Array_in_buf<unsigned, unsigned long> const &axes,
00068                           L4::Ipc::Array_ref<Event_absinfo, unsigned long> &info)
00069     {
00070         unsigned naxes = cxx::min<unsigned>(L4RE_ABS_MAX, axes.length);
00071
00072         info.length = 0;
00073
00074         Event_absinfo _info[L4RE_ABS_MAX];
00075         int r = static_cast<SVR*>(this)->get_axis_info(id, naxes, axes.data, _info);
00076         if (r < 0)
00077             return r;
00078
00079         for (unsigned i = 0; i < naxes; ++i)
00080             info.data[i] = _info[i];
00081
00082         info.length = naxes;
00083         return r;
00084     }
00085
00086     long op_get_stream_state_for_id(L4Re::Event::Rights, l4_umword_t stream_id,
00087                                     Event_stream_state &state)
00088     { return static_cast<SVR*>(this)->get_stream_state_for_id(stream_id, &state); }
00089
00090     int get_num_streams() const { return 0; }
00091     int get_stream_info(int, L4Re::Event_stream_info *)
00092     { return -L4_EINVAL; }
00093     int get_stream_info_for_id(l4_umword_t, L4Re::Event_stream_info *)
00094     { return -L4_EINVAL; }
00095     int get_axis_info(l4_umword_t, unsigned /*naxes*/, unsigned const * /*axes*/,
00096                       L4Re::Event_absinfo *)
00097     { return -L4_EINVAL; }
00098     int get_stream_state_for_id(l4_umword_t, L4Re::Event_stream_state *)
00099     { return -L4_EINVAL; }
00100 };
00101
00102 }
```

## 16.391 icu\_svr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2009-2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018
00019 #pragma once
00020
00021
00022 #include <l4/sys/types.h>
00023
00024 #include <l4/sys/icu>
00025 #include <l4/sys/task>
00026 #include <l4/re/env>
00027 #include <l4/re/util/cap_alloc>
00028 #include <l4/sys/cxx/ipc_legacy>
00029
00030 namespace L4Re { namespace Util {
00031
00032 template< typename ICU >
00033 class Icu_svr
00034 {
00035 private:
00036     ICU const *this_icu() const { return static_cast<ICU const *>(this); }
00037     ICU *this_icu() { return static_cast<ICU*>(this); }
00038
00039 public:
00040     L4_RPC_LEGACY_DISPATCH(L4::Icu);
00041
00042     int op_bind(L4::Icu::Rights, l4_umword_t irqnum,
00043                L4::Ipc::Snd_fpage irq_fp);
00044     int op_unbind(L4::Icu::Rights, l4_umword_t irqnum,
00045                  L4::Ipc::Snd_fpage irq_fp);
00046     int op_info(L4::Icu::Rights, L4::Icu::_Info &info);
00047     int op_msi_info(L4::Icu::Rights, l4_umword_t irqnum,
00048                     l4_uint64_t source, l4_icu_msi_info_t &info);
00049     int op_mask(L4::Icu::Rights, l4_umword_t irqnum);
00050     int op_unmask(L4::Icu::Rights, l4_umword_t irqnum);
00051     int op_set_mode(L4::Icu::Rights, l4_umword_t, l4_umword_t)
00052     { return 0; }
00053 };
00054
00055 template<typename ICU> inline
00056 int
00057 Icu_svr<ICU>::op_bind(L4::Icu::Rights, l4_umword_t irqnum,
00058                       L4::Ipc::Snd_fpage irq_fp)
00059 {
00060     typename ICU::Irq *irq = this_icu()->icu_get_irq(irqnum);
00061     if (!irq)
00062         return -L4_EINVAL;
00063     return irq->bind(this_icu(), irq_fp);
00064 }
00065
00066 template<typename ICU> inline
00067 int
00068 Icu_svr<ICU>::op_unbind(L4::Icu::Rights, l4_umword_t irqnum,
00069                          L4::Ipc::Snd_fpage irq_fp)
00070 {
00071     typename ICU::Irq *irq = this_icu()->icu_get_irq(irqnum);
00072     if (!irq)
00073         return -L4_EINVAL;
00074     return irq->unbind(this_icu(), irq_fp);
00075 }
00076
00077 template<typename ICU> inline
00078 int
00079 Icu_svr<ICU>::op_info(L4::Icu::Rights, L4::Icu::_Info &info)
00080 {
00081     l4_icu_info_t i;
00082     this_icu()->icu_get_info(&i);
00083     info.features = i.features;
00084 }

```

```

00086     info.nr_irqs = i.nr_irqs;
00087     info.nr_msis = i.nr_msis;
00088     return 0;
00089 }
00090
00091 template<typename ICU> inline
00092 int
00093 Icu_svr<ICU>::op_msi_info(L4::Icu::Rights, l4_umword_t irqnum,
00094                          l4_uint64_t source, l4_icu_msi_info_t &info)
00095 {
00096     typename ICU::Irq *irq = this_icu()->icu_get_irq(irqnum);
00097     if (!irq)
00098         return -L4_EINVAL;
00099     return irq->msi_info(source, &info);
00100 }
00101
00102 template<typename ICU> inline
00103 int
00104 Icu_svr<ICU>::op_mask(L4::Icu::Rights, l4_umword_t irqnum)
00105 {
00106     typename ICU::Irq *irq = this_icu()->icu_get_irq(irqnum);
00107     if (irq)
00108         irq->mask(true);
00109     return -L4_ENOREPLY;
00110 }
00111
00112 template<typename ICU> inline
00113 int
00114 Icu_svr<ICU>::op_unmask(L4::Icu::Rights, l4_umword_t irqnum)
00115 {
00116     typename ICU::Irq *irq = this_icu()->icu_get_irq(irqnum);
00117     if (irq)
00118         irq->mask(false);
00119     return -L4_ENOREPLY;
00120 }
00121
00122
00123 template< typename ICU >
00124 class Icu_cap_array_svr : public Icu_svr<ICU>
00125 {
00126 protected:
00127     static void free_irq_cap(L4::Cap<L4::Irq> &cap)
00128     {
00129         if (cap)
00130         {
00131             L4Re::Util::cap_alloc.free(cap);
00132             cap.invalidate();
00133         }
00134     }
00135
00136 public:
00137     class Irq
00138     {
00139     public:
00140         Irq() {}
00141         ~Irq() { ICU::free_irq_cap(_cap); }
00142
00143         void trigger() const
00144         {
00145             if (_cap)
00146                 _cap->trigger();
00147         }
00148
00149         int bind(ICU *, L4::Ipc::Snd_fpage const &irq_fp);
00150         int unbind(ICU *, L4::Ipc::Snd_fpage const &irq_fp);
00151         void mask(bool /**mask*/ ) const
00152         { }
00153
00154         int msi_info(l4_uint64_t, l4_icu_msi_info_t *) const
00155         { return -L4_EINVAL; }
00156
00157         L4::Cap<L4::Irq> cap() const { return _cap; }
00158
00159     private:
00160         L4::Cap<L4::Irq> _cap;
00161     };
00162
00163 private:
00164     Irq *_irqs;
00165     unsigned _nr_irqs;
00166
00167 public:
00168
00169     Icu_cap_array_svr(unsigned nr_irqs, Irq *irqs)
00170     : _irqs(irqs), _nr_irqs(nr_irqs)
00171     {}
00172

```

```

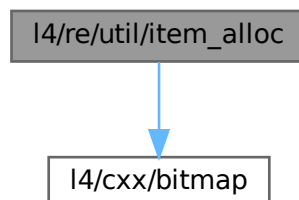
00173 Irq *icu_get_irq(l4_umword_t irqnum)
00174 {
00175     if (irqnum >= _nr_irqs)
00176         return 0;
00177     return _irqs + irqnum;
00178 }
00179
00180 void icu_get_info(l4_icu_info_t *inf)
00181 {
00182     inf->features = 0;
00183     inf->nr_irqs = _nr_irqs;
00184     inf->nr_msis = 0;
00185 }
00186 };
00187
00188 template< typename ICU >
00189 int
00190 Icu_cap_array_svr<ICU>::Irq::bind(ICU *cfb, L4::Ipc::Snd_fpage const &irq_fp)
00191 {
00192     if (!irq_fp.cap_received())
00193         return -L4_EINVAL;
00194     L4::Cap<L4::Irq> irq = cfb->server_iface()->template_rcv_cap<L4::Irq>(0);
00195     if (!irq)
00196         return -L4_EINVAL;
00197     int r = cfb->server_iface()->realloc_rcv_cap(0);
00198     if (r < 0)
00199         return r;
00200     ICU::free_irq_cap(_cap);
00201     _cap = irq;
00202     return 0;
00203 }
00204
00205 template< typename ICU >
00206 int
00207 Icu_cap_array_svr<ICU>::Irq::unbind(ICU *, L4::Ipc::Snd_fpage const &/*irq_fp*/)
00208 {
00209     ICU::free_irq_cap(_cap);
00210     _cap = L4::Cap<L4::Irq>::Invalid;
00211     return 0;
00212 }
00213
00214 {}
00215
00216 }
00217
00218 }
00219 }

```

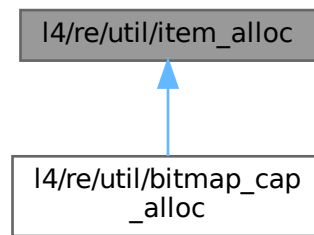
## 16.392 I4/re/util/item\_alloc File Reference

Item allocator.

```
#include <l4/cxx/bitmap>
Include dependency graph for item_alloc:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4Re::Util::Item\\_alloc\\_base](#)  
*Item allocator.*

## Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*
- namespace [L4Re::Util](#)  
*Documentation of the [L4 Runtime Environment](#) utility functionality in C++.*

## 16.392.1 Detailed Description

Item allocator.

Definition in file [item\\_alloc](#).

## 16.393 item\_alloc

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
  
```

```

00023  * the GNU General Public License.
00024  */
00025
00026 #pragma once
00027
00028 #include <l4/cxx/bitmap>
00029
00030 namespace L4Re { namespace Util {
00031
00032 using cxx::Bitmap_base;
00033 using cxx::Bitmap;
00034
00038 class Item_alloc_base
00039 {
00040 private:
00041     long _capacity;
00042     long _free_hint;
00043     Bitmap_base _bits;
00044
00045     void hint(long hint)
00046     { __atomic_store_n(&_free_hint, hint, __ATOMIC_RELAXED); }
00047
00048 public:
00049     bool is_allocated(long item) const noexcept
00050     { return _bits[item]; }
00051
00052     long hint() const { return __atomic_load_n(&_free_hint, __ATOMIC_RELAXED); }
00053
00054     bool alloc(long item) noexcept
00055     {
00056         return !_bits.atomic_get_and_set(item);
00057     }
00058
00059     void free(long item) noexcept
00060     {
00061         if (item < hint())
00062             hint(item);
00063
00064         _bits.atomic_clear_bit(item);
00065     }
00066
00067     Item_alloc_base(long size, void *mem) noexcept
00068         : _capacity(size), _free_hint(0), _bits(mem)
00069     {}
00070
00071     long alloc() noexcept
00072     {
00073         long free_hint = hint();
00074
00075         for (long i = free_hint; i < _capacity; ++i)
00076             if (alloc(i))
00077             {
00078                 hint(i + 1);
00079                 return i;
00080             }
00081
00082         // _free_hint is not necessarily correct in case of multi-threading! Make
00083         // sure we don't miss any potentially free slots.
00084         for (long i = 0; i < free_hint && i < _capacity; ++i)
00085             if (alloc(i))
00086             {
00087                 hint(i + 1);
00088                 return i;
00089             }
00090
00091         return -1;
00092     }
00093
00094     long size() const noexcept
00095     {
00096         return _capacity;
00097     }
00098 };
00099
00100 template< long Bits >
00101 class Item_alloc : public Item_alloc_base
00102 {
00103 private:
00104     typename Bitmap_base::Word<Bits>::Type _bits[Bitmap_base::Word<Bits>::Size];
00105
00106 public:
00107     Item_alloc() noexcept : Item_alloc_base(Bits, _bits) {}
00108 };
00109
00110 {}

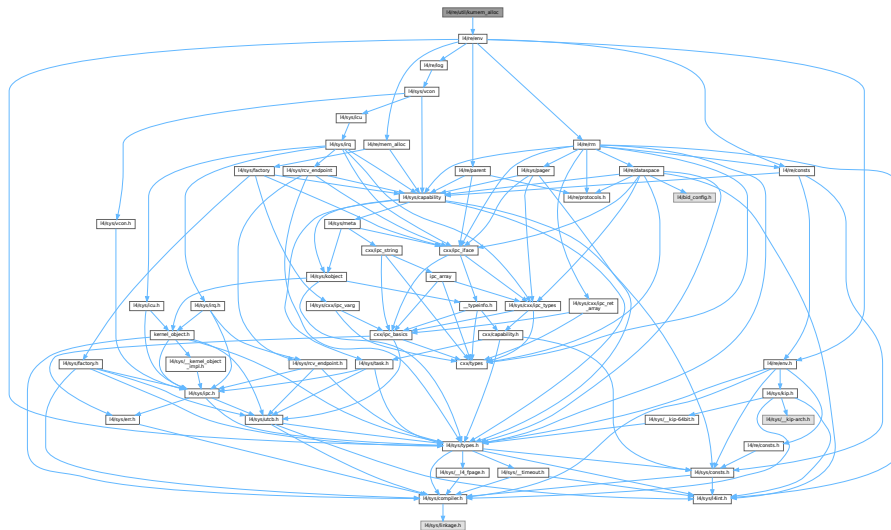
```

## 16.394 l4/re/util/kumem\_alloc File Reference

Kumem allocator helper.

```
#include <l4/re/env>
```

Include dependency graph for kumem\_alloc:



### Namespaces

- namespace [L4Re](#)  
*[L4Re](#) C++ Interfaces.*
- namespace [L4Re::Util](#)  
*Documentation of the [L4](#) Runtime Environment utility functionality in C++.*

### Functions

- int [L4Re::Util::kumem\\_alloc](#) (l4\_addr\_t \*mem, unsigned pages\_order, [L4::Cap](#)< [L4::Task](#) > task=[L4Re::Env::env](#)() ->task(), [L4::Cap](#)< [L4Re::Rm](#) > rm=[L4Re::Env::env](#)() ->rm()) noexcept  
*Allocate state area.*

### 16.394.1 Detailed Description

Kumem allocator helper.

Definition in file [kumem\\_alloc](#).

## 16.395 kumem\_alloc

[Go to the documentation of this file.](#)

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025
00026 #pragma once
00027
00028 #include <l4/re/env>
00029
00030 namespace L4Re { namespace Util {
00031
00055 int
00056 kumem_alloc(l4_addr_t *mem, unsigned pages_order,
00057             L4::Cap<L4::Task> task = L4Re::Env::env()->task(),
00058             L4::Cap<L4Re::Rm> rm = L4Re::Env::env()->rm()) noexcept;
00059
00061 }}
```

## 16.396 meta

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002
00003 #pragma once
00004
00005 #include <l4/sys/meta>
00006 #include <l4/sys/typeinfo_svr>
00007
00008 namespace L4Re { namespace Util {
00009 using L4::Util::handle_meta_request;
00010 }}
```

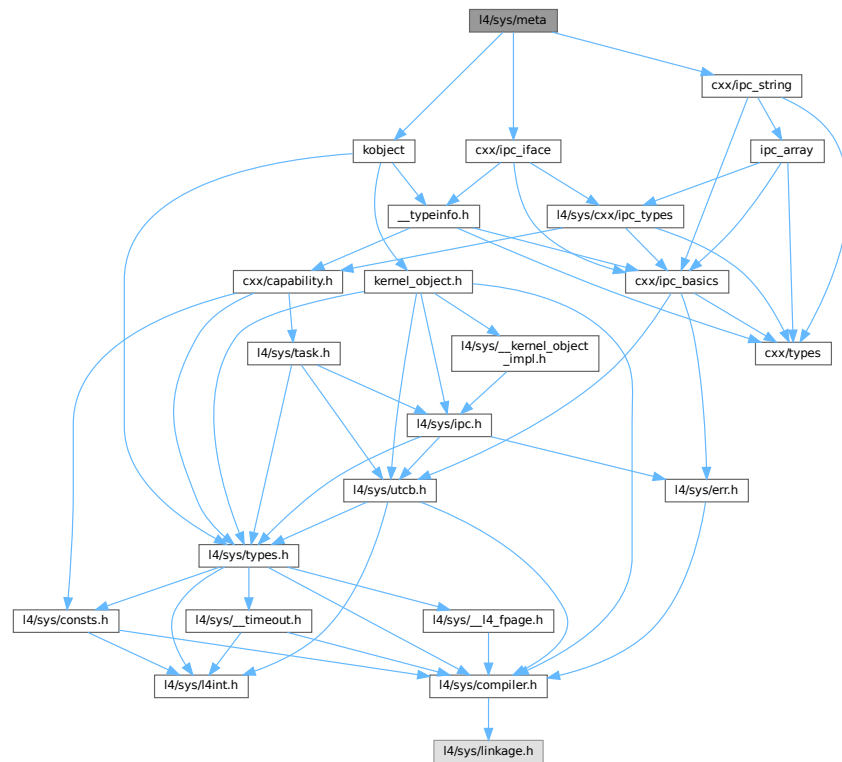
## 16.397 l4/sys/meta File Reference

Meta interface for getting dynamic type information about objects behind capabilities.

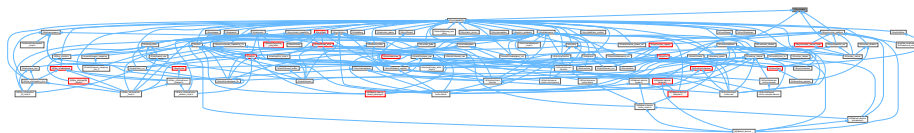
```
#include "kobject"
#include "cxx/ipc_iface"
#include "cxx/ipc_string"
```



Include dependency graph for meta:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::Meta](#)

*Meta* interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.

## Namespaces

- namespace [L4](#)

*L4* low-level kernel interface.

## 16.397.1 Detailed Description

Meta interface for getting dynamic type information about objects behind capabilities.

Definition in file [meta](#).

## 16.398 meta

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019 #pragma once
00020 #include "kobject"
00021 #include "cxx/ipc_iface"
00022 #include "cxx/ipc_string"
00023 namespace L4 {
00024     class Meta : public Kobject_t<Meta, Kobject, L4_PROTO_META>
00025     {
00026     public:
00027         L4_INLINE_RPC(l4_msgtag_t, num_interfaces, ());
00028         L4_INLINE_RPC(l4_msgtag_t, interface, (l4_umword_t idx, long *proto,
00029             L4::Ipc::String<char> *name));
00030         L4_INLINE_RPC(l4_msgtag_t, supports, (l4_mword_t protocol));
00031         typedef L4::Typeid::Rpc<num_interfaces_t, interface_t, supports_t> Rpc;
00032     };
00033 }
```

## 16.399 name\_space\_svr

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020 #pragma once
00021 #include <l4/cxx/avl_tree>
00022 #include <l4/cxx/std_ops>
00023 #include <l4/sys/cxx/ipc_epiface>
00024 #include <l4/cxx/string>
00025 #include <l4/re/util/debug>
00026 #include <l4/sys/capability>
00027 #include <l4/re/namespace>
00028 namespace L4Re { namespace Util { namespace Names {
```

```

00035
00039 class Name : public cxx::String
00040 {
00041 public:
00042
00043     Name(const char *name = "") : String(name, __builtin_strlen(name)) {}
00044     Name(const char *name, unsigned long len) : String(name, len) {}
00045     Name(cxx::String const &n) : String(n) {}
00046     char const *name() const { return start(); }
00047     bool operator < (Name const &r) const
00048     {
00049         unsigned long l = cxx::min(len(), r.len());
00050         int v = memcmp(start(), r.start(), l);
00051         return v < 0 || (v == 0 && len() < r.len());
00052     }
00053 };
00054
00055
00059 class Obj
00060 {
00061 protected:
00062     unsigned _f;
00063     union
00064     {
00065         l4_cap_idx_t _cap;
00066         L4::Epiface *_obj;
00067     };
00068
00069 public:
00070     enum Flags
00071     {
00072         F_rw          = L4Re::Namespace::Rw,
00073         F_strong       = L4Re::Namespace::Strong,
00074
00075         F_trusted      = L4Re::Namespace::Trusted,
00076
00077         F_rights_mask = F_rw | F_strong | F_trusted,
00078
00079         F_cap          = 0x100,
00080         F_local        = 0x200,
00081         F_replacable   = 0x400,
00082         F_base_mask    = 0xf00,
00083     };
00084 };
00085
00086
00087 unsigned flags() const { return _f; }
00088 void restrict_flags(unsigned max_rights)
00089 { _f &= (~F_rights_mask | (max_rights & F_rights_mask)); }
00090
00091 bool is_rw() const { return (_f & F_rw) == F_rw; }
00092 bool is_strong() const { return _f & F_strong; }
00093
00094 bool is_valid() const { return _f & F_cap; }
00095 bool is_complete() const { return is_valid(); }
00096 bool is_local() const { return _f & F_local; }
00097 bool is_replacable() const { return _f & F_replacable; }
00098 bool is_trusted() const { return _f & F_trusted; }
00099
00100 L4::Epiface *obj() const { if (is_local()) return _obj; return 0; }
00101 L4::Cap<void> cap() const
00102 {
00103     if (!is_local())
00104         return L4::Cap<void>(_cap);
00105     if (!_obj)
00106         return L4::Cap<void>::Invalid;
00107     return _obj->obj_cap();
00108 }
00109
00110
00111 void set(Obj const &o, unsigned flags)
00112 {
00113     *this = o;
00114     restrict_flags(flags);
00115 }
00116
00117 explicit Obj(unsigned flags = 0)
00118 : _f(flags), _cap(L4_INVALID_CAP)
00119 {}
00120
00121 Obj(unsigned f, L4::Cap<void> const &cap)
00122 : _f((f & ~F_base_mask) | F_cap), _cap(cap.cap())
00123 {}
00124
00125 Obj(unsigned f, L4::Epiface *o)
00126 : _f((f & ~F_base_mask) | F_cap | F_local), _obj(o)
00127 {}

```

```

00128
00129 void reset(unsigned flags)
00130 {
00131     _f = (_f & F_replacable) | (flags & ~(F_cap | F_local));
00132     _cap = L4_INVALIDID_CAP;
00133 }
00134
00135
00136 };
00137
00138
00142 class Entry : public cxx::Avl_tree_node
00143 {
00144 private:
00145     friend class Name_space;
00146     Name _n;
00147     Obj _o;
00148
00149     bool _dynamic;
00150
00151 public:
00152     Entry(Name const &n, Obj const &o, bool dynamic = false)
00153         : _n(n), _o(o), _dynamic(dynamic) {}
00154
00155     Name const &name() const { return _n; }
00156     Obj const *obj() const { return &_amp;o; }
00157     Obj *obj() { return &_amp;o; }
00158     void obj(Obj const &o) { _o = o; }
00159
00160     bool is_placeholder() const
00161     { return !obj()->is_complete(); }
00162
00163     bool is_dynamic() const { return _dynamic; }
00164
00165     void set(Obj const &o)
00166     {
00167         obj()->set(o, obj()->flags());
00168     }
00169
00170 private:
00171     void * operator new (size_t s);
00172     void operator delete(void *b);
00173
00174 };
00175
00176 struct Names_get_key
00177 {
00178     typedef Name Key_type;
00179     static Key_type const &key_of(Entry const *e)
00180     { return e->name(); }
00181 };
00182
00183
00191 class Name_space
00192 {
00193     friend class Entry;
00194
00195 private:
00196     typedef cxx::Avl_tree<Entry, Names_get_key> Tree;
00197     Tree _tree;
00198
00199 protected:
00200     L4Re::Util::Dbg const &_dbg;
00201     L4Re::Util::Err const &_err;
00202
00203 public:
00204
00205     typedef Tree::Const_iterator Const_iterator;
00206
00207     Const_iterator begin() const { return _tree.begin(); }
00208     Const_iterator end() const { return _tree.end(); }
00209
00210     Name_space(L4Re::Util::Dbg const &dbg, L4Re::Util::Err const &err)
00211         : _dbg(dbg), _err(err)
00212     {}
00213
00217     virtual ~Name_space() {}
00218
00219     Entry *find(Name const &name) const { return _tree.find_node(name); }
00220     Entry *remove(Name const &name) { return _tree.remove(name); }
00221     Entry *find_iter(Name const &pname) const
00222     {
00223         Name name = pname;
00224         _dbg.printf("resolve '%.*s': ", name.len(), name.start());
00225         Name_space const *ns = this;
00226         while (ns)
00227         {

```

```

00228         cxx::String::Index sep = name.find("/");
00229         cxx::String part;
00230         if (!name.eof(sep))
00231             part = name.head(sep);
00232         else
00233             part = name;
00234
00235         _dbg.cprintf(" '%.s'", part.len(), part.start());
00236         Entry *o = ns->find(Name(part.start(), part.len()));
00237
00238         if (!o)
00239         {
00240             _dbg.cprintf(": resolution failed: '%.s' remaining\n",
00241                 name.len(), name.start());
00242             return 0;
00243         }
00244
00245         auto const *obj = o->obj()->obj();
00246         ns = dynamic_cast<Name_space const *>(obj);
00247         if (ns)
00248         {
00249             if (!name.eof(sep))
00250             {
00251                 name = name.substr(sep + 1);
00252                 continue;
00253             }
00254         }
00255
00256         _dbg.cprintf(": found object: %p (%s)\n",
00257             obj, obj ? typeid(*obj).name() : "");
00258
00259         return o;
00260     }
00261
00262     return 0;
00263 }
00264
00265 bool insert(Entry *e) { return _tree.insert(e).second; }
00266
00267 void dump(bool rec = false, int indent = 0) const;
00268
00269 protected:
00270 // server support -----
00283 virtual Entry *alloc_dynamic_entry(Name const &n, unsigned flags) = 0;
00284
00290 virtual void free_dynamic_entry(Entry *e) = 0;
00291
00314 virtual int get_epiface(l4_umword_t data, bool is_local, L4::Epiface **lo) = 0;
00315
00328 virtual int copy_receive_cap(L4::Cap<void> *cap) = 0;
00329
00338 virtual void free_capability(L4::Cap<void> cap) = 0;
00339
00348 virtual void free_epiface(L4::Epiface *epiface) = 0;
00349
00350 int insert_entry(Name const &name, unsigned flags, Entry **e)
00351 {
00352     Entry *n = find(name);
00353     if (n && n->obj()->is_valid())
00354     {
00355         if (!(flags & L4Re::Namespace::Overwrite)
00356             && n->obj()->cap().validate(L4_BASE_TASK_CAP).label() > 0)
00357             return -L4_EEXIST;
00358
00359         if (n->obj()->is_local())
00360             free_epiface(n->obj()->obj());
00361         else
00362             free_capability(n->obj()->cap());
00363
00364         if (n->is_dynamic())
00365         {
00366             remove(n->name());
00367             free_dynamic_entry(n);
00368             n = 0;
00369         }
00370         else
00371         {
00372             if (!n->obj()->is_replacable())
00373                 return -L4_EEXIST;
00374             n->obj()->reset(Obj::F_rw);
00375         }
00376     }
00377
00378     flags &= L4Re::Namespace::Cap_flags;
00379     if (!n)
00380     {
00381         if (!(n = alloc_dynamic_entry(name, flags)))

```

```

00382         return -L4_ENOMEM;
00383     else
00384     {
00385         if (!insert(n))
00386         {
00387             free_dynamic_entry(n);
00388             return -L4_EEXIST;
00389         }
00390     }
00391 }
00392
00393 *e = n;
00394 return 0;
00395 }
00396
00397 public:
00398 // server interface -----
00399 int op_query(L4Re::Namespace::Rights,
00400             L4::Ipc::Array_in_buf<char, unsigned long> const &name,
00401             L4::Ipc::Snd_fpage &snd_cap, L4::Ipc::Opt<L4::Opcode> &dummy,
00402             L4::Ipc::Opt<L4::Ipc::Array_ref<char, unsigned long> > &out_name)
00403 {
00404     #if 1
00405     _dbg.printf("query: [%ld] '%.*s'\n", name.length,
00406                static_cast<int>(name.length), name.data);
00407     #endif
00408
00409     char const *sep
00410     = static_cast<char const*>(memchr(name.data, '/', name.length));
00411     unsigned long part;
00412     if (sep)
00413         part = sep - name.data;
00414     else
00415         part = name.length;
00416
00417     Entry *n = find(Name(name.data, part));
00418     if (!n)
00419         return -L4_ENOENT;
00420     else if (!n->obj()->is_valid())
00421         return -L4_EAGAIN;
00422     else
00423     {
00424         if (n->obj()->cap().validate(L4_BASE_TASK_CAP).label() <= 0)
00425         {
00426             if (n->obj()->is_local())
00427                 free_epiface(n->obj()->obj());
00428             else
00429                 free_capability(n->obj()->cap());
00430
00431             if (n->is_dynamic())
00432             {
00433                 remove(n->name());
00434                 free_dynamic_entry(n);
00435             }
00436             return -L4_ENOENT;
00437         }
00438
00439         // make picky clients happy
00440         dummy.set_valid();
00441
00442         l4_umword_t result = 0;
00443
00444         out_name.set_valid();
00445         if (part < name.length)
00446         {
00447             result |= L4Re::Namespace::Partly_resolved;
00448             memcpy(out_name->data, name.data + part + 1, name.length - part - 1);
00449             out_name->length = name.length - part - 1;
00450         }
00451         else
00452             out_name->length = 0;
00453
00454         unsigned flags = L4_CAP_FPAGE_R;
00455         if (n->obj()->is_rw()) flags |= L4_CAP_FPAGE_W;
00456         if (n->obj()->is_strong()) flags |= L4_CAP_FPAGE_S;
00457
00458         snd_cap = L4::Ipc::Snd_fpage(n->obj()->cap(), flags);
00459         _dbg.printf(" result = %lx flgs=%x strg=%d\n",
00460                    result, flags, static_cast<int>(n->obj()->is_strong()));
00461         return result;
00462     }
00463 }
00464
00465 int op_register_obj(L4Re::Namespace::Rights, unsigned flags,
00466                   L4::Ipc::Array_in_buf<char, unsigned long> const &name,
00467                   L4::Ipc::Snd_fpage &cap)
00468 {

```

```

00469     if (name.length == 0 || memchr(name.data, '/', name.length))
00470         return -L4_EINVAL;
00471
00472     L4::Cap<void> reg_cap(L4_INVALID_CAP);
00473     L4::Epiface *src_o = 0;
00474
00475     // Did we receive something we have handed out ourselves? If yes,
00476     // register the object under the given name but do not allocate
00477     // anything more.
00478     if (cap.id_received() || cap.local_id_received())
00479     {
00480         if (int ret = get_epiface(cap.data(), cap.local_id_received(), &src_o))
00481             return ret;
00482
00483         // Make sure rights are restricted to the mapped rights.
00484         flags &= (cap.data() & 0x3UL) | ~0x3UL;
00485     }
00486     else if (cap.cap_received())
00487     {
00488         if (int ret = copy_receive_cap(&reg_cap))
00489             return ret;
00490     }
00491     else if (!cap.is_valid())
00492     {
00493         reg_cap = L4::Cap<void>::Invalid;
00494     }
00495     else
00496         return -L4_EINVAL;
00497
00498     // got a valid entry to register
00499     _dbg.printf("register: '%.*s' flags=%x\n", static_cast<int>(name.length),
00500               name.data, flags);
00501
00502     Name _name(name.data, name.length);
00503
00504     Entry *n;
00505     if (int r = insert_entry(_name, flags, &n))
00506     {
00507         if (cap.cap_received())
00508             free_capability(reg_cap);
00509         if (src_o)
00510             free_epiface(src_o);
00511
00512         return r;
00513     }
00514
00515     if (src_o)
00516         n->set(Namespace::Obj(flags & L4Re::Namespace::Cap_flags, src_o));
00517     else if (reg_cap.is_valid())
00518         n->set(Namespace::Obj(flags & L4Re::Namespace::Cap_flags, reg_cap));
00519
00520     return 0;
00521 }
00522
00523 int op_unlink(L4Re::Namespace::Rights,
00524             L4::Ipc::Array_in_buf<char, unsigned long> const &name)
00525 {
00526     #if 1
00527     _dbg.printf("unlink: [%ld] '%.*s'\n", name.length,
00528               static_cast<int>(name.length), name.data);
00529     #endif
00530
00531     char const *sep
00532         = static_cast<char const*>(memchr(name.data, '/', name.length));
00533     unsigned long part;
00534     if (sep)
00535         part = sep - name.data;
00536     else
00537         part = name.length;
00538
00539     Entry *n = find(Name(name.data, part));
00540     if (!n || !n->obj()->is_valid())
00541         return -L4_ENOENT;
00542
00543     if (n->obj()->is_local())
00544         free_epiface(n->obj()->obj());
00545     else
00546         free_capability(n->obj()->cap());
00547
00548     if (n->is_dynamic())
00549     {
00550         remove(n->name());
00551         free_dynamic_entry(n);
00552     }
00553     else
00554         return -L4_EACCESS;
00555

```

```

00556
00557     return 0;
00558 }
00559 };
00560
00561 }}}
00562

```

## 16.400 object\_registry

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020
00021 #pragma once
00022
00023 #include <l4/re/util/cap_alloc>
00024 #include <l4/re/util/unique_cap>
00025 #include <l4/re/consts>
00026 #include <l4/re/env>
00027
00028 #include <l4/sys/cxx/ipc_server_loop>
00029 #include <l4/sys/factory>
00030 #include <l4/sys/task>
00031 #include <l4/sys/thread>
00032 #include <l4/sys/ipc_gate>
00033
00034 #include <l4/cxx/exceptions>
00035
00036 namespace L4Re { namespace Util {
00037
00052 class Object_registry :
00053     public L4::Basic_registry,
00054     public L4::Registry_iface
00055 {
00060     struct Null_handler : L4::Epiface_t<Null_handler, L4::Kobject>
00061     {};
00062
00063 protected:
00064     L4::Cap<L4::Thread> _server;
00065     L4::Cap<L4::Factory> _factory;
00066     L4::Ipc_svr::Server_iface *_sif;
00067
00068 private:
00069     Null_handler _null_handler;
00070
00071 public:
00072     explicit
00073     Object_registry(L4::Ipc_svr::Server_iface *sif)
00074     : _server(L4Re::Env::env()->main_thread()),
00075       _factory(L4Re::Env::env()->factory()),
00076       _sif(sif)
00077     {}
00078
00092     Object_registry(L4::Ipc_svr::Server_iface *sif,
00093                     L4::Cap<L4::Thread> server,
00094                     L4::Cap<L4::Factory> factory)
00095     : _server(server), _factory(factory), _sif(sif)
00096     {}
00097
00098 private:
00099     typedef L4::Ipc_svr::Server_iface Server_iface;
00100     typedef Server_iface::Demand Demand;
00101
00102     L4::Cap<L4::Rcv_endpoint>
00103     _register_ep(L4::Epiface *o, L4::Cap<L4::Rcv_endpoint> ep,
00104                 Demand const &demand)

```



```

00105 {
00106     int err = _sif->alloc_buffer_demand(demand);
00107     if (err < 0)
00108         return L4::Cap<L4::Rcv_endpoint>(err | L4_INVALID_CAP_BIT);
00109
00110     err = o->set_server(_sif, ep);
00111     if (err < 0)
00112         return L4::Cap<L4::Rcv_endpoint>(err | L4_INVALID_CAP_BIT);
00113
00114     l4_umword_t id = l4_umword_t(o);
00115     err = l4_error(ep->bind_thread(_server, id));
00116     if (err < 0)
00117         return L4::Cap<L4::Rcv_endpoint>(err | L4_INVALID_CAP_BIT);
00118
00119     return ep;
00120 }
00121
00122 L4::Cap<void> _register_ep(L4::Epiface *o, char const *service,
00123                          Demand const &demand)
00124 {
00125     L4::Cap<L4::Rcv_endpoint> cap = L4Re::Env::env()->get_cap<L4::Rcv_endpoint>(service);
00126     if (!cap.is_valid())
00127         return cap;
00128
00129     return _register_ep(o, cap, demand);
00130 }
00131
00132 L4::Cap<void> _register_gate(L4::Epiface *o, Demand const &demand)
00133 {
00134     int err = _sif->alloc_buffer_demand(demand);
00135     if (err < 0)
00136         return L4::Cap<void>(err | L4_INVALID_CAP_BIT);
00137
00138     auto cap = L4Re::Util::make_unique_cap<L4::Kobject>();
00139
00140     if (!cap.is_valid())
00141         return cap.get();
00142
00143     l4_umword_t id = l4_umword_t(o);
00144     err = l4_error(_factory->create_gate(cap.get(), _server, id));
00145     if (err < 0)
00146         return L4::Cap<void>(err | L4_INVALID_CAP_BIT);
00147
00148     err = o->set_server(_sif, cap.get(), true);
00149     if (err < 0)
00150         return L4::Cap<void>(err | L4_INVALID_CAP_BIT);
00151
00152     return cap.release();
00153 }
00154
00155 L4::Cap<L4::Irq> _register_irq(L4::Epiface *o,
00156                              Demand const &demand)
00157 {
00158     int err = _sif->alloc_buffer_demand(demand);
00159     if (err < 0)
00160         return L4::Cap<L4::Irq>(err | L4_INVALID_CAP_BIT);
00161
00162     auto cap = L4Re::Util::make_unique_cap<L4::Irq>();
00163
00164     if (!cap.is_valid())
00165         return cap.get();
00166
00167     l4_umword_t id = l4_umword_t(o);
00168     err = l4_error(_factory->create(cap.get()));
00169     if (err < 0)
00170         return L4::Cap<L4::Irq>(err | L4_INVALID_CAP_BIT);
00171
00172     err = o->set_server(_sif, cap.get(), true);
00173     if (err < 0)
00174         return L4::Cap<L4::Irq>(err | L4_INVALID_CAP_BIT);
00175
00176     err = l4_error(cap->bind_thread(_server, id));
00177     if (err < 0)
00178         return L4::Cap<L4::Irq>(err | L4_INVALID_CAP_BIT);
00179
00180     return cap.release();
00181 }
00182
00183 static Demand _get_buffer_demand(L4::Epiface *o)
00184 { return o->get_buffer_demand(); }
00185
00186 template<typename T>
00187 static Demand _get_buffer_demand(T *,
00188                                  typename L4::Kobject_typeid<typename T::Interface>::Demand
00189                                  d = typename L4::Kobject_typeid<typename T::Interface>::Demand())
00190 { return d; }
00191

```

```

00192 public:
00205 L4::Cap<void> register_obj(L4::Epiface *o, char const *service) override
00206 {
00207     return _register_ep(o, service, _get_buffer_demand(o));
00208 }
00209
00222 L4::Cap<void> register_obj(L4::Epiface *o) override
00223 {
00224     return _register_gate(o, _get_buffer_demand(o));
00225 }
00226
00238 L4::Cap<L4::Irq> register_irq_obj(L4::Epiface *o) override
00239 {
00240     return _register_irq(o, _get_buffer_demand(o));
00241 }
00242
00243 // pass access to deprecated register_irq_obj
00244 using L4::Registry_iface::register_irq_obj;
00245
00259 L4::Cap<L4::Rcv_endpoint>
00260 register_obj(L4::Epiface *o, L4::Cap<L4::Rcv_endpoint> ep) override
00261 {
00262     return _register_ep(o, ep, _get_buffer_demand(o));
00263 }
00264
00265
00276 void unregister_obj(L4::Epiface *o, bool unmap = true) override
00277 {
00278     L4::Epiface::Stored_cap c;
00279
00280     if (!o || !o->obj_cap().is_valid())
00281         return;
00282
00283     c = o->obj_cap();
00284
00285     if (unmap)
00286         L4::Cap<L4::Task> (L4Re::This_task)->unmap(c.fpage(), L4_FP_ALL_SPACES);
00287
00288     // make sure unhandled ipc ends up with the null handler
00289     L4::Thread::Modify_senders todo;
00290     todo.add(~3UL, reinterpret_cast<l4_umword_t>(o),
00291             ~0UL, reinterpret_cast<l4_umword_t>
00292                 (static_cast<L4::Epiface *>(&_null_handler)));
00293     _server->modify_senders(todo);
00294
00295     // we use bit 4 to indicated an internally allocated cap
00296     if (c.managed())
00297         cap_alloc.free(c);
00298
00299     o->set_server(0, L4::Cap<void>::Invalid);
00300 }
00301 };
00302
00306 template< typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks >
00307 class Registry_server : public L4::Server<LOOP_HOOKS>
00308 {
00309 private:
00310     typedef L4::Server<LOOP_HOOKS> Base;
00311     Object_registry _registry;
00312
00313 public:
00319     Registry_server() : _registry(this)
00320     {}
00321
00331     Registry_server(l4_utcb_t *, L4::Cap<L4::Thread> server,
00332                     L4::Cap<L4::Factory> factory) L4_DEPRECATED("Omit UTCB pointer argument")
00333     : _registry(this, server, factory)
00334     {}
00335
00342     Registry_server(L4::Cap<L4::Thread> server,
00343                     L4::Cap<L4::Factory> factory)
00344     : _registry(this, server, factory)
00345     {}
00346
00348     Object_registry const *registry() const { return &_registry; }
00350     Object_registry *registry() { return &_registry; }
00351
00358     void L4_NORETURN loop(l4_utcb_t *utcb = l4_utcb())
00359     { Base::template loop<L4::Runtime_error, Object_registry &>(_registry, utcb); }
00360 };
00361
00362 }

```

## 16.401 poll\_timeout\_kipclock

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2012 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU Lesser General Public License 2.1.
00007  * Please see the COPYING-LGPL-2.1 file for details.
00008  */
00009 #pragma once
00010
00011 #include <cassert>
00012 #include <l4/sys/kip.h>
00013 #include <l4/re/env.h>
00014
00015 namespace L4 {
00016
00017 class Poll_timeout_kipclock
00018 {
00019 public:
00020     Poll_timeout_kipclock(unsigned poll_time_us)
00021     {
00022         set(poll_time_us);
00023     }
00024
00025     void set(unsigned poll_time_us)
00026     {
00027         _timeout = l4_kip_clock(l4re_kip()) + poll_time_us;
00028         _last_check = true;
00029     }
00030
00031     bool test(bool expression = true)
00032     {
00033         if (!expression)
00034             return false;
00035
00036         return _last_check = l4_kip_clock(l4re_kip()) < _timeout;
00037     }
00038
00039     bool timed_out() const { return !_last_check; }
00040
00041 private:
00042     l4_cpu_time_t _timeout;
00043     bool _last_check;
00044 };
00045
00046 }

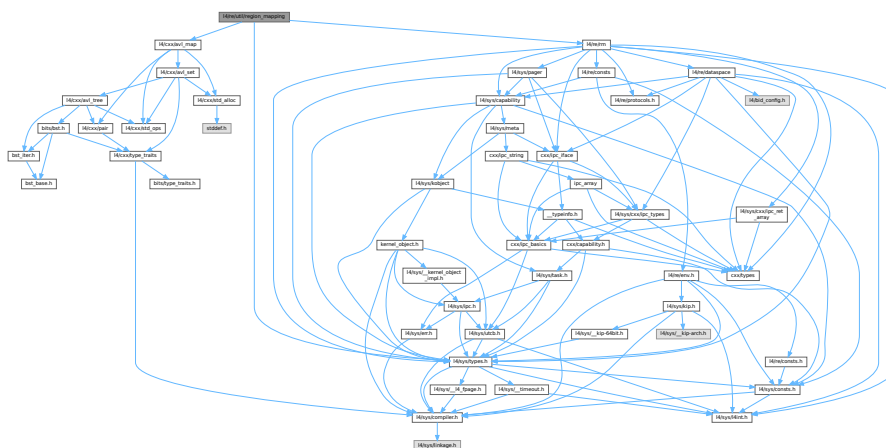
```

## 16.402 I4/re/util/region\_mapping File Reference

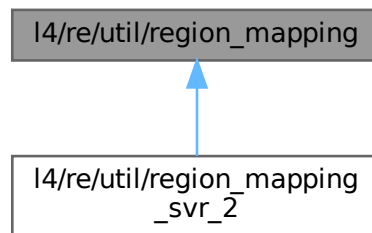
### Region handling.

```
#include <l4/cxx/avl_map>
#include <l4/sys/types.h>
#include <l4/re/rm>
```

Include dependency graph for region mapping:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [L4Re](#)  
[L4Re](#) C++ Interfaces.
- namespace [L4Re::Util](#)  
 Documentation of the [L4](#) Runtime Environment utility functionality in C++.

## 16.402.1 Detailed Description

Region handling.

Definition in file [region\\_mapping](#).

## 16.403 region\_mapping

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00005  *                Alexander Warg <warg@os.inf.tu-dresden.de>,
00006  *                Björn Döbel <doebel@os.inf.tu-dresden.de>
00007  *                economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #pragma once
00023 #include <l4/cxx/avl_map>
00024 #include <l4/sys/types.h>
00025 #include <l4/re/rm>
00026

```

```

00033
00034 namespace L4Re { namespace Util {
00035 class Region
00036 {
00037 private:
00038     l4_addr_t _start, _end;
00039
00040 public:
00041     Region() noexcept : _start(~0UL), _end(~0UL) {}
00042     Region(l4_addr_t addr) noexcept : _start(addr), _end(addr) {}
00043     Region(l4_addr_t start, l4_addr_t end) noexcept
00044         : _start(start), _end(end) {}
00045     l4_addr_t start() const noexcept { return _start; }
00046     l4_addr_t end() const noexcept { return _end; }
00047     unsigned long size() const noexcept { return end() - start() + 1; }
00048     bool invalid() const noexcept { return _start == ~0UL && _end == ~0UL; }
00049     bool operator < (Region const &o) const noexcept
00050     { return end() < o.start(); }
00051     bool contains(Region const &o) const noexcept
00052     { return o.start() >= start() && o.end() <= end(); }
00053     bool operator == (Region const &o) const noexcept
00054     { return o.start() == start() && o.end() == end(); }
00055     ~Region() noexcept {}
00056 };
00057
00058 template< typename DS, typename OPS >
00059 class Region_handler
00060 {
00061 private:
00062     L4Re::Rm::Offset _offs;
00063     DS _mem;
00064     l4_cap_idx_t _client_cap = L4_INVALID_CAP;
00065     L4Re::Rm::Region_flags _flags;
00066
00067 public:
00068     typedef DS Dataspace;
00069     typedef OPS Ops;
00070     typedef typename OPS::Map_result Map_result;
00071
00072     Region_handler() noexcept : _offs(0), _mem(), _flags() {}
00073     Region_handler(Dataspace const &mem, l4_cap_idx_t client_cap,
00074         L4Re::Rm::Offset offset = 0,
00075         L4Re::Rm::Region_flags flags = L4Re::Rm::Region_flags(0)) noexcept
00076     : _offs(offset), _mem(mem), _client_cap(client_cap), _flags(flags)
00077     {}
00078
00079     Dataspace const &memory() const noexcept
00080     {
00081         return _mem;
00082     }
00083
00084     l4_cap_idx_t client_cap_idx() const noexcept
00085     {
00086         return _client_cap;
00087     }
00088
00089     L4Re::Rm::Offset offset() const noexcept
00090     {
00091         return _offs;
00092     }
00093
00094     constexpr bool is_ro() const noexcept
00095     {
00096         return !(_flags & L4Re::Rm::F::W);
00097     }
00098
00099     L4Re::Rm::Region_flags caching() const noexcept
00100     {
00101         return _flags & L4Re::Rm::F::Caching_mask;
00102     }
00103
00104     L4Re::Rm::Region_flags flags() const noexcept
00105     {
00106         return _flags;
00107     }
00108
00109     Region_handler operator + (l4_int64_t offset) const noexcept
00110     {
00111         Region_handler n = *this; n._offs += offset; return n;
00112     }
00113
00114     void free(l4_addr_t start, unsigned long size) const noexcept
00115     {
00116         Ops::free(this, start, size);
00117     }
00118
00119     int map(l4_addr_t addr, Region const &r, bool writable,

```

```

00120         Map_result *result) const
00121     {
00122         return Ops::map(this, addr, r, writable, result);
00123     }
00124
00125     int map_info(l4_addr_t *start_addr, l4_addr_t *end_addr) const
00126     {
00127         return Ops::map_info(this, start_addr, end_addr);
00128     }
00129
00130 };
00131
00132
00133 template< typename Hdlr, template<typename T> class Alloc >
00134 class Region_map
00135 {
00136 protected:
00137     typedef cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc > Tree;
00138     Tree _rm;
00139     Tree _am;
00140
00141 private:
00142     l4_addr_t _start;
00143     l4_addr_t _end;
00144
00145 protected:
00146     void set_limits(l4_addr_t start, l4_addr_t end) noexcept
00147     {
00148         _start = start;
00149         _end = end;
00150     }
00151
00152 public:
00153     typedef typename Tree::Item_type Item;
00154     typedef typename Tree::Node Node;
00155     typedef typename Tree::Key_type Key_type;
00156     typedef Hdlr Region_handler;
00157
00158     typedef typename Tree::Iterator Iterator;
00159     typedef typename Tree::Const_iterator Const_iterator;
00160     typedef typename Tree::Rev_iterator Rev_iterator;
00161     typedef typename Tree::Const_rev_iterator Const_rev_iterator;
00162
00163     Iterator begin() noexcept { return _rm.begin(); }
00164     Const_iterator begin() const noexcept { return _rm.begin(); }
00165     Iterator end() noexcept { return _rm.end(); }
00166     Const_iterator end() const noexcept { return _rm.end(); }
00167
00168     Iterator area_begin() noexcept { return _am.begin(); }
00169     Const_iterator area_begin() const noexcept { return _am.begin(); }
00170     Iterator area_end() noexcept { return _am.end(); }
00171     Const_iterator area_end() const noexcept { return _am.end(); }
00172     Node area_find(Key_type const &c) const noexcept { return _am.find_node(c); }
00173
00174     l4_addr_t min_addr() const noexcept { return _start; }
00175     l4_addr_t max_addr() const noexcept { return _end; }
00176
00177
00178     Region_map(l4_addr_t start, l4_addr_t end) noexcept : _start(start), _end(end) {}
00179
00180     Node find(Key_type const &key) const noexcept
00181     {
00182         Node n = _rm.find_node(key);
00183         if (!n)
00184             return Node();
00185
00186         // 'find' should find any region overlapping with the searched one, the
00187         // caller should check for further requirements
00188         if (0)
00189             if (!n->first.contains(key))
00190                 return Node();
00191
00192         return n;
00193     }
00194
00195     Node lower_bound(Key_type const &key) const noexcept
00196     {
00197         Node n = _rm.lower_bound_node(key);
00198         return n;
00199     }
00200
00201     Node lower_bound_area(Key_type const &key) const noexcept
00202     {
00203         Node n = _am.lower_bound_node(key);
00204         return n;
00205     }
00206

```

```

00207  l4_addr_t attach_area(l4_addr_t addr, unsigned long size,
00208                      L4Re::Rm::Flags flags = L4Re::Rm::Flags(0),
00209                      unsigned char align = L4_PAGESHIFT) noexcept
00210  {
00211      if (size < 2)
00212          return L4_INVALID_ADDR;
00213
00214      Region c;
00215
00216      if (!(flags & L4Re::Rm::F::Search_addr))
00217      {
00218          c = Region(addr, addr + size - 1);
00219          Node r = _am.find_node(c);
00220          if (r)
00221              return L4_INVALID_ADDR;
00222      }
00223
00224      while (flags & L4Re::Rm::F::Search_addr)
00225      {
00226          if (addr < min_addr() || (addr + size - 1) > max_addr())
00227              addr = min_addr();
00228          addr = find_free(addr, max_addr(), size, align, flags);
00229          if (addr == L4_INVALID_ADDR)
00230              return L4_INVALID_ADDR;
00231          c = Region(addr, addr + size - 1);
00232          Node r = _am.find_node(c);
00233          if (!r)
00234              break;
00235          if (r->first.end() >= max_addr())
00236              return L4_INVALID_ADDR;
00237          addr = r->first.end() + 1;
00238      }
00239      if (_am.insert(c, Hdlr(typename Hdlr::Dataspace(), 0, 0, flags.region_flags())).second == 0)
00240          return addr;
00241      return L4_INVALID_ADDR;
00242  }
00243
00244  bool detach_area(l4_addr_t addr) noexcept
00245  {
00246      if (_am.remove(addr))
00247          return false;
00248      return true;
00249  }
00250
00251  void *attach(void *addr, unsigned long size, Hdlr const &hdlr,
00252              L4Re::Rm::Flags flags = L4Re::Rm::Flags(0),
00253              unsigned char align = L4_PAGESHIFT) noexcept
00254  {
00255      if (size < 2)
00256          return L4_INVALID_PTR;
00257
00258      l4_addr_t beg, end;
00259      int err = hdlr.map_info(&beg, &end);
00260      if (err > 0)
00261      {
00262          // Mapping address determined by underlying dataspace. Make sure we
00263          // prevent any additional alignment. We already know the place!
00264          beg += hdlr.offset();
00265          end = beg + size - 1U;
00266          align = L4_PAGESHIFT;
00267
00268          // In case of exact mappings, the supplied address must match because
00269          // we cannot remap.
00270          if (!(flags & L4Re::Rm::F::Search_addr)
00271              && reinterpret_cast<l4_addr_t>(addr) != beg)
00272              return L4_INVALID_PTR;
00273
00274          // When searching for a suitable address, the start must cover the
00275          // dataspace beginning to "find" the right spot.
00276          if ((flags & L4Re::Rm::F::Search_addr)
00277              && reinterpret_cast<l4_addr_t>(addr) > beg)
00278              return L4_INVALID_PTR;
00279      }
00280      else if (err == 0)
00281      {
00282          beg = reinterpret_cast<l4_addr_t>(addr);
00283          end = max_addr();
00284      }
00285      else if (err < 0)
00286          return L4_INVALID_PTR;
00287  }

```

```

00294
00295     if (flags & L4Re::Rm::F::In_area)
00296     {
00297         Node r = _am.find_node(Region(beg, beg + size - 1));
00298         if (!r || (r->second.flags() & L4Re::Rm::F::Reserved))
00299             return L4_INVALID_PTR;
00300
00301         end = r->first.end();
00302     }
00303
00304     if (flags & L4Re::Rm::F::Search_addr)
00305     {
00306         beg = find_free(beg, end, size, align, flags);
00307         if (beg == L4_INVALID_ADDR)
00308             return L4_INVALID_PTR;
00309     }
00310
00311     if (!(flags & (L4Re::Rm::F::Search_addr | L4Re::Rm::F::In_area))
00312         && _am.find_node(Region(beg, beg + size - 1)))
00313         return L4_INVALID_PTR;
00314
00315     if (beg < min_addr() || beg + size - 1 > end)
00316         return L4_INVALID_PTR;
00317
00318     if (_rm.insert(Region(beg, beg + size - 1), hdlr).second == 0)
00319         return reinterpret_cast<void*>(beg);
00320
00321     return L4_INVALID_PTR;
00322 }
00323
00324 int detach(void *addr, unsigned long sz, unsigned flags,
00325            Region *reg, Hdlr *hdlr) noexcept
00326 {
00327     l4_addr_t a = reinterpret_cast<l4_addr_t>(addr);
00328     Region dr(a, a + sz - 1);
00329     Region res(~0UL, 0);
00330
00331     Node r = find(dr);
00332     if (!r)
00333         return -L4_ENOENT;
00334
00335     Region g = r->first;
00336     Hdlr const &h = r->second;
00337
00338     if (flags & L4Re::Rm::Detach_overlap || dr.contains(g))
00339     {
00340         // successful removal of the AVL tree item also frees the node
00341         Hdlr h_copy = h;
00342
00343         if (_rm.remove(g))
00344             return -L4_ENOENT;
00345
00346         if (!(flags & L4Re::Rm::Detach_keep) && (h_copy.flags() & L4Re::Rm::F::Detach_free))
00347             h_copy.free(0, g.size());
00348
00349         if (hdlr)
00350             *hdlr = h_copy;
00351         if (reg)
00352             *reg = g;
00353
00354         if (find(dr))
00355             return Rm::Detached_ds | Rm::Detach_again;
00356         else
00357             return Rm::Detached_ds;
00358     }
00359     else if (dr.start() <= g.start())
00360     {
00361         // move the start of a region
00362
00363         if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::F::Detach_free))
00364             h.free(0, dr.end() + 1 - g.start());
00365
00366         unsigned long sz = dr.end() + 1 - g.start();
00367         Item &cn = const_cast<Item &>(*r);
00368         cn.first = Region(dr.end() + 1, g.end());
00369         cn.second = cn.second + sz;
00370         if (hdlr)
00371             *hdlr = Hdlr();
00372         if (reg)
00373             *reg = Region(g.start(), dr.end());
00374         if (find(dr))
00375             return Rm::Kept_ds | Rm::Detach_again;
00376         else
00377             return Rm::Kept_ds;
00378     }
00379     else if (dr.end() >= g.end())
00380     {

```



```

00381         // move the end of a region
00382
00383         if (!(flags & L4Re::Rm::Detach_keep)
00384             && (h.flags() & L4Re::Rm::F::Detach_free))
00385             h.free(dr.start() - g.start(), g.end() + 1 - dr.start());
00386
00387         Item &cn = const_cast<Item &>(*r);
00388         cn.first = Region(g.start(), dr.start() - 1);
00389         if (hdlr)
00390             *hdlr = Hdlr();
00391         if (reg)
00392             *reg = Region(dr.start(), g.end());
00393
00394         if (find(dr))
00395             return Rm::Kept_ds | Rm::Detach_again;
00396         else
00397             return Rm::Kept_ds;
00398     }
00399     else if (g.contains(dr))
00400     {
00401         // split a single region that contains the new region
00402
00403         if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::F::Detach_free))
00404             h.free(dr.start() - g.start(), dr.size());
00405
00406         // first move the end off the existing region before the new one
00407         Item &cn = const_cast<Item &>(*r);
00408         cn.first = Region(g.start(), dr.start()-1);
00409
00410         int err;
00411
00412         // insert a second region for the remaining tail of
00413         // the old existing region
00414         err = _rm.insert(Region(dr.end() + 1, g.end()),
00415                         h + (dr.end() + 1 - g.start())).second;
00416
00417         if (err)
00418             return err;
00419
00420         if (hdlr)
00421             *hdlr = h;
00422         if (reg)
00423             *reg = dr;
00424         return Rm::Split_ds;
00425     }
00426     return -L4_ENOENT;
00427 }
00428
00429 l4_addr_t find_free(l4_addr_t start, l4_addr_t end, l4_addr_t size,
00430                    unsigned char align, L4Re::Rm::Flags flags) const noexcept;
00431
00432 };
00433
00434
00435 template< typename Hdlr, template<typename T> class Alloc >
00436 l4_addr_t
00437 Region_map<Hdlr, Alloc>::find_free(l4_addr_t start, l4_addr_t end,
00438                                   unsigned long size, unsigned char align, L4Re::Rm::Flags flags) const noexcept
00439 {
00440     l4_addr_t addr = start;
00441
00442     if (addr == ~0UL || addr < min_addr() || addr >= end)
00443         addr = min_addr();
00444
00445     addr = l4_round_size(addr, align);
00446     Node r;
00447
00448     for(;;)
00449     {
00450         if (addr > 0 && addr - 1 > end - size)
00451             return L4_INVALID_ADDR;
00452
00453         Region c(addr, addr + size - 1);
00454         r = _rm.find_node(c);
00455
00456         if (!r)
00457         {
00458             if (!(flags & L4Re::Rm::F::In_area) && (r = _am.find_node(c)))
00459             {
00460                 if (r->first.end() > end - size)
00461                     return L4_INVALID_ADDR;
00462
00463                 addr = l4_round_size(r->first.end() + 1, align);
00464                 continue;
00465             }
00466             break;
00467         }
00468     }

```

```

00468         else if (r->first.end() > end - size)
00469             return L4_INVALID_ADDR;
00470
00471         addr = l4_round_size(r->first.end() + 1, align);
00472     }
00473
00474     if (!r)
00475         return addr;
00476
00477     return L4_INVALID_ADDR;
00478 }
00479
00480 }}

```

## 16.404 region\_mapping\_svr\_2

```

00001 // vi:set ft=c++ -- Mode: C++ --
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019
00020 #include <l4/sys/types.h>
00021 #include <l4/re/rm>
00022 #include <l4/re/util/region_mapping>
00023
00024 namespace L4Re { namespace Util {
00025
00026     template<typename DERIVED, typename Dbg>
00027     struct Rm_server
00028     {
00029     private:
00030         DERIVED *rm() { return static_cast<DERIVED*>(this); }
00031         DERIVED const *rm() const { return static_cast<DERIVED const*>(this); }
00032     public:
00033
00034         long op_attach(L4Re::Rm::Rights, l4_addr_t &_start,
00035                       unsigned long size, Rm::Flags flags,
00036                       L4::Ipc::Snd_fpage ds_cap, L4Re::Rm::Offset offs,
00037                       unsigned char align, l4_cap_idx_t client_cap_idx)
00038         {
00039             typename DERIVED::Dataspace ds;
00040
00041             if (!(flags & Rm::F::Reserved))
00042             {
00043                 if (long r = rm()->validate_ds
00044                     (static_cast<DERIVED*>(this)->server_iface(), ds_cap,
00045                      flags.region_flags(), &ds))
00046                     return r;
00047             }
00048
00049             size = l4_round_page(size);
00050             l4_addr_t start = l4_trunc_page(_start);
00051
00052             if (size < L4_PAGESIZE)
00053                 return -L4_EINVAL;
00054
00055             Rm::Region_flags r_flags = flags.region_flags();
00056             Rm::Attach_flags a_flags = flags.attach_flags();
00057
00058             typename DERIVED::Region_handler handler(ds, client_cap_idx, offs, r_flags);
00059             start = l4_addr_t(rm()->attach(reinterpret_cast<void*>(start), size,
00060                                           handler, a_flags, align));
00061
00062             if (start == L4_INVALID_ADDR)
00063                 return -L4_EADDRNOTAVAIL;
00064
00065             _start = start;
00066             return L4_EOK;
00067         }
00068     };
00069 }

```

```

00076     }
00077
00081     long op_free_area(L4Re::Rm::Rights, l4_addr_t start)
00082     {
00083         if (!rm()->detach_area(start))
00084             return -L4_ENOENT;
00085
00086         return L4_EOK;
00087     }
00088
00092     long op_find(L4Re::Rm::Rights, l4_addr_t &addr, unsigned long &size,
00093                 L4Re::Rm::Flags &flags, L4Re::Rm::Offset &offset,
00094                 L4::Cap<L4Re::Dataspace> &m)
00095     {
00096         if (!DERIVED::Have_find)
00097             return -L4_EPERM;
00098
00099         Rm::Flags flag_area { 0 };
00100
00101         typename DERIVED::Node r = rm()->find(Region(addr, addr + size - 1));
00102         if (!r)
00103         {
00104             r = rm()->area_find(Region(addr, addr + size - 1));
00105             if (!r)
00106                 return -L4_ENOENT;
00107             flag_area = Rm::F::In_area;
00108         }
00109
00110         addr = r->first.start();
00111         size = r->first.end() + 1 - addr;
00112
00113         flags = r->second.flags() | flag_area;
00114         offset = r->second.offset();
00115         m = L4::Cap<L4Re::Dataspace>(DERIVED::find_res(r->second.memory()));
00116         return L4_EOK;
00117     }
00118
00122     long op_detach(L4Re::Rm::Rights, l4_addr_t addr,
00123                 unsigned long size, unsigned flags,
00124                 l4_addr_t &start, l4_addr_t &rsz,
00125                 l4_cap_idx_t &mem_cap)
00126     {
00127         Region r;
00128         typename DERIVED::Region_handler h;
00129         int err = rm()->detach(reinterpret_cast<void*>(addr), size, flags, &r, &h);
00130         if (err < 0)
00131         {
00132             start = rsz = 0;
00133             mem_cap = L4_INVALID_CAP;
00134             return err;
00135         }
00136
00137         if (r.invalid())
00138         {
00139             start = rsz = 0;
00140             mem_cap = L4_INVALID_CAP;
00141             return -L4_ENOENT;
00142         }
00143
00144         start = r.start();
00145         rsz = r.size();
00146         mem_cap = h.client_cap_idx();
00147         return err;
00148     }
00149
00153     long op_reserve_area(L4Re::Rm::Rights, l4_addr_t &start, unsigned long size,
00154                        L4Re::Rm::Flags flags, unsigned char align)
00155     {
00156         start = rm()->attach_area(start, size, flags, align);
00157         if (start == L4_INVALID_ADDR)
00158             return -L4_EADDRNOTAVAIL;
00159         return L4_EOK;
00160     }
00161
00165     long op_get_regions(L4Re::Rm::Rights, l4_addr_t addr,
00166                       L4::Ipc::Ret_array<L4Re::Rm::Region> regions)
00167     {
00168         typename DERIVED::Node r;
00169         unsigned num = 0;
00170         while ((r = rm()->lower_bound(Region(addr)))
00171             {
00172             Rm::Region &x = regions.value[num];
00173             x.start = r->first.start();
00174             x.end = r->first.end();
00175
00176             if (++num >= regions.max)
00177                 break;

```

```

00178
00179         if (x.end >= rm()->max_addr())
00180             break;
00181         addr = x.end + 1;
00182     }
00183     return num;
00184 }
00185
00186 long op_get_areas(L4Re::Rm::Rights, l4_addr_t addr,
00187                 L4::Ipc::Ret_array<L4Re::Rm::Area> areas)
00188 {
00189     typename DERIVED::Node r;
00190     unsigned num = 0;
00191     while ((r = rm()->lower_bound_area(Region(addr))))
00192     {
00193         Rm::Area &x = areas.value[num];
00194         x.start = r->first.start();
00195         x.end = r->first.end();
00196
00197         if (++num >= areas.max)
00198             break;
00199
00200         if (x.end >= rm()->max_addr())
00201             break;
00202
00203         addr = x.end + 1;
00204     }
00205     return num;
00206 }
00207
00208 private:
00209     static void pager_set_result(L4::Ipc::Opt<L4::Ipc::Snd_fpage> *fp,
00210                                 L4::Ipc::Snd_fpage const &f)
00211     { *fp = f; }
00212
00213     static void pager_set_result(L4::Ipc::Opt<L4::Ipc::Snd_fpage> *, ...)
00214     {}
00215
00216 public:
00217     long op_io_page_fault(L4::Io_pager::Rights, l4_fpage_t, l4_umword_t,
00218                          L4::Ipc::Opt<L4::Ipc::Snd_fpage> &)
00219     {
00220         // generate exception
00221         return -L4_ENOMEM;
00222     }
00223
00224     long op_page_fault(L4::Pager::Rights, l4_umword_t addr, l4_umword_t pc,
00225                       L4::Ipc::Opt<L4::Ipc::Snd_fpage> &fp)
00226     {
00227         Dbg(Dbg::Server).printf("page fault: %lx pc=%lx\n", addr, pc);
00228
00229         bool need_w = addr & 2;
00230         bool need_x = addr & 4;
00231
00232         typename DERIVED::Node n = rm()->find(addr);
00233
00234         if (!n || !n->second.memory())
00235         {
00236             Dbg(Dbg::Warn, "rm").printf("unhandled %s page fault at 0x%lx pc=0x%lx\n",
00237                                         need_w ? "write" :
00238                                         need_x ? "instruction" : "read", addr, pc);
00239
00240             // generate exception
00241             return -L4_ENOMEM;
00242         }
00243
00244         if (!(n->second.flags() & L4Re::Rm::F::W) && need_w)
00245         {
00246             Dbg(Dbg::Warn, "rm").printf("write page fault in readonly region at 0x%lx pc=0x%lx\n",
00247                                         addr, pc);
00248
00249             // generate exception
00250             return -L4_EACCESS;
00251         }
00252
00253         if (!(n->second.flags() & L4Re::Rm::F::X) && need_x)
00254         {
00255             Dbg(Dbg::Warn, "rm").printf("instruction page fault in non-exec region at 0x%lx pc=0x%lx\n",
00256                                         addr, pc);
00257
00258             // generate exception
00259             return -L4_EACCESS;
00260         }
00261
00262         typename DERIVED::Region_handler::Ops::Map_result map_res;
00263         if (int err = n->second.map(addr, n->first, need_w, &map_res))
00264         {
00265             Dbg(Dbg::Warn, "rm").printf("mapping for page fault failed with error %d at 0x%lx pc=0x%lx\n",
00266                                         err, addr, pc);
00267
00268             // generate exception
00269
00270

```

```

00271         return -L4_ENOMEM;
00272     }
00273
00274     pager_set_result(&fp, map_res);
00275     return L4_EOK;
00276 }
00277 };
00278
00279 }}

```

## 16.405 vcon\_svr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2011 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *                      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *                      Adam Lackorzysnski <adam@os.inf.tu-dresden.de>
00006  *                      economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU General Public License 2.
00010  * Please see the COPYING-GPL-2 file for details.
00011  *
00012  * As a special exception, you may use this file as part of a free software
00013  * library without restriction. Specifically, if other files instantiate
00014  * templates or use macros or inline functions from this file, or you compile
00015  * this file and link it with other files to produce an executable, this
00016  * file does not by itself cause the resulting executable to be covered by
00017  * the GNU General Public License. This exception does not however
00018  * invalidate any other reasons why the executable file might be covered by
00019  * the GNU General Public License.
00020  */
00021 #pragma once
00022
00023 #include <l4/sys/types.h>
00024 #include <l4/sys/vcon>
00025 #include <l4/sys/cxx/ipc_legacy>
00026 #include <l4/cxx/minmax>
00027
00028 namespace L4Re { namespace Util {
00029
00046 template< typename SVR >
00047 class Vcon_svr
00048 {
00049 public:
00050     L4_RPC_LEGACY_DISPATCH(L4::Vcon);
00051
00052     l4_msgtag_t op_dispatch(l4_utcb_t *utcb, l4_msgtag_t tag, L4::Vcon::Rights)
00053     {
00054         l4_msg_regst_t *m = l4_utcb_mr_u(utcb);
00055         L4::Opcode op = m->mr[0];
00056
00057         switch (op)
00058         {
00059             case L4_VCON_WRITE_OP:
00060                 if (tag.words() < 3)
00061                     return l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00062
00063                 this_vcon()->vcon_write(reinterpret_cast<char const *>(&m->mr[2]),
00064                                         m->mr[1]);
00065                 return l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00066
00067             case L4_VCON_SET_ATTR_OP:
00068                 {
00069                     if (tag.words() < 4)
00070                         return l4_msgtag(-L4_EINVAL, 0, 0, 0);
00071
00072                     auto attr = reinterpret_cast<l4_vcon_attr_t const *>(&m->mr[1]);
00073                     return l4_msgtag(this_vcon()->vcon_set_attr(attr), 0, 0, 0);
00074                 }
00075             case L4_VCON_GET_ATTR_OP:
00076                 {
00077                     auto attr = reinterpret_cast<l4_vcon_attr_t *>(&m->mr[1]);
00078                     return l4_msgtag(this_vcon()->vcon_get_attr(attr), 4, 0, 0);
00079                 }
00080
00081             default:
00082                 break;
00083         }
00084
00085         unsigned const max_size = sizeof(l4_utcb_mr()->mr) - sizeof(l4_utcb_mr()->mr[0]);
00086         char buf[max_size];
00087

```

```

00088     unsigned size = cxx::min<unsigned>(op > 16, max_size);
00089
00090     // Hmm, could we avoid the double copy here?
00091     l4_umword_t v = this_vcon()->vcon_read(buf, size);
00092     unsigned bytes = v & L4_VCON_READ_SIZE_MASK;
00093
00094     if (bytes <= size)
00095         v |= L4_VCON_READ_STAT_DONE;
00096
00097     m->mr[0] = v;
00098     __builtin_memcpy(&m->mr[1], buf, bytes);
00099
00100     return l4_msgtag(0, l4_bytes_to_mwords(bytes) + 1, 0, 0);
00101 }
00102
00103 unsigned vcon_read(char *buf, unsigned size) noexcept;
00104 void vcon_write(const char *buf, unsigned size) noexcept;
00105 int vcon_set_attr(l4_vcon_attr_t const *) noexcept
00106 { return -L4_EOK; }
00107 int vcon_get_attr(l4_vcon_attr_t *attr) noexcept
00108 {
00109     attr->l_flags = attr->o_flags = attr->i_flags = 0;
00110     return -L4_EOK;
00111 }
00112
00113 private:
00114     SVR const *this_vcon() const { return static_cast<SVR const *>(this); }
00115     SVR *this_vcon() { return static_cast<SVR *>(this); }
00116 };
00117
00118 }}

```

## 16.406 goos\_fb

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020 #pragma once
00021
00022 #include <l4/re/env>
00023 #include <l4/re/error_helper>
00024 #include <l4/re/namespace>
00025 #include <l4/re/rm>
00026 #include <l4/re/util/cap_alloc>
00027 #include <l4/re/util/env_ns>
00028 #include <l4/re/util/video/goos_fb>
00029 #include <l4/re/video/goos>
00030
00031 namespace L4Re { namespace Util { namespace Video {
00032
00033     class Goos_fb
00034     {
00035     private:
00036         L4::Cap<L4Re::Video::Goos> _goos;
00037         L4Re::Video::View _view;
00038         L4::Cap<L4Re::Dataspace> _buffer;
00039
00040         enum Flags
00041         {
00042             F_dyn_buffer = 0x01,
00043             F_dyn_view   = 0x02,
00044             F_dyn_goos    = 0x04,
00045         };
00046         unsigned _flags;
00047
00048         unsigned _buffer_index;
00049

```

```

00050 private:
00051     void init()
00052     {
00053         using namespace L4Re::Video;
00054         using L4Re::chksys;
00055         using L4Re::chkcapi;
00056
00057         Goos::Info gi;
00058         chksys(_goos->info(&gi), "requesting goos info");
00059
00060         if (gi.has_dynamic_views())
00061         {
00062             chksys(_goos->create_view(&_view), "creating dynamic goos view");
00063             _flags |= F_dyn_view;
00064         }
00065         else // we just assume view 0 to be our's and ignore other possible views
00066             _view = _goos->view(0);
00067
00068         View::Info vi;
00069         chksys(_view.info(&vi), "requesting goos view information");
00070
00071         _buffer = chkcapi(cap_alloc.alloc<L4Re::Dataspace>()),
00072             "allocating goos buffer cap");
00073
00074         if (vi.has_static_buffer())
00075             chksys(_goos->get_static_buffer(vi.buffer_index, _buffer),
00076                 "requesting static goos buffer");
00077         else
00078         {
00079             unsigned long buffer_sz = gi.pixel_info.bytes_per_pixel() * gi.width
00080                                     * gi.height;
00081             _buffer_index = chksys(_goos->create_buffer(buffer_sz, _buffer),
00082                 "allocating goos buffer");
00083             _flags |= F_dyn_buffer;
00084
00085             // use the allocated buffer, at offset 0
00086             vi.buffer_index = _buffer_index;
00087             vi.buffer_offset = 0;
00088             vi.pixel_info = gi.pixel_info;
00089             vi.bytes_per_line = gi.width * gi.pixel_info.bytes_per_pixel();
00090
00091             // we want a fullscreen view
00092             vi.xpos = 0;
00093             vi.ypos = 0;
00094             vi.width = gi.width;
00095             vi.height = gi.height;
00096
00097             chksys(_view.set_info(vi), "setting up dynamic view");
00098             chksys(_view.push_top(), "bringing view to top");
00099         }
00100     }
00101
00102     Goos_fb(Goos_fb const &);
00103     void operator = (Goos_fb const &);
00104
00105 public:
00106     Goos_fb()
00107     : _goos(L4_INVALID_CAP), _buffer(L4_INVALID_CAP), _flags(0), _buffer_index(0)
00108     {}
00109
00110     explicit Goos_fb(L4::Cap<L4Re::Video::Goos> goos)
00111     : _goos(goos), _buffer(L4_INVALID_CAP), _flags(0)
00112     { init(); }
00113
00114     explicit Goos_fb(char const *name)
00115     : _goos(L4_INVALID_CAP), _buffer(L4_INVALID_CAP), _flags(0)
00116     { setup(name); }
00117
00118     void setup(L4::Cap<L4Re::Video::Goos> goos)
00119     {
00120         _goos = goos;
00121         init();
00122     }
00123
00124     void setup(char const *name)
00125     {
00126         Env_ns ns;
00127         // _goos = chkcapi(cap_alloc.alloc<L4Re::Video::Goos>()), "allocating goos cap");
00128         _goos = chkcapi(ns.query<L4Re::Video::Goos>(name), "requesting goos cap", 0);
00129         _flags |= F_dyn_goos;
00130
00131         // chksys(L4Re::Env::env()->names()->query(name, _goos), "requesting goos service");
00132         init();
00133     }
00134
00135     ~Goos_fb()
00136     {}

```

```

00137     if (!_goos.is_valid())
00138         return;
00139
00140     if (_flags & F_dyn_view)
00141         _goos->delete_view(_view);
00142
00143     if (_flags & F_dyn_buffer)
00144         _goos->delete_buffer(_buffer_index);
00145
00146     if (_buffer.is_valid())
00147         cap_alloc.free(_buffer);
00148
00149     if (_flags & F_dyn_goos)
00150         cap_alloc.free(_goos);
00151 }
00152
00153 int view_info(L4Re::Video::View::Info *info)
00154 { return _view.info(info); }
00155
00156 L4Re::Video::View const *view() const { return &_view; }
00157 L4Re::Video::View *view() { return &_view; }
00158
00159 L4::Cap<L4Re::Dataspace> buffer() const { return _buffer; }
00160 void *attach_buffer()
00161 {
00162     void *fb_addr = 0;
00163     L4Re::chkcap(_goos);
00164     L4Re::chksys(L4Re::Env::env()->rm()
00165         ->attach(&fb_addr, _buffer->size(),
00166             L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW, _buffer,
00167             0, L4_SUPERPAGESHIFT), "attaching frame-buffer memory");
00168     return fb_addr;
00169 }
00170
00171 int refresh(int x, int y, int w, int h)
00172 { return _view.refresh(x, y, w, h); }
00173
00174 L4::Cap<L4Re::Video::Goos> goos() const { return _goos; }
00175 };
00176 }}}

```

## 16.407 goos\_svr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020
00021 #pragma once
00022
00023 #include <l4/re/dataspace>
00024 #include <l4/re/video/goos>
00025 #include <l4/re/video/goos-sys.h>
00026
00027 #include <l4/sys/capability>
00028 #include <l4/sys/cxx/ipc_legacy>
00029
00030 namespace L4Re { namespace Util { namespace Video {
00031
00036 class Goos_svr
00037 {
00038     typedef L4Re::Video::Goos::Rights Rights;
00039 protected:
00041     L4::Cap<L4Re::Dataspace> _fb_ds;
00043     L4Re::Video::Goos::Info _screen_info;
00045     L4Re::Video::View::Info _view_info;
00046
00047 public:

```



```

00048 L4_RPC_LEGACY_DISPATCH(L4Re::Video::Goos);
00053 L4::Cap<L4Re::Dataspace> get_fb() const { return _fb_ds; }
00054
00059 L4Re::Video::Goos::Info const *screen_info() const { return &_screen_info; }
00060
00065 L4Re::Video::View::Info const *view_info() const { return &_view_info; }
00066
00077 virtual int refresh(int x, int y, int w, int h)
00078 { (void)x; (void)y; (void)w; (void)h; return -L4_ENOSYS; }
00079
00080
00089 void init_infos()
00090 {
00091     using L4Re::Video::View;
00092
00093     _view_info.flags = View::F_none;
00094
00095     _view_info.view_index = 0;
00096     _view_info.xpos = 0;
00097     _view_info.ypos = 0;
00098     _view_info.width = _screen_info.width;
00099     _view_info.height = _screen_info.height;
00100     _view_info.pixel_info = _screen_info.pixel_info;
00101     _view_info.buffer_index = 0;
00102 }
00103
00107 virtual ~Goos_svr() {}
00108
00109 long op_view_info(Rights, unsigned idx, L4Re::Video::View::Info &info)
00110 {
00111     if (idx != 0)
00112         return -L4_ERANGE;
00113
00114     info = _view_info;
00115     return L4_EOK;
00116 }
00117
00118 long op_info(Rights, L4Re::Video::Goos::Info &info)
00119 {
00120     info = _screen_info;
00121     return L4_EOK;
00122 }
00123
00124 long op_get_static_buffer(Rights, unsigned idx,
00125                           L4::Ipc::Cap<L4Re::Dataspace> &ds)
00126 {
00127     if (idx != 0)
00128         return -L4_ERANGE;
00129
00130     ds = L4::Ipc::Cap<L4Re::Dataspace>(_fb_ds, L4_CAP_FPAGE_RW);
00131     return L4_EOK;
00132 }
00133
00134 long op_refresh(Rights, int x, int y, int w, int h)
00135 { return refresh(x, y, w, h); }
00136
00137 long op_view_refresh(Rights, unsigned idx, int x, int y, int w, int h)
00138 {
00139     if (idx != 0)
00140         return -L4_ERANGE;
00141
00142     return refresh(x, y, w, h);
00143 }
00144
00145 long op_set_view_info(Rights, unsigned, L4Re::Video::View::Info)
00146 { return -L4_ENOSYS; }
00147
00148 long op_view_stack(Rights, unsigned, unsigned, bool)
00149 { return -L4_ENOSYS; }
00150
00151 long op_delete_view(Rights, unsigned)
00152 { return -L4_ENOSYS; }
00153
00154 long op_create_view(Rights)
00155 { return -L4_ENOSYS; }
00156
00157 long op_create_buffer(Rights, unsigned long,
00158                      L4::Ipc::Cap<L4Re::Dataspace> &)
00159 { return -L4_ENOSYS; }
00160
00161 long op_delete_buffer(Rights, unsigned)
00162 { return -L4_ENOSYS; }
00163 };
00164
00165
00166 }}}

```

## 16.408 colors

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020
00021 #pragma once
00022
00023 #include <l4/sys/compiler.h>
00024 #include <l4/cxx/minmax>
00025
00026 namespace L4Re { namespace Video {
00027
00032 class L4_EXPORT Color_component
00033 {
00034 private:
00035     unsigned char _bits;
00036     unsigned char _shift;
00037
00038 public:
00040     Color_component() : _bits(0), _shift(0) {}
00041
00047     Color_component(unsigned char bits, unsigned char shift)
00048     : _bits(bits), _shift(shift) {}
00049
00054     unsigned char size() const { return _bits; }
00055
00060     unsigned char shift() const { return _shift; }
00061
00066     bool operator == (Color_component const &o) const
00067     { return _shift == o._shift && _bits == o._bits; }
00068
00074     int get(unsigned long v) const
00075     {
00076         return ((v > _shift) & ~(~0UL << _bits)) << (16UL - _bits);
00077     }
00078
00084     long unsigned set(int v) const
00085     { return (static_cast<unsigned long>(v) > (16UL - _bits)) << _shift; }
00086
00091     template< typename OUT >
00092     void dump(OUT &s) const
00093     {
00094         s.printf("%d(%d)", static_cast<int>(size()), static_cast<int>(shift()));
00095     }
00096 } __attribute__((packed));
00097
00105 class L4_EXPORT Pixel_info
00106 {
00107 private:
00108     Color_component _r, _g, _b, _a;
00109     unsigned char _bpp;
00110
00111 public:
00116     Color_component const &r() const { return _r; }
00117
00122     Color_component const &g() const { return _g; }
00123
00128     Color_component const &b() const { return _b; }
00129
00134     Color_component const &a() const { return _a; }
00135
00142     Color_component const padding() const
00143     {
00144         unsigned char top_bit = cxx::max<unsigned char>(_r.size() + _r.shift(),
00145                                                         _g.size() + _g.shift());
00146         top_bit = cxx::max<unsigned char>(top_bit, _b.size() + _b.shift());
00147         top_bit = cxx::max<unsigned char>(top_bit, _a.size() + _a.shift());
00148
00149         unsigned char bits = _bpp * 8;
00150

```

```

00151     if (top_bit < bits)
00152         return Color_component(bits - top_bit, top_bit);
00153
00154     return Color_component(0, 0);
00155 }
00156
00161 unsigned char bytes_per_pixel() const { return _bpp; }
00162
00167 unsigned char bits_per_pixel() const
00168 { return _r.size() + _g.size() + _b.size() + _a.size(); }
00169
00174 bool has_alpha() const { return _a.size() > 0; }
00175
00180 void r(Color_component const &c) { _r = c; }
00181
00186 void g(Color_component const &c) { _g = c; }
00187
00192 void b(Color_component const &c) { _b = c; }
00193
00198 void a(Color_component const &c) { _a = c; }
00199
00204 void bytes_per_pixel(unsigned char bpp) { _bpp = bpp; }
00205
00209 Pixel_info() : _bpp(0) {};
00210
00223 Pixel_info(unsigned char bpp, char r, char rs, char g, char gs,
00224             char b, char bs, char a = 0, char as = 0)
00225 : _r(r, rs), _g(g, gs), _b(b, bs), _a(a, as), _bpp(bpp)
00226 {}
00227
00234 template<typename VBI>
00235 explicit Pixel_info(VBI const *vbi)
00236 : _r(vbi->red_mask_size, vbi->red_field_position),
00237   _g(vbi->green_mask_size, vbi->green_field_position),
00238   _b(vbi->blue_mask_size, vbi->blue_field_position),
00239   _bpp((vbi->bits_per_pixel + 7) / 8)
00240 {}
00241
00247 bool operator == (Pixel_info const &o) const
00248 {
00249     return _r == o._r && _g == o._g && _b == o._b && _a == o._a && _bpp == o._bpp;
00250 }
00251
00256 template< typename OUT >
00257 void dump(OUT &s) const
00258 {
00259     s.printf("RGBA(%d):%d(%d):%d(%d):%d(%d):%d(%d)",
00260             static_cast<int>(bytes_per_pixel()),
00261             static_cast<int>(r().size()), static_cast<int>(r().shift()),
00262             static_cast<int>(g().size()), static_cast<int>(g().shift()),
00263             static_cast<int>(b().size()), static_cast<int>(b().shift()),
00264             static_cast<int>(a().size()), static_cast<int>(a().shift()));
00265 }
00266 };
00267
00268
00269 }}
00270
00271

```

## 16.409 goos

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020 #pragma once
00021

```

```

00022 #include <l4/sys/capability>
00023 #include <l4/re/dataspace>
00024 #include <l4/re/video/colors>
00025 #include <l4/sys/cxx/ipc_iface>
00026
00027 namespace L4Re { namespace Video {
00028
00029 class L4_EXPORT Goos;
00030
00042 class L4_EXPORT View
00043 {
00044 private:
00045     friend class Goos;
00046
00047     L4::Cap<Goos> _goos;
00048     unsigned _view_idx;
00049
00050     View(l4_cap_idx_t goos, unsigned idx)
00051     : _goos(goos), _view_idx(_goos.is_valid() ? idx : ~0U) {}
00052
00053     unsigned view_index() const noexcept
00054     { return _goos.is_valid() ? _view_idx : ~0U; }
00055
00056 public:
00057     View() : _goos(L4::Cap<Goos>::Invalid), _view_idx(~0U) {}
00058
00062     enum Flags
00063     {
00064         F_none                = 0x00,
00065         F_set_buffer           = 0x01,
00066         F_set_buffer_offset    = 0x02,
00067         F_set_bytes_per_line   = 0x04,
00068         F_set_pixel            = 0x08,
00069         F_set_position         = 0x10,
00070         F_dyn_allocated        = 0x20,
00071         F_set_background       = 0x40,
00072         F_set_flags            = 0x80,
00073
00075         F_fully_dynamic        = F_set_buffer | F_set_buffer_offset | F_set_bytes_per_line
00076                               | F_set_pixel | F_set_position | F_dyn_allocated,
00077     };
00078
00085     enum V_flags
00086     {
00087         F_above                = 0x1000,
00088         F_flags_mask           = 0xff000,
00089     };
00090
00094     struct Info
00095     {
00096         unsigned flags          = 0;
00097         unsigned view_index     = 0;
00098
00099         unsigned long xpos      = 0;
00100         unsigned long ypos      = 0;
00101         unsigned long width     = 0;
00102         unsigned long height    = 0;
00103         unsigned long buffer_offset = 0;
00104         unsigned long bytes_per_line = 0;
00105         Pixel_info pixel_info;
00106         unsigned buffer_index    = 0;
00107
00109         bool has_static_buffer() const { return !(flags & F_set_buffer); }
00111         bool has_static_buffer_offset() const { return !(flags & F_set_buffer_offset); }
00112
00114         bool has_set_buffer() const { return flags & F_set_buffer; }
00116         bool has_set_buffer_offset() const { return flags & F_set_buffer_offset; }
00118         bool has_set_bytes_per_line() const { return flags & F_set_bytes_per_line; }
00120         bool has_set_pixel() const { return flags & F_set_pixel; }
00122         bool has_set_position() const { return flags & F_set_position; }
00123
00125     template< typename OUT >
00126     void dump(OUT &s) const
00127     {
00128         s.printf("View::Info:\n"
00129                 "  flags: %x\n"
00130                 "  size: %ldx%ld\n"
00131                 "  pos: %ldx%ld\n"
00132                 "  bytes_per_line: %ld\n"
00133                 "  buffer_offset: %lx\n"
00134                 "  ",
00135                 flags, width, height, xpos, ypos,
00136                 bytes_per_line, buffer_offset);
00137         pixel_info.dump(s);
00138         s.printf("\n");
00139     }
00140 };

```

```

00141
00149     int info(Info *info) const noexcept;
00150
00161     int set_info(Info const &info) const noexcept;
00162
00174     int set_viewport(int scr_x, int scr_y, int w, int h, unsigned long buf_offset) const noexcept;
00175
00185     int stack(View const &pivot, bool behind = true) const noexcept;
00186
00188     int push_top() const noexcept
00189     { return stack(View(), true); }
00190
00192     int push_bottom() const noexcept
00193     { return stack(View(), false); }
00194
00205     int refresh(int x, int y, int w, int h) const noexcept;
00206
00208     bool valid() const { return _goos.is_valid(); }
00209 };
00210
00211
00226 class L4_EXPORT Goos :
00227     public L4::Kobject_t<Goos, L4::Kobject, L4RE_PROTO_GOOS>
00228 {
00229 public:
00231     enum Flags
00232     {
00233         F_auto_refresh      = 0x01,
00234         F_pointer           = 0x02,
00235         F_dynamic_views     = 0x04,
00236         F_dynamic_buffers   = 0x08,
00237     };
00238
00240     struct Info
00241     {
00242         unsigned long width;
00243         unsigned long height;
00244         unsigned flags;
00245         unsigned num_static_views;
00246         unsigned num_static_buffers;
00247         Pixel_info pixel_info;
00248
00251         bool auto_refresh() const { return flags & F_auto_refresh; }
00253         bool has_pointer() const { return flags & F_pointer; }
00255         bool has_dynamic_views() const { return flags & F_dynamic_views; }
00257         bool has_dynamic_buffers() const { return flags & F_dynamic_buffers; }
00258
00259         Info()
00260             : width(0), height(0), flags(0), num_static_views(0),
00261               num_static_buffers(0) {}
00262     };
00263
00271     L4_INLINE_RPC(long, info, (Info *info));
00272
00281     L4_RPC(long, get_static_buffer, (unsigned idx,
00282                                     L4::Ipc::Out<L4::Cap<L4Re::Dataspace> > rbuf));
00283
00292     L4_RPC(long, create_buffer, (unsigned long size,
00293                                 L4::Ipc::Out<L4::Cap<L4Re::Dataspace> > rbuf));
00294
00302     L4_INLINE_RPC(long, delete_buffer, (unsigned idx));
00303
00304     // Use a wrapper for this RPC as we encapsulate the View
00305     L4_INLINE_RPC_NF(long, create_view, ());
00306
00315     int create_view(View *view, l4_utcb_t *utcb = l4_utcb()) const noexcept
00316     {
00317         long r = create_view_t::call(c(), utcb);
00318         if (r < 0)
00319             return r;
00320         *view = View(cap(), r);
00321         return r;
00322     }
00323
00324     // Use a wrapper as Views are encapsulated
00325     L4_INLINE_RPC_NF(long, delete_view, (unsigned index));
00326
00335     int delete_view(View const &v, l4_utcb_t *utcb = l4_utcb()) const noexcept
00336     {
00337         return delete_view_t::call(c(), v._view_idx, utcb);
00338     }
00339
00345     View view(unsigned index) const noexcept;
00346
00350     L4_INLINE_RPC(long, refresh, (int x, int y, int w, int h));
00351
00352     // those are used by the View

```

```

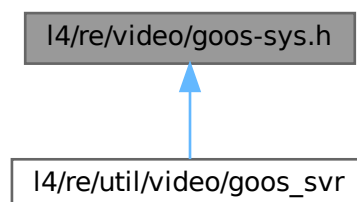
00353 L4_INLINE_RPC(long, view_info, (unsigned index, View::Info *info));
00354 L4_INLINE_RPC(long, set_view_info, (unsigned index, View::Info const &info));
00355 L4_INLINE_RPC(long, view_stack, (unsigned index, unsigned pivot, bool behind));
00356 L4_INLINE_RPC(long, view_refresh, (unsigned index, int x, int y, int w, int h));
00357
00358 typedef L4::Typeid::Rpc<
00359     info_t, get_static_buffer_t, create_buffer_t, create_view_t, delete_buffer_t,
00360     delete_view_t, view_info_t, set_view_info_t, view_stack_t, view_refresh_t,
00361     refresh_t
00362 > Rpc<
00363 >;
00364
00365 inline View
00366 Goos::view(unsigned index) const noexcept
00367 { return View(cap(), index); }
00368
00369 inline int
00370 View::info(Info *info) const noexcept
00371 { return _goos->view_info(_view_idx, info); }
00372
00373 inline int
00374 View::set_info(Info const &info) const noexcept
00375 { return _goos->set_view_info(_view_idx, info); }
00376
00377 inline int
00378 View::stack(View const &pivot, bool behind) const noexcept
00379 { return _goos->view_stack(_view_idx, pivot._view_idx, behind); }
00380
00381 inline int
00382 View::refresh(int x, int y, int w, int h) const noexcept
00383 { return _goos->view_refresh(_view_idx, x, y, w, h); }
00384
00385 inline int
00386 View::set_viewport(int scr_x, int scr_y, int w, int h,
00387                    unsigned long buf_offset) const noexcept
00388 {
00389     Info i;
00390     i.flags = F_set_buffer_offset | F_set_position;
00391     i.buffer_offset = buf_offset;
00392     i.buffer_index = 0;
00393     i.view_index = 0;
00394     i.bytes_per_line = 0;
00395     i.pixel_info = Pixel_info();
00396     i.xpos = scr_x;
00397     i.ypos = scr_y;
00398     i.width = w;
00399     i.height = h;
00400     return set_info(i);
00401 }
00402
00403 {}

```

## 16.410 I4/re/video/goos-sys.h File Reference

Goos protocol definition.

This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*

## Enumerations

- enum [L4Re::Video::Goos\\_::Opcodes](#)  
*Frame buffer communication-protocol opcodes.*

### 16.410.1 Detailed Description

Goos protocol definition.

Definition in file [goos-sys.h](#).

## 16.411 goos-sys.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025
00026 namespace L4Re { namespace Video {
00027     namespace Goos_
00028     {
00034         enum Opcodes
00035         {
00036             Info, Get_buffer, Create_buffer, Create_view,
00037             Delete_buffer, Delete_view,
00038             View_info, View_set_info, View_stack, View_refresh,
00039             Screen_refresh
00040         };
00041     };
00042 }}
```

## 16.412 view

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020 #pragma once
00021
00022 #include <l4/re/video/goos>
00023

```

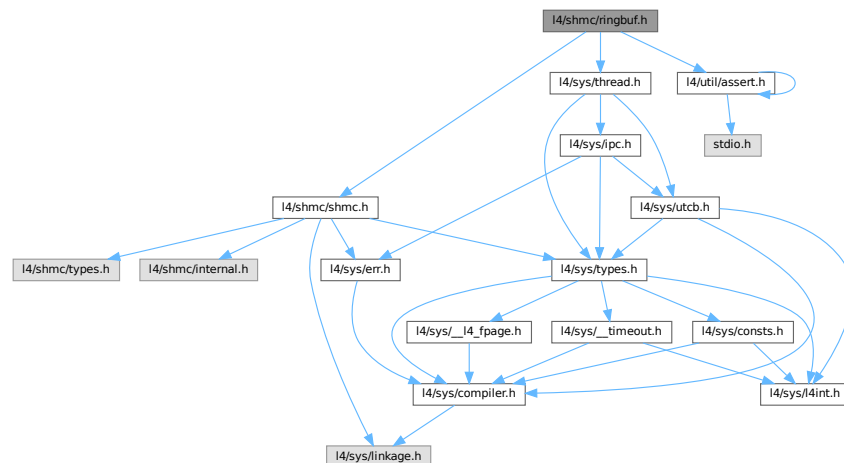
## 16.413 I4/shmc/ringbuf.h File Reference

```

#include <l4/shmc/shmc.h>
#include <l4/util/assert.h>
#include <l4/sys/thread.h>

```

Include dependency graph for ringbuf.h:



### Data Structures

- struct [I4shmc\\_ringbuf\\_head\\_t](#)  
*Head field of a ring buffer.*
- struct [I4shmc\\_ringbuf\\_t](#)  
*Ring buffer.*



## Macros

- `#define L4SHMC_RINGBUF_HEAD(ringbuf) ((l4shmc_ringbuf_head_t*)((ringbuf)->_addr))`  
*Get ring buffer head pointer.*
- `#define L4SHMC_RINGBUF_DATA(ringbuf) (L4SHMC_RINGBUF_HEAD(ringbuf)->data)`  
*Get ring buffer data pointer.*
- `#define L4SHMC_RINGBUF_DATA_SIZE(ringbuf) ((ringbuf)->_size - sizeof(l4shmc_ringbuf_head_t))`  
*Get size of data area.*

## Functions

- `int l4shmc_rb_init_buffer (l4shmc_ringbuf_t *buf, l4shmc_area_t *area, char const *chunk_name, char const *signal_name, unsigned size)`  
*Initialize a ring buffer by creating an SHMC chunk and the corresponding signals.*
- `void l4shmc_rb_deinit_buffer (l4shmc_ringbuf_t *buf)`  
*De-init a ring buffer.*
- `int l4shmc_rb_attach_sender (l4shmc_ringbuf_t *buf, char const *signal_name, l4_cap_idx_t owner)`  
*Attach to sender signal of a ring buffer.*
- `char * l4shmc_rb_sender_alloc_packet (l4shmc_ringbuf_head_t *head, unsigned psize)`  
*Allocate a packet of a given size within the ring buffer.*
- `void l4shmc_rb_sender_put_data (l4shmc_ringbuf_t *buf, char *addr, char *data, unsigned dsize)`  
*Copy data into a previously allocated packet.*
- `int l4shmc_rb_sender_next_copy_in (l4shmc_ringbuf_t *buf, char *data, unsigned size, int block_if_necessary)`  
*Copy in packet from an external data source.*
- `void l4shmc_rb_sender_commit_packet (l4shmc_ringbuf_t *buf)`  
*Tell the consumer that new data is available.*
- `int l4shmc_rb_init_receiver (l4shmc_ringbuf_t *buf, l4shmc_area_t *area, char const *chunk_name, char const *signal_name)`  
*Initialize receive buffer.*
- `void l4shmc_rb_attach_receiver (l4shmc_ringbuf_t *buf, l4_cap_idx_t owner)`  
*Attach to receiver signal of a ring buffer.*
- `int l4shmc_rb_receiver_wait_for_data (l4shmc_ringbuf_t *buf, int blocking)`  
*Check if (and optionally block until) new data is ready.*
- `int l4shmc_rb_receiver_copy_out (l4shmc_ringbuf_head_t *head, char *target, unsigned *tsize)`  
*Copy data out of the buffer.*
- `void l4shmc_rb_receiver_notify_done (l4shmc_ringbuf_t *buf)`  
*Notify producer that space is available.*
- `int l4shmc_rb_receiver_read_next_size (l4shmc_ringbuf_head_t *head)`  
*Have a look at the ring buffer and see which size the next packet to be read has.*

## 16.413.1 Function Documentation

### 16.413.1.1 l4shmc\_rb\_attach\_receiver()

```
void l4shmc_rb_attach_receiver (
    l4shmc_ringbuf_t * buf,
    l4_cap_idx_t owner )
```

Attach to receiver signal of a ring buffer.

Attach owner to the receiver-side signal of a ring buffer, which is triggered whenever new data has been produced.

This is split from initialization, because you may not know the owner cap when initializing the buffer.

## Parameters

<i>buf</i>	pointer to ring buffer struct
<i>owner</i>	owner thread

**16.413.1.2 l4shmc\_rb\_attach\_sender()**

```
int l4shmc_rb_attach_sender (
    l4shmc_ringbuf_t * buf,
    char const * signal_name,
    l4_cap_idx_t owner )
```

Attach to sender signal of a ring buffer.

Attach owner to the sender-side signal of a ring buffer, which is triggered whenever new space has been freed in the buffer for the sender to write to.

This is split from initialization, because you may not know the owner cap when initializing the buffer.

## Parameters

<i>buf</i>	pointer to ring buffer struct
<i>signal_name</i>	signal base name
<i>owner</i>	owner thread

## Returns

0 on success, error otherwise

**16.413.1.3 l4shmc\_rb\_deinit\_buffer()**

```
void l4shmc_rb_deinit_buffer (
    l4shmc_ringbuf_t * buf )
```

De-init a ring buffer.

## Parameters

<i>buf</i>	pointer to ring buffer struct
------------	-------------------------------

**16.413.1.4 l4shmc\_rb\_init\_buffer()**

```
int l4shmc_rb_init_buffer (
    l4shmc_ringbuf_t * buf,
    l4shmc_area_t * area,
    char const * chunk_name,
```

```
char const * signal_name,  
unsigned size )
```

Initialize a ring buffer by creating an SHMC chunk and the corresponding signals.

This needs to be done by one of the participating parties when setting up communication channel.

#### Precondition

area has been attached using [l4shmc\\_attach\(\)](#).

#### Parameters

<i>buf</i>	pointer to ring buffer struct
<i>area</i>	pointer to SHMC area
<i>chunk_name</i>	name of SHMC chunk to create in area
<i>signal_name</i>	base name for SHMC signals to create
<i>size</i>	chunk size

#### Returns

0 on success, error otherwise

#### 16.413.1.5 l4shmc\_rb\_init\_receiver()

```
int l4shmc_rb_init_receiver (  
    l4shmc_ringbuf_t * buf,  
    l4shmc_area_t * area,  
    char const * chunk_name,  
    char const * signal_name )
```

Initialize receive buffer.

Initialize the receiver-side of a ring buffer. This requires the underlying SHMC chunk and the corresponding signals to be valid already (read: to be initialized by the sender).

#### Precondition

chunk & signals have been created and initialized by the sender side

#### Parameters

<i>buf</i>	pointer to ring buffer struct
<i>area</i>	pointer to SHMC area
<i>chunk_name</i>	name of SHMC chunk to create in area
<i>signal_name</i>	base name for SHMC signals to create

**Returns**

0 on success, error otherwise

**16.413.1.6 l4shmc\_rb\_receiver\_copy\_out()**

```
int l4shmc_rb_receiver_copy_out (
    l4shmc_ringbuf_head_t * head,
    char * target,
    unsigned * tsize )
```

Copy data out of the buffer.

**Parameters**

	<i>head</i>	ring buffer head pointer
	<i>target</i>	valid target buffer
<i>in, out</i>	<i>tsize</i>	size of target buffer (must be $\geq$ packet size!); contains the real data size

**Returns**

0 on success, negative error otherwise

**16.413.1.7 l4shmc\_rb\_receiver\_notify\_done()**

```
void l4shmc_rb_receiver_notify_done (
    l4shmc_ringbuf_t * buf )
```

Notify producer that space is available.

**Parameters**

<i>buf</i>	pointer to ring buffer struct
------------	-------------------------------

**16.413.1.8 l4shmc\_rb\_receiver\_read\_next\_size()**

```
int l4shmc_rb_receiver_read_next_size (
    l4shmc_ringbuf_head_t * head )
```

Have a look at the ring buffer and see which size the next packet to be read has.

Does not modify anything.

**Returns**

size of next buffer or -1 if no data available

### 16.413.1.9 l4shmc\_rb\_receiver\_wait\_for\_data()

```
int l4shmc_rb_receiver_wait_for_data (
    l4shmc_ringbuf_t * buf,
    int blocking )
```

Check if (and optionally block until) new data is ready.

#### Parameters

<i>buf</i>	pointer to ring buffer struct
<i>blocking</i>	block if data is not available immediately

Returns immediately, if data is available.

#### Returns

0 success, data available, != 0 otherwise

### 16.413.1.10 l4shmc\_rb\_sender\_alloc\_packet()

```
char * l4shmc_rb_sender_alloc_packet (
    l4shmc_ringbuf_head_t * head,
    unsigned psize )
```

Allocate a packet of a given size within the ring buffer.

This packet may wrap around at the end of the buffer. Users need to be aware of that.

#### Parameters

<i>head</i>	ring buffer head pointer
<i>psize</i>	packet size

#### Returns

valid address on success

#### Return values

<i>NULL</i>	if not enough space available
-------------	-------------------------------

### 16.413.1.11 l4shmc\_rb\_sender\_commit\_packet()

```
void l4shmc_rb_sender_commit_packet (
    l4shmc_ringbuf_t * buf )
```

Tell the consumer that new data is available.

## Parameters

<i>buf</i>	pointer to ring buffer struct
------------	-------------------------------

**16.413.1.12 l4shmc\_rb\_sender\_next\_copy\_in()**

```
int l4shmc_rb_sender_next_copy_in (
    l4shmc_ringbuf_t * buf,
    char * data,
    unsigned size,
    int block_if_necessary )
```

Copy in packet from an external data source.

This is the function you'll want to use. Just pass it a buffer pointer and let the lib do the work.

## Parameters

<i>buf</i>	pointer to ring buffer struct
<i>data</i>	valid buffer
<i>size</i>	data size
<i>block_if_necessary</i>	bool: block if buffer currently full

## Return values

0	on success
-L4_ENOMEM	if block == false and no space available

**16.413.1.13 l4shmc\_rb\_sender\_put\_data()**

```
void l4shmc_rb_sender_put_data (
    l4shmc_ringbuf_t * buf,
    char * addr,
    char * data,
    unsigned dsize )
```

Copy data into a previously allocated packet.

This function is wrap-around aware.

## Parameters

<i>buf</i>	pointer to ring buffer struct
<i>addr</i>	valid destination (allocate with alloc_packet())
<i>data</i>	data source
<i>dsize</i>	data size

## 16.414 ringbuf.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2010 Björn Döbel <doebel@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  * This file is part of TUD:OS and distributed under the terms of the
00005  * GNU Lesser General Public License 2.1.
00006  * Please see the COPYING-LGPL-2.1 file for details.
00007  */
00008
00012 #pragma once
00013
00014 #include <l4/shmc/shmc.h>
00015 #include <l4/util/assert.h>
00016 #include <l4/sys/thread.h>
00017
00018 __BEGIN_DECLS
00019
00046 /*
00047  * Turn on ringbuf poisoning. This will add magic values to the ringbuf
00048  * header as well as each packet header and check that these values are
00049  * valid all the time.
00050  */
00051 #define L4SHMC_RINGBUF_POISONING 1
00052
00058 typedef struct
00059 {
00060     volatile l4_uint32_t lock;
00061     unsigned data_size;
00062     #if L4SHMC_RINGBUF_POISONING
00063     char magic1;
00064     #endif
00065     unsigned next_read;
00066     unsigned next_write;
00067     #if L4SHMC_RINGBUF_POISONING
00068     char magic2;
00069     #endif
00070     unsigned bytes_filled;
00071     unsigned sender_waits;
00072     #if L4SHMC_RINGBUF_POISONING
00073     char magic3;
00074     #endif
00075     char data[];
00076 } l4shmc_ringbuf_head_t;
00077
00078
00084 typedef struct
00085 {
00086     l4shmc_area_t *_area;
00087     l4_cap_idx_t _owner;
00088     l4shmc_chunk_t _chunk;
00089     unsigned _size;
00090     char *_chunkname;
00091     char *_signame;
00092     l4shmc_ringbuf_head_t *_addr;
00093     l4shmc_signal_t _signal_full;
00094     l4shmc_signal_t _signal_empty;
00095 } l4shmc_ringbuf_t;
00096
00097
00104 #define L4SHMC_RINGBUF_HEAD(ringbuf) ((l4shmc_ringbuf_head_t*) ((ringbuf)->_addr))
00105
00106
00113 #define L4SHMC_RINGBUF_DATA(ringbuf) (L4SHMC_RINGBUF_HEAD(ringbuf)->data)
00114
00115
00122 #define L4SHMC_RINGBUF_DATA_SIZE(ringbuf) ((ringbuf)->_size - sizeof(l4shmc_ringbuf_head_t))
00123
00124 enum lock_content
00125 {
00126     lock_cont_min = 4,
00127     locked = 5,
00128     unlocked = 6,
00129     lock_cont_max = 7,
00130 };
00131
00132 static L4_CV inline void l4shmc_rb_lock(l4shmc_ringbuf_head_t *head)
00133 {
00134     ASSERT_NOT_NULL(head);
00135     ASSERT_ASSERT(head->lock > lock_cont_min);
00136     ASSERT_ASSERT(head->lock < lock_cont_max);
00137
00138     while (!l4util_cmpxchg32(&head->lock, unlocked, locked))
00139         l4_thread_yield();

```

```

00140 }
00141
00142
00143 static L4_CV inline void l4shmc_rb_unlock(l4shmc_ringbuf_head_t *head)
00144 {
00145     ASSERT_NOT_NULL(head);
00146     ASSERT_ASSERT(head->lock > lock_cont_min);
00147     ASSERT_ASSERT(head->lock < lock_cont_max);
00148
00149     head->lock = unlocked;
00150 }
00151
00152 /*****
00153  * Initialization *
00154  *****/
00155
00172 L4_CV int l4shmc_rb_init_buffer(l4shmc_ringbuf_t *buf, l4shmc_area_t *area,
00173                                char const *chunk_name,
00174                                char const *signal_name, unsigned size);
00175
00181 L4_CV void l4shmc_rb_deinit_buffer(l4shmc_ringbuf_t *buf);
00182
00183
00184
00185 /*****
00186  * RINGBUF SENDER *
00187  *****/
00188
00205 L4_CV int l4shmc_rb_attach_sender(l4shmc_ringbuf_t *buf, char const *signal_name,
00206                                    l4_cap_idx_t owner);
00207
00208
00221 L4_CV char *l4shmc_rb_sender_alloc_packet(l4shmc_ringbuf_head_t *head,
00222                                             unsigned psize);
00223
00224
00235 L4_CV void l4shmc_rb_sender_put_data(l4shmc_ringbuf_t *buf, char *addr,
00236                                       char *data, unsigned dsize);
00237
00238
00253 L4_CV int l4shmc_rb_sender_next_copy_in(l4shmc_ringbuf_t *buf, char *data,
00254                                           unsigned size, int block_if_necessary);
00255
00256
00262 L4_CV void l4shmc_rb_sender_commit_packet(l4shmc_ringbuf_t *buf);
00263
00264
00265 /*****
00266  * RINGBUF RECEIVER *
00267  *****/
00268
00285 L4_CV int l4shmc_rb_init_receiver(l4shmc_ringbuf_t *buf, l4shmc_area_t *area,
00286                                    char const *chunk_name,
00287                                    char const *signal_name);
00288
00289
00302 L4_CV void l4shmc_rb_attach_receiver(l4shmc_ringbuf_t *buf, l4_cap_idx_t owner);
00303
00304
00315 L4_CV int l4shmc_rb_receiver_wait_for_data(l4shmc_ringbuf_t *buf, int blocking);
00316
00317
00327 L4_CV int l4shmc_rb_receiver_copy_out(l4shmc_ringbuf_head_t *head, char *target,
00328                                         unsigned *tsize);
00329
00330
00336 L4_CV void l4shmc_rb_receiver_notify_done(l4shmc_ringbuf_t *buf);
00337
00338
00345 L4_CV int l4shmc_rb_receiver_read_next_size(l4shmc_ringbuf_head_t *head);
00346
00347 __END_DECLS

```

## 16.415 l4/shmc/shmc.h File Reference

Shared memory library header file.

```

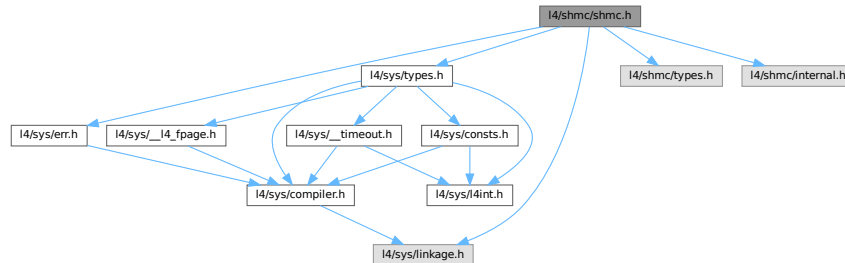
#include <l4/sys/linkage.h>
#include <l4/sys/types.h>

```

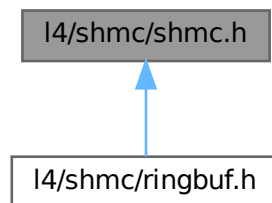


```
#include <l4/sys/err.h>
#include <l4/shmc/types.h>
#include <l4/shmc/internal.h>
```

Include dependency graph for shmc.h:



This graph shows which files directly or indirectly include this file:



## Functions

- long [l4shmc\\_create](#) (char const \*shmc\_name)  
*Create a shared memory area.*
- long [l4shmc\\_attach](#) (char const \*shmc\_name, l4shmc\_area\_t \*shmarea)  
*Attach to a shared memory area.*
- long [l4shmc\\_get\\_client\\_nr](#) (l4shmc\_area\_t const \*shmarea)  
*Determine the client number of the shared memory region.*
- long [l4shmc\\_mark\\_client\\_initialized](#) (l4shmc\_area\_t \*shmarea)  
*Mark this shared memory client as 'initialized'.*
- long [l4shmc\\_get\\_initialized\\_clients](#) (l4shmc\_area\_t \*shmarea, l4\_umword\_t \*bitmask)  
*Fetch the `_clients_init_done` bitmask of the shared memory area.*
- long [l4shmc\\_add\\_chunk](#) (l4shmc\_area\_t \*shmarea, char const \*chunk\_name, l4\_umword\_t chunk\_capacity, l4shmc\_chunk\_t \*chunk)  
*Add a chunk in the shared memory area.*
- long [l4shmc\\_add\\_signal](#) (l4shmc\_area\_t \*shmarea, char const \*signal\_name, l4shmc\_signal\_t \*signal)  
*Add a signal for the shared memory area.*
- long [l4shmc\\_trigger](#) (l4shmc\_signal\_t \*signal)  
*Trigger a signal.*

- long [l4shmc\\_chunk\\_try\\_to\\_take](#) (l4shmc\_chunk\_t \*chunk)  
*Try to mark chunk busy.*
- long [l4shmc\\_chunk\\_try\\_to\\_take\\_for\\_writing](#) (l4shmc\_chunk\_t \*chunk)  
*Try to mark chunk busy writing.*
- long [l4shmc\\_chunk\\_try\\_to\\_take\\_for\\_overwriting](#) (l4shmc\_chunk\_t \*chunk)  
*Try to mark the chunk busy writing after it was ready for reading.*
- long [l4shmc\\_chunk\\_try\\_to\\_take\\_for\\_reading](#) (l4shmc\_chunk\_t \*chunk)  
*Try to mark chunk busy reading.*
- long [l4shmc\\_chunk\\_ready](#) (l4shmc\_chunk\_t \*chunk, l4\_umword\_t size)  
*Mark chunk as filled (ready).*
- long [l4shmc\\_chunk\\_ready\\_sig](#) (l4shmc\_chunk\_t \*chunk, l4\_umword\_t size)  
*Mark chunk as filled (ready) and signal consumer.*
- long [l4shmc\\_get\\_chunk](#) (l4shmc\_area\_t \*shmarea, char const \*chunk\_name, l4shmc\_chunk\_t \*chunk)  
*Get chunk out of shared memory area.*
- long [l4shmc\\_get\\_chunk\\_to](#) (l4shmc\_area\_t \*shmarea, char const \*chunk\_name, l4\_umword\_t timeout\_ms, l4shmc\_chunk\_t \*chunk)  
*Get chunk out of shared memory area, with timeout.*
- long [l4shmc\\_iterate\\_chunk](#) (l4shmc\_area\_t const \*shmarea, char const \*\*chunk\_name, long offs)  
*Iterate over names of all existing chunks.*
- long [l4shmc\\_attach\\_signal](#) (l4shmc\_area\_t \*shmarea, char const \*signal\_name, l4\_cap\_idx\_t thread, l4shmc\_signal\_t \*signal)  
*Attach to signal.*
- long [l4shmc\\_get\\_signal](#) (l4shmc\_area\_t \*shmarea, char const \*signal\_name, l4shmc\_signal\_t \*signal)  
*Get signal object from the shared memory area.*
- long [l4shmc\\_enable\\_signal](#) (l4shmc\_signal\_t \*signal)  
*Enable a signal.*
- long [l4shmc\\_enable\\_chunk](#) (l4shmc\_chunk\_t \*chunk)  
*Enable a signal connected with a chunk.*
- long [l4shmc\\_wait\\_any](#) (l4shmc\_signal\_t \*\*retsignal)  
*Wait on any signal.*
- long [l4shmc\\_wait\\_any\\_try](#) (l4shmc\_signal\_t \*\*retsignal)  
*Check whether any waited signal has an event pending.*
- long [l4shmc\\_wait\\_any\\_to](#) (l4\_timeout\_t timeout, l4shmc\_signal\_t \*\*retsignal)  
*Wait for any signal with timeout.*
- long [l4shmc\\_wait\\_signal](#) (l4shmc\_signal\_t \*signal)  
*Wait on a specific signal.*
- long [l4shmc\\_wait\\_signal\\_to](#) (l4shmc\_signal\_t \*signal, l4\_timeout\_t timeout)  
*Wait on a specific signal, with timeout.*
- long [l4shmc\\_wait\\_signal\\_try](#) (l4shmc\_signal\_t \*signal)  
*Check whether a specific signal has an event pending.*
- long [l4shmc\\_wait\\_chunk](#) (l4shmc\_chunk\_t \*chunk)  
*Wait on a specific chunk.*
- long [l4shmc\\_wait\\_chunk\\_to](#) (l4shmc\_chunk\_t \*chunk, l4\_timeout\_t timeout)  
*Check whether a specific chunk has an event pending, with timeout.*
- long [l4shmc\\_wait\\_chunk\\_try](#) (l4shmc\_chunk\_t \*chunk)  
*Check whether a specific chunk has an event pending.*
- long [l4shmc\\_chunk\\_consumed](#) (l4shmc\_chunk\_t \*chunk)  
*Mark a chunk as free.*
- long [l4shmc\\_connect\\_chunk\\_signal](#) (l4shmc\_chunk\_t \*chunk, l4shmc\_signal\_t \*signal)  
*Connect a signal with a chunk.*
- long [l4shmc\\_is\\_chunk\\_ready](#) (l4shmc\_chunk\_t const \*chunk)

- Check whether data is available.*
- `long l4shmc_is_chunk_clear` (`l4shmc_chunk_t` const \*`chunk`)
- Check whether chunk is free.*
- `void * l4shmc_chunk_ptr` (`l4shmc_chunk_t` const \*`chunk`)
- Get data pointer to chunk.*
- `long l4shmc_chunk_size` (`l4shmc_chunk_t` const \*`chunk`)
- Get current size of a chunk.*
- `long l4shmc_chunk_capacity` (`l4shmc_chunk_t` const \*`chunk`)
- Get capacity of a chunk.*
- `l4shmc_signal_t * l4shmc_chunk_signal` (`l4shmc_chunk_t` const \*`chunk`)
- Get the registered signal of a chunk.*
- `l4_cap_idx_t l4shmc_signal_cap` (`l4shmc_signal_t` const \*`signal`)
- Get the signal capability of a signal.*
- `long l4shmc_check_magic` (`l4shmc_chunk_t` const \*`chunk`)
- Check magic value of a chunk.*
- `long l4shmc_area_size` (`l4shmc_area_t` const \*`shmarea`)
- Get size of shared memory area.*
- `long l4shmc_area_size_free` (`l4shmc_area_t` const \*`shmarea`)
- Get free size of shared memory area.*
- `long l4shmc_area_overhead` (`void`)
- Get memory overhead per area that is not available for chunks.*
- `long l4shmc_chunk_overhead` (`void`)
- Get memory overhead required in addition to the chunk capacity for adding one chunk.*

### 16.415.1 Detailed Description

Shared memory library header file.

Definition in file [shmc.h](#).

## 16.416 shmc.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU Lesser General Public License 2.1.
00011  * Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/linkage.h>
00016 #include <l4/sys/types.h>
00017 #include <l4/sys/err.h>
00018
00069 #define __INCLUDED_FROM_L4SHMC_H__
00070 #include <l4/shmc/types.h>
00071
00072 __BEGIN_DECLS
00073
00086 L4_CV long
00087 l4shmc_create(char const *shm_name);
00088
00110 L4_CV long
00111 l4shmc_attach(char const *shm_name, l4shmc_area_t *shmarea);
00112

```

```

00121 L4_CV long
00122 l4shmc_get_client_nr(l4shmc_area_t const *shmarea);
00123
00136 L4_CV long
00137 l4shmc_mark_client_initialized(l4shmc_area_t *shmarea);
00138
00151 L4_CV long
00152 l4shmc_get_initialized_clients(l4shmc_area_t *shmarea, l4_umword_t *bitmask);
00153
00166 L4_CV long
00167 l4shmc_add_chunk(l4shmc_area_t *shmarea, char const *chunk_name,
00168                  l4_umword_t chunk_capacity, l4shmc_chunk_t *chunk);
00169
00181 L4_CV long
00182 l4shmc_add_signal(l4shmc_area_t *shmarea, char const *signal_name,
00183                  l4shmc_signal_t *signal);
00184
00194 L4_CV L4_INLINE long
00195 l4shmc_trigger(l4shmc_signal_t *signal);
00196
00206 L4_CV L4_INLINE long
00207 l4shmc_chunk_try_to_take(l4shmc_chunk_t *chunk);
00208
00220 L4_CV L4_INLINE long
00221 l4shmc_chunk_try_to_take_for_writing(l4shmc_chunk_t *chunk);
00222
00237 L4_CV L4_INLINE long
00238 l4shmc_chunk_try_to_take_for_overwriting(l4shmc_chunk_t *chunk);
00239
00249 L4_CV L4_INLINE long
00250 l4shmc_chunk_try_to_take_for_reading(l4shmc_chunk_t *chunk);
00251
00262 L4_CV L4_INLINE long
00263 l4shmc_chunk_ready(l4shmc_chunk_t *chunk, l4_umword_t size);
00264
00275 L4_CV L4_INLINE long
00276 l4shmc_chunk_ready_sig(l4shmc_chunk_t *chunk, l4_umword_t size);
00277
00289 L4_CV L4_INLINE long
00290 l4shmc_get_chunk(l4shmc_area_t *shmarea, char const *chunk_name,
00291                  l4shmc_chunk_t *chunk);
00292
00306 L4_CV long
00307 l4shmc_get_chunk_to(l4shmc_area_t *shmarea, char const *chunk_name,
00308                      l4_umword_t timeout_ms, l4shmc_chunk_t *chunk);
00309
00323 L4_CV long
00324 l4shmc_iterate_chunk(l4shmc_area_t const *shmarea, char const **chunk_name,
00325                      long offs);
00326
00339 L4_CV long
00340 l4shmc_attach_signal(l4shmc_area_t *shmarea, char const *signal_name,
00341                      l4_cap_idx_t thread, l4shmc_signal_t *signal);
00342
00343
00355 L4_CV long
00356 l4shmc_get_signal(l4shmc_area_t *shmarea, char const *signal_name,
00357                   l4shmc_signal_t *signal);
00358
00372 L4_CV long
00373 l4shmc_enable_signal(l4shmc_signal_t *signal);
00374
00388 L4_CV long
00389 l4shmc_enable_chunk(l4shmc_chunk_t *chunk);
00390
00400 L4_CV L4_INLINE long
00401 l4shmc_wait_any(l4shmc_signal_t **retsignal);
00402
00416 L4_CV L4_INLINE long
00417 l4shmc_wait_any_try(l4shmc_signal_t **retsignal);
00418
00433 L4_CV long
00434 l4shmc_wait_any_to(l4_timeout_t timeout, l4shmc_signal_t **retsignal);
00435
00445 L4_CV L4_INLINE long
00446 l4shmc_wait_signal(l4shmc_signal_t *signal);
00447
00458 L4_CV long
00459 l4shmc_wait_signal_to(l4shmc_signal_t *signal, l4_timeout_t timeout);
00460
00474 L4_CV L4_INLINE long
00475 l4shmc_wait_signal_try(l4shmc_signal_t *signal);
00476
00486 L4_CV L4_INLINE long
00487 l4shmc_wait_chunk(l4shmc_chunk_t *chunk);
00488
00503 L4_CV long

```

```

00504 l4shmc_wait_chunk_to(l4shmc_chunk_t *chunk, l4_timeout_t timeout);
00505
00519 L4_CV L4_INLINE long
00520 l4shmc_wait_chunk_try(l4shmc_chunk_t *chunk);
00521
00531 L4_CV L4_INLINE long
00532 l4shmc_chunk_consumed(l4shmc_chunk_t *chunk);
00533
00544 L4_CV long
00545 l4shmc_connect_chunk_signal(l4shmc_chunk_t *chunk, l4shmc_signal_t *signal);
00546
00556 L4_CV L4_INLINE long
00557 l4shmc_is_chunk_ready(l4shmc_chunk_t const *chunk);
00558
00568 L4_CV L4_INLINE long
00569 l4shmc_is_chunk_clear(l4shmc_chunk_t const *chunk);
00570
00579 L4_CV L4_INLINE void *
00580 l4shmc_chunk_ptr(l4shmc_chunk_t const *chunk);
00581
00590 L4_CV L4_INLINE long
00591 l4shmc_chunk_size(l4shmc_chunk_t const *chunk);
00592
00601 L4_CV L4_INLINE long
00602 l4shmc_chunk_capacity(l4shmc_chunk_t const *chunk);
00603
00613 L4_CV L4_INLINE l4shmc_signal_t *
00614 l4shmc_chunk_signal(l4shmc_chunk_t const *chunk);
00615
00624 L4_CV L4_INLINE l4_cap_idx_t
00625 l4shmc_signal_cap(l4shmc_signal_t const *signal);
00626
00636 L4_CV L4_INLINE long
00637 l4shmc_check_magic(l4shmc_chunk_t const *chunk);
00638
00648 L4_CV long
00649 l4shmc_area_size(l4shmc_area_t const *shmarea);
00650
00660 L4_CV long
00661 l4shmc_area_size_free(l4shmc_area_t const *shmarea);
00662
00669 L4_CV long
00670 l4shmc_area_overhead(void);
00671
00679 L4_CV long
00680 l4shmc_chunk_overhead(void);
00681
00682 #include <l4/shmc/internal.h>
00683
00684 __END_DECLS

```

## 16.417 l4/sigma0/sigma0.h File Reference

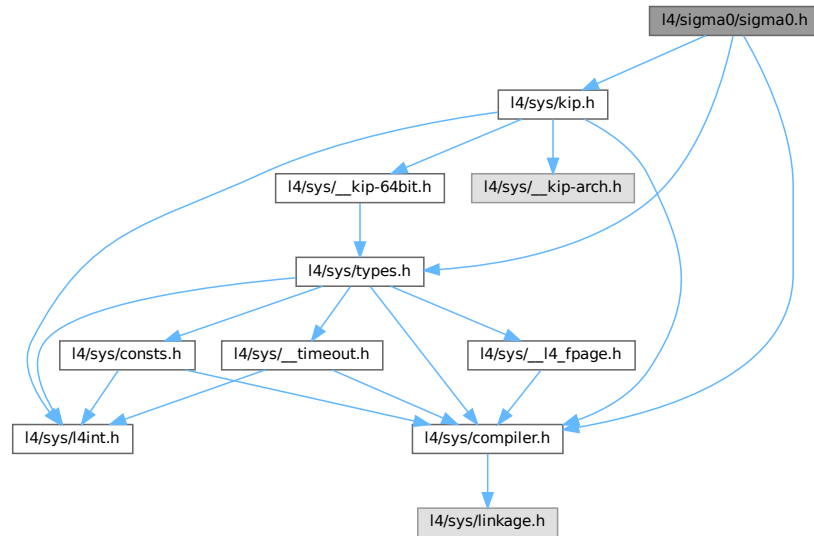
Sigma0 interface.

```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/sys/kip.h>

```

Include dependency graph for sigma0.h:



## Macros

- **#define SIGMA0\_REQ\_MAGIC** ~0xFFUL  
*Request magic.*
- **#define SIGMA0\_REQ\_MASK** ~0xFFUL  
*Request mask.*
- **#define SIGMA0\_REQ\_ID\_MASK** 0xF0  
*ID mask.*
- **#define SIGMA0\_REQ\_ID\_FPAGE\_RAM** 0x60  
*RAM.*
- **#define SIGMA0\_REQ\_ID\_FPAGE\_IOMEM** 0x70  
*I/O memory.*
- **#define SIGMA0\_REQ\_ID\_FPAGE\_IOMEM\_CACHED** 0x80  
*Cached I/O memory.*
- **#define SIGMA0\_REQ\_ID\_FPAGE\_ANY** 0x90  
*Any.*
- **#define SIGMA0\_REQ\_ID\_KIP** 0xA0  
*KIP.*
- **#define SIGMA0\_REQ\_ID\_DEBUG\_DUMP** 0xC0  
*Debug dump.*
- **#define SIGMA0\_IS\_MAGIC\_REQ(d1)** ((d1 & SIGMA0\_REQ\_MASK) == SIGMA0\_REQ\_MAGIC)  
*Check if magic.*
- **#define SIGMA0\_REQ(x)** (SIGMA0\_REQ\_MAGIC + SIGMA0\_REQ\_ID\_ ## x)  
*Construct.*
- **#define SIGMA0\_REQ\_FPAGE\_RAM** (SIGMA0\_REQ(FPAGE\_RAM))  
*RAM.*
- **#define SIGMA0\_REQ\_FPAGE\_IOMEM** (SIGMA0\_REQ(FPAGE\_IOMEM))  
*I/O memory.*
- **#define SIGMA0\_REQ\_FPAGE\_IOMEM\_CACHED** (SIGMA0\_REQ(FPAGE\_IOMEM\_CACHED))

- Cache I/O memory.
- #define **SIGMA0\_REQ\_FPAGE\_ANY** ([SIGMA0\\_REQ\(FPAGE\\_ANY\)](#))  
Any.
- #define **SIGMA0\_REQ\_KIP** ([SIGMA0\\_REQ\(KIP\)](#))  
KIP.
- #define **SIGMA0\_REQ\_DEBUG\_DUMP** ([SIGMA0\\_REQ\(DEBUG\\_DUMP\)](#))  
Debug dump.

## Enumerations

- enum [l4sigma0\\_return\\_flags\\_t](#) {  
[L4SIGMA0\\_OK](#) , [L4SIGMA0\\_NOTALIGNED](#) , [L4SIGMA0\\_IPCERROR](#) , [L4SIGMA0\\_NOFPAGE](#) ,  
[L4SIGMA0\\_4](#) , [L4SIGMA0\\_5](#) , [L4SIGMA0\\_SMALLERFPAGE](#) }  
Return flags of libsigma0 functions.

## Functions

- [l4\\_kernel\\_info\\_t](#) \* [l4sigma0\\_map\\_kip](#) ([l4\\_cap\\_idx\\_t](#) sigma0, void \*addr, unsigned log2\_size)  
Map the kernel info page from sigma0 to addr.
- int [l4sigma0\\_map\\_mem](#) ([l4\\_cap\\_idx\\_t](#) sigma0, [l4\\_addr\\_t](#) phys, [l4\\_addr\\_t](#) virt, [l4\\_addr\\_t](#) size)  
Request a memory mapping from sigma0.
- int [l4sigma0\\_map\\_iomem](#) ([l4\\_cap\\_idx\\_t](#) sigma0, [l4\\_addr\\_t](#) phys, [l4\\_addr\\_t](#) virt, [l4\\_addr\\_t](#) size, int cached)  
Request IO memory from sigma0.
- int [l4sigma0\\_map\\_anypage](#) ([l4\\_cap\\_idx\\_t](#) sigma0, [l4\\_addr\\_t](#) map\_area, unsigned log2\_map\_size, [l4\\_addr\\_t](#) \*base, unsigned sz)  
Request an arbitrary free page of RAM.
- void [l4sigma0\\_debug\\_dump](#) ([l4\\_cap\\_idx\\_t](#) sigma0)  
Request sigma0 to dump internal debug information.
- char const \* [l4sigma0\\_map\\_errstr](#) (int err)  
Get user readable error messages for the return codes.

### 16.417.1 Detailed Description

Sigma0 interface.

Definition in file [sigma0.h](#).

## 16.418 sigma0.h

[Go to the documentation of this file.](#)

```
00001
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *           Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *           Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011  *           economic rights: Technische Universität Dresden (Germany)
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU Lesser General Public License 2.1.
00014  * Please see the COPYING-LGPL-2.1 file for details.
00015  */
00016 #ifndef __L4_SIGMA0_SIGMA0_H
00017 #define __L4_SIGMA0_SIGMA0_H
00018
```

```

00027 #include <l4/sys/compiler.h>
00028 #include <l4/sys/types.h>
00029 #include <l4/sys/kip.h>
00030
00037 #undef SIGMA0_REQ_MAGIC
00038 #undef SIGMA0_REQ_MASK
00039
00040 # define SIGMA0_REQ_MAGIC    ~0xFFFUL
00041 # define SIGMA0_REQ_MASK    ~0xFFFUL
00043 /* Starting with 0x60 allows to detect components which still use the old
00044  * constants (0x00 ... 0x50) */
00045 #define SIGMA0_REQ_ID_MASK    0xF0
00046 #define SIGMA0_REQ_ID_FPAGE_RAM    0x60
00047 #define SIGMA0_REQ_ID_FPAGE_IOMEM    0x70
00048 #define SIGMA0_REQ_ID_FPAGE_IOMEM_CACHED    0x80
00049 #define SIGMA0_REQ_ID_FPAGE_ANY    0x90
00050 #define SIGMA0_REQ_ID_KIP    0xA0
00051 #define SIGMA0_REQ_ID_DEBUG_DUMP    0xC0
00053 #define SIGMA0_IS_MAGIC_REQ(d1) \
00054     ((d1 & SIGMA0_REQ_MASK) == SIGMA0_REQ_MAGIC)
00056 #define SIGMA0_REQ(x) \
00057     (SIGMA0_REQ_MAGIC + SIGMA0_REQ_ID_ ## x)
00059 /* Use these constants in your code! */
00060 #define SIGMA0_REQ_FPAGE_RAM    (SIGMA0_REQ(FPAGE_RAM))
00061 #define SIGMA0_REQ_FPAGE_IOMEM    (SIGMA0_REQ(FPAGE_IOMEM))
00062 #define SIGMA0_REQ_FPAGE_IOMEM_CACHED    (SIGMA0_REQ(FPAGE_IOMEM_CACHED))
00063 #define SIGMA0_REQ_FPAGE_ANY    (SIGMA0_REQ(FPAGE_ANY))
00064 #define SIGMA0_REQ_KIP    (SIGMA0_REQ(KIP))
00065 #define SIGMA0_REQ_DEBUG_DUMP    (SIGMA0_REQ(DEBUG_DUMP))
00076 enum l4sigma0_return_flags_t {
00077     L4SIGMA0_OK,
00078     L4SIGMA0_NOTALIGNED,
00079     L4SIGMA0_IPCERROR,
00080     L4SIGMA0_NOFPAGE,
00081     L4SIGMA0_4,
00082     L4SIGMA0_5,
00083     L4SIGMA0_SMALLERFPAGE,
00084 };
00085
00086 EXTERN_C_BEGIN
00087
00097 L4_CV l4_kernel_info_t *
00098 l4sigma0_map_kip(l4_cap_idx_t sigma0, void *addr, unsigned log2_size);
00099
00129 L4_CV int l4sigma0_map_mem(l4_cap_idx_t sigma0,
00130     l4_addr_t phys, l4_addr_t virt, l4_addr_t size);
00131
00154 L4_CV int l4sigma0_map_iomem(l4_cap_idx_t sigma0, l4_addr_t phys,
00155     l4_addr_t virt, l4_addr_t size, int cached);
00180 L4_CV int l4sigma0_map_anypage(l4_cap_idx_t sigma0, l4_addr_t map_area,
00181     unsigned log2_map_size, l4_addr_t *base,
00182     unsigned sz);
00183
00192 L4_CV void l4sigma0_debug_dump(l4_cap_idx_t sigma0);
00193
00201 L4_INLINE char const *l4sigma0_map_errstr(int err);
00202
00206 /* Implementations */
00207
00208 L4_INLINE char const *l4sigma0_map_errstr(int err)
00209 {
00210     switch (err)
00211     {
00212     case 0: return "No error";
00213     case -1: return "Phys, virt or size not aligned";
00214     case -2: return "IPC error";
00215     case -3: return "No fpage received";
00216 #ifndef SIGMA0_REQ_MAGIC
00217     case -4: return "Bad physical address (old protocol only)";
00218 #endif
00219     case -6: return "Superpage requested but smaller flexpage received";
00220     case -7: return "Cannot map I/O memory cacheable (old protocol only)";
00221     default: return "Unknown error";
00222     }
00223 }
00224
00225
00226 EXTERN_C_END
00227
00228 #endif /* ! __L4_SIGMA0_SIGMA0_H */

```

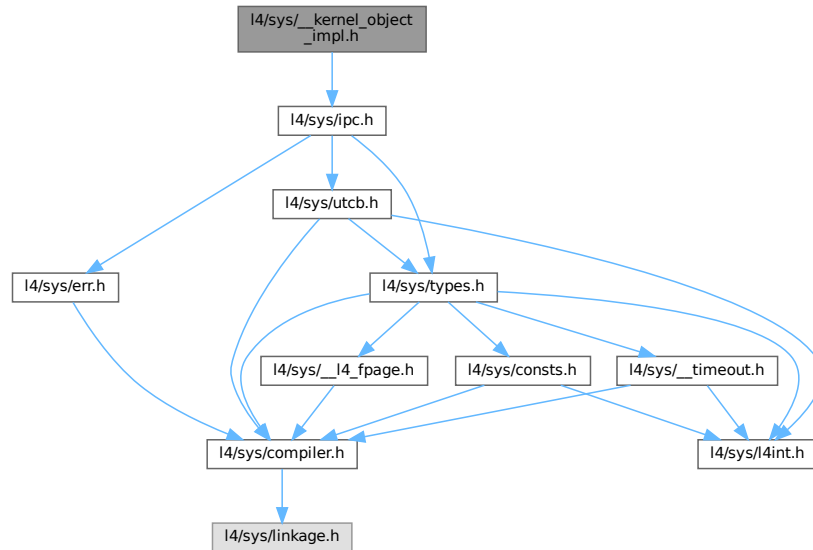


## 16.419 l4/sys/\_\_kernel\_object\_impl.h File Reference

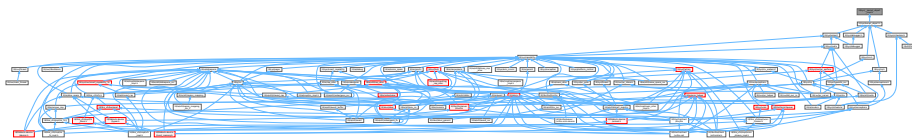
Low-level kernel debugger functions.

```
#include <l4/sys/ipc.h>
```

Include dependency graph for \_\_kernel\_object\_impl.h:



This graph shows which files directly or indirectly include this file:



### 16.419.1 Detailed Description

Low-level kernel debugger functions.

Definition in file [\\_\\_kernel\\_object\\_impl.h](#).

## 16.420 \_\_kernel\_object\_impl.h

[Go to the documentation of this file.](#)

```

00001
00005 #pragma once
00006
00007 #include <l4/sys/ipc.h>
00008
00009 L4_INLINE l4_msgtag_t
00010 l4_invoke_debugger(l4_cap_idx_t obj, l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW

```

```

00011 {
00012     l4_msgtag_t t2;
00013     unsigned const words = l4_msgtag_words(tag);
00014     l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00015
00016     if (l4_is_invalid_cap(obj))
00017         return l4_msgtag(~L4_EINVAL, 0, 0, 0);
00018
00019     if (words + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00020         return l4_msgtag(~L4_EMSTOOLONG, 0, 0, 0);
00021
00022     mr->mr[0] += 0x100;
00023     mr->mr[words] = L4_ITEM_MAP;
00024     mr->mr[words + 1] = l4_obj_fpage(obj, 0, L4_CAP_FPAGE_RWS).raw;
00025     t2 = l4_msgtag(L4_PROTO_DEBUGGER, words, 1, l4_msgtag_flags(tag));
00026
00027     return l4_ipc_call(L4_BASE_DEBUGGER_CAP, utcb, t2, L4_IPC_NEVER);
00028 }
00029

```

## 16.421 \_\_kip-32bit.h

```

00001
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *               Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011  *               Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00012  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>,
00013  *               Martin Pohlack <mp26@os.inf.tu-dresden.de>,
00014  *               Lars Reuther <reuther@os.inf.tu-dresden.de>
00015  *               economic rights: Technische Universität Dresden (Germany)
00016  *
00017  * This file is part of TUD:OS and distributed under the terms of the
00018  * GNU General Public License 2.
00019  * Please see the COPYING-GPL-2 file for details.
00020  *
00021  * As a special exception, you may use this file as part of a free software
00022  * library without restriction. Specifically, if other files instantiate
00023  * templates or use macros or inline functions from this file, or you compile
00024  * this file and link it with other files to produce an executable, this
00025  * file does not by itself cause the resulting executable to be covered by
00026  * the GNU General Public License. This exception does not however
00027  * invalidate any other reasons why the executable file might be covered by
00028  * the GNU General Public License.
00029  */
00030 #pragma once
00031
00032 #include <l4/sys/types.h>
00033
00038 typedef struct l4_kernel_info_t
00039 {
00040     /* offset 0x00 */
00041     l4_uint32_t magic;
00042     l4_uint32_t version;
00043     l4_uint8_t offset_version_strings;
00044     l4_uint8_t fill0[3];
00045     l4_uint8_t kip_sys_calls;
00046     l4_uint8_t node;
00047     l4_uint8_t fill1[2];
00048
00049     /* the following stuff is undocumented; we assume that the kernel
00050      * info page is located at offset 0x1000 into the L4 kernel boot
00051      * image so that these declarations are consistent with section 2.9
00052      * of the L4 Reference Manual */
00053
00054     /* offset 0x10 */
00055     /* Kernel debugger */
00056     l4_umword_t scheduler_granularity;
00057     l4_umword_t _res00[3];
00058
00059     /* offset 0x20 */
00060     /* Sigma0 */
00061     l4_umword_t sigma0_esp;
00062     l4_umword_t sigma0_eip;
00063     l4_umword_t _res01[2];
00064
00065     /* offset 0x30 */
00066     /* Sigma1 */
00067     l4_umword_t sigma1_esp;
00068     l4_umword_t sigma1_eip;
00069     l4_umword_t _res02[2];
00070
00071
00072

```

```

00073  /* offset 0x40 */
00074  /* Root task */
00075  l4_umword_t          root_esp;
00076  l4_umword_t          root_eip;
00077  l4_umword_t          _res03[2];
00078
00079  /* offset 0x50 */
00080  /* L4 configuration */
00081  l4_umword_t          _res50[1];
00082  l4_umword_t          mem_info;
00083  l4_umword_t          _res58[2];
00084
00085  /* offset 0x60 */
00086  l4_umword_t          _res04[16];
00087
00088  /* offset 0xA0 */
00089  volatile l4_cpu_time_t _clock_val;
00090  l4_umword_t          _res05[2];
00091
00092  /* offset 0xB0 */
00093  l4_umword_t          frequency_cpu;
00094  l4_umword_t          frequency_bus;
00095
00096  /* offset 0xB8 */
00097  l4_umword_t          _res06[10];
00098
00099  /* offset 0xE0 */
00100  l4_umword_t          user_ptr;
00101  l4_umword_t          vhw_offset;
00102  l4_umword_t          _res07[2];
00103
00104  /* offset 0xF0 */
00105  struct l4_kip_platform_info platform_info;
00106 } l4_kernel_info_t;

```

## 16.422 \_\_kip-64bit.h

```

00001
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *                Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *                Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011  *                Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00012  *                Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00013  *                economic rights: Technische Universität Dresden (Germany)
00014  *
00015  * This file is part of TUD:OS and distributed under the terms of the
00016  * GNU General Public License 2.
00017  * Please see the COPYING-GPL-2 file for details.
00018  *
00019  * As a special exception, you may use this file as part of a free software
00020  * library without restriction. Specifically, if other files instantiate
00021  * templates or use macros or inline functions from this file, or you compile
00022  * this file and link it with other files to produce an executable, this
00023  * file does not by itself cause the resulting executable to be covered by
00024  * the GNU General Public License. This exception does not however
00025  * invalidate any other reasons why the executable file might be covered by
00026  * the GNU General Public License.
00027  */
00028 #pragma once
00029
00030 #include <l4/sys/types.h>
00031
00036 typedef struct l4_kernel_info_t
00037 {
00038     /* offset 0x00 */
00039     l4_uint64_t          magic;
00042     l4_uint64_t          version;
00043     l4_uint8_t           offset_version_strings;
00044     l4_uint8_t           fill2[7];
00045     l4_uint8_t           kip_sys_calls;
00046     l4_uint8_t           node;
00047     l4_uint8_t           fill3[6];
00048
00049     /* the following stuff is undocumented; we assume that the kernel
00050        info page is located at offset 0x1000 into the L4 kernel boot
00051        image so that these declarations are consistent with section 2.9
00052        of the L4 Reference Manual */
00053
00054     /* offset 0x20 */
00055     /* Kernel debugger */
00056     l4_umword_t          scheduler_granularity;
00057     l4_umword_t          _res00[3];

```

```

00058
00059 /* offset 0x40 */
00060 /* Sigma0 */
00061 l4_umword_t          sigma0_esp;
00062 l4_umword_t          sigma0_eip;
00063 l4_umword_t          _res01[2];
00064
00065 /* offset 0x60 */
00066 /* Sigma1 */
00067 l4_umword_t          sigma1_esp;
00068 l4_umword_t          sigma1_eip;
00069 l4_umword_t          _res02[2];
00070
00071 /* offset 0x80 */
00072 /* Root task */
00073 l4_umword_t          root_esp;
00074 l4_umword_t          root_eip;
00075 l4_umword_t          _res03[2];
00076
00077 /* offset 0xA0 */
00078 /* L4 configuration */
00079 l4_umword_t          _res_a0[1];
00080 l4_umword_t          mem_info;
00081 l4_umword_t          _res_b0[2];
00082
00083 /* offset 0xC0 */
00084 l4_umword_t          _res04[16];
00085
00086 /* offset 0x140 */
00087 volatile l4_cpu_time_t _clock_val;
00088 l4_umword_t          _res05[1];
00089 l4_umword_t          frequency_cpu;
00090 l4_umword_t          frequency_bus;
00091
00092 /* offset 0x160 */
00093 l4_umword_t          _res06[12];
00094
00095 /* offset 0x1C0 */
00096 l4_umword_t          user_ptr;
00097 l4_umword_t          vhw_offset;
00098 l4_umword_t          _res07[2];
00099
00100 /* offset 0x1E0 */
00101 struct l4_kip_platform_info platform_info;
00102 } l4_kernel_info_t;

```

## 16.423 l4/sys/\_\_ktrace-impl.h File Reference

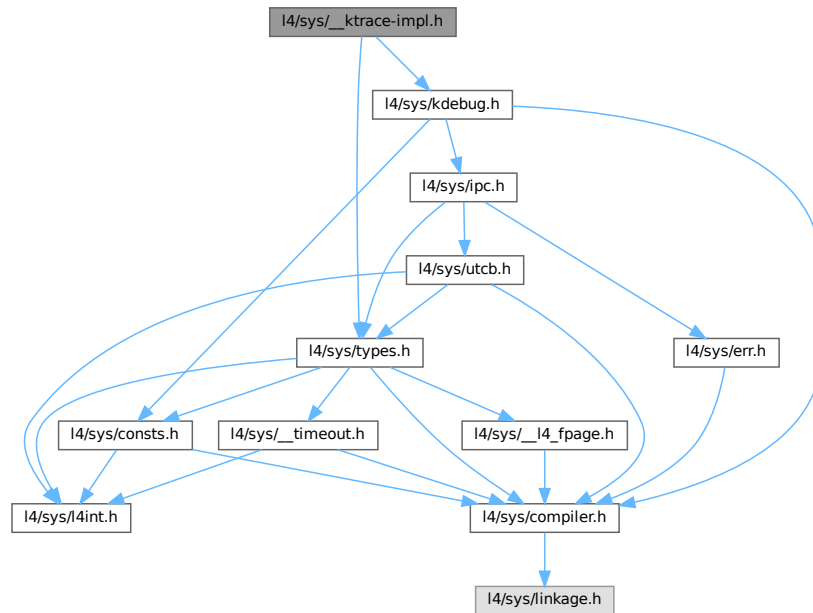
L4 kernel event tracing.

```

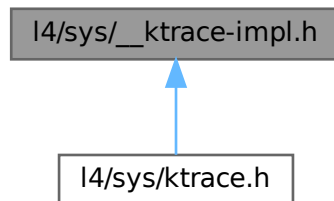
#include <l4/sys/types.h>
#include <l4/sys/kdebug.h>

```

Include dependency graph for \_\_\_ktrace-impl.h:



This graph shows which files directly or indirectly include this file:



## Functions

- `l4_umword_t fiasco_tbuf_log` (const char \*text)  
Create new trace-buffer entry with describing <text>.
- `l4_umword_t fiasco_tbuf_log_3val` (const char \*text, `l4_umword_t` v1, `l4_umword_t` v2, `l4_umword_t` v3)  
Create new trace-buffer entry with describing <text> and three additional values.
- void `fiasco_tbuf_clear` (void)  
Clear trace-buffer.
- void `fiasco_tbuf_dump` (void)  
Dump trace-buffer to kernel console.
- `l4_umword_t fiasco_tbuf_log_binary` (const unsigned char \*data)  
Create new trace-buffer entry with binary data.

## 16.423.1 Detailed Description

[L4](#) kernel event tracing.

Definition in file [\\_\\_ktrace-impl.h](#).

## 16.424 \_\_ktrace-impl.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *          Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009  *          Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *          economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/sys/types.h>
00028 #include <l4/sys/kdebug.h>
00029
00030 /*****
00031  *** Implementation
00032  *****/
00033
00034 L4_INLINE l4_umword_t
00035 fiasco_tbuf_log(const char *text)
00036 {
00037     enum { TBUF_LOG = 0x201 };
00038     return l4_error(__kdebug_text(TBUF_LOG, text, __builtin_strlen(text)));
00039 }
00040
00041 L4_INLINE l4_umword_t
00042 fiasco_tbuf_log_3val(const char *text, l4_umword_t v1, l4_umword_t v2,
00043                     l4_umword_t v3)
00044 {
00045     enum { TBUF_LOG_3VAL = 0x204 };
00046     return l4_error(__kdebug_3_text(TBUF_LOG_3VAL, text,
00047                                     __builtin_strlen(text), v1, v2, v3));
00048 }
00049
00050 L4_INLINE void
00051 fiasco_tbuf_clear(void)
00052 {
00053     enum { TBUF_CLEAR = 0x202 };
00054     __kdebug_op(TBUF_CLEAR);
00055 }
00056
00057 L4_INLINE void
00058 fiasco_tbuf_dump(void)
00059 {
00060     enum { TBUF_DUMP = 0x203 };
00061     __kdebug_op(TBUF_DUMP);
00062 }
00063
00064 L4_INLINE l4_umword_t
00065 fiasco_tbuf_log_binary(const unsigned char *data)
00066 {
00067     enum { TBUF_LOG_BIN = 0x208 };
00068     return l4_error(__kdebug_text(TBUF_LOG_BIN, (const char *)data, 24));
00069 }
00070

```

## 16.425 \_\_l4\_fpage.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00010  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction. Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License. This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 #pragma once
00027
00028 #include <l4/sys/compiler.h>
00029
00057 enum L4_fpage_consts
00058 {
00059     L4_FPAGE_RIGHTS_SHIFT = 0,
00060     L4_FPAGE_TYPE_SHIFT  = 4,
00061     L4_FPAGE_SIZE_SHIFT  = 6,
00062     L4_FPAGE_ADDR_SHIFT  = 12,
00063
00064     L4_FPAGE_RIGHTS_BITS = 4,
00065     L4_FPAGE_TYPE_BITS  = 2,
00066     L4_FPAGE_SIZE_BITS  = 6,
00067     L4_FPAGE_ADDR_BITS  = L4_MWORD_BITS - L4_FPAGE_ADDR_SHIFT,
00068
00070     L4_FPAGE_RIGHTS_MASK = ((1UL < L4_FPAGE_RIGHTS_BITS) - 1)
00071                          < L4_FPAGE_RIGHTS_SHIFT,
00072     L4_FPAGE_TYPE_MASK  = ((1UL < L4_FPAGE_TYPE_BITS) - 1)
00073                          < L4_FPAGE_TYPE_SHIFT,
00074     L4_FPAGE_SIZE_MASK  = ((1UL < L4_FPAGE_SIZE_BITS) - 1)
00075                          < L4_FPAGE_SIZE_SHIFT,
00076     L4_FPAGE_ADDR_MASK  = ~0UL < L4_FPAGE_ADDR_SHIFT,
00078     L4_FPAGE_RIGHTS_ALL = L4_FPAGE_RIGHTS_MASK,
00079 };
00080
00085 typedef union {
00086     l4_umword_t fpage;
00087     l4_umword_t raw;
00088 } l4_fpage_t;
00089
00093 enum
00094 {
00101     L4_WHOLE_ADDRESS_SPACE = 63
00102 };
00103
00108 typedef struct {
00109     l4_umword_t snd_base;
00110     l4_fpage_t fpage;
00111 } l4_snd_fpage_t;
00112
00113
00124 enum L4_fpage_rights
00125 {
00126     L4_FPAGE_X      = 1,
00127     L4_FPAGE_W      = 2,
00128     L4_FPAGE_RO     = 4,
00129     L4_FPAGE_RW     = L4_FPAGE_RO | L4_FPAGE_W,
00130     L4_FPAGE_RX     = L4_FPAGE_RO | L4_FPAGE_X,
00131     L4_FPAGE_RWX    = L4_FPAGE_RW | L4_FPAGE_X,
00132 };
00133
00151 enum L4_cap_fpage_rights
00152 {
00160     L4_CAP_FPAGE_W      = 0x1,
00172     L4_CAP_FPAGE_S      = 0x2,
00178     L4_CAP_FPAGE_R      = 0x4,
00179     L4_CAP_FPAGE_RO     = 0x4,
00188     L4_CAP_FPAGE_D      = 0x8,
00195     L4_CAP_FPAGE_RW     = L4_CAP_FPAGE_R | L4_CAP_FPAGE_W,
00202     L4_CAP_FPAGE_RS     = L4_CAP_FPAGE_R | L4_CAP_FPAGE_S,
00209     L4_CAP_FPAGE_RWS    = L4_CAP_FPAGE_RW | L4_CAP_FPAGE_S,
00215     L4_CAP_FPAGE_RWSD   = L4_CAP_FPAGE_RWS | L4_CAP_FPAGE_D,
00221     L4_CAP_FPAGE_RWD    = L4_CAP_FPAGE_RW | L4_CAP_FPAGE_D,

```

```

00227 L4_CAP_FPAGE_RSD = L4_CAP_FPAGE_RS | L4_CAP_FPAGE_D,
00228 };
00229
00233 enum L4_fpage_type
00234 {
00235     L4_FPAGE_SPECIAL = 0,
00236     L4_FPAGE_MEMORY = 1,
00237     L4_FPAGE_IO = 2,
00238     L4_FPAGE_OBJ = 3,
00239 };
00240
00244 enum L4_fpage_control
00245 {
00248     L4_FPAGE_CONTROL_OFFSET_SHIFT = 12,
00251     L4_FPAGE_CONTROL_MASK = ~0UL << L4_FPAGE_CONTROL_OFFSET_SHIFT,
00252 };
00253
00263 enum L4_obj_fpage_ctl
00264 {
00265     L4_FPAGE_C_REF_CNT = 0x00,
00266     L4_FPAGE_C_NO_REF_CNT = 0x10,
00267
00268     L4_FPAGE_C_OBJ_RIGHT1 = 0x20,
00269     L4_FPAGE_C_OBJ_RIGHT2 = 0x40,
00270     L4_FPAGE_C_OBJ_RIGHT3 = 0x80,
00271     L4_FPAGE_C_OBJ_RIGHTS = 0xe0,
00272
00278     L4_FPAGE_C_IPCGATE_SVR = L4_FPAGE_C_OBJ_RIGHT1
00279 };
00280
00281
00285 enum l4_fpage_cacheability_opt_t
00286 {
00288     L4_FPAGE_CACHE_OPT = 0x1,
00289
00291     L4_FPAGE_CACHEABLE = 0x3,
00292
00294     L4_FPAGE_BUFFERABLE = 0x5,
00295
00297     L4_FPAGE_UNCACHEABLE = 0x1
00298 };
00299
00300
00304 enum
00305 {
00309     L4_WHOLE_IOADDRESS_SPACE = 16,
00310
00312     L4_IOPORT_MAX = (1L << L4_WHOLE_IOADDRESS_SPACE)
00313 };
00314
00315
00316
00331 L4_INLINE l4_fpage_t
00332 l4_fpage(l4_addr_t address, unsigned int size, unsigned char rights) L4_NOTHROW;
00333
00345 L4_INLINE l4_fpage_t
00346 l4_fpage_all(void) L4_NOTHROW;
00347
00354 L4_INLINE l4_fpage_t
00355 l4_fpage_invalid(void) L4_NOTHROW;
00356
00357
00368 L4_INLINE l4_fpage_t
00369 l4_iofpage(unsigned long port, unsigned int size) L4_NOTHROW;
00370
00371
00384 L4_INLINE l4_fpage_t
00385 l4_obj_fpage(l4_cap_idx_t obj, unsigned int order, unsigned char rights) L4_NOTHROW;
00386
00396 L4_INLINE int
00397 l4_is_fpage_writable(l4_fpage_t fp) L4_NOTHROW;
00398
00399
00434 L4_INLINE l4_umword_t
00435 l4_map_control(l4_umword_t spot, unsigned char cache, unsigned grant) L4_NOTHROW;
00436
00449 L4_INLINE l4_umword_t
00450 l4_map_obj_control(l4_umword_t spot, unsigned grant) L4_NOTHROW;
00451
00460 L4_INLINE unsigned
00461 l4_fpage_rights(l4_fpage_t f) L4_NOTHROW;
00462
00471 L4_INLINE unsigned
00472 l4_fpage_type(l4_fpage_t f) L4_NOTHROW;
00473
00484 L4_INLINE unsigned
00485 l4_fpage_size(l4_fpage_t f) L4_NOTHROW;

```



```

00486
00497 L4_INLINE unsigned long
00498 l4_fpage_page(l4_fpage_t f) L4_NOTHROW;
00499
00513 L4_INLINE l4_addr_t
00514 l4_fpage_memaddr(l4_fpage_t f) L4_NOTHROW;
00515
00529 L4_INLINE l4_cap_idx_t
00530 l4_fpage_obj(l4_fpage_t f) L4_NOTHROW;
00531
00545 L4_INLINE unsigned long
00546 l4_fpage_ioport(l4_fpage_t f) L4_NOTHROW;
00547
00557 L4_INLINE l4_fpage_t
00558 l4_fpage_set_rights(l4_fpage_t src, unsigned char new_rights) L4_NOTHROW;
00559
00571 L4_INLINE int
00572 l4_fpage_contains(l4_fpage_t fpage, l4_addr_t addr, unsigned size) L4_NOTHROW;
00573
00590 L4_INLINE unsigned char
00591 l4_fpage_max_order(unsigned char order, l4_addr_t addr,
00592                    l4_addr_t min_addr, l4_addr_t max_addr,
00593                    l4_addr_t hotspot L4_DEFAULT_PARAM(0));
00594
00595 /*****
00596  * Implementations
00597  *****/
00598
00599 L4_INLINE unsigned
00600 l4_fpage_rights(l4_fpage_t f) L4_NOTHROW
00601 {
00602     return (f.raw & L4_FPAGE_RIGHTS_MASK) » L4_FPAGE_RIGHTS_SHIFT;
00603 }
00604
00605 L4_INLINE unsigned
00606 l4_fpage_type(l4_fpage_t f) L4_NOTHROW
00607 {
00608     return (f.raw & L4_FPAGE_TYPE_MASK) » L4_FPAGE_TYPE_SHIFT;
00609 }
00610
00611 L4_INLINE unsigned
00612 l4_fpage_size(l4_fpage_t f) L4_NOTHROW
00613 {
00614     return (f.raw & L4_FPAGE_SIZE_MASK) » L4_FPAGE_SIZE_SHIFT;
00615 }
00616
00617 L4_INLINE unsigned long
00618 l4_fpage_page(l4_fpage_t f) L4_NOTHROW
00619 {
00620     return (f.raw & L4_FPAGE_ADDR_MASK) » L4_FPAGE_ADDR_SHIFT;
00621 }
00622
00623 L4_INLINE unsigned long
00624 l4_fpage_ioport(l4_fpage_t f) L4_NOTHROW
00625 {
00626     return (f.raw & L4_FPAGE_ADDR_MASK) » L4_FPAGE_ADDR_SHIFT;
00627 }
00628
00629 L4_INLINE l4_addr_t
00630 l4_fpage_memaddr(l4_fpage_t f) L4_NOTHROW
00631 {
00632     return f.raw & L4_FPAGE_ADDR_MASK;
00633 }
00634
00635 L4_INLINE l4_cap_idx_t
00636 l4_fpage_obj(l4_fpage_t f) L4_NOTHROW
00637 {
00638     return f.raw & L4_FPAGE_ADDR_MASK;
00639 }
00640
00642 L4_INLINE l4_fpage_t
00643 __l4_fpage_generic(unsigned long address, unsigned int type,
00644                   unsigned int size, unsigned char rights) L4_NOTHROW;
00645
00646 L4_INLINE l4_fpage_t
00647 __l4_fpage_generic(unsigned long address, unsigned int type,
00648                   unsigned int size, unsigned char rights) L4_NOTHROW
00649 {
00650     l4_fpage_t t;
00651     t.raw = ((rights « L4_FPAGE_RIGHTS_SHIFT) & L4_FPAGE_RIGHTS_MASK)
00652           | ((type « L4_FPAGE_TYPE_SHIFT) & L4_FPAGE_TYPE_MASK)
00653           | ((size « L4_FPAGE_SIZE_SHIFT) & L4_FPAGE_SIZE_MASK)
00654           | ((address & L4_FPAGE_ADDR_MASK));
00655     return t;
00656 }
00657
00658 L4_INLINE l4_fpage_t

```

```

00659 l4_fpage_set_rights(l4_fpage_t src, unsigned char new_rights) L4_NOTHROW
00660 {
00661     l4_fpage_t f;
00662     f.raw = ((L4_FPAGE_TYPE_MASK | L4_FPAGE_SIZE_MASK | L4_FPAGE_ADDR_MASK) & src.raw)
00663         | ((new_rights < L4_FPAGE_RIGHTS_SHIFT) & L4_FPAGE_RIGHTS_MASK);
00664     return f;
00665 }
00666
00667 L4_INLINE l4_fpage_t
00668 l4_fpage(l4_addr_t address, unsigned int size, unsigned char rights) L4_NOTHROW
00669 {
00670     return __l4_fpage_generic(address, L4_FPAGE_MEMORY, size, rights);
00671 }
00672
00673 L4_INLINE l4_fpage_t
00674 l4_iofpage(unsigned long port, unsigned int size) L4_NOTHROW
00675 {
00676     return __l4_fpage_generic(port << L4_FPAGE_ADDR_SHIFT, L4_FPAGE_IO, size, L4_FPAGE_RW);
00677 }
00678
00679 L4_INLINE l4_fpage_t
00680 l4_obj_fpage(l4_cap_idx_t obj, unsigned int order, unsigned char rights) L4_NOTHROW
00681 {
00682     static_assert((unsigned long)L4_CAP_SHIFT >= L4_FPAGE_ADDR_SHIFT,
00683         "Capability index does not fit into fpage.");
00684     return __l4_fpage_generic(obj, L4_FPAGE_OBJ, order, rights);
00685 }
00686
00687 L4_INLINE l4_fpage_t
00688 l4_fpage_all(void) L4_NOTHROW
00689 {
00690     return __l4_fpage_generic(0, L4_FPAGE_SPECIAL, L4_WHOLE_ADDRESS_SPACE, 0);
00691 }
00692
00693 L4_INLINE l4_fpage_t
00694 l4_fpage_invalid(void) L4_NOTHROW
00695 {
00696     return __l4_fpage_generic(0, L4_FPAGE_SPECIAL, 0, 0);
00697 }
00698
00699
00700 L4_INLINE int
00701 l4_is_fpage_writable(l4_fpage_t fp) L4_NOTHROW
00702 {
00703     return l4_fpage_rights(fp) & L4_FPAGE_W;
00704 }
00705
00706 L4_INLINE l4_umword_t
00707 l4_map_control(l4_umword_t snd_base, unsigned char cache, unsigned grant) L4_NOTHROW
00708 {
00709     return (snd_base & L4_FPAGE_CONTROL_MASK)
00710         | ((l4_umword_t)cache << 4) | L4_ITEM_MAP | grant;
00711 }
00712
00713 L4_INLINE l4_umword_t
00714 l4_map_obj_control(l4_umword_t snd_base, unsigned grant) L4_NOTHROW
00715 {
00716     return l4_map_control(snd_base, 0, grant);
00717 }
00718
00719 L4_INLINE int
00720 l4_fpage_contains(l4_fpage_t fpage, l4_addr_t addr, unsigned log2size) L4_NOTHROW
00721 {
00722     l4_addr_t fa = l4_fpage_memaddr(fpage);
00723     return (fa <= addr)
00724         && (fa + (1UL << l4_fpage_size(fpage)) >= addr + (1UL << log2size));
00725 }
00726
00727 L4_INLINE unsigned char
00728 l4_fpage_max_order(unsigned char order, l4_addr_t addr,
00729     l4_addr_t min_addr, l4_addr_t max_addr,
00730     l4_addr_t hotspot)
00731 {
00732     while (order < 30 /* limit to 1GB flexpages */)
00733     {
00734         l4_addr_t mask;
00735         l4_addr_t base = l4_trunc_size(addr, order + 1);
00736         if (base < min_addr)
00737             return order;
00738
00739         if (base + (1UL << (order + 1)) - 1 > max_addr - 1)
00740             return order;
00741
00742         mask = ~(~0UL << (order + 1));
00743         if (hotspot == ~0UL || ((addr ^ hotspot) & mask))
00744             break;
00745     }

```

```

00746     ++order;
00747     }
00748
00749     return order;
00750 }

```

## 16.426 \_\_platform\_control-arm.h

```

00001 /*
00002  * Copyright (C) 2024 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/types.h>
00010
00011
00012 L4_INLINE l4_msgtag_t
00013 l4_platform_ctl_set_task_asid(l4_cap_idx_t pfc,
00014                               l4_cap_idx_t task,
00015                               l4_umword_t asid) L4_NOTHROW;
00016
00017 L4_INLINE l4_msgtag_t
00018 l4_platform_ctl_set_task_asid_u(l4_cap_idx_t pfc,
00019                                 l4_cap_idx_t task,
00020                                 l4_umword_t asid,
00021                                 l4_utcb_t *utcb) L4_NOTHROW;
00022
00023 /* IMPLEMENTATION -----*/
00024 L4_INLINE l4_msgtag_t
00025 l4_platform_ctl_set_task_asid_u(l4_cap_idx_t pfc,
00026                                 l4_cap_idx_t task,
00027                                 l4_umword_t asid,
00028                                 l4_utcb_t *utcb) L4_NOTHROW
00029 {
00030     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00031     v->mr[0] = L4_PLATFORM_CTL_SET_TASK_ASID_OP;
00032     v->mr[1] = asid;
00033     v->mr[2] = l4_map_obj_control(0,0);
00034     v->mr[3] = l4_obj_fpage(task, 0, L4_CAP_FPAGE_RWS).raw;
00035     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 1, 0),
00036                       L4_IPC_NEVER);
00037 }
00038
00039 L4_INLINE l4_msgtag_t
00040 l4_platform_ctl_set_task_asid(l4_cap_idx_t pfc,
00041                               l4_cap_idx_t task,
00042                               l4_umword_t asid) L4_NOTHROW
00043 {
00044     return l4_platform_ctl_set_task_asid_u(pfc, task, asid, l4_utcb());
00045 }

```

## 16.427 \_\_task-arm.h

```

00001 /*
00002  * (c) 2018 Adam Lackorzynski <adam@l4re.org>
00003  *
00004  * This file is part of L4Re and distributed under the terms of the
00005  * GNU General Public License 2.
00006  * Please see the COPYING-GPL-2 file for details.
00007  *
00008  * As a special exception, you may use this file as part of a free software
00009  * library without restriction. Specifically, if other files instantiate
00010  * templates or use macros or inline functions from this file, or you compile
00011  * this file and link it with other files to produce an executable, this
00012  * file does not by itself cause the resulting executable to be covered by
00013  * the GNU General Public License. This exception does not however
00014  * invalidate any other reasons why the executable file might be covered by
00015  * the GNU General Public License.
00016  */
00017 #pragma once
00018
00019 #include <l4/sys/types.h>
00020
00021 L4_INLINE l4_msgtag_t

```

```

00025 l4_task_vgicc_map_u(l4_cap_idx_t task, l4_fpage_t vgicc_fpage,
00026                      l4_utcb_t *u) L4_NOTHROW;
00027
00039 L4_INLINE l4_msgtag_t
00040 l4_task_vgicc_map(l4_cap_idx_t task, l4_fpage_t vgicc_fpage) L4_NOTHROW;
00041
00042 /* IMPLEMENTATION ----- */
00043
00044 #include <l4/sys/ipc.h>
00045
00046 L4_INLINE l4_msgtag_t
00047 l4_task_vgicc_map_u(l4_cap_idx_t task, l4_fpage_t vgicc_fpage,
00048                    l4_utcb_t *u) L4_NOTHROW
00049 {
00050     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00051     v->mr[0] = L4_TASK_MAP_VGICC_ARM_OP;
00052     v->mr[1] = vgicc_fpage.raw;
00053     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0), L4_IPC_NEVER);
00054 }
00055
00056 L4_INLINE l4_msgtag_t
00057 l4_task_vgicc_map(l4_cap_idx_t task, l4_fpage_t vgicc_fpage) L4_NOTHROW
00058 {
00059     return l4_task_vgicc_map_u(task, vgicc_fpage, l4_utcb());
00060 }

```

## 16.428 \_\_timeout.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #ifndef L4_SYS_TIMEOUT_H__
00026 #define L4_SYS_TIMEOUT_H__
00027
00028 #include <l4/sys/l4int.h>
00029 #include <l4/sys/compiler.h>
00030
00048 typedef struct l4_timeout_s {
00049     l4_uint16_t t;
00050 } __attribute__((packed)) l4_timeout_s;
00051
00052
00060 typedef union l4_timeout_t
00061 {
00062     l4_uint32_t raw;
00063     struct
00064     {
00065         #ifdef __BIG_ENDIAN__
00066             l4_timeout_s snd;
00067             l4_timeout_s rcv;
00068         #else
00069             l4_timeout_s rcv;
00070             l4_timeout_s snd;
00071         #endif
00072     } p;
00073 } l4_timeout_t;
00074
00075
00081 #define L4_IPC_TIMEOUT_0 ((l4_timeout_s){0x0400})
00082 #define L4_IPC_TIMEOUT_NEVER ((l4_timeout_s){0})
00083 #define L4_IPC_NEVER_INITIALIZER {0}
00084 #define L4_IPC_NEVER ((l4_timeout_t){0})
00085 #define L4_IPC_RECV_TIMEOUT_0 ((l4_timeout_t){0x00000400})
00086 #define L4_IPC_SEND_TIMEOUT_0 ((l4_timeout_t){0x04000000})
00087 #define L4_IPC_BOTH_TIMEOUT_0 ((l4_timeout_t){0x04000400})
00093 #define L4_TIMEOUT_US_NEVER (~0ULL)

```

```

00094
00099 #define L4_TIMEOUT_US_MAX    ((1ULL < 41) - 1)
00100
00112 L4_CONSTEXPR L4_INLINE
00113 l4_timeout_s l4_timeout_rel(unsigned man, unsigned exp) L4_NOTHROW;
00114
00115
00125 L4_CONSTEXPR L4_INLINE
00126 l4_timeout_t l4_ipc_timeout(unsigned snd_man, unsigned snd_exp,
00127                             unsigned rcv_man, unsigned rcv_exp) L4_NOTHROW;
00128
00138 L4_CONSTEXPR L4_INLINE
00139 l4_timeout_t l4_timeout(l4_timeout_s snd, l4_timeout_s rcv) L4_NOTHROW;
00140
00148 L4_CONSTEXPR L4_INLINE
00149 void l4_snd_timeout(l4_timeout_s snd, l4_timeout_t *to) L4_NOTHROW;
00150
00158 L4_CONSTEXPR L4_INLINE
00159 void l4_rcv_timeout(l4_timeout_s rcv, l4_timeout_t *to) L4_NOTHROW;
00160
00169 L4_CONSTEXPR L4_INLINE
00170 l4_kernel_clock_t l4_timeout_rel_get(l4_timeout_s to) L4_NOTHROW;
00171
00172
00181 L4_CONSTEXPR L4_INLINE
00182 unsigned l4_timeout_is_absolute(l4_timeout_s to) L4_NOTHROW;
00183
00193 L4_CONSTEXPR L4_INLINE
00194 l4_kernel_clock_t l4_timeout_get(l4_kernel_clock_t cur, l4_timeout_s to) L4_NOTHROW;
00195
00203 L4_CONSTEXPR L4_INLINE
00204 l4_timeout_s l4_timeout_from_us(l4_uint64_t us) L4_NOTHROW;
00205
00206 /*
00207  * Implementation
00208  */
00209
00210 L4_CONSTEXPR L4_INLINE
00211 l4_timeout_t l4_ipc_timeout(unsigned snd_man, unsigned snd_exp,
00212                             unsigned rcv_man, unsigned rcv_exp) L4_NOTHROW
00213 {
00214     l4_uint16_t snd = (snd_man & 0x3ff) | ((snd_exp < 10) & 0x7c00);
00215     l4_uint16_t rcv = (rcv_man & 0x3ff) | ((rcv_exp < 10) & 0x7c00);
00216     return l4_timeout((l4_timeout_s){snd}, (l4_timeout_s){rcv});
00217 }
00218
00219
00220 L4_CONSTEXPR L4_INLINE
00221 l4_timeout_t l4_timeout(l4_timeout_s snd, l4_timeout_s rcv) L4_NOTHROW
00222 {
00223     return (l4_timeout_t){ ((l4_uint32_t){snd.t} < 16) | rcv.t };
00224 }
00225
00226
00227 L4_CONSTEXPR L4_INLINE
00228 void l4_snd_timeout(l4_timeout_s snd, l4_timeout_t *to) L4_NOTHROW
00229 {
00230     to->p.snd = snd;
00231 }
00232
00233
00234 L4_CONSTEXPR L4_INLINE
00235 void l4_rcv_timeout(l4_timeout_s rcv, l4_timeout_t *to) L4_NOTHROW
00236 {
00237     to->p.rcv = rcv;
00238 }
00239
00240
00241 L4_CONSTEXPR L4_INLINE
00242 l4_timeout_s l4_timeout_rel(unsigned man, unsigned exp) L4_NOTHROW
00243 {
00244     return (l4_timeout_s){(l4_uint16_t)((man & 0x3ff) | ((exp < 10) & 0x7c00))};
00245 }
00246
00247
00248 L4_CONSTEXPR L4_INLINE
00249 l4_kernel_clock_t l4_timeout_rel_get(l4_timeout_s to) L4_NOTHROW
00250 {
00251     if (to.t == 0)
00252         return ~0ULL;
00253     return (l4_kernel_clock_t)(to.t & 0x3ff) << ((to.t > 10) & 0x1f);
00254 }
00255
00256
00257 L4_CONSTEXPR L4_INLINE
00258 unsigned l4_timeout_is_absolute(l4_timeout_s to) L4_NOTHROW
00259 {

```

```

00260     return to.t & 0x8000;
00261 }
00262
00263
00264 L4_CONSTEXPR L4_INLINE
00265 l4_kernel_clock_t l4_timeout_get(l4_kernel_clock_t cur, l4_timeout_s to) L4_NOTHROW
00266 {
00267     if (l4_timeout_is_absolute(to))
00268         return 0; /* We cannot retrieve the value ... */
00269     else
00270         return cur + l4_timeout_rel_get(to);
00271 }
00272
00273 L4_CONSTEXPR L4_INLINE
00274 l4_timeout_s l4_timeout_from_us(l4_uint64_t us) L4_NOTHROW
00275 {
00276     if (us == 0)
00277         return L4_IPC_TIMEOUT_0;
00278     else if (us == L4_TIMEOUT_US_NEVER || us > L4_TIMEOUT_US_MAX)
00279         return L4_IPC_TIMEOUT_NEVER;
00280     else
00281     {
00282         /* Here it is certain that at least one bit in 'us' is set. */
00283
00284         l4_uint16_t m = 0; // initialization required by constexpr, optimized away
00285         l4_uint16_t v = 0; // initialization required by constexpr, optimized away
00286         int e = (63 - __builtin_clzll(us)) - 9;
00287         if (e < 0)
00288             e = 0;
00289
00290         /* Here it is certain that '0 <= e <= 31' and '1 <= 2^e <= 2^31':
00291          * L4_TIMEOUT_US_MAX = 2^41-1 = 0x000001fffffffffff => e = 31.
00292          * Note: 2^41-1 (0x000001fffffffffff) > 1023*2^31 (0x00001fff800000000). */
00293
00294         m = us >> e;
00295
00296         /* Here it is certain that '1 <= m <= 1023. Consider the following cases:
00297          * o 1 <= us <= 1023: e = 0; 2^e = 1; 1 <= us/1 <= 1023
00298          * o 1024 <= us <= 2047: e = 1; 2^e = 2; 512 <= us/2 <= 1023
00299          * o 2048 <= us <= 4095: e = 2; 2^e = 4; 512 <= us/4 <= 1023
00300          * ...
00301          * o 2^31 <= us <= 2^32-1: e = 22; 512 <= us/2^22 <= 1023
00302          * o 2^40 <= us <= 2^41-1: e = 31; 512 <= us/2^31 <= 1023
00303          *
00304          * Dividing by (1<e) ensures that for all us < 2^41: m < 2^10.
00305          *
00306          * Maximum possible timeout using this format: L4_TIMEOUT_US_MAX = 2^41-1:
00307          * e = 31, m = 1023 => 2'196'875'771'904 us = 610h 14m 35s.
00308          */
00309
00310         /* Without introducing 'v' we had to type-cast the expression to
00311          * l4_uint16_t. This cannot be avoided by declaring m and e_pow_10 as
00312          * l4_uint16_t due to C++ integer promotion. */
00313         v = (e << 10) | m;
00314         return (l4_timeout_s){v};
00315     }
00316 }
00317
00318 #endif

```

## 16.429 l4/sys/\_\_typeinfo.h File Reference

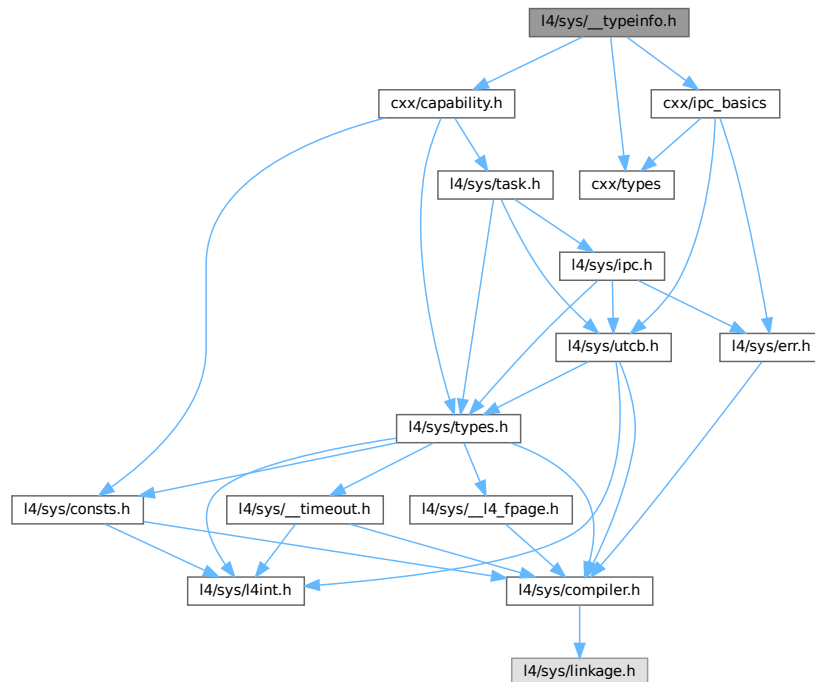
Type information handling.

```

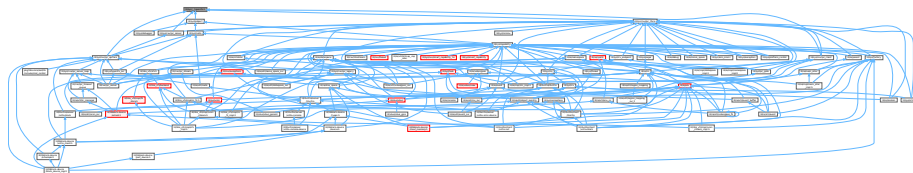
#include "cxx/types"
#include "cxx/ipc_basics"
#include "cxx/capability.h"

```

Include dependency graph for \_\_typeinfo.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [L4::Typeid::P\\_dispatch< LIST >](#)  
*Use for protocol based dispatch stage.*
- struct [L4::Typeid::Detail::Rpc\\_end](#)  
*Internal end-of-list marker.*
- struct [L4::Typeid::Detail::\\_Rpc< OPCODE, O, X >](#)  
*Empty list of RPCs.*
- struct [L4::Typeid::Detail::\\_Rpc< OPCODE, O, R, X... >](#)  
*Non-empty list of RPCs.*
- struct [L4::Typeid::Detail::\\_Rpc< OPCODE, O, R, X... >::Rpc< Y >](#)  
*Find the given RPC in the list.*
- struct [L4::Typeid::Detail::\\_Rpc< OPCODE, O, Default\\_op< R > >::Rpc< Y >](#)  
*Find the given RPC in the list.*
- struct [L4::Typeid::Raw\\_ipc< CLASS >](#)

- *RPCs list for passing raw incoming IPC to the server object.*
- struct [L4::Typeid::Rpc< RPCS >](#)
  - *Standard list of RPCs of an interface.*
- struct [L4::Typeid::Rpc\\_code< OPCODE\\_TYPE >](#)
  - *List of RPCs of an interface using a special opcode type.*
- struct [L4::Typeid::Rpc\\_code< OPCODE\\_TYPE >::F< RPCS >](#)
- struct [L4::Typeid::Rpc\\_nocode< OPERATION >](#)
  - *List of RPCs of an interface using a single operation without an opcode.*
- struct [L4::Typeid::Rpc\\_sys< ARG >](#)
  - *List of RPCs typically used for kernel interfaces.*
- struct [L4::Type\\_info](#)
  - *Dynamic Type Information for [L4Re](#) Interfaces.*
- class [L4::Type\\_info::Demand](#)
  - *Data type for expressing the needed receive buffers at the server-side of an interface.*
- struct [L4::Type\\_info::Demand\\_t< CAPS, FLAGS, MEM, PORTS >](#)
  - *Template type statically describing demand of receive buffers.*
- struct [L4::Type\\_info::Demand\\_union\\_t< D1, D2 >](#)
  - *Template type statically describing the combination of two [Demand](#) object.*
- struct [L4::Kobject\\_typeid< T >](#)
  - *[Meta](#) object for handling access to type information of [Kobjects](#).*
- struct [L4::Kobject\\_typeid< void >](#)
  - *Minimalistic ID for `void` interface.*
- class [L4::Kobject\\_t< Derived, Base, PROTO, S\\_DEMAND >](#)
  - *Helper class to create an [L4Re](#) interface class that is derived from a single base class.*
- class [L4::Kobject\\_2t< Derived, Base1, Base2, PROTO, S\\_DEMAND >](#)
  - *Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject\\_t](#)).*
- struct [L4::Kobject\\_3t< Derived, Base1, Base2, Base3, PROTO, S\\_DEMAND >](#)
  - *Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject\\_t](#)).*
- struct [L4::Proto\\_t< P >](#)
  - *Data type for defining protocol numbers.*

## Namespaces

- namespace [L4](#)
  - *[L4](#) low-level kernel interface.*
- namespace [L4::Typeid](#)
  - *Definition of interface data-type helpers.*

## Typedefs

- typedef int [L4::Opcode](#)
  - *Data type for RPC opcodes.*

## Enumerations

- enum { [L4::PROTO\\_ANY](#) = 0 , [L4::PROTO\\_EMPTY](#) = -19 }



## Functions

- `template<typename T>`  
`Type_info` const \* `L4::kobject_typeid` () noexcept  
*Get the `L4::Type_info` for the `L4Re` interface given in `T`.*

### 16.429.1 Detailed Description

Type information handling.

Definition in file `__typeinfo.h`.

## 16.430 \_\_typeinfo.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * Copyright (C) 2014-2017, 2019, 2022 Kernkonzept GmbH.
00007  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00008  */
00009 /*
00010  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction. Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License. This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 #pragma once
00027 #pragma GCC system_header
00028
00029 #include "cxx/types"
00030 #include "cxx/ipc_basics"
00031 #include "cxx/capability.h"
00032
00033 #if defined(__GXX_RTTI) && !defined(L4_NO_RTTI)
00034 #   include <typeinfo>
00035 #   typedef std::type_info const *L4_std_type_info_ptr;
00036 #   define L4_KOBJECT_META_RTTI(type) (&typeid(type))
00037 #   inline char const *L4_kobject_type_name(L4_std_type_info_ptr n) noexcept
00038 #   { return n ? n->name() : 0; }
00039 #else
00040 #   typedef void const *L4_std_type_info_ptr;
00041 #   define L4_KOBJECT_META_RTTI(type) (0)
00042 #   inline char const *L4_kobject_type_name(L4_std_type_info_ptr) noexcept
00043 #   { return 0; }
00044 #endif
00045
00046 namespace L4 {
00047     typedef int Opcode;
00048     // internal max helpers
00049     namespace __I {
00050         // internal max of A and B helper
00051         template< unsigned char A, unsigned char B>
00052         struct Max { enum { Res = A > B ? A : B }; };
00053     } // namespace __I
00054
00055     enum
00056     {
00057         PROTO_ANY = 0,
00058         PROTO_EMPTY = -19,
00059     };
00060
00061 namespace Typeid {
```

```

00083     using namespace L4::Types;
00084
00085     /*****/
00093     template<long P, typename T>
00094     struct Iface
00095     {
00096         typedef Iface type;
00097         typedef T iface_type;
00098         enum { Proto = P };
00099     };
00100
00101
00102     /*****/
00108     struct Iface_list_end
00109     {
00110         typedef Iface_list_end type;
00111         static bool contains(long) noexcept { return false; }
00112     };
00113
00114
00122     template<typename I, typename N = Iface_list_end>
00123     struct Iface_list
00124     {
00125         typedef Iface_list<I, N> type;
00126
00127         typedef typename I::iface_type iface_type;
00128         typedef N Next;
00129
00130         enum { Proto = I::Proto };
00131
00132         static bool contains(long proto) noexcept
00133         { return (proto == Proto) || Next::contains(proto); }
00134     };
00135
00136     // do not insert PROTO_EMPTY interfaces
00137     template<typename I, typename N>
00138     struct Iface_list<Iface<PROTO_EMPTY, I>, N> : N {};
00139
00140     // do not insert 'void' type interfaces
00141     template<long P, typename N>
00142     struct Iface_list<Iface<P, void>, N> : N {};
00143
00144
00145     /*****/
00146     /*
00147     * \internal
00148     * Test if an interface I is in list L
00149     * \tparam I   Interface for lookup
00150     * \tparam L   Iface_list for search
00151     */
00152     template< typename I, typename L >
00153     struct _In_list;
00154
00155     template< typename I >
00156     struct _In_list<I, Iface_list_end> : False {};
00157
00158     template< typename I, typename N >
00159     struct _In_list<I, Iface_list<I, N> > : True {};
00160
00161     template< typename I, typename I2, typename N >
00162     struct _In_list<I, Iface_list<I2, N> > : _In_list<I, typename N::type> {};
00163
00164     template<typename I, typename L>
00165     struct In_list : _In_list<typename I::type, typename L::type> {};
00166
00167
00168     /*****/
00169     /*
00170     * \internal
00171     * Add Helper: add I to interface list L if ADD is true
00172     * \ingroup l4_cxx_ipc_internal
00173     */
00174     template< bool ADD, typename I, typename L>
00175     struct _Iface_list_add;
00176
00177     template< typename I, typename L>
00178     struct _Iface_list_add<false, I, L> : L {};
00179
00180     template< typename I, typename L>
00181     struct _Iface_list_add<true, I, L> : Iface_list<I, L> {};
00182
00183     /*
00184     * \internal
00185     * Add Helper: add I to interface list L if not already in L.
00186     * \ingroup l4_cxx_ipc_internal
00187     */
00188     template< typename I, typename L >

```

```

00189 struct Iface_list_add :
00190     _Iface_list_add<
00191         !In_list<I, typename L::type>::value, I, typename L::type>
00192     >;
00193
00194 /*****/
00195 /*
00196  * \internal
00197  * Helper: checking for a conflict between I2 and I2.
00198  * A conflict means I1 and I2 have the same protocol ID but a different
00199  * iface_type.
00200  */
00201 template< typename I1, typename I2 >
00202 struct __Iface_conflict : Bool<I1::Proto != PROTO_EMPTY && I1::Proto == I2::Proto> {};
00203
00204 template< typename I >
00205 struct __Iface_conflict<I, I> : False {};
00206
00207 /*
00208  * \internal
00209  * Helper: checking for a conflict between I and any interface in LIST.
00210  */
00211 template< typename I, typename LIST >
00212 struct _Iface_conflict;
00213
00214 template< typename I >
00215 struct _Iface_conflict<I, Iface_list_end> : False {};
00216
00217 template< typename I, typename I2, typename LIST >
00218 struct _Iface_conflict<I, Iface_list<I2, LIST> > :
00219     Bool<__Iface_conflict<I, I2>::value || _Iface_conflict<I, typename LIST::type>::value>
00220     >;
00221
00222 template< typename I, typename LIST >
00223 struct Iface_conflict : _Iface_conflict<typename I::type, typename LIST::type> {};
00224
00225 /*****/
00226 /*
00227  * \internal
00228  * Helper: merge two interface lists
00229  */
00230 template< typename L1, typename L2 >
00231 struct _Merge_list;
00232
00233 template< typename L >
00234 struct _Merge_list<Iface_list_end, L> : L {};
00235
00236 template< typename I, typename L1, typename L2 >
00237 struct _Merge_list<Iface_list<I, L1>, L2> :
00238     _Merge_list<typename L1::type, typename Iface_list_add<I, L2>::type> {};
00239
00240 template<typename L1, typename L2>
00241 struct Merge_list : _Merge_list<typename L1::type, typename L2::type> {};
00242
00243 /*****/
00244 /*
00245  * \internal
00246  * check for conflicts among all interfaces in L1 with any interfaces in L2.
00247  */
00248 template< typename L1, typename L2 >
00249 struct _Conflict;
00250
00251 template< typename L >
00252 struct _Conflict<Iface_list_end, L> : False {};
00253
00254 template< typename I, typename L1, typename L2 >
00255 struct _Conflict<Iface_list<I, L1>, L2> :
00256     Bool<Iface_conflict<I, typename L2::type>::value
00257         || _Conflict<typename L1::type, typename L2::type>::value> {};
00258
00259 template< typename L1, typename L2 >
00260 struct Conflict : _Conflict<typename L1::type, typename L2::type> {};
00261
00262 // to be removed -----
00263 // p_dispatch code -- for legacy dispatch -----
00264 /*****/
00265 /*
00266  * \internal
00267  * helper: Dispatch helper for calling server-side p_dispatch() functions.
00268  */
00269 template<typename LIST>
00270 struct _P_dispatch;
00271
00272 // No matching dispatcher found
00273 template<>
00274 struct _P_dispatch<Iface_list_end>
00275 {

```

```

00280     template< typename THIS, typename A1, typename A2 >
00281     static int f(THIS *, long, A1, A2 &) noexcept
00282     { return -L4_EBADPROTO; }
00283 };
00284
00285
00286 // call matching p_dispatch() function
00287 template< typename I, typename LIST >
00288 struct _P_dispatch<Iface_list<I, LIST> >
00289 {
00290     // special handling for the meta protocol, to avoid 'using' murx
00291     template< typename THIS, typename A1, typename A2 >
00292     static int _f(THIS self, A1, A2 &a2, True::type)
00293     {
00294         return self->dispatch_meta_request(a2);
00295     }
00296
00297     // normal p_dispatch() dispatching
00298     template< typename THIS, typename A1, typename A2 >
00299     static int _f(THIS self, A1 a1, A2 &a2, False::type)
00300     {
00301         return self->p_dispatch(reinterpret_cast<typename I::iface_type *>(0),
00302                                 a1, a2);
00303     }
00304
00305     // dispatch function with switch for meta protocol
00306     template< typename THIS, typename A1, typename A2 >
00307     static int f(THIS *self, long proto, A1 a1, A2 &a2)
00308     {
00309         if (I::Proto == proto)
00310             return _f(self, a1, a2,
00311                       Bool<I::Proto == static_cast<long>(L4_PROTO_META)>());
00312
00313         return _P_dispatch<typename LIST::type>::f(self, proto, a1, a2);
00314     }
00315 };
00316
00317 template<typename LIST>
00318 struct P_dispatch : _P_dispatch<typename LIST::type> {};
00319 // end: p_dispatch -----
00320 // end: to be removed -----
00321
00322 template<typename RPC> struct Default_op;
00323
00324 namespace Detail {
00325
00326     struct Rpcs_end
00327     {
00328         typedef void opcode_type;
00329         typedef Rpcs_end rpc;
00330         typedef Rpcs_end type;
00331     };
00332
00333     template<typename O1, typename O2, typename RPCS>
00334     struct _Rpc : _Rpc<typename RPCS::next::rpc, O2, typename RPCS::next::type> {};
00335
00336     template<typename O1, typename O2>
00337     struct _Rpc<O1, O2, Rpcs_end> {};
00338
00339     template<typename OP, typename RPCS>
00340     struct _Rpc<OP, OP, RPCS> : RPCS
00341     {
00342         typedef _Rpc type;
00343     };
00344
00345     template<typename OP, typename RPCS>
00346     struct Rpc : _Rpc<typename RPCS::rpc, OP, RPCS> {};
00347
00348     template<typename T, unsigned CODE>
00349     struct _Get_opcode
00350     {
00351         template<bool, typename> struct Invalid_opcode {};
00352         template<typename X> struct Invalid_opcode<true, X>;
00353
00354     private:
00355         template<typename U, U> struct _chk;
00356         template<typename U> static long _opc(_chk<int, U::Opcode> *);
00357         template<typename U> static char _opc(...);
00358
00359         template<unsigned SZ, typename U>
00360         struct _Opc { enum { value = CODE }; };
00361
00362         template<typename U>
00363         struct _Opc<sizeof(long), U> { enum { value = U::Opcode }; };
00364
00365     public:
00366         enum { value = _Opc<sizeof(_opc<T>()), T::value };
00367

```

```

00371     Invalid_opcode<(value < CODE), T> invalid_opcode;
00372 };
00373
00375 template<typename OPCODE, unsigned O, typename ...X>
00376 struct _Rpc : Rpc_end {};
00377
00379 template<typename OPCODE, unsigned O, typename R, typename ...X>
00380 struct _Rpc<OPCODE, O, R, X...>
00381 {
00383     typedef _Rpc type;
00385     typedef OPCODE opcode_type;
00387     typedef R rpc;
00389     typedef typename _Rpc<OPCODE, _Get_opcode<R, O>::value + 1, X...>::type next;
00391     enum { Opcode = _Get_opcode<R, O>::value };
00393     template<typename Y> struct Rpc : Typeid::Detail::Rpc<Y, _Rpc> {};
00394 };
00395
00396 template<typename OPCODE, unsigned O, typename R>
00397 struct _Rpc<OPCODE, O, Default_op<R> >
00398 {
00400     typedef _Rpc type;
00402     typedef void opcode_type;
00404     typedef R rpc;
00406     typedef Rpc_end next;
00408     enum { Opcode = -99 };
00410     template<typename Y> struct Rpc : Typeid::Detail::Rpc<Y, _Rpc> {};
00411 };
00412
00413 } // namespace Detail
00414
00422 template<typename CLASS>
00423 struct Raw_ipc
00424 {
00425     typedef Raw_ipc type;
00426     typedef Detail::Rpc_end next;
00427     typedef void opcode_type;
00428 };
00429
00438 template<typename ...RPCS>
00439 struct Rpc : Detail::_Rpc<L4::Opcode, 0, RPCS...> {};
00440
00449 template<typename OPCODE_TYPE>
00450 struct Rpc_code
00451 {
00455     template<typename ...RPCS>
00456     struct F : Detail::_Rpc<OPCODE_TYPE, 0, RPCS...> {};
00457 };
00458
00464 template<typename OPERATION>
00465 struct Rpc_nocode : Detail::_Rpc<void, 0, OPERATION> {};
00466
00475 template<typename ...ARG>
00476 struct Rpc_sys : Detail::_Rpc<l4_umword_t, 0, ARG...> {};
00477
00478 template<typename CLASS>
00479 struct Rights
00480 {
00481     unsigned rights;
00482     Rights(unsigned rights) noexcept : rights(rights) {}
00483     unsigned operator & (unsigned rhs) const noexcept { return rights & rhs; }
00484 };
00485
00486 } // namespace Typeid
00487
00510 struct L4_EXPORT Type_info
00511 {
00517     class L4_EXPORT Demand
00518     {
00519     private:
00521         static unsigned char max(unsigned char a, unsigned char b) noexcept
00522         { return a > b ? a : b; }
00523
00524     public:
00525         unsigned char caps;
00526         unsigned char flags;
00527         unsigned char mem;
00528         unsigned char ports;
00529
00537         explicit
00538         Demand(unsigned char caps = 0, unsigned char flags = 0,
00539                unsigned char mem = 0, unsigned char ports = 0) noexcept
00540             : caps(caps), flags(flags), mem(mem), ports(ports) {}
00541
00543         bool no_demand() const noexcept
00544         { return caps == 0 && mem == 0 && ports == 0 && flags == 0; }
00545
00547         Demand operator | (Demand const &rhs) const noexcept

```

```

00548     {
00549         return Demand(max(caps, rhs.caps), flags | rhs.flags,
00550                       max(mem, rhs.mem), max(ports, rhs.ports));
00551     }
00552 };
00553
00562 template<unsigned char CAPS = 0, unsigned char FLAGS = 0,
00563         unsigned char MEM = 0, unsigned char PORTS = 0>
00564 struct Demand_t : Demand
00565 {
00566     enum
00567     {
00568         Caps = CAPS,
00569         Flags = FLAGS,
00570         Mem = MEM,
00571         Ports = PORTS
00572     };
00573     Demand_t() noexcept : Demand(CAPS, FLAGS, MEM, PORTS) {}
00574 };
00575
00583 template<typename D1, typename D2>
00584 struct Demand_union_t : Demand_t<__I::Max<D1::Caps, D2::Caps>::Res,
00585                                   D1::Flags | D2::Flags,
00586                                   __I::Max<D1::Mem, D2::Mem>::Res,
00587                                   __I::Max<D1::Ports, D2::Ports>::Res>
00588 {};
00589
00590 L4_std_type_info_ptr _type;
00591 Type_info const *const *_bases;
00592 unsigned _num_bases;
00593 long _proto;
00594
00595 L4_std_type_info_ptr type() const noexcept { return _type; }
00596 Type_info const *base(unsigned idx) const noexcept { return _bases[idx]; }
00597 unsigned num_bases() const noexcept { return _num_bases; }
00598 long proto() const noexcept { return _proto; }
00599 char const *name() const noexcept { return L4_kobject_type_name(type()); }
00600 bool has_proto(long proto) const noexcept
00601 {
00602     if (_proto && _proto == proto)
00603         return true;
00604
00605     if (!proto)
00606         return false;
00607
00608     for (unsigned i = 0; i < _num_bases; ++i)
00609         if (base(i)->has_proto(proto))
00610             return true;
00611
00612     return false;
00613 }
00614 };
00615
00621 template<typename T> struct Kobject_typeid
00622 {
00633     typedef typename T::__Kobject_typeid::Demand Demand;
00634     typedef typename T::__Iface::iface_type Iface;
00635     typedef typename T::__Iface_list Iface_list;
00636
00641     static Type_info const *id() noexcept { return &T::__Kobject_typeid::_m; }
00642
00650     static Type_info::Demand demand() noexcept
00651     { return T::__Kobject_typeid::Demand(); }
00652
00653     // to be removed -----
00654     // p_dispatch -----
00670     template<typename THIS, typename A1, typename A2>
00671     static int proto_dispatch(THIS *self, long proto, A1 a1, A2 &a2)
00672     { return typeid::P_dispatch<typename T::__Iface_list>::f(self, proto, a1, a2); }
00673     // p_dispatch -----
00674     // end: to be removed -----
00675 };
00676
00678 template<> struct Kobject_typeid<void>
00679 {
00680     typedef Type_info::Demand_t<> Demand;
00681 };
00682
00691 template<typename T>
00692 inline
00693 Type_info const *kobject_typeid() noexcept
00694 { return Kobject_typeid<T>::id(); }
00695
00700 #define L4___GEN_TI(t...) //
00701 Type_info const t::__Kobject_typeid::_m = //
00702 { //
00703     L4_KOBJECT_META_RTTI(Derived), //

```

```

00704     &t::__Kobject_typeid::_b[0],
00705     sizeof(t::__Kobject_typeid::_b) / sizeof(t::__Kobject_typeid::_b[0]),
00706     PROTO
00707 }
00708
00713 #define L4___GEN_TI_MEMBERS(BASE_DEMAND...)
00714 private:
00715     template< typename T > friend struct Kobject_typeid;
00716 protected:
00717     struct __Kobject_typeid {
00718         typedef Type_info::Demand_union_t<S_DEMAND, BASE_DEMAND> Demand;
00719         static Type_info const *const _b[];
00720         static Type_info const _m;
00721     };
00722 public:
00723     static long const Protocol = PROTO;
00724     typedef L4::Typeid::Rights<Class> Rights;
00725
00754 template<
00755     typename Derived,
00756     typename Base,
00757     long PROTO = PROTO_ANY,
00758     typename S_DEMAND = Type_info::Demand_t<>
00759 >
00760 class Kobject_t : public Base
00761 {
00762 protected:
00764     typedef Derived Class;
00766     typedef Typeid::Iface<PROTO, Derived> __Iface;
00768     typedef Typeid::Merge_list<
00769         Typeid::Iface_list<__Iface>, typename Base::__Iface_list
00770     > __Iface_list;
00771
00773     static void __check_protocols__() noexcept
00774     {
00775         typedef Typeid::Iface_conflict<__Iface, typename Base::__Iface_list> Base_conflict;
00776         static_assert(!Base_conflict::value, "ambiguous protocol ID: protocol also used by Base");
00777     }
00778
00780     L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }
00781
00782     // Generate the remaining type information
00783     L4___GEN_TI_MEMBERS(typename Base::__Kobject_typeid::Demand)
00784 };
00785
00786
00788 template< typename Derived, typename Base, long PROTO, typename S_DEMAND>
00789 Type_info const *const
00790 Kobject_t<Derived, Base, PROTO, S_DEMAND>::
00791     __Kobject_typeid::_b[] = { &Base::__Kobject_typeid::_m };
00793
00798 template< typename Derived, typename Base, long PROTO, typename S_DEMAND>
00799 L4___GEN_TI(Kobject_t<Derived, Base, PROTO, S_DEMAND>);
00800
00801
00831 template<
00832     typename Derived,
00833     typename Base1,
00834     typename Base2,
00835     long PROTO = PROTO_ANY,
00836     typename S_DEMAND = Type_info::Demand_t<>
00837 >
00838 class Kobject_2t : public Base1, public Base2
00839 {
00840 protected:
00842     typedef Derived Class;
00844     typedef Typeid::Iface<PROTO, Derived> __Iface;
00846     typedef Typeid::Merge_list<
00847         Typeid::Iface_list<__Iface>,
00848         Typeid::Merge_list<
00849             typename Base1::__Iface_list,
00850             typename Base2::__Iface_list
00851         >
00852     > __Iface_list;
00853
00855     static void __check_protocols__() noexcept
00856     {
00857         typedef typename Base1::__Iface_list Base1_proto_list;
00858         typedef typename Base2::__Iface_list Base2_proto_list;
00859
00860         typedef Typeid::Iface_conflict<__Iface, Base1_proto_list> Base1_conflict;
00861         typedef Typeid::Iface_conflict<__Iface, Base2_proto_list> Base2_conflict;
00862         static_assert(!Base1_conflict::value, "ambiguous protocol ID, also in Base1");
00863         static_assert(!Base2_conflict::value, "ambiguous protocol ID, also in Base2");
00864
00865         typedef Typeid::Conflict<Base1_proto_list, Base2_proto_list> Bases_conflict;
00866         static_assert(!Bases_conflict::value, "ambiguous protocol IDs in base classes");

```

```

00867     }
00868
00869     // disambiguate cap()
00870     l4_cap_idx_t cap() const noexcept
00871     { return Basel::cap(); }
00872
00873     L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }
00874
00875     L4__GEN_TI_MEMBERS(Type_info::Demand_union_t<
00876         typename Basel::__Kobject_typeid::Demand,
00877         typename Base2::__Kobject_typeid::Demand>
00878     )
00879
00880 public:
00881     // Provide non-ambiguous conversion to Kobject
00882     operator Kobject const & () const noexcept
00883     { return *static_cast<Basel const *>(this); }
00884
00885     // Provide non-ambiguous access of dec_refcnt()
00886     l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
00887     noexcept (noexcept (static_cast<Basel*>(nullptr)->dec_refcnt(diff, utcb)))
00888     { return Basel::dec_refcnt(diff, utcb); }
00889 };
00890
00891
00892
00893 template< typename Derived, typename Basel, typename Base2,
00894     long PROTO, typename S_DEMAND >
00895 Type_info const *const
00896 Kobject_2t<Derived, Basel, Base2, PROTO, S_DEMAND>::__Kobject_typeid::_b[] =
00897 {
00898     &Basel::__Kobject_typeid::_m,
00899     &Base2::__Kobject_typeid::_m
00900 };
00901
00902
00903 template< typename Derived, typename Basel, typename Base2,
00904     long PROTO, typename S_DEMAND >
00905 L4__GEN_TI(Kobject_2t<Derived, Basel, Base2, PROTO, S_DEMAND>);
00906
00907
00908 template<
00909     typename Derived,
00910     typename Basel,
00911     typename Base2,
00912     typename Base3,
00913     long PROTO = PROTO_ANY,
00914     typename S_DEMAND = Type_info::Demand_t<>
00915 >
00916 struct Kobject_3t : Basel, Base2, Base3
00917 {
00918 protected:
00919     typedef Derived Class;
00920     typedef Typeid::Iface<PROTO, Derived> __Iface;
00921     typedef Typeid::Merge_list<
00922         Typeid::Iface_list<__Iface>,
00923         Typeid::Merge_list<
00924             typename Basel::__Iface_list,
00925             Typeid::Merge_list<
00926                 typename Base2::__Iface_list,
00927                 typename Base3::__Iface_list
00928             >
00929         >
00930     > __Iface_list;
00931
00932     static void __check_protocols__() noexcept
00933     {
00934         typedef typename Basel::__Iface_list Basel_proto_list;
00935         typedef typename Base2::__Iface_list Base2_proto_list;
00936         typedef typename Base3::__Iface_list Base3_proto_list;
00937
00938         typedef Typeid::Iface_conflict<__Iface, Basel_proto_list> Basel_conflict;
00939         typedef Typeid::Iface_conflict<__Iface, Base2_proto_list> Base2_conflict;
00940         typedef Typeid::Iface_conflict<__Iface, Base3_proto_list> Base3_conflict;
00941
00942         static_assert(!Basel_conflict::value, "ambiguous protocol ID, also in Basel");
00943         static_assert(!Base2_conflict::value, "ambiguous protocol ID, also in Base2");
00944         static_assert(!Base3_conflict::value, "ambiguous protocol ID, also in Base3");
00945
00946         typedef Typeid::Conflict<Basel_proto_list, Base2_proto_list> Conflict_bases12;
00947         typedef Typeid::Conflict<Basel_proto_list, Base3_proto_list> Conflict_bases13;
00948         typedef Typeid::Conflict<Base2_proto_list, Base3_proto_list> Conflict_bases23;
00949
00950         static_assert(!Conflict_bases12::value, "ambiguous protocol IDs in base classes: Basel and Base2");
00951         static_assert(!Conflict_bases13::value, "ambiguous protocol IDs in base classes: Basel and Base3");
00952         static_assert(!Conflict_bases23::value, "ambiguous protocol IDs in base classes: Base2 and

```



```

        Base3");
00982     }
00983
00984     // disambiguate cap()
00985     l4_cap_idx_t cap() const noexcept
00986     { return Base1::cap(); }
00987
00988     L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }
00989
00990     L4__GEN_TI_MEMBERS(Type_info::Demand_union_t<Type_info::Demand_union_t<
00991         typename Base1::__Kobject_typeid::Demand,
00992         typename Base2::__Kobject_typeid::Demand>,
00993         typename Base3::__Kobject_typeid::Demand>
00994     )
00995 }
00996
00997 public:
00998     // Provide non-ambiguous conversion to Kobject
00999     operator Kobject const & () const noexcept
01000     { return *static_cast<Base1 const *>(this); }
01001
01002     // Provide non-ambiguous access of dec_refcnt()
01003     l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
01004     noexcept(noexcept(static_cast<Base1*>(nullptr)->dec_refcnt(diff, utcb)))
01005     { return Base1::dec_refcnt(diff, utcb); }
01006 };
01007
01008
01009 template< typename Derived, typename Base1, typename Base2, typename Base3,
01010     long PROTO, typename S_DEMAND >
01011 Type_info const *const
01012 Kobject_3t<Derived, Base1, Base2, Base3, PROTO, S_DEMAND>::__Kobject_typeid::_b[] =
01013 {
01014     &Base1::__Kobject_typeid::_m,
01015     &Base2::__Kobject_typeid::_m,
01016     &Base3::__Kobject_typeid::_m
01017 };
01018
01019 template< typename Derived, typename Base1, typename Base2, typename Base3,
01020     long PROTO, typename S_DEMAND >
01021 L4__GEN_TI(Kobject_3t<Derived, Base1, Base2, Base3, PROTO, S_DEMAND>);
01022
01023 }
01024
01025 #if __cplusplus >= 201103L
01026 namespace L4 {
01027
01028     template< typename ...T >
01029     struct Kobject_demand;
01030
01031     template<>
01032     struct Kobject_demand<> : Type_info::Demand_t<> {};
01033
01034     template<typename T>
01035     struct Kobject_demand<T> : Kobject_typeid<T>::Demand {};
01036
01037     template<typename T1, typename ...T2>
01038     struct Kobject_demand<T1, T2...> :
01039         Type_info::Demand_union_t<typename Kobject_typeid<T1>::Demand,
01040             Kobject_demand<T2...> >
01041     {};
01042
01043 namespace Typeid_xx {
01044
01045     template<typename ...LISTS>
01046     struct Merge_list;
01047
01048     template<typename L>
01049     struct Merge_list<L> : L {};
01050
01051     template<typename L1, typename L2>
01052     struct Merge_list<L1, L2> : Typeid::Merge_list<L1, L2> {};
01053
01054     template<typename L1, typename L2, typename ...LISTS>
01055     struct Merge_list<L1, L2, LISTS...> :
01056         Merge_list<typename Typeid::Merge_list<L1, L2>::type, LISTS...> {};
01057
01058     template< typename I, typename ...LIST >
01059     struct Iface_conflict;
01060
01061     template< typename I >
01062     struct Iface_conflict<I> : Typeid::False {};
01063
01064     template< typename I, typename L, typename ...LIST >
01065     struct Iface_conflict<I, L, LIST...> :
01066         Typeid::Bool<Typeid::Iface_conflict<I::type, typename L::type>::value
01067             || Iface_conflict<I, LIST...>::value>
01068     {};
01069
01070 }
01071
01072 #endif

```

```

01081     {};
```

```

01082
01083     template< typename ...LIST >
01084     struct Conflict;
```

```

01085
01086     template< typename L >
01087     struct Conflict<L> : Typeid::False {};
```

```

01088
01089     template< typename L1, typename L2, typename ...LIST >
01090     struct Conflict<L1, L2, LIST...> :
01091         Typeid::Bool<Typeid::Conflict<typename L1::type, typename L2::type>::value
01092             || Conflict<L1, LIST...>::value
01093             || Conflict<L2, LIST...>::value>
01094     {};
```

```

01095
01096     template< typename T >
01097     struct Is_demand
01098     {
01099         static long test(Type_info::Demand const *);
01100         static char test(...);
01101         enum { value = sizeof(test(static_cast<T*>(nullptr))) == sizeof(long) };
01102     };
01103
01104     template< typename T, typename ... >
01105     struct First : T { typedef T type; };
01106 } // Typeid
01107
01113 template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>
01114 struct __Kobject_base : BASES...
01115 {
01116 protected:
01117     typedef Derived Class;
01118     typedef Typeid::Iface<PROTO, Derived> __Iface;
01119     typedef Typeid_xx::Merge_list<
01120         Typeid::Iface_list<__Iface>,
01121         typename BASES::__Iface_list...
01122     > __Iface_list;
01123
01124     static void __check_protocols__() noexcept
01125     {
01126         typedef Typeid_xx::Iface_conflict<__Iface, typename BASES::__Iface_list...> Conflict;
01127         static_assert(!Conflict::value, "ambiguous protocol ID, protocol also used in base class");
01128
01129         typedef Typeid_xx::Conflict<typename BASES::__Iface_list...> Base_conflict;
01130         static_assert(!Base_conflict::value, "ambiguous protocol IDs in base classes");
01131     }
01132
01133     // disambiguate cap()
01134     l4_cap_idx_t cap() const noexcept
01135     { return Typeid_xx::First<BASES...>::type::cap(); }
01136
01137     L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }
01138
01139     L4__GEN_TI_MEMBERS(Kobject_demand<BASES...>)
01140
01141 private:
01142     // This function returns the first base class (used below)
01143     template<typename B1, typename ...> struct Basel { typedef B1 type; };
01144
01145 public:
01146     // Provide non-ambiguous conversion to Kobject
01147     operator Kobject const & () const noexcept
01148     { return *static_cast<typename Basel<BASES...>::type const *>(this); }
01149
01150     // Provide non-ambiguous access of dec_refcnt()
01151     l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
01152     noexcept(noexcept(static_cast<typename Basel<BASES...>::type *>(nullptr)
01153         ->dec_refcnt(diff, utcb)))
01154     { return Basel<BASES...>::type::dec_refcnt(diff, utcb); }
01155 };
01156
01158 template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>
01159 Type_info const *const
01160 __Kobject_base<Derived, PROTO, S_DEMAND, BASES...>::__Kobject_typeid::b[] =
01161 {
01162     (&BASES::__Kobject_typeid::_m)...
01163 };
01164
01166 template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>
01167 L4__GEN_TI(__Kobject_base<Derived, PROTO, S_DEMAND, BASES...>);
01168
01169
01170 // Test if there is a Demand argument to Kobject_x
01171 template< typename Derived, long PROTO, bool HAS_DEMAND, typename DEMAND, typename ...ARGS >
01172 struct __Kobject_x_proto;
01173
01174 // YES: pass it to __Kobject_base
```

```

01175 template< typename Derived, long PROTO, typename DEMAND, typename ...BASES>
01176 struct __Kobject_x_proto<Derived, PROTO, true, DEMAND, BASES...> :
01177     __Kobject_base<Derived, PROTO, DEMAND, BASES...> {};
01178
01179 // NO: pass it empty Type_info::Demand_t
01180 template< typename Derived, long PROTO, typename B1, typename ...BASES>
01181 struct __Kobject_x_proto<Derived, PROTO, false, B1, BASES...> :
01182     __Kobject_base<Derived, PROTO, Type_info::Demand_t<>, B1, BASES...> {};
01183
01191 template< long P = PROTO_EMPTY >
01192 struct Proto_t {};
01193
01207 template< typename Derived, typename ...ARGS >
01208 struct Kobject_x;
01209
01210 template< typename Derived, typename A, typename ...ARGS >
01211 struct Kobject_x<Derived, A, ARGS...> :
01212     __Kobject_x_proto<Derived, PROTO_ANY, Typeid_xx::Is_demand<A>::value, A, ARGS...>
01213 {};
01214
01215 template< typename Derived, long PROTO, typename A, typename ...ARGS >
01216 struct Kobject_x<Derived, Proto_t<PROTO>, A, ARGS...> :
01217     __Kobject_x_proto<Derived, PROTO, Typeid_xx::Is_demand<A>::value, A, ARGS...>
01218 {};
01219
01220 }
01221 #endif
01222
01223 #undef L4___GEN_TI
01224 #undef L4___GEN_TI_MEMBERS

```

## 16.431 \_\_vcpu-arm.h

```

00001 /*
00002  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00003  *
00004  * This file is part of L4Re and distributed under the terms of the
00005  * GNU General Public License 2.
00006  * Please see the COPYING-GPL-2 file for details.
00007  *
00008  * As a special exception, you may use this file as part of a free software
00009  * library without restriction. Specifically, if other files instantiate
00010  * templates or use macros or inline functions from this file, or you compile
00011  * this file and link it with other files to produce an executable, this
00012  * file does not by itself cause the resulting executable to be covered by
00013  * the GNU General Public License. This exception does not however
00014  * invalidate any other reasons why the executable file might be covered by
00015  * the GNU General Public License.
00016  */
00017 #pragma once
00018
00019 typedef struct l4_arm_vcpu_e_info_t
00020 {
00021     l4_uint8_t version; // must be 0
00022     l4_uint8_t gic_version;
00023     l4_uint8_t _rsvd0[2];
00024     l4_uint32_t features;
00025     l4_uint32_t _rsvd1[14];
00026     l4_umword_t user[8];
00027 } l4_arm_vcpu_e_info_t;
00028
00029 L4_INLINE void *l4_vcpu_e_ptr(void const *vcpu, unsigned id) L4_NOTHROW;
00030
00031 enum L4_vcpu_e_consts
00032 {
00033     L4_VCPU_E_NUM_LR = 4,
00034 };
00035
00036 L4_INLINE l4_arm_vcpu_e_info_t const *
00037 l4_vcpu_e_info(void const *vcpu) L4_NOTHROW;
00038
00039 L4_INLINE l4_umword_t *
00040 l4_vcpu_e_info_user(void *vcpu) L4_NOTHROW;
00041
00042 L4_INLINE l4_umword_t *
00043 l4_vcpu_e_info_user(void *vcpu) L4_NOTHROW
00044 {
00045     return ((l4_arm_vcpu_e_info_t *)l4_vcpu_e_info(vcpu))->user;
00046 }
00047
00048
00056 L4_INLINE l4_uint32_t
00057 l4_vcpu_e_read_32(void const *vcpu, unsigned id) L4_NOTHROW;

```

```

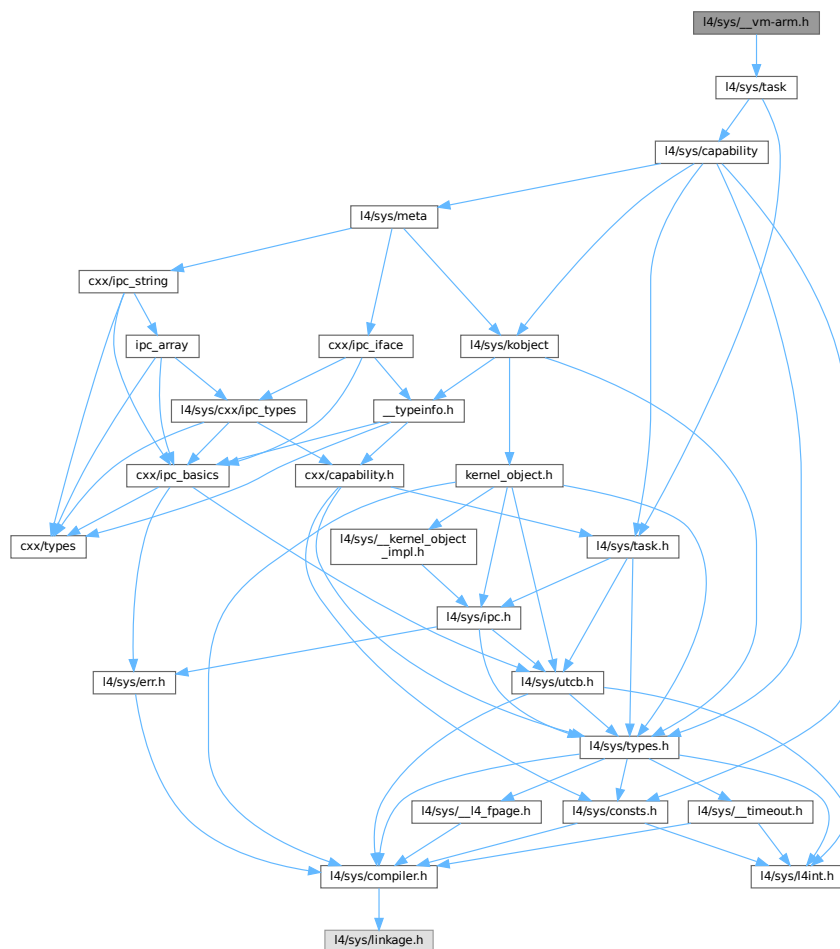
00058
00059 L4_INLINE l4_uint32_t
00060 l4_vcpu_e_read_32(void const *vcpu, unsigned id) L4_NOTHROW
00061 { return *(l4_uint32_t const *)l4_vcpu_e_ptr(vcpu, id); }
00062
00070 L4_INLINE void
00071 l4_vcpu_e_write_32(void *vcpu, unsigned id, l4_uint32_t val) L4_NOTHROW;
00072
00073 L4_INLINE void
00074 l4_vcpu_e_write_32(void *vcpu, unsigned id, l4_uint32_t val) L4_NOTHROW
00075 { *((l4_uint32_t *)l4_vcpu_e_ptr(vcpu, + id)) = val; }
00076
00084 L4_INLINE l4_uint64_t
00085 l4_vcpu_e_read_64(void const *vcpu, unsigned id) L4_NOTHROW;
00086
00087 L4_INLINE l4_uint64_t
00088 l4_vcpu_e_read_64(void const *vcpu, unsigned id) L4_NOTHROW
00089 { return *(l4_uint64_t const *)l4_vcpu_e_ptr(vcpu, id); }
00090
00098 L4_INLINE void
00099 l4_vcpu_e_write_64(void *vcpu, unsigned id, l4_uint64_t val) L4_NOTHROW;
00100
00101 L4_INLINE void
00102 l4_vcpu_e_write_64(void *vcpu, unsigned id, l4_uint64_t val) L4_NOTHROW
00103 { *((l4_uint64_t *)l4_vcpu_e_ptr(vcpu, id)) = val; }
00104
00112 L4_INLINE l4_umword_t
00113 l4_vcpu_e_read(void const *vcpu, unsigned id) L4_NOTHROW;
00114
00115 L4_INLINE l4_umword_t
00116 l4_vcpu_e_read(void const *vcpu, unsigned id) L4_NOTHROW
00117 { return *(l4_umword_t const *)l4_vcpu_e_ptr(vcpu, id); }
00118
00126 L4_INLINE void
00127 l4_vcpu_e_write(void *vcpu, unsigned id, l4_umword_t val) L4_NOTHROW;
00128
00129 L4_INLINE void
00130 l4_vcpu_e_write(void *vcpu, unsigned id, l4_umword_t val) L4_NOTHROW
00131 { *((l4_umword_t *)l4_vcpu_e_ptr(vcpu, id)) = val; }

```

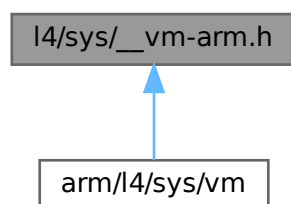
## 16.432 l4/sys/\_\_vm-arm.h File Reference

Virtualization interface.

Include dependency graph for \_\_vm-arm.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4::Vm`

*Virtual machine host address space.*

## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

## 16.432.1 Detailed Description

Virtualization interface.

Definition in file [\\_\\_vm-arm.h](#).

## 16.433 \_\_vm-arm.h

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2018 Adam Lackorzynski <adam@l4re.org>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #pragma once
00023
00024 #include <l4/sys/task>
00025
00026 namespace L4 {
00027
00028 class Vm : public Kobject_t<Vm, Task, L4_PROTO_VM>
00029 {
00030 public:
00041     l4_msgtag_t vgicc_map(l4_fpage_t const vgicc_fpage,
00042                          l4_utcb_t *utcb = l4_utcb()) noexcept
00043     { return l4_task_vgicc_map_u(cap(), vgicc_fpage, utcb); }
00044
00045 protected:
00046     Vm();
00047
00048 private:
00049     Vm(Vm const &);
00050     void operator = (Vm const &);
00051 };
00052
00053 }
```

## 16.434 \_\_vm-svm.h

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
```

```

00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/sys/types.h>
00027
00039 typedef struct l4_vm_svm_vmcb_control_area
00040 {
00041     l4_uint16_t intercept_rd_crX;
00042     l4_uint16_t intercept_wr_crX;
00043
00044     l4_uint16_t intercept_rd_drX;
00045     l4_uint16_t intercept_wr_drX;
00046
00047     l4_uint32_t intercept_exceptions;
00048
00049     l4_uint32_t intercept_instruction0;
00050     l4_uint32_t intercept_instruction1;
00051
00052     l4_uint8_t _reserved0[40];
00053
00054     l4_uint16_t pause_filter_threshold;
00055     l4_uint16_t pause_filter_count;
00056
00057     l4_uint64_t iopm_base_pa;
00058     l4_uint64_t msrpm_base_pa;
00059     l4_uint64_t tsc_offset;
00060     l4_uint64_t guest_asid_tlb_ctl;
00061     l4_uint64_t interrupt_ctl;
00062     l4_uint64_t interrupt_shadow;
00063     l4_uint64_t exitcode;
00064     l4_uint64_t exitinfo1;
00065     l4_uint64_t exitinfo2;
00066     l4_uint64_t exitintinfo;
00067     l4_uint64_t np_enable;
00068
00069     l4_uint8_t _reserved1[16];
00070
00071     l4_uint64_t eventinj;
00072     l4_uint64_t n_cr3;
00073     l4_uint64_t lbr_virtualization_enable;
00074     l4_uint64_t clean_bits;
00075     l4_uint64_t n_rip;
00076
00077     l4_uint8_t _reserved2[816];
00078 } __attribute__((packed)) l4_vm_svm_vmcb_control_area_t;
00079
00084 typedef struct l4_vm_svm_vmcb_state_save_area_seg
00085 {
00086     l4_uint16_t selector;
00087     l4_uint16_t attrib;
00088     l4_uint32_t limit;
00089     l4_uint64_t base;
00090 } __attribute__((packed)) l4_vm_svm_vmcb_state_save_area_seg_t;
00091
00096 typedef struct l4_vm_svm_vmcb_state_save_area
00097 {
00098     struct l4_vm_svm_vmcb_state_save_area_seg es;
00099     struct l4_vm_svm_vmcb_state_save_area_seg cs;
00100     struct l4_vm_svm_vmcb_state_save_area_seg ss;
00101     struct l4_vm_svm_vmcb_state_save_area_seg ds;
00102     struct l4_vm_svm_vmcb_state_save_area_seg fs;
00103     struct l4_vm_svm_vmcb_state_save_area_seg gs;
00104     struct l4_vm_svm_vmcb_state_save_area_seg gdt;
00105     struct l4_vm_svm_vmcb_state_save_area_seg ldtr;
00106     struct l4_vm_svm_vmcb_state_save_area_seg idtr;
00107     struct l4_vm_svm_vmcb_state_save_area_seg tr;
00108
00109     l4_uint8_t _reserved0[43];
00110
00111     l4_uint8_t cpl;
00112
00113     l4_uint32_t _reserved1;
00114
00115     l4_uint64_t efer;
00116
00117     l4_uint8_t _reserved2[112];
00118
00119     l4_uint64_t cr4;
00120     l4_uint64_t cr3;
00121     l4_uint64_t cr0;
00122     l4_uint64_t dr7;
00123     l4_uint64_t dr6;

```

```

00124  l4_uint64_t rflags;
00125  l4_uint64_t rip;
00126
00127  l4_uint8_t _reserved3[88];
00128
00129  l4_uint64_t rsp;
00130
00131  l4_uint8_t _reserved4[24];
00132
00133  l4_uint64_t rax;
00134  l4_uint64_t star;
00135  l4_uint64_t lstar;
00136  l4_uint64_t cstar;
00137  l4_uint64_t sfmask;
00138  l4_uint64_t kernelgsbase;
00139  l4_uint64_t sysenter_cs;
00140  l4_uint64_t sysenter_esp;
00141  l4_uint64_t sysenter_eip;
00142  l4_uint64_t cr2;
00143
00144  l4_uint8_t _reserved5[32];
00145
00146  l4_uint64_t g_pat;
00147  l4_uint64_t dbgctl;
00148  l4_uint64_t br_from;
00149  l4_uint64_t br_to;
00150  l4_uint64_t lastexcpfrom;
00151  l4_uint64_t last_excpto;
00152
00153  // this field is _NOT_ part of the official VMCB specification
00154  // a (userlevel) VMM needs this for proper FPU state virtualization
00155  l4_uint64_t xcr0;
00156
00157  l4_uint8_t _reserved6[2400];
00158 } __attribute__((packed)) l4_vm_svm_vmc_b_state_save_area_t;
00159
00160
00161 typedef struct l4_vm_svm_vmc_b_t
00162 {
00163     l4_vm_svm_vmc_b_control_area_t    control_area;
00164     l4_vm_svm_vmc_b_state_save_area_t state_save_area;
00165 } l4_vm_svm_vmc_b_t;

```

## 16.435 \_\_vm-vmx.h

```

00001
00006 /*
00007  * (c) 2010-2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/sys/vcpu.h>
00027
00039 enum l4_vm_vmx_caps_regs
00040 {
00041     L4_VM_VMX_BASIC_REG          = 0,
00042     L4_VM_VMX_TRUE_PINBASED_CTLS_REG = 1,
00043     L4_VM_VMX_TRUE_PROCBASED_CTLS_REG = 2,
00044     L4_VM_VMX_TRUE_EXIT_CTLS_REG   = 3,
00045     L4_VM_VMX_TRUE_ENTRY_CTLS_REG  = 4,
00046     L4_VM_VMX_MISC_REG             = 5,
00047     L4_VM_VMX_CR0_FIXED0_REG       = 6,
00048     L4_VM_VMX_CR0_FIXED1_REG       = 7,
00049     L4_VM_VMX_CR4_FIXED0_REG       = 8,
00050     L4_VM_VMX_CR4_FIXED1_REG       = 9,
00051     L4_VM_VMX_VMCS_ENUM_REG        = 0xa,
00052     L4_VM_VMX_PROCBASED_CTLS2_REG  = 0xb,
00053     L4_VM_VMX_EPT_VPID_CAP_REG     = 0xc,

```



```

00054 L4_VM_VMX_NESTED_REVISION          = 0xd,
00055 L4_VM_VMX_NUM_CAPS_REGS
00056 };
00057
00058
00063 enum L4_vm_vmx_dfl1_regs
00064 {
00065     L4_VM_VMX_PINBASED_CTL5_DFL1_REG = 0x1,
00066     L4_VM_VMX_PROCBASED_CTL5_DFL1_REG = 0x2,
00067     L4_VM_VMX_EXIT_CTL5_DFL1_REG     = 0x3,
00068     L4_VM_VMX_ENTRY_CTL5_DFL1_REG    = 0x4,
00069     L4_VM_VMX_NUM_DFL1_REGS
00070 };
00071
00080 L4_INLINE
00081 l4_uint64_t
00082 l4_vm_vmx_get_caps(void const *vcpu_state, unsigned cap_msr) L4_NOTHROW;
00083
00092 L4_INLINE
00093 l4_uint32_t
00094 l4_vm_vmx_get_caps_default1(void const *vcpu_state, unsigned cap_msr) L4_NOTHROW;
00095
00096
00106 enum L4_vm_vmx_sw_fields
00107 {
00115     L4_VM_VMX_VMCS_CR2          = 0x683e,
00116     // Custom argument passed from kernel to user space
00117     L4_VM_VMX_VMCS_NAT_ARG0     = 0x6840,
00118     // Custom argument passed from kernel to user space
00119     L4_VM_VMX_VMCS_NAT_ARG1     = 0x6842,
00120     // Custom argument passed from kernel to user space
00121     L4_VM_VMX_VMCS_NAT_ARG2     = 0x6844,
00122     // Custom argument passed from kernel to user space
00123     L4_VM_VMX_VMCS_NAT_ARG3     = 0x6846,
00125     L4_VM_VMX_VMCS_XCR0         = 0x2840,
00127     L4_VM_VMX_VMCS_MSR_SYSCALL_MASK = 0x2842,
00129     L4_VM_VMX_VMCS_MSR_LSTAR    = 0x2844,
00131     L4_VM_VMX_VMCS_MSR_CSTAR    = 0x2846,
00133     L4_VM_VMX_VMCS_MSR_TSC_AUX  = 0x2848,
00135     L4_VM_VMX_VMCS_MSR_STAR     = 0x284a,
00137     L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE = 0x284c,
00138 };
00139
00186 typedef struct l4_vmx_offset_table_t
00187 {
00188     l4_uint8_t offsets[4][4];
00189     l4_uint8_t limits[4][4];
00190     l4_uint8_t index_shifts[4];
00191     l4_uint8_t base_offset;
00192     l4_uint8_t size;
00193
00194     l4_uint8_t reserved[2];
00195 } l4_vmx_offset_table_t;
00196
00197 enum L4_vm_vmx_vmcs_sizes
00198 {
00199     L4_VM_VMX_VMCS_SIZE_VALUES = 2560,
00200     L4_VM_VMX_VMCS_SIZE_DIRTY_BITMAP = 320,
00201 };
00202
00231 typedef struct l4_ext_vcpu_state_vmx_t
00232 {
00233     l4_uint64_t reserved0;
00234
00235     l4_uint64_t user_data;
00236     l4_uint32_t cr2_index;
00237     l4_uint8_t reserved1[4];
00238
00239     l4_cap_idx_t vmcs;
00240
00241     /*
00242      * Since the capability type size depends on the platform, we add a 32-bit
00243      * padding if necessary.
00244      */
00245
00246     #if L4_MWORD_BITS == 32
00247         l4_uint32_t padding0;
00248     #elif L4_MWORD_BITS == 64
00249         /* No padding needed. */
00250     #else
00251         #error Unsupported machine word size.
00252     #endif
00253
00254     l4_vmx_offset_table_t offset_table;
00255     l4_uint8_t reserved2[120];
00256
00257     l4_uint8_t values[L4_VM_VMX_VMCS_SIZE_VALUES];

```

```

00258     l4_uint8_t dirty_bitmap[L4_VM_VMX_VMCS_SIZE_DIRTY_BITMAP];
00259 } l4_ext_vcpu_state_vmx_t;
00260
00261
00269 L4_INLINE
00270 unsigned
00271 l4_vm_vmx_field_len(unsigned field) L4_NOTHROW;
00272
00280 L4_INLINE
00281 unsigned
00282 l4_vm_vmx_field_order(unsigned field) L4_NOTHROW;
00283
00298 L4_INLINE
00299 void *
00300 l4_vm_vmx_field_ptr(void *vmcs, unsigned field) L4_NOTHROW;
00301
00311 L4_INLINE
00312 void
00313 l4_vm_vmx_clear(void *vmcs, void *user_vmcs) L4_NOTHROW;
00314
00324 L4_INLINE
00325 void
00326 l4_vm_vmx_ptr_load(void *vmcs, void *user_vmcs) L4_NOTHROW;
00327
00328
00343 L4_INLINE
00344 l4_uint32_t
00345 l4_vm_vmx_get_cr2_index(void const *vmcs) L4_NOTHROW;
00346
00356 L4_INLINE
00357 l4_umword_t
00358 l4_vm_vmx_read_nat(void *vmcs, unsigned field) L4_NOTHROW;
00359
00369 L4_INLINE
00370 l4_uint16_t
00371 l4_vm_vmx_read_16(void *vmcs, unsigned field) L4_NOTHROW;
00372
00382 L4_INLINE
00383 l4_uint32_t
00384 l4_vm_vmx_read_32(void *vmcs, unsigned field) L4_NOTHROW;
00385
00395 L4_INLINE
00396 l4_uint64_t
00397 l4_vm_vmx_read_64(void *vmcs, unsigned field) L4_NOTHROW;
00398
00408 L4_INLINE
00409 l4_uint64_t
00410 l4_vm_vmx_read(void *vmcs, unsigned field) L4_NOTHROW;
00411
00420 L4_INLINE
00421 void
00422 l4_vm_vmx_write_nat(void *vmcs, unsigned field, l4_umword_t val) L4_NOTHROW;
00423
00432 L4_INLINE
00433 void
00434 l4_vm_vmx_write_16(void *vmcs, unsigned field, l4_uint16_t val) L4_NOTHROW;
00435
00444 L4_INLINE
00445 void
00446 l4_vm_vmx_write_32(void *vmcs, unsigned field, l4_uint32_t val) L4_NOTHROW;
00447
00456 L4_INLINE
00457 void
00458 l4_vm_vmx_write_64(void *vmcs, unsigned field, l4_uint64_t val) L4_NOTHROW;
00459
00468 L4_INLINE
00469 void
00470 l4_vm_vmx_write(void *vmcs, unsigned field, l4_uint64_t val) L4_NOTHROW;
00471
00497 L4_INLINE
00498 void
00499 l4_vm_vmx_set_hw_vmcs(void *vmcs, l4_cap_idx_t vmcs_cap) L4_NOTHROW;
00500
00509 L4_INLINE
00510 l4_cap_idx_t
00511 l4_vm_vmx_get_hw_vmcs(void *vmcs) L4_NOTHROW;
00512
00513
00514 /* Implementations */
00515
00516 L4_INLINE
00517 unsigned
00518 l4_vm_vmx_field_order(unsigned field) L4_NOTHROW
00519 {
00520     unsigned size = (field >> 13) & 0x03U;
00521
00522     switch (size)

```

```

00523     {
00524         case 0: return 1; /* 16 bits */
00525         case 1: return 3; /* 64 bits */
00526         case 2: return 2; /* 32 bits */
00527         case 3: return (sizeof(l4_umword_t) == 8) ? 3 : 2; /* Natural width */
00528     }
00529
00530     __builtin_trap();
00531 }
00532
00533 L4_INLINE
00534 unsigned
00535 l4_vm_vmx_field_len(unsigned field) L4_NOTHROW
00536 {
00537     return 1 << l4_vm_vmx_field_order(field);
00538 }
00539
00540 L4_INLINE
00541 unsigned
00542 l4_vm_vmx_field_offset(void const *vmcs, unsigned field) L4_NOTHROW
00543 {
00544     l4_ext_vcpu_state_vmx_t const *state = (l4_ext_vcpu_state_vmx_t const *)vmcs;
00545
00546     unsigned index = field & 0x3feU;
00547     unsigned size = (field >> 13) & 0x03U;
00548     unsigned group = (field >> 10) & 0x03U;
00549
00550     unsigned shifted_index = index << state->offset_table.index_shifts[size];
00551
00552     if (shifted_index >= (unsigned)state->offset_table.limits[size][group] * 64)
00553         return ~0U;
00554
00555     return (unsigned)state->offset_table.offsets[size][group] * 64
00556         + shifted_index;
00557 }
00558
00559 L4_INLINE
00560 void *
00561 l4_vm_vmx_field_ptr(void *vmcs, unsigned field) L4_NOTHROW
00562 {
00563     l4_ext_vcpu_state_vmx_t *state = (l4_ext_vcpu_state_vmx_t *)vmcs;
00564
00565     unsigned offset = l4_vm_vmx_field_offset(vmcs, field);
00566     if (offset == ~0U)
00567         return 0;
00568
00569     return (void *) (state->values + offset);
00570 }
00571
00572 L4_INLINE
00573 void *
00574 l4_vm_vmx_field_ptr_offset(void *vmcs, unsigned field, unsigned *offset) L4_NOTHROW
00575 {
00576     l4_ext_vcpu_state_vmx_t *state = (l4_ext_vcpu_state_vmx_t *)vmcs;
00577
00578     *offset = l4_vm_vmx_field_offset(vmcs, field);
00579     if (*offset == ~0U)
00580         return 0;
00581
00582     return (void *) (state->values + *offset);
00583 }
00584
00585 L4_INLINE
00586 void
00587 l4_vm_vmx_offset_dirty(void *vmcs, unsigned offset) L4_NOTHROW
00588 {
00589     l4_ext_vcpu_state_vmx_t *state = (l4_ext_vcpu_state_vmx_t *)vmcs;
00590
00591     state->dirty_bitmap[offset / 8] |= 1U << (offset % 8);
00592 }
00593
00594 L4_INLINE
00595 void
00596 l4_vm_vmx_copy_values(l4_ext_vcpu_state_vmx_t const *state, l4_uint8_t *_dst,
00597     l4_uint8_t const *_src) L4_NOTHROW
00598 {
00599     unsigned base_offset = state->offset_table.base_offset * 64;
00600     unsigned size = state->offset_table.size * 64;
00601
00602     void *dst = _dst + base_offset;
00603     void const *src = _src + base_offset;
00604     __builtin_memcpy(dst, src, size);
00605 }
00606
00607 L4_INLINE
00608 void
00609 l4_vm_vmx_clear(void *vmcs, void *user_vmcs) L4_NOTHROW

```

```

00624 {
00625     l4_ext_vcpu_state_vmx_t *state = (l4_ext_vcpu_state_vmx_t *)vmcs;
00626     l4_ext_vcpu_state_vmx_t *user_state = (l4_ext_vcpu_state_vmx_t *)user_vmcs;
00627
00628     void **current_vmcs_ptr = (void **)&state->user_data;
00629     if (*current_vmcs_ptr != user_vmcs)
00630         return;
00631
00632     l4_vm_vmx_set_hw_vmcs(user_state, l4_vm_vmx_get_hw_vmcs(state));
00633     l4_vm_vmx_copy_values(state, user_state->values, state->values);
00634
00635     /* Due to its size, the dirty bitmap is always compiled in its entirety. */
00636     __builtin_memcpy(user_state->dirty_bitmap, state->dirty_bitmap,
00637         L4_VM_VMX_VMCS_SIZE_DIRTY_BITMAP);
00638
00639     *current_vmcs_ptr = 0;
00640 }
00641
00642 L4_INLINE
00643 void
00644 l4_vm_vmx_ptr_load(void *vmcs, void *user_vmcs) L4_NOTHROW
00645 {
00646     l4_ext_vcpu_state_vmx_t *state = (l4_ext_vcpu_state_vmx_t *)vmcs;
00647     l4_ext_vcpu_state_vmx_t *user_state = (l4_ext_vcpu_state_vmx_t *)user_vmcs;
00648
00649     void **current_vmcs_ptr = (void **)&state->user_data;
00650
00651     if (*current_vmcs_ptr == user_vmcs)
00652         return;
00653
00654     if (*current_vmcs_ptr && *current_vmcs_ptr != user_vmcs)
00655         l4_vm_vmx_clear(vmcs, *current_vmcs_ptr);
00656
00657     *current_vmcs_ptr = user_vmcs;
00658
00659     l4_vm_vmx_set_hw_vmcs(state, l4_vm_vmx_get_hw_vmcs(user_state));
00660     l4_vm_vmx_copy_values(state, state->values, user_state->values);
00661
00662     /* Due to its size, the dirty bitmap is always compiled in its entirety. */
00663     __builtin_memcpy(state->dirty_bitmap, user_state->dirty_bitmap,
00664         L4_VM_VMX_VMCS_SIZE_DIRTY_BITMAP);
00665 }
00666
00667 L4_INLINE
00668 l4_umword_t
00669 l4_vm_vmx_read_nat(void *vmcs, unsigned field) L4_NOTHROW
00670 {
00671     l4_umword_t *ptr = (l4_umword_t *)l4_vm_vmx_field_ptr(vmcs, field);
00672     if (!ptr)
00673         return 0;
00674
00675     return *ptr;
00676 }
00677
00678 L4_INLINE
00679 l4_uint16_t
00680 l4_vm_vmx_read_16(void *vmcs, unsigned field) L4_NOTHROW
00681 {
00682     l4_uint16_t *ptr = (l4_uint16_t *)l4_vm_vmx_field_ptr(vmcs, field);
00683     if (!ptr)
00684         return 0;
00685
00686     return *ptr;
00687 }
00688
00689 L4_INLINE
00690 l4_uint32_t
00691 l4_vm_vmx_read_32(void *vmcs, unsigned field) L4_NOTHROW
00692 {
00693     l4_uint32_t *ptr = (l4_uint32_t *)l4_vm_vmx_field_ptr(vmcs, field);
00694     if (!ptr)
00695         return 0;
00696
00697     return *ptr;
00698 }
00699
00700 L4_INLINE
00701 l4_uint64_t
00702 l4_vm_vmx_read_64(void *vmcs, unsigned field) L4_NOTHROW
00703 {
00704     l4_uint64_t *ptr = (l4_uint64_t *)l4_vm_vmx_field_ptr(vmcs, field);
00705     if (!ptr)
00706         return 0;
00707
00708     return *ptr;
00709 }
00710

```

```

00711 L4_INLINE
00712 l4_uint64_t
00713 l4_vm_vmx_read(void *vmcs, unsigned field) L4_NOTHROW
00714 {
00715     unsigned size = (field >> 13) & 0x03U;
00716
00717     switch (size)
00718     {
00719         case 0: return l4_vm_vmx_read_16(vmcs, field);
00720         case 1: return l4_vm_vmx_read_64(vmcs, field);
00721         case 2: return l4_vm_vmx_read_32(vmcs, field);
00722         case 3: return l4_vm_vmx_read_nat(vmcs, field);
00723     }
00724
00725     __builtin_trap();
00726 }
00727
00728 L4_INLINE
00729 void
00730 l4_vm_vmx_write_nat(void *vmcs, unsigned field, l4_umword_t val) L4_NOTHROW
00731 {
00732     unsigned offset;
00733     l4_umword_t *ptr = (l4_umword_t *)l4_vm_vmx_field_ptr_offset(vmcs, field,
00734                                                                    &offset);
00735
00736     if ((ptr) && (*ptr != val))
00737     {
00738         *ptr = val;
00739         l4_vm_vmx_offset_dirty(vmcs, offset);
00740     }
00741 }
00742
00743 L4_INLINE
00744 void
00745 l4_vm_vmx_write_16(void *vmcs, unsigned field, l4_uint16_t val) L4_NOTHROW
00746 {
00747     unsigned offset;
00748     l4_uint16_t *ptr = (l4_uint16_t *)l4_vm_vmx_field_ptr_offset(vmcs, field,
00749                                                                    &offset);
00750
00751     if ((ptr) && (*ptr != val))
00752     {
00753         *ptr = val;
00754         l4_vm_vmx_offset_dirty(vmcs, offset);
00755     }
00756 }
00757
00758 L4_INLINE
00759 void
00760 l4_vm_vmx_write_32(void *vmcs, unsigned field, l4_uint32_t val) L4_NOTHROW
00761 {
00762     unsigned offset;
00763     l4_uint32_t *ptr = (l4_uint32_t *)l4_vm_vmx_field_ptr_offset(vmcs, field,
00764                                                                    &offset);
00765
00766     if ((ptr) && (*ptr != val))
00767     {
00768         *ptr = val;
00769         l4_vm_vmx_offset_dirty(vmcs, offset);
00770     }
00771 }
00772
00773 L4_INLINE
00774 void
00775 l4_vm_vmx_write_64(void *vmcs, unsigned field, l4_uint64_t val) L4_NOTHROW
00776 {
00777     unsigned offset;
00778     l4_uint64_t *ptr = (l4_uint64_t *)l4_vm_vmx_field_ptr_offset(vmcs, field,
00779                                                                    &offset);
00780
00781     if ((ptr) && (*ptr != val))
00782     {
00783         *ptr = val;
00784         l4_vm_vmx_offset_dirty(vmcs, offset);
00785     }
00786 }
00787
00788 L4_INLINE
00789 void
00790 l4_vm_vmx_write(void *vmcs, unsigned field, l4_uint64_t val) L4_NOTHROW
00791 {
00792     unsigned size = (field >> 13) & 0x03U;
00793
00794     switch (size)
00795     {
00796         case 0: l4_vm_vmx_write_16(vmcs, field, val); break;
00797         case 1: l4_vm_vmx_write_64(vmcs, field, val); break;

```

```

00798     case 2: l4_vm_vmx_write_32(vmcs, field, val); break;
00799     case 3: l4_vm_vmx_write_nat(vmcs, field, val); break;
00800     }
00801 }
00802
00803 L4_INLINE
00804 l4_uint64_t
00805 l4_vm_vmx_get_caps(void const *vcpu_state, unsigned cap_msr) L4_NOTHROW
00806 {
00807     l4_uint64_t const *caps = (l4_uint64_t const *)((char const *) (vcpu_state) +
00808     L4_VCPU_OFFSET_EXT_INFOS);
00809     return caps[cap_msr & 0xfU];
00810 }
00811 L4_INLINE
00812 l4_uint32_t
00813 l4_vm_vmx_get_caps_default1(void const *vcpu_state, unsigned cap_msr) L4_NOTHROW
00814 {
00815     l4_uint32_t const *caps = (l4_uint32_t const *)((char const *) (vcpu_state) +
00816     L4_VCPU_OFFSET_EXT_INFOS);
00817     return caps[L4_VM_VMX_NUM_CAPS_REGS * 2 + ((cap_msr & 0xfU) - L4_VM_VMX_PINBASED_CTL5_DFL1_REG)];
00818 }
00819 L4_INLINE
00820 l4_uint32_t
00821 l4_vm_vmx_get_cr2_index(void const *vmcs) L4_NOTHROW
00822 {
00823     l4_ext_vcpu_state_vmx_t const *state = (l4_ext_vcpu_state_vmx_t const *) vmcs;
00824     return state->cr2_index;
00825 }
00826
00827 L4_INLINE
00828 void
00829 l4_vm_vmx_set_hw_vmcs(void *vmcs, l4_cap_idx_t vmcs_cap) L4_NOTHROW
00830 {
00831     l4_ext_vcpu_state_vmx_t *state = (l4_ext_vcpu_state_vmx_t *) vmcs;
00832     state->vmcs = vmcs_cap;
00833 }
00834
00835 L4_INLINE
00836 l4_cap_idx_t
00837 l4_vm_vmx_get_hw_vmcs(void *vmcs) L4_NOTHROW
00838 {
00839     l4_ext_vcpu_state_vmx_t *state = (l4_ext_vcpu_state_vmx_t *) vmcs;
00840     return state->vmcs & L4_CAP_MASK;
00841 }

```

## 16.436 l4/sys/arm\_smccc File Reference

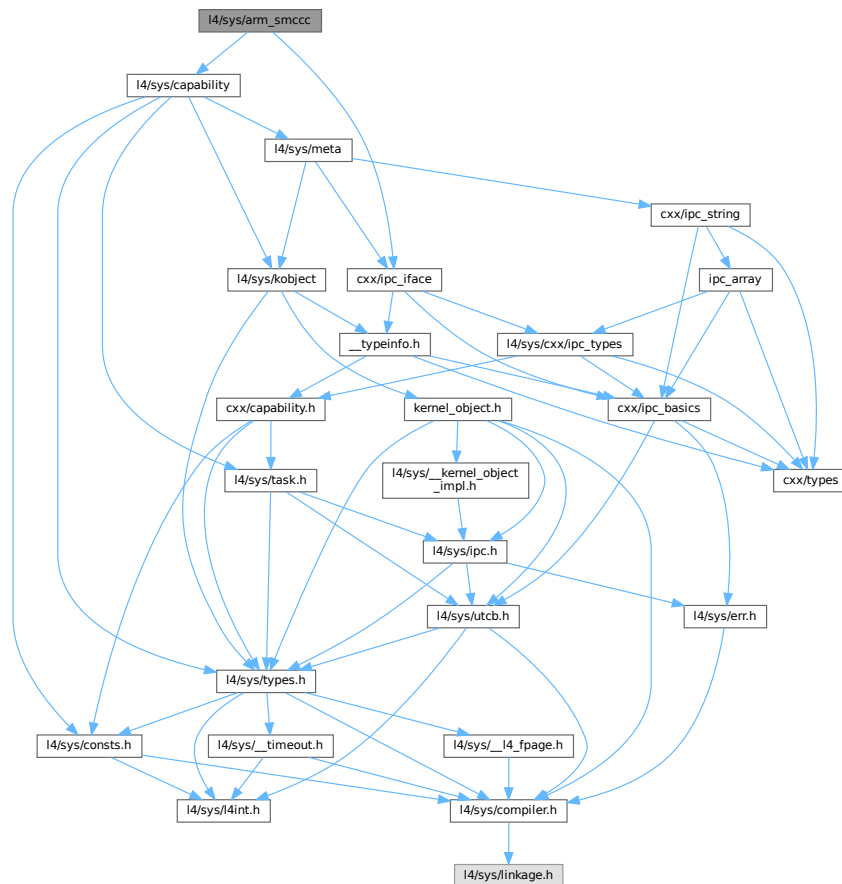
ARM secure monitor call functions.

```

#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>

```

Include dependency graph for arm\_smccc:



## Data Structures

- class [L4::Arm\\_smccc](#)

*Wrapper for function calls that follow the ARM SMC/HVC calling convention.*

## Namespaces

- namespace [L4](#)

*[L4](#) low-level kernel interface.*

## 16.436.1 Detailed Description

ARM secure monitor call functions.

Definition in file [arm\\_smccc](#).

## 16.437 arm\_smccc

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2018, 2022 Kernkonzept GmbH.
00004  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00005  *
00006  * This file is distributed under the terms of the GNU General Public
00007  * License, version 2. Please see the COPYING-GPL-2 file for details.
00008  */
00013 #pragma once
00014
00015 #include <l4/sys/capability>
00016 #include <l4/sys/cxx/ipc_iface>
00017
00018 namespace L4 {
00019
00024 class L4_EXPORT Arm_smccc : public Kobject_0t<Arm_smccc, L4_PROTO_SMCCC>
00025 {
00026 public:
00063     L4_INLINE_RPC(l4_msgtag_t, call,
00064                   (l4_umword_t func, l4_umword_t in0, l4_umword_t in1,
00065                    l4_umword_t in2, l4_umword_t in3, l4_umword_t in4,
00066                    l4_umword_t in5, l4_umword_t *out0, l4_umword_t *out1,
00067                    l4_umword_t *out2, l4_umword_t *out3,
00068                    l4_umword_t client_id));
00069
00070     typedef L4::Typeid::Rpc_nocode<call_t> Rpcs;
00071 };
00072
00073 }

```

## 16.438 l4/sys/arm\_smccc.h File Reference

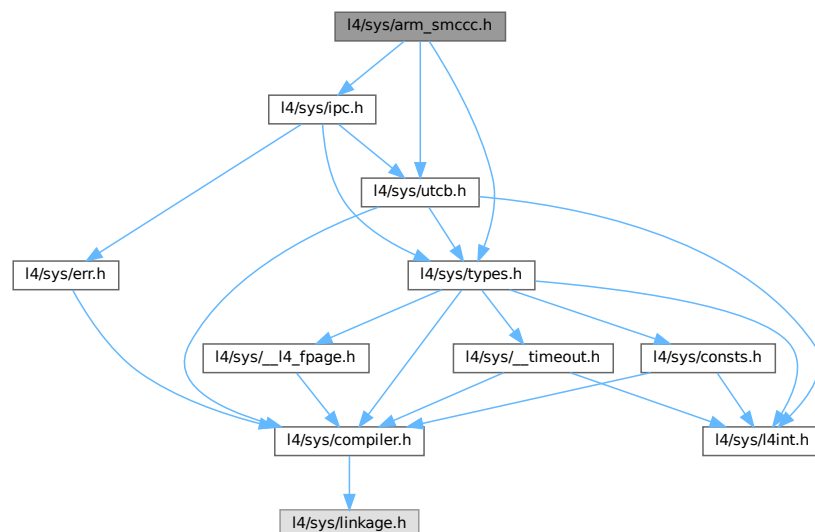
ARM secure monitor call functions.

```

#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for arm\_smccc.h:





## Functions

- [l4\\_msgtag\\_t l4\\_arm\\_smccc\\_call](#) ([l4\\_cap\\_idx\\_t](#) pfc, [l4\\_umword\\_t](#) func, [l4\\_umword\\_t](#) in0, [l4\\_umword\\_t](#) in1, [l4\\_umword\\_t](#) in2, [l4\\_umword\\_t](#) in3, [l4\\_umword\\_t](#) in4, [l4\\_umword\\_t](#) in5, [l4\\_umword\\_t](#) \*out0, [l4\\_umword\\_t](#) \*out1, [l4\\_umword\\_t](#) \*out2, [l4\\_umword\\_t](#) \*out3, [l4\\_umword\\_t](#) client\_id) [L4\\_NOTHROW](#)

*C interface for calling the ARM secure monitor, see [L4::Arm\\_smccc::call\(\)](#) for the C++ interface.*

### 16.438.1 Detailed Description

ARM secure monitor call functions.

Definition in file [arm\\_smccc.h](#).

### 16.438.2 Function Documentation

#### 16.438.2.1 l4\_arm\_smccc\_call()

```
l4_msgtag_t l4_arm_smccc_call (
    l4_cap_idx_t pfc,
    l4_umword_t func,
    l4_umword_t in0,
    l4_umword_t in1,
    l4_umword_t in2,
    l4_umword_t in3,
    l4_umword_t in4,
    l4_umword_t in5,
    l4_umword_t * out0,
    l4_umword_t * out1,
    l4_umword_t * out2,
    l4_umword_t * out3,
    l4_umword_t client_id ) [inline]
```

C interface for calling the ARM secure monitor, see [L4::Arm\\_smccc::call\(\)](#) for the C++ interface.

#### Parameters

<i>pfc</i>	Capability of the SMC kernel object.
------------	--------------------------------------

The input parameters consist of a function identifier, 6 arguments and a client id. Results are returned in 4 output parameters.

## Parameters

	<i>func</i>	Function identifier. <ul style="list-style-type: none"> <li>• Bit 31 has to be set: This marks the call as <i>Fast Call</i>. <i>Yielding Calls</i> (bit 31 unset) are rejected by the kernel.</li> <li>• Bit 30 defines the calling convention:</li> <li>• Bit 30 == 1: 64-bit calling convention.</li> <li>• Bit 30 == 0: 32-bit calling convention.</li> <li>• Bits 24..29 determine the service call ID. Only service IDs <math>\geq 0x30000000</math> (<i>Trusted Application Calls</i> and <i>Trusted OS Calls</i>) are allowed.</li> </ul>
in	<i>in0</i>	First input parameter.
in	<i>in1</i>	Second input parameter.
in	<i>in2</i>	Third input parameter.
in	<i>in3</i>	Fourth input parameter.
in	<i>in4</i>	Fifth input parameter.
in	<i>in5</i>	Sixth input parameter.
out	<i>out0</i>	First output parameter.
out	<i>out1</i>	Second output parameter.
out	<i>out2</i>	Third output parameter.
out	<i>out3</i>	Fourth output parameter.
in	<i>client_id</i>	Client ID. According to the specification, this value might be ignored by certain functions.

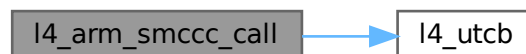
## Return values

-L4_ENOSYS	Either bit 31 of the function call not set or service ID $< 0x30000000$ .
-L4_EINVAL	Invalid number of parameters.
$< 0$	Other L4 error.
0	Success.

Definition at line 43 of file [arm\\_smccc.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



## 16.439 arm\_smccc.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (C) 2018, 2022 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * This file is distributed under the terms of the GNU General Public
00006  * License, version 2. Please see the COPYING-GPL-2 file for details.
00007  */
00012 #pragma once
00013
00014 #include <l4/sys/types.h>
00015 #include <l4/sys/utcb.h>
00016
00017 L4_INLINE l4_msgtag_t
00018 l4_arm_smccc_call(l4_cap_idx_t pfc, l4_umword_t func, l4_umword_t in0,
00019                 l4_umword_t in1, l4_umword_t in2, l4_umword_t in3,
00020                 l4_umword_t in4, l4_umword_t in5, l4_umword_t *out0,
00021                 l4_umword_t *out1, l4_umword_t *out2, l4_umword_t *out3,
00022                 l4_umword_t client_id) L4_NOTHROW;
00023
00024 L4_INLINE l4_msgtag_t
00025 l4_arm_smccc_call_u(l4_cap_idx_t pfc, l4_umword_t func, l4_umword_t in0,
00026                   l4_umword_t in1, l4_umword_t in2, l4_umword_t in3,
00027                   l4_umword_t in4, l4_umword_t in5, l4_umword_t *out0,
00028                   l4_umword_t *out1, l4_umword_t *out2, l4_umword_t *out3,
00029                   l4_umword_t client_id, l4_utcb_t *utcb) L4_NOTHROW;
00030
00031 /* IMPLEMENTATION -----*/
00032
00033 #include <l4/sys/ipc.h>
00034
00042 L4_INLINE l4_msgtag_t
00043 l4_arm_smccc_call(l4_cap_idx_t pfc, l4_umword_t func,
00044                 l4_umword_t in0, l4_umword_t in1,
00045                 l4_umword_t in2, l4_umword_t in3,
00046                 l4_umword_t in4, l4_umword_t in5,
00047                 l4_umword_t *out0, l4_umword_t *out1,
00048                 l4_umword_t *out2, l4_umword_t *out3,
00049                 l4_umword_t client_id) L4_NOTHROW
00050 {
00051     return l4_arm_smccc_call_u(pfc, func, in0, in1, in2, in3, in4, in5,
00052                               out0, out1, out2, out3, client_id, l4_utcb());
00053 }
00054
00055 L4_INLINE l4_msgtag_t
00056 l4_arm_smccc_call_u(l4_cap_idx_t pfc, l4_umword_t func, l4_umword_t in0,
00057                   l4_umword_t in1, l4_umword_t in2, l4_umword_t in3,
00058                   l4_umword_t in4, l4_umword_t in5, l4_umword_t *out0,
00059                   l4_umword_t *out1, l4_umword_t *out2, l4_umword_t *out3,
00060                   l4_umword_t client_id, l4_utcb_t *utcb) L4_NOTHROW
00061 {
00062     {
00063         l4_msgtag_t ret;
00064         l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00065         v->mr[0] = func;
00066         v->mr[1] = in0;
00067         v->mr[2] = in1;
00068         v->mr[3] = in2;
00069         v->mr[4] = in3;
00070         v->mr[5] = in4;
00071         v->mr[6] = in5;
00072         v->mr[7] = client_id;
00073
00074         ret = l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_SMCCC, 8, 0, 0),
00075                         L4_IPC_NEVER);
00076
00077         if (l4_error(ret) >= 0)
00078         {
00079             *out0 = v->mr[0];
00080             *out1 = v->mr[1];
00081             *out2 = v->mr[2];
00082             *out3 = v->mr[3];
00083         }
00084
00085         return ret;
00086     }

```

## 16.440 l4/sys/assert.h File Reference

Low-level assert implementation.

```

#include <l4/sys/compiler.h>
#include <l4/sys/thread.h>

```



## 16.440.1 Detailed Description

Low-level assert implementation.

Definition in file [assert.h](#).

## 16.440.2 Macro Definition Documentation

### 16.440.2.1 l4\_assert

```
#define l4_assert(  
    expr )
```

**Value:**

```
l4_assert_fn(!!(expr), __FILE__ ":" L4_stringify(__LINE__) ": Assertion \"" \
    L4_stringify(expr) "\" failed.\n")
```

Low-level assert.

**Parameters**

<i>expr</i>	Expression to be evaluate for the assertion.
-------------	--

This assertion is a low-level implementation that directly uses kernel primitives. Only use [l4\\_assert\(\)](#) when the standard `assert()` functionality is not available.

Definition at line 43 of file [assert.h](#).

## 16.441 assert.h

[Go to the documentation of this file.](#)

```
00001
00002 /*
00003  * (c) 2015 Adam Lackorzynski <adam@l4re.org>
00004  *
00005  * This file is part of L4Re and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019
00020 #ifdef NDEBUB
00021
00022 #define l4_assert(x) do { } while (0)
00023 #define l4_check(x) do { (void)(x); } while (0)
00024
00025 #else
00026
00027 #include <l4/sys/compiler.h>
00028 #include <l4/sys/thread.h>
00029 #include <l4/sys/vcon.h>
00030
00031 #define l4_assert(expr) \
```

```

00044  l4_assert_fn(!!(expr), __FILE__ ":" L4_stringify(__LINE__) ": Assertion \"" \
00045                      L4_stringify(expr) "\" failed.\n")
00046
00047  #define l4_check(expr) l4_assert(expr)
00048
00052  L4_ALWAYS_INLINE
00053  void l4_assert_fn(unsigned expr, const char *text) L4_NOTHROW;
00054
00058  L4_INLINE L4_NORETURN
00059  void l4_assert_abort(const char *text) L4_NOTHROW;
00060
00061
00062  /* IMPLEMENTATION ----- */
00063
00064  L4_INLINE L4_NORETURN
00065  void l4_assert_abort(const char *text) L4_NOTHROW
00066  {
00067      l4_vcon_write(L4_BASE_LOG_CAP, text, __builtin_strlen(text));
00068      for (;;)
00069          l4_thread_ex_regs(L4_INVALID_CAP, ~0UL, ~0UL,
00070                          L4_THREAD_EX_REGS_TRIGGER_EXCEPTION);
00071  }
00072
00073  L4_ALWAYS_INLINE
00074  void l4_assert_fn(unsigned expr, const char *text) L4_NOTHROW
00075  {
00076      if (L4_LIKELY(expr))
00077          return;
00078      l4_assert_abort(text);
00079  }
00080  }
00081
00082  #endif /* NDEBUG */

```

## 16.442 l4/util/assert.h File Reference

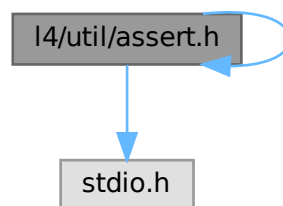
Some useful assert-style macros.

```

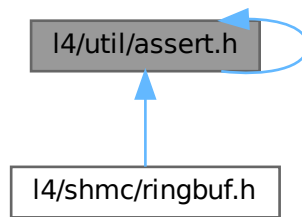
#include <stdio.h>
#include <assert.h>

```

Include dependency graph for assert.h:



This graph shows which files directly or indirectly include this file:



### 16.442.1 Detailed Description

Some useful assert-style macros.

#### Date

09/2009

#### Author

Bjoern Doebel [doebel@tudos.org](mailto:doebel@tudos.org)

Definition in file [assert.h](#).

## 16.443 assert.h

[Go to the documentation of this file.](#)

```

00001 /*****
00009  */
00010  * (c) 2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU Lesser General Public License 2.1.
00014  * Please see the COPYING-LGPL-2.1 file for details.
00015  */
00016
00017 /*****
00018 #pragma once
00019
00020 #ifndef NDEBUG
00021
00022 #define DO_NOTHING          do { } while (0)
00023 #define ASSERT_ASSERT(x)    DO_NOTHING
00024 #define ASSERT_VALID(c)     DO_NOTHING
00025 #define ASSERT_EQUAL(a,b)   DO_NOTHING
00026 #define ASSERT_NOT_EQUAL(a,b) DO_NOTHING
00027 #define ASSERT_LOWER_EQ(a,b) DO_NOTHING
00028 #define ASSERT_GREATER_EQ(a,b) DO_NOTHING
00029 #define ASSERT_BETWEEN(a,b,c) DO_NOTHING
00030 #define ASSERT_IPC_OK(i)    DO_NOTHING
00031 #define ASSERT_OK(e)        do { (void)e; } while (0)
00032 #define ASSERT_NOT_NULL(p)  DO_NOTHING
00033 #ifndef assert
00034 #define assert(cond)        DO_NOTHING
00035 #endif

```

```

00036
00037 #else // NDEBUG
00038
00039 #ifndef ASSERT_PRINTF
00040 #include <stdio.h>
00041 #define ASSERT_PRINTF printf
00042 #endif
00043 #ifndef ASSERT_ASSERT
00044 #include <assert.h>
00045 #define ASSERT_ASSERT(x) assert(x)
00046 #endif
00047
00048 #define ASSERT_VALID(cap) \
00049     do { \
00050         typeof(cap) _cap = cap; \
00051         if (l4_is_invalid_cap(_cap)) { \
00052             ASSERT_PRINTF("%s: Cap invalid.\n", __func__); \
00053             ASSERT_ASSERT(!l4_is_invalid_cap(_cap)); \
00054         } \
00055     } while (0)
00056
00057
00058 #define ASSERT_EQUAL(a, b) \
00059     do { \
00060         typeof(a) _a = a; \
00061         typeof(b) _b = b; \
00062         if (_a != _b) { \
00063             ASSERT_PRINTF("%s:\n", __func__); \
00064             ASSERT_PRINTF("    "#a" (%lx) != "#b" (%lx)\n", (unsigned long)_a, (unsigned long)_b); \
00065             ASSERT_ASSERT(_a == _b); \
00066         } \
00067     } while (0)
00068
00069
00070 #define ASSERT_NOT_EQUAL(a, b) \
00071     do { \
00072         typeof(a) _a = a; \
00073         typeof(b) _b = b; \
00074         if (_a == _b) { \
00075             ASSERT_PRINTF("%s:\n", __func__); \
00076             ASSERT_PRINTF("    "#a" (%lx) == "#b" (%lx)\n", (unsigned long)_a, (unsigned long)_b); \
00077             ASSERT_ASSERT(_a != _b); \
00078         } \
00079     } while (0)
00080
00081
00082 #define ASSERT_LOWER_EQ(val, max) \
00083     do { \
00084         typeof(val) _val = val; \
00085         typeof(max) _max = max; \
00086         if (_val > _max) { \
00087             ASSERT_PRINTF("%s:\n", __func__); \
00088             ASSERT_PRINTF("    "#val" (%lx) > "#max" (%lx)\n", (unsigned long)_val, (unsigned long)_max); \
00089             ASSERT_ASSERT(_val <= _max); \
00090         } \
00091     } while (0)
00092
00093
00094 #define ASSERT_GREATER_EQ(val, min) \
00095     do { \
00096         typeof(val) _val = val; \
00097         typeof(min) _min = min; \
00098         if (_val < _min) { \
00099             ASSERT_PRINTF("%s:\n", __func__); \
00100             ASSERT_PRINTF("    "#val" (%lx) < "#min" (%lx)\n", (unsigned long)_val, (unsigned long)_min); \
00101             ASSERT_ASSERT(_val >= _min); \
00102         } \
00103     } while (0)
00104
00105
00106 #define ASSERT_BETWEEN(val, min, max) \
00107     ASSERT_LOWER_EQ((val), (max)); \
00108     ASSERT_GREATER_EQ((val), (min));
00109
00110
00111 #define ASSERT_IPC_OK(msgtag) \
00112     do { \
00113         int _r = l4_ipc_error(msgtag, l4_utcb()); \
00114         if (_r) { \
00115             ASSERT_PRINTF("%s: IPC Error: %lx\n", __func__, _r); \
00116             ASSERT_ASSERT(_r == 0); \
00117         } \
00118     } while (0)
00119
00120 #define ASSERT_OK(val)          ASSERT_EQUAL((val), 0)
00121 #define ASSERT_NOT_NULL(ptr)    ASSERT_NOT_EQUAL((ptr), (void *)0)
00122

```



```
00123 #endif // NDEBUG
```

## 16.444 amd64/l4/sys/cache.h File Reference

Cache functions.

### Functions

- `int l4_cache_clean_data` (unsigned long start, unsigned long end) `L4_NOTHROW`  
*Cache clean a range in D-cache; writes back to PoC.*
- `int l4_cache_flush_data` (unsigned long start, unsigned long end) `L4_NOTHROW`  
*Cache flush a range; writes back to PoC.*
- `int l4_cache_inv_data` (unsigned long start, unsigned long end) `L4_NOTHROW`  
*Cache invalidate a range; might write back to PoC.*
- `int l4_cache_coherent` (unsigned long start, unsigned long end) `L4_NOTHROW`  
*Make memory coherent between I-cache and D-cache; writes back to PoU.*
- `int l4_cache_dma_coherent` (unsigned long start, unsigned long end) `L4_NOTHROW`  
*Make memory coherent for use with external memory; writes back to PoC.*
- `int l4_cache_dma_coherent_full` (void) `L4_NOTHROW`  
*Make memory coherent for use with external memory; writes back to PoC.*

### 16.444.1 Detailed Description

Cache functions.

Definition in file `cache.h`.

## 16.445 cache.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #ifndef __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__
00023 #define __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__
00024
00025 #include_next <l4/sys/cache.h>
00026
00027 L4_INLINE int
00028 l4_cache_clean_data(unsigned long start,
00029                    unsigned long end) L4_NOTHROW
00030 {
```

```

00031     (void)start; (void)end;
00032     return 0;
00033 }
00034
00035 L4_INLINE int
00036 l4_cache_flush_data(unsigned long start,
00037                     unsigned long end) L4_NOTHROW
00038 {
00039     (void)start; (void)end;
00040     return 0;
00041 }
00042
00043 L4_INLINE int
00044 l4_cache_inv_data(unsigned long start,
00045                  unsigned long end) L4_NOTHROW
00046 {
00047     (void)start; (void)end;
00048     return 0;
00049 }
00050
00051 L4_INLINE int
00052 l4_cache_coherent(unsigned long start,
00053                  unsigned long end) L4_NOTHROW
00054 {
00055     (void)start; (void)end;
00056     return 0;
00057 }
00058
00059 L4_INLINE int
00060 l4_cache_dma_coherent(unsigned long start,
00061                      unsigned long end) L4_NOTHROW
00062 {
00063     (void)start; (void)end;
00064     return 0;
00065 }
00066
00067 L4_INLINE int
00068 l4_cache_dma_coherent_full(void) L4_NOTHROW
00069 {
00070     return 0;
00071 }
00072
00073 #endif /* ! __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__ */

```

## 16.446 arm/l4/sys/cache.h File Reference

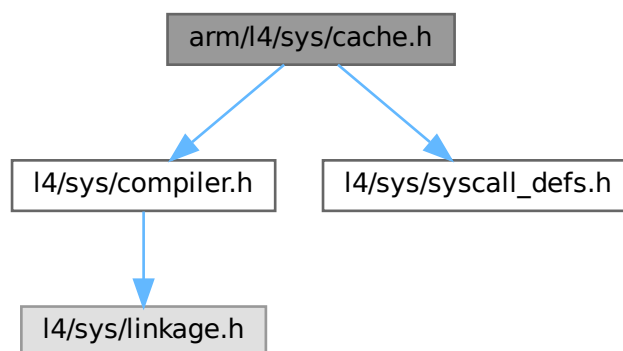
Cache functions.

```

#include <l4/sys/compiler.h>
#include <l4/sys/syscall_defs.h>

```

Include dependency graph for cache.h:



## Functions

- int [l4\\_cache\\_clean\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache clean a range in D-cache; writes back to PoC.*
- int [l4\\_cache\\_flush\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache flush a range; writes back to PoC.*
- int [l4\\_cache\\_inv\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache invalidate a range; might write back to PoC.*
- int [l4\\_cache\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent between I-cache and D-cache; writes back to PoU.*
- int [l4\\_cache\\_dma\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory; writes back to PoC.*
- int [l4\\_cache\\_dma\\_coherent\\_full](#) (void) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory; writes back to PoC.*

## 16.446.1 Detailed Description

Cache functions.

### Date

2007-11

### Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file [cache.h](#).

## 16.447 cache.h

[Go to the documentation of this file.](#)

```
00001
00002 /*
00003  * (c) 2007-2009 Author(s)
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019 #ifndef __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__
00020 #define __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__
00021
00022 #include <l4/sys/compiler.h>
00023 #include <l4/sys/syscall_defs.h>
00024
00025 #include_next <l4/sys/cache.h>
00026
00027 L4_INLINE void
00028 l4_cache_op_arm_call(unsigned long op,
```

```

00039             unsigned long start,
00040             unsigned long end);
00041
00042 L4_INLINE void
00043 l4_cache_op_arm_call(unsigned long op,
00044                     unsigned long start,
00045                     unsigned long end)
00046 {
00047     register unsigned long _op    __asm__ ("r0") = op;
00048     register unsigned long _start __asm__ ("r1") = start;
00049     register unsigned long _end   __asm__ ("r2") = end;
00050
00051     __asm__ __volatile__
00052     ("@ l4_cache_op_arm_call(start) \n\t"
00053      "mov    r5, %[sc] \n\t"
00054      "blx    __l4_sys_syscall \n\t"
00055      "@ l4_cache_op_arm_call(end) \n\t"
00056      :
00057      "=r" (_op),
00058      "=r" (_start),
00059      "=r" (_end)
00060      :
00061      [sc] "i" (L4_SYSCALL_MEM_OP),
00062      "0" (_op),
00063      "1" (_start),
00064      "2" (_end)
00065      :
00066      "cc", "memory", "r5", "ip", "lr"
00067      );
00068 }
00069
00070 enum L4_mem_cache_ops
00071 {
00072     L4_MEM_CACHE_OP_CLEAN_DATA      = 0,
00073     L4_MEM_CACHE_OP_FLUSH_DATA      = 1,
00074     L4_MEM_CACHE_OP_INV_DATA        = 2,
00075     L4_MEM_CACHE_OP_COHERENT        = 3,
00076     L4_MEM_CACHE_OP_DMA_COHERENT    = 4,
00077     L4_MEM_CACHE_OP_DMA_COHERENT_FULL = 5,
00078 };
00079
00080 L4_INLINE int
00081 l4_cache_clean_data(unsigned long start,
00082                    unsigned long end) L4_NOTHROW
00083 {
00084     l4_cache_op_arm_call(L4_MEM_CACHE_OP_CLEAN_DATA, start, end);
00085     return 0;
00086 }
00087
00088 L4_INLINE int
00089 l4_cache_flush_data(unsigned long start,
00090                    unsigned long end) L4_NOTHROW
00091 {
00092     l4_cache_op_arm_call(L4_MEM_CACHE_OP_FLUSH_DATA, start, end);
00093     return 0;
00094 }
00095
00096 L4_INLINE int
00097 l4_cache_inv_data(unsigned long start,
00098                  unsigned long end) L4_NOTHROW
00099 {
00100     l4_cache_op_arm_call(L4_MEM_CACHE_OP_INV_DATA, start, end);
00101     return 0;
00102 }
00103
00104 L4_INLINE int
00105 l4_cache_coherent(unsigned long start,
00106                  unsigned long end) L4_NOTHROW
00107 {
00108     l4_cache_op_arm_call(L4_MEM_CACHE_OP_COHERENT, start, end);
00109     return 0;
00110 }
00111
00112 L4_INLINE int
00113 l4_cache_dma_coherent(unsigned long start,
00114                      unsigned long end) L4_NOTHROW
00115 {
00116     l4_cache_op_arm_call(L4_MEM_CACHE_OP_DMA_COHERENT, start, end);
00117     return 0;
00118 }
00119
00120 L4_INLINE int
00121 l4_cache_dma_coherent_full(void) L4_NOTHROW
00122 {
00123     l4_cache_op_arm_call(L4_MEM_CACHE_OP_DMA_COHERENT_FULL, 0, 0);
00124     return 0;
00125 }

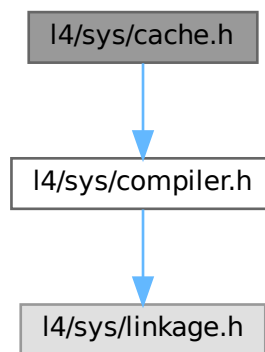
```

```
00126
00127 #endif /* ! __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__ */
```

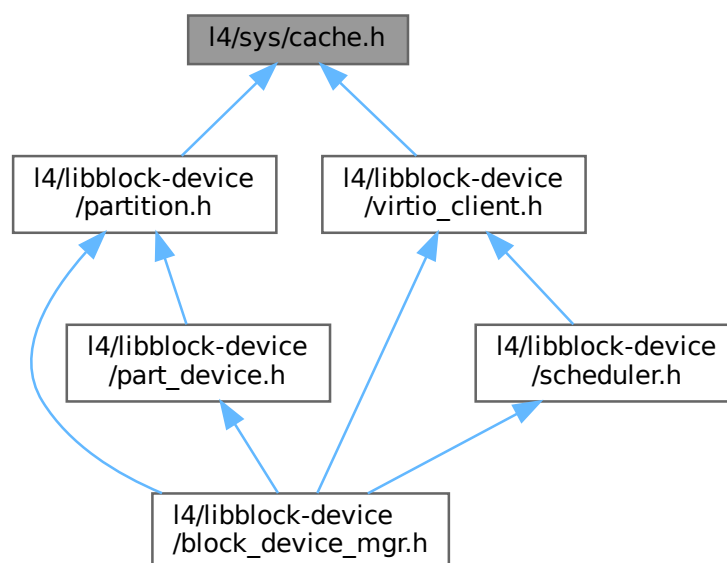
## 16.448 l4/sys/cache.h File Reference

Cache-consistency functions.

```
#include <l4/sys/compiler.h>
Include dependency graph for cache.h:
```



This graph shows which files directly or indirectly include this file:



## Functions

- int [l4\\_cache\\_clean\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache clean a range in D-cache; writes back to PoC.*
- int [l4\\_cache\\_flush\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache flush a range; writes back to PoC.*
- int [l4\\_cache\\_inv\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache invalidate a range; might write back to PoC.*
- int [l4\\_cache\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent between I-cache and D-cache; writes back to PoU.*
- int [l4\\_cache\\_dma\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory; writes back to PoC.*
- int [l4\\_cache\\_dma\\_coherent\\_full](#) (void) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory; writes back to PoC.*

## 16.448.1 Detailed Description

Cache-consistency functions.

### Date

2007-11

### Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file [cache.h](#).

## 16.449 cache.h

[Go to the documentation of this file.](#)

```

00001
00010 /*
00011  * (c) 2007-2009 Author(s)
00012  *      economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * This file is part of TUD:OS and distributed under the terms of the
00015  * GNU General Public License 2.
00016  * Please see the COPYING-GPL-2 file for details.
00017  *
00018  * As a special exception, you may use this file as part of a free software
00019  * library without restriction. Specifically, if other files instantiate
00020  * templates or use macros or inline functions from this file, or you compile
00021  * this file and link it with other files to produce an executable, this
00022  * file does not by itself cause the resulting executable to be covered by
00023  * the GNU General Public License. This exception does not however
00024  * invalidate any other reasons why the executable file might be covered by
00025  * the GNU General Public License.
00026  */
00027
00028 #ifndef __L4SYS__INCLUDE__CACHE_H__
00029 #define __L4SYS__INCLUDE__CACHE_H__
00030
00031 #include <l4/sys/compiler.h>
00032
00048 EXTERN_C_BEGIN
00049
00064 L4_INLINE int
00065 l4_cache_clean_data(unsigned long start,
```

```

00066             unsigned long end) L4_NOTHROW;
00067
00082 L4_INLINE int
00083 l4_cache_flush_data(unsigned long start,
00084                     unsigned long end) L4_NOTHROW;
00085
00104 L4_INLINE int
00105 l4_cache_inv_data(unsigned long start,
00106                  unsigned long end) L4_NOTHROW;
00107
00119 L4_INLINE int
00120 l4_cache_coherent(unsigned long start,
00121                  unsigned long end) L4_NOTHROW;
00122
00134 L4_INLINE int
00135 l4_cache_dma_coherent(unsigned long start,
00136                      unsigned long end) L4_NOTHROW;
00137
00138 #if !defined(ARCH_arm64)
00143 L4_INLINE int
00144 l4_cache_dma_coherent_full(void) L4_NOTHROW;
00145 #endif
00146
00147 EXTERN_C_END
00148
00149 #endif /* ! __L4SYS__INCLUDE__CACHE_H__ */

```

## 16.450 x86/I4/sys/cache.h File Reference

Cache functions.

### Functions

- int [l4\\_cache\\_clean\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache clean a range in D-cache; writes back to PoC.*
- int [l4\\_cache\\_flush\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache flush a range; writes back to PoC.*
- int [l4\\_cache\\_inv\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache invalidate a range; might write back to PoC.*
- int [l4\\_cache\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent between I-cache and D-cache; writes back to PoU.*
- int [l4\\_cache\\_dma\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory; writes back to PoC.*
- int [l4\\_cache\\_dma\\_coherent\\_full](#) (void) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory; writes back to PoC.*

### 16.450.1 Detailed Description

Cache functions.

Definition in file [cache.h](#).

## 16.451 cache.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #ifndef __L4SYS__INCLUDE__ARCH_X86__CACHE_H__
00023 #define __L4SYS__INCLUDE__ARCH_X86__CACHE_H__
00024
00025 #include_next <l4/sys/cache.h>
00026
00027 L4_INLINE int
00028 l4_cache_clean_data(unsigned long start,
00029                    unsigned long end) L4_NOTHROW
00030 {
00031     (void)start; (void)end;
00032     return 0;
00033 }
00034
00035 L4_INLINE int
00036 l4_cache_flush_data(unsigned long start,
00037                    unsigned long end) L4_NOTHROW
00038 {
00039     (void)start; (void)end;
00040     return 0;
00041 }
00042
00043 L4_INLINE int
00044 l4_cache_inv_data(unsigned long start,
00045                  unsigned long end) L4_NOTHROW
00046 {
00047     (void)start; (void)end;
00048     return 0;
00049 }
00050
00051 L4_INLINE int
00052 l4_cache_coherent(unsigned long start,
00053                  unsigned long end) L4_NOTHROW
00054 {
00055     (void)start; (void)end;
00056     return 0;
00057 }
00058
00059 L4_INLINE int
00060 l4_cache_dma_coherent(unsigned long start,
00061                      unsigned long end) L4_NOTHROW
00062 {
00063     (void)start; (void)end;
00064     return 0;
00065 }
00066
00067 L4_INLINE int
00068 l4_cache_dma_coherent_full(void) L4_NOTHROW
00069 {
00070     return 0;
00071 }
00072
00073 #endif /* ! __L4SYS__INCLUDE__ARCH_X86__CACHE_H__ */

```

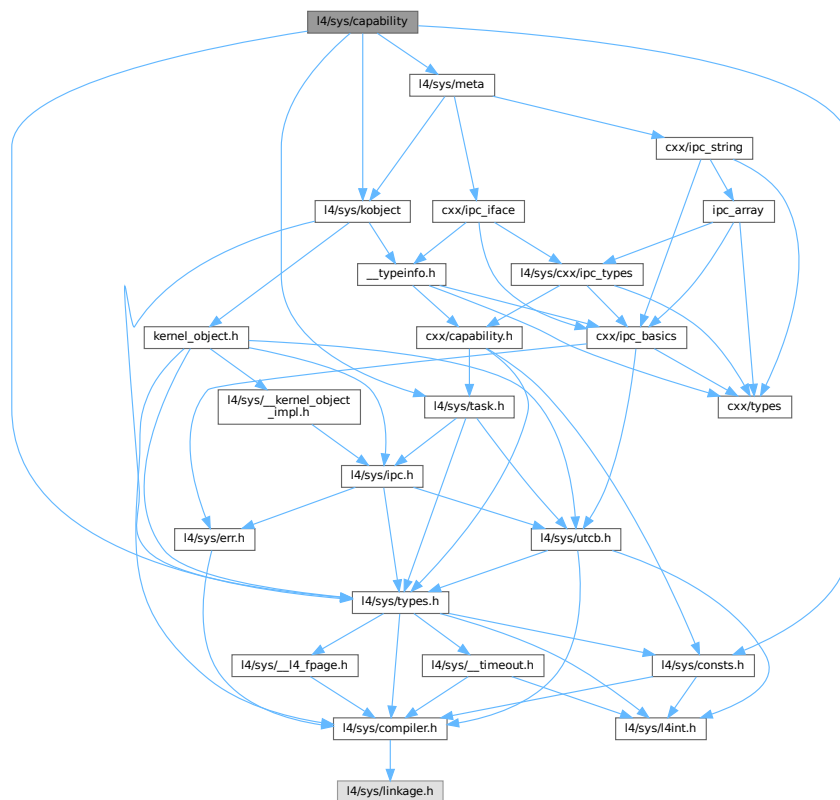
## 16.452 l4/sys/capability File Reference

[L4::Cap](#) related definitions.

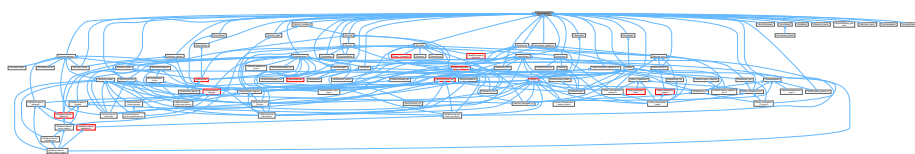


```
#include <l4/sys/consts.h>
#include <l4/sys/types.h>
#include <l4/sys/kobject>
#include <l4/sys/task.h>
#include <l4/sys/meta>
```

Include dependency graph for capability:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

## Macros

- #define [L4\\_DISABLE\\_COPY](#)(\_class)  
*Disable copy of a class.*

## Functions

- `template<typename T, typename F >`  
`Cap< T > L4::cap_dynamic_cast (Cap< F > const &c) noexcept`  
*dynamic\_cast for capabilities.*

### 16.452.1 Detailed Description

[L4::Cap](#) related definitions.

#### Author

Alexander Warg [alexander.warg@os.inf.tu-dresden.de](mailto:alexander.warg@os.inf.tu-dresden.de)

Definition in file [capability](#).

### 16.452.2 Macro Definition Documentation

#### 16.452.2.1 L4\_DISABLE\_COPY

```
#define L4_DISABLE_COPY(  
    _class )
```

#### Value:

```
public:
    _class(_class const &) = delete; \
    _class operator = (_class const &) = delete; \
private:
```

Disable copy of a class.

#### Parameters

<code>_class</code>	Name of the class that shall not have value copy semantics.
---------------------	---

The typical use of this is:

```
class Non_value
{
    L4_DISABLE_COPY(Non_value)
    ...
}
```

Definition at line 62 of file [capability](#).

## 16.453 capability

[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00009 /*
00010  * (c) 2008-2009,2015 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
```

```

00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction. Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License. This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 #pragma once
00027
00028 #include <l4/sys/consts.h>
00029 #include <l4/sys/types.h>
00030 #include <l4/sys/kobject>
00031 #include <l4/sys/task.h>
00032
00033 namespace L4
00034 {
00035
00036 /* Forward declarations for our kernel object classes. */
00037 class Task;
00038 class Thread;
00039 class Factory;
00040 class Irq;
00041 class Log;
00042 class Vm;
00043 class Vcpu_context;
00044 class Kobject;
00045
00061 #if __cplusplus >= 201103L
00062 # define L4_DISABLE_COPY(_class) \
00063     public: \
00064         _class(_class const &) = delete; \
00065         _class operator = (_class const &) = delete; \
00066     private:
00067 #else
00068 # define L4_DISABLE_COPY(_class) \
00069     private: \
00070         _class(_class const &); \
00071         _class operator = (_class const &);
00072 #endif
00073
00074 #define L4_KOBJECT_DISABLE_COPY(_class) \
00075     protected: \
00076         _class(); \
00077         L4_DISABLE_COPY(_class)
00078
00081 #define L4_KOBJECT(_class) L4_KOBJECT_DISABLE_COPY(_class)
00082
00083 inline l4_msgtag_t
00084 Cap_base::validate(Cap<Task> task, l4_utcb_t *u) const noexcept
00085 {
00086     return is_valid() ? l4_task_cap_valid_u(task.cap(), _c, u)
00087         : l4_msgtag(0, 0, 0, 0);
00088 }
00089
00090 inline l4_msgtag_t
00091 Cap_base::validate(l4_utcb_t *u) const noexcept
00092 {
00093     return is_valid() ? l4_task_cap_valid_u(L4_BASE_TASK_CAP, _c, u)
00094         : l4_msgtag(0, 0, 0, 0);
00095 }
00096
00097 }; // namespace L4
00098
00099 #include <l4/sys/meta>
00100
00101 namespace L4 {
00102
00124 template< typename T, typename F >
00125 inline
00126 Cap<T> cap_dynamic_cast(Cap<F> const &c) noexcept
00127 {
00128     if (!c.is_valid())
00129         return Cap<T>::Invalid;
00130
00131     Cap<Meta> mc = cap_reinterpret_cast<Meta>(c);
00132     Type_info const *m = kobject_typeid<T>();
00133     if (m->proto() && l4_error(mc->supports(m->proto())) > 0)
00134         return Cap<T>(c.cap());
00135 }

```

```

00136 // FIXME: use generic checker
00137 #if 0
00138 if (l4_error(mc->supports(T::kobject_proto())) > 0)
00139     return Cap<T>(c.cap());
00140 #endif
00141
00142 return Cap<T>::Invalid;
00143 }
00144
00145 }

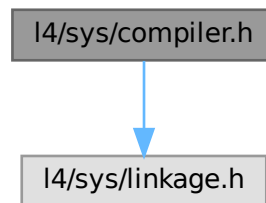
```

## 16.454 l4/sys/compiler.h File Reference

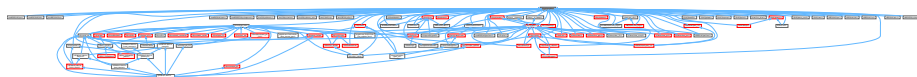
L4 compiler related defines.

```
#include <l4/sys/linkage.h>
```

Include dependency graph for compiler.h:



This graph shows which files directly or indirectly include this file:



### Macros

- **#define L4\_INLINE**  
*L4 Inline function attribute.*
- **#define L4\_ALWAYS\_INLINE**  
*Always inline a function.*
- **#define L4\_NOTHROW**  
*Mark a function declaration and definition as never throwing an exception.*
- **#define EXTERN\_C\_BEGIN**  
*Start section with C types and functions.*
- **#define EXTERN\_C\_END**  
*End section with C types and functions.*
- **#define EXTERN\_C**  
*Mark C types and functions.*
- **#define \_\_BEGIN\_DECLS**

- Start section with C types and functions.*

  - **#define `__END_DECLS`**

*End section with C types and functions.*
- **#define `L4_CONSTEXPR`**

*Constexpr function attribute.*
- **#define `L4_NORETURN`**

*Noreturn function attribute.*
- **#define `L4_NOINSTRUMENT`**

*No instrumentation function attribute.*
- **#define `L4_HIDDEN`**

*Attribute to mark functions, variables, and data types as being explicitly hidden from users of a library.*
- **#define `L4_EXPORT`**

*Attribute to mark functions, variables, and data types as being exported from a library.*
- **#define `L4_LIKELY(x)`**

*Expression is likely to execute.*
- **#define `L4_UNLIKELY(x)`**

*Expression is unlikely to execute.*
- **#define `L4_STICKY(x)`**

*Mark symbol sticky (even not there)*
- **#define `L4_DEPRECATED(s)`**

*Mark symbol deprecated.*
- **#define `L4_stringify_helper(x)`**

*stringify helper.*
- **#define `L4_stringify(x)`**

*stringify.*

## Functions

- unsigned long `l4_align_stack_for_direct_fncall` (unsigned long stack)

*Specify the desired alignment of the stack pointer.*
- void `l4_barrier` (void)

*Memory barrier.*
- void `l4_mb` (void)

*Memory barrier.*
- void `l4_wmb` (void)

*Write memory barrier.*
- `L4_NORETURN` void `l4_infinite_loop` (void)

*Infinite loop.*

## 16.454.1 Detailed Description

[L4](#) compiler related defines.

Definition in file [compiler.h](#).

## 16.455 compiler.h

[Go to the documentation of this file.](#)

```

00001 /*****
00007 */
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 *      Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00011 *      Jork Löser <jork@os.inf.tu-dresden.de>,
00012 *      Ronald Aigner <ra3@os.inf.tu-dresden.de>
00013 *      economic rights: Technische Universität Dresden (Germany)
00014 *
00015 * This file is part of TUD:OS and distributed under the terms of the
00016 * GNU General Public License 2.
00017 * Please see the COPYING-GPL-2 file for details.
00018 *
00019 * As a special exception, you may use this file as part of a free software
00020 * library without restriction. Specifically, if other files instantiate
00021 * templates or use macros or inline functions from this file, or you compile
00022 * this file and link it with other files to produce an executable, this
00023 * file does not by itself cause the resulting executable to be covered by
00024 * the GNU General Public License. This exception does not however
00025 * invalidate any other reasons why the executable file might be covered by
00026 * the GNU General Public License.
00027 */
00028 /*****
00029 #ifndef __L4_COMPILER_H__
00030 #define __L4_COMPILER_H__
00031
00032 #if !defined(__ASSEMBLY__) && !defined(__ASSEMBLER__)
00033
00045 #ifndef L4_INLINE
00046 #ifndef __cplusplus
00047 #   ifdef __OPTIMIZE__
00048 #       define L4_INLINE_STATIC static __inline__
00049 #       define L4_INLINE_EXTERN extern __inline__
00050 #   else
00051 #       define L4_INLINE L4_INLINE_STATIC
00052 #   else
00053 #       define L4_INLINE L4_INLINE_EXTERN
00054 #   endif
00055 # else /* ! __OPTIMIZE__ */
00056 #   define L4_INLINE static
00057 #   endif /* ! __OPTIMIZE__ */
00058 #else /* __cplusplus */
00059 #   define L4_INLINE inline
00060 #endif /* __cplusplus */
00061 #elif defined DOXYGEN
00062 #   define L4_INLINE inline
00063 #endif /* L4_INLINE */
00064
00069 #define L4_ALWAYS_INLINE L4_INLINE __attribute__((__always_inline__))
00070
00071
00072 #define L4_DECLARE_CONSTRUCTOR(func, prio) \
00073     static inline __attribute__((constructor(prio))) void func ## _ctor_func(void) { func(); }
00074
00075
00173 #ifndef __cplusplus
00174 #   define L4_NOTHROW_A      __attribute__((nothrow))
00175 #   define L4_NOTHROW
00176 #   define EXTERN_C_BEGIN
00177 #   define EXTERN_C_END
00178 #   define EXTERN_C
00179 #   ifdef __BEGIN_DECLS
00180 #       define __BEGIN_DECLS
00181 #   endif
00182 #   ifdef __END_DECLS
00183 #       define __END_DECLS
00184 #   endif
00185 #   define L4_DEFAULT_PARAM(x)
00186 #else /* __cplusplus */
00187 #   if __cplusplus >= 201103L
00188 #       define L4_NOTHROW noexcept
00189 #   else /* C++ < 11 */
00190 #       define L4_NOTHROW throw()
00191 #   endif
00192 #   define EXTERN_C_BEGIN extern "C" {
00193 #   define EXTERN_C_END }
00194 #   define EXTERN_C extern "C"
00195 #   if !defined __BEGIN_DECLS || defined DOXYGEN
00196 #       define __BEGIN_DECLS extern "C" {
00197 #   endif
00198 #   if !defined __END_DECLS || defined DOXYGEN
00199 #       define __END_DECLS }

```

```

00200 # endif
00201 # define L4_DEFAULT_PARAM(x) = x
00202 #endif /* __cplusplus */
00203
00208 #if defined __cplusplus && __cplusplus >= 201402L
00209 # define L4_CONSTEXPR constexpr
00210 #else
00211 # define L4_CONSTEXPR
00212 #endif
00213
00218 #define L4_NORETURN __attribute__((noreturn))
00219
00220 #define L4_PURE __attribute__((pure))
00221
00226 #define L4_NOINSTRUMENT __attribute__((no_instrument_function))
00227 #ifndef L4_HIDDEN
00228 # define L4_HIDDEN __attribute__((visibility("hidden")))
00229 #endif
00230 #if !defined L4_EXPORT || defined DOXYGEN
00231 # define L4_EXPORT __attribute__((visibility("default")))
00232 #endif
00233 #ifndef L4_EXPORT_TYPE
00234 # ifdef __cplusplus
00235 # define L4_EXPORT_TYPE __attribute__((visibility("default")))
00236 # else
00237 # define L4_EXPORT_TYPE
00238 # endif
00239 #endif
00240 #define L4_STRONG_ALIAS(name, aliasname) L4__STRONG_ALIAS(name, aliasname)
00241 #define L4__STRONG_ALIAS(name, aliasname) \
00242     extern __typeof (name) aliasname __attribute__((alias (#name)));
00243
00251 #if defined(__i386__) || defined(__amd64__) || \
00252     defined(__arm__) || defined(__aarch64__) || \
00253     defined(__mips__) || defined(__riscv__) || \
00254     defined(__powerpc__) || defined(__sparc__)
00255 # define L4_STACK_ALIGN __BIGGEST_ALIGNMENT__
00256 #else
00257 # error Define L4_STACK_ALIGN for this target!
00258 #endif
00259
00277 #if defined(__i386__) || defined(__amd64__)
00278 L4_INLINE unsigned long l4_align_stack_for_direct_fncall(unsigned long stack)
00279 {
00280     if ((stack & (L4_STACK_ALIGN - 1)) == (L4_STACK_ALIGN - sizeof(unsigned long)))
00281         return stack;
00282     return (stack & ~(L4_STACK_ALIGN)) - sizeof(unsigned long);
00283 }
00284 #else
00285 L4_INLINE unsigned long l4_align_stack_for_direct_fncall(unsigned long stack)
00286 {
00287     return stack & ~(L4_STACK_ALIGN);
00288 }
00289 #endif
00290
00291 #endif /* !__ASSEMBLY__ */
00292
00293 #include <l4/sys/linkage.h>
00294
00295 #define L4_LIKELY(x) __builtin_expect((x),1)
00296 #define L4_UNLIKELY(x) __builtin_expect((x),0)
00297
00298 /* Make sure that the function is not removed by optimization. Without the
00299  * "used" attribute, unreferenced static functions are removed. */
00300 #define L4_STICKY(x) __attribute__((used)) x
00301 #define L4_DEPRECATED(s) __attribute__((deprecated(s)))
00302
00303 #ifndef static_assert
00304 # if !defined(__cplusplus)
00305 # define static_assert(x, y) _Static_assert(x, y)
00306 # elif __cplusplus < 201103L
00307 # define static_assert(x, y) \
00308     extern int l4_static_assert[-(!x)] __attribute__((unused))
00309 # endif
00310 #endif
00311
00312 #define L4_stringify_helper(x) #x
00313 #define L4_stringify(x) L4_stringify_helper(x)
00314
00315 #ifndef __has_builtin
00316 #define L4_HAS_BUILTIN(def) __has_builtin(def)
00317 #else
00318 #define L4_HAS_BUILTIN(def) 0
00319 #endif
00320
00321 #ifndef __ASSEMBLER__
00325 L4_INLINE void l4_barrier(void);

```

```

00326
00330 L4_INLINE void l4_mb(void);
00331
00335 L4_INLINE void l4_wmb(void);
00336
00340 L4_INLINE L4_NORETURN void l4_infinite_loop(void);
00341
00342
00343 /* Implementations */
00344 L4_INLINE void l4_barrier(void)
00345 {
00346     __asm__ __volatile__ (" : : : "memory");
00347 }
00348
00349 L4_INLINE void l4_mb(void)
00350 {
00351     __asm__ __volatile__ (" : : : "memory");
00352 }
00353
00354 L4_INLINE void l4_wmb(void)
00355 {
00356     __asm__ __volatile__ (" : : : "memory");
00357 }
00358
00359 L4_INLINE L4_NORETURN void l4_infinite_loop(void)
00360 {
00361     while (1)
00362         l4_barrier();
00363 }
00364 #endif
00365
00368 #endif /* !__L4_COMPILER_H__ */

```

## 16.456 capability.h

```

00001
00002 #pragma once
00003
00004 #include <l4/sys/consts.h>
00005 #include <l4/sys/types.h>
00006 #include <l4/sys/task.h>
00007
00008 namespace L4 {
00009
00010 class Task;
00011 class Kobject;
00012
00013 template< typename T > class L4_EXPORT Cap;
00014
00025 class L4_EXPORT Cap_base
00026 {
00027 public:
00029     enum No_init_type
00030     {
00034         No_init
00035     };
00036
00040     enum Cap_type
00041     {
00042         Invalid = L4_INVALID_CAP
00043     };
00044
00049     l4_cap_idx_t cap() const noexcept { return _c; }
00050
00057     bool is_valid() const noexcept { return !(_c & L4_INVALID_CAP_BIT); }
00058
00059     explicit operator bool () const noexcept
00060     { return !(_c & L4_INVALID_CAP_BIT); }
00061
00069     l4_fpage_t fpage(unsigned rights = L4_CAP_FPAGE_RWS) const noexcept
00070     { return l4_obj_fpage(_c, 0, rights); }
00071
00081     l4_umword_t snd_base(unsigned grant = L4_MAP_ITEM_MAP,
00082                          l4_cap_idx_t base = L4_INVALID_CAP) const noexcept
00083     {
00084         if (base == L4_INVALID_CAP)
00085             base = _c;
00086         return l4_map_obj_control(base, grant);
00087     }
00088
00089
00093     bool operator == (Cap_base const &o) const noexcept
00094     { return _c == o._c; }

```



```

00095
00099 bool operator != (Cap_base const &o) const noexcept
00100 { return _c != o._c; }
00101
00115 inline l4_msgtag_t validate(l4_utcb_t *u = l4_utcb()) const noexcept;
00116
00131 inline l4_msgtag_t validate(Cap<Task> task,
00132                             l4_utcb_t *u = l4_utcb()) const noexcept;
00133
00137 void invalidate() noexcept { _c = L4_INVALID_CAP; }
00138 protected:
00144 explicit Cap_base(l4_cap_idx_t c) noexcept : _c(c) {}
00148 explicit Cap_base(Cap_type cap) noexcept : _c(cap) {}
00149
00155 explicit Cap_base(l4_default_caps_t cap) noexcept : _c(cap) {}
00156
00160 explicit Cap_base() noexcept {}
00161
00171 void move(Cap_base const &src) const
00172 {
00173     if (!is_valid() || !src.is_valid())
00174         return;
00175
00176     l4_task_map(L4_BASE_TASK_CAP, L4_BASE_TASK_CAP, src.fpage(L4_CAP_FPAGE_RWSD),
00177                 snd_base(L4_MAP_ITEM_GRANT) | L4_FPAGE_C_OBJ_RIGHTS);
00178 }
00179
00187 void copy(Cap_base const &src) const
00188 {
00189     if (!is_valid() || !src.is_valid())
00190         return;
00191
00192     l4_task_map(L4_BASE_TASK_CAP, L4_BASE_TASK_CAP, src.fpage(L4_CAP_FPAGE_RWSD),
00193                 snd_base() | L4_FPAGE_C_OBJ_RIGHTS);
00194 }
00195
00198 l4_cap_idx_t _c;
00199 };
00200
00201
00217 template< typename T >
00218 class L4_EXPORT Cap : public Cap_base
00219 {
00220 private:
00221     friend class L4::Kobject;
00222
00234 explicit Cap(T const *p) noexcept
00235 : Cap_base(reinterpret_cast<l4_cap_idx_t>(p)) {}
00236
00237 public:
00238
00245 template< typename From >
00246 static void check_convertible_from() noexcept
00247 {
00248     using To = T;
00249     [[maybe_unused]] To* t = static_cast<From*>(nullptr);
00250 }
00251
00258 template< typename From >
00259 static void check_castable_from() noexcept
00260 {
00261     using To = T;
00262     [[maybe_unused]] To *t = static_cast<To *>(static_cast<From *>(nullptr));
00263 }
00264
00269 template< typename O >
00270 Cap(Cap<O> const &o) noexcept : Cap_base(o.cap())
00271 { check_convertible_from<O>(); }
00272
00277 Cap(Cap_type cap) noexcept : Cap_base(cap) {}
00278
00283 Cap(l4_default_caps_t cap) noexcept : Cap_base(cap) {}
00284
00289 explicit Cap(l4_cap_idx_t idx = L4_INVALID_CAP) noexcept : Cap_base(idx) {}
00290
00294 explicit Cap(No_init_type) noexcept {}
00295
00302 Cap move(Cap const &src) const
00303 {
00304     Cap_base::move(src);
00305     return *this;
00306 }
00307
00312 Cap copy(Cap const &src) const
00313 {
00314     Cap_base::copy(src);
00315     return *this;

```

```

00316     }
00317
00321 T *operator -> () const noexcept { return reinterpret_cast<T*>(_c); }
00322 };
00323
00324
00335 template<>
00336 class L4_EXPORT Cap<void> : public Cap_base
00337 {
00338 public:
00339
00340     explicit Cap(void const *p) noexcept
00341     : Cap_base(reinterpret_cast<l4_cap_idx_t>(p)) {}
00342
00346     Cap(Cap_type cap) noexcept : Cap_base(cap) {}
00347
00352     Cap(l4_default_caps_t cap) noexcept : Cap_base(cap) {}
00353
00358     explicit Cap(l4_cap_idx_t idx = L4_INVALID_CAP) noexcept : Cap_base(idx) {}
00359     explicit Cap(No_init_type) noexcept {}
00360
00361     template< typename From >
00362     static void check_convertible_from() noexcept {}
00363
00364     template< typename From >
00365     static void check_castable_from() noexcept {}
00366
00373     Cap move(Cap const &src) const
00374     {
00375         Cap_base::move(src);
00376         return *this;
00377     }
00378
00383     Cap copy(Cap const &src) const
00384     {
00385         Cap_base::copy(src);
00386         return *this;
00387     }
00388
00389     template< typename T >
00390     Cap(Cap<T> const &o) noexcept : Cap_base(o.cap()) {}
00391 };
00392
00409 template< typename T, typename F >
00410 inline
00411 Cap<T> cap_cast(Cap<F> const &c) noexcept
00412 {
00413     Cap<T>::template check_castable_from<F>();
00414     return Cap<T>(c.cap());
00415 }
00416
00417 // gracefully deal with L4::Kobject ambiguity
00418 template< typename T >
00419 inline
00420 Cap<T> cap_cast(Cap<L4::Kobject> const &c) noexcept
00421 {
00422     return Cap<T>(c.cap());
00423 }
00424
00440 template< typename T, typename F >
00441 inline
00442 Cap<T> cap_reinterpret_cast(Cap<F> const &c) noexcept
00443 {
00444     return Cap<T>(c.cap());
00445 }
00446
00447 }

```

## 16.457 ipc\_array

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by

```

```

00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018  #pragma once
00019
00020  #include "types"
00021  #include "ipc_basics"
00022  #include "ipc_types"
00023
00024  namespace L4 { namespace Ipc L4_EXPORT {
00025
00027  typedef unsigned short Array_len_default;
00028
00038  template< typename ELEM_TYPE, typename LEN_TYPE = Array_len_default >
00039  struct Array_ref
00040  {
00041      typedef ELEM_TYPE *ptr_type;
00042      typedef LEN_TYPE len_type;
00043
00044      len_type length;
00045      ptr_type data;
00046      Array_ref() = default;
00047      Array_ref(len_type length, ptr_type data)
00048      : length(length), data(data)
00049      {}
00050
00051      template<typename X> struct Non_const
00052      { typedef Array_ref<X, LEN_TYPE> type; };
00053
00054      template<typename X> struct Non_const<X const>
00055      { typedef Array_ref<X, LEN_TYPE> type; };
00056
00057      Array_ref(typename Non_const<ELEM_TYPE>::type const &other)
00058      : length(other.length), data(other.data)
00059      {}
00060
00061      Array_ref &operator = (typename Non_const<ELEM_TYPE>::type const &other)
00062      {
00063          this->length = other.length;
00064          this->data = other.data;
00065          return *this;
00066      }
00067  };
00068
00091  template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default>
00092  struct Array : Array_ref<ELEM_TYPE, LEN_TYPE>
00093  {
00095      Array() {}
00097      Array(LEN_TYPE length, ELEM_TYPE *data)
00098      : Array_ref<ELEM_TYPE, LEN_TYPE>(length, data)
00099      {}
00100
00101      template<typename X> struct Non_const
00102      { typedef Array<X, LEN_TYPE> type; };
00103
00104      template<typename X> struct Non_const<X const>
00105      { typedef Array<X, LEN_TYPE> type; };
00106
00108      Array(typename Non_const<ELEM_TYPE>::type const &other)
00109      : Array_ref<ELEM_TYPE, LEN_TYPE>(other.length, other.data)
00110      {}
00111
00112      Array &operator = (typename Non_const<ELEM_TYPE>::type const &other)
00113      {
00114          this->length = other.length;
00115          this->data = other.data;
00116          return *this;
00117      }
00118  };
00119
00133  template< typename ELEM_TYPE,
00134             typename LEN_TYPE = Array_len_default,
00135             LEN_TYPE MAX      = (L4_UTCB_GENERIC_DATA_SIZE *
00136                                 sizeof(l4_umword_t)) / sizeof(ELEM_TYPE) >
00137  struct Array_in_buf
00138  {
00139      typedef Array_ref<ELEM_TYPE, LEN_TYPE> array;
00140      typedef Array_ref<ELEM_TYPE const, LEN_TYPE> const_array;
00141
00143      ELEM_TYPE data[MAX];
00145      LEN_TYPE length;
00146
00148      void copy_in(const_array a)
00149      {
00150          length = a.length;
00151          if (length > MAX)

```

```

00152     length = MAX;
00153
00154     for (LEN_TYPE i = 0; i < length; ++i)
00155         data[i] = a.data[i];
00156     }
00157
00159     Array_in_buf(const_array a) { copy_in(a); }
00161     Array_in_buf(array a) { copy_in(a); }
00162 };
00163
00164 // implementation details for transmission
00165 namespace Msg {
00166
00168     template<typename A, typename LEN>
00169     struct Elem< Array<A, LEN> >
00170     {
00172         typedef Array<A, LEN> arg_type;
00174         typedef Array_ref<A, LEN> svr_type;
00175         typedef svr_type svr_arg_type;
00176         enum { Is_optional = false };
00177     };
00178
00180     template<typename A, typename LEN>
00181     struct Elem< Array<A, LEN> & >
00182     {
00184         typedef Array<A, LEN> &arg_type;
00186         typedef Array_ref<A, LEN> svr_type;
00188         typedef svr_type &svr_arg_type;
00189         enum { Is_optional = false };
00190     };
00191
00193     template<typename A, typename LEN>
00194     struct Elem< Array_ref<A, LEN> & >
00195     {
00197         typedef Array_ref<A, LEN> &arg_type;
00199         typedef Array_ref<typename L4::Types::Remove_const<A>::type, LEN> svr_type;
00201         typedef svr_type &svr_arg_type;
00202         enum { Is_optional = false };
00203     };
00204
00205     template<typename A> struct Class<Array<A> > : Class<A>::type {};
00206     template<typename A> struct Class<Array_ref<A> > : Class<A>::type {};
00207
00208     namespace Detail {
00209
00210         template<typename A, typename LEN, typename ARRAY, bool REF>
00211         struct Clnt_val_ops_d_in : Clnt_noops<ARRAY>
00212         {
00213             using Clnt_noops<ARRAY>::to_msg;
00214             static int to_msg(char *msg, unsigned offset, unsigned limit,
00215                             ARRAY a, Dir_in, Cls_data)
00216             {
00217                 offset = align_to<LEN>(offset);
00218                 if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00219                     return -L4_MSGTOOLONG;
00220                 *reinterpret_cast<LEN*>(msg + offset) = a.length;
00221                 offset = align_to<A>(offset + sizeof(LEN));
00222                 if (L4_UNLIKELY(!check_size<A>(offset, limit, a.length)))
00223                     return -L4_MSGTOOLONG;
00224                 typedef typename L4::Types::Remove_const<A>::type elem_type;
00225                 elem_type *data = reinterpret_cast<elem_type*>(msg + offset);
00226
00227                 // we do not correctly handle overlaps
00228                 if (!REF || data != a.data)
00229                 {
00230                     for (LEN i = 0; i < a.length; ++i)
00231                         data[i] = a.data[i];
00232                 }
00233                 return offset + a.length * sizeof(A);
00234             }
00235         };
00236     };
00237 } // namespace Detail
00238
00239 template<typename A, typename LEN>
00240 struct Clnt_val_ops<Array<A, LEN>, Dir_in, Cls_data> :
00241     Detail::Clnt_val_ops_d_in<A, LEN, Array<A, LEN>, false> {};
00242
00243 template<typename A, typename LEN>
00244 struct Clnt_val_ops<Array_ref<A, LEN>, Dir_in, Cls_data> :
00245     Detail::Clnt_val_ops_d_in<A, LEN, Array_ref<A, LEN>, true> {};
00246
00247 template<typename A, typename LEN, typename CLASS>
00248 struct Svr_val_ops< Array_ref<A, LEN>, Dir_in, CLASS >
00249 : Svr_noops< Array_ref<A, LEN> >
00250 {
00251     typedef Array_ref<A, LEN> svr_type;

```

```

00252
00253 using Svr_noops<svr_type>::to_svr;
00254 static int to_svr(char *msg, unsigned offset, unsigned limit,
00255                  svr_type &a, Dir_in, Cls_data)
00256 {
00257     offset = align_to<LEN>(offset);
00258     if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00259         return -L4_MSGTOOSHORT;
00260     a.length = *reinterpret_cast<LEN*>(msg + offset);
00261     offset = align_to<A>(offset + sizeof(LEN));
00262     if (L4_UNLIKELY(!check_size<A>(offset, limit, a.length)))
00263         return -L4_MSGTOOSHORT;
00264     a.data = reinterpret_cast<A*>(msg + offset);
00265     return offset + a.length * sizeof(A);
00266 }
00267 };
00268
00269 template<typename A, typename LEN>
00270 struct Svr_xmit<Array<A, LEN> > : Svr_xmit<Array_ref<A, LEN> > {};
00271
00272 template<typename A, typename LEN>
00273 struct Clnt_val_ops<Array<A, LEN>, Dir_out, Cls_data> : Clnt_noops<Array<A, LEN> >
00274 {
00275     typedef Array<A, LEN> type;
00276
00277     using Clnt_noops<type>::from_msg;
00278     static int from_msg(char *msg, unsigned offset, unsigned limit, long,
00279                        type &a, Dir_out, Cls_data)
00280     {
00281         offset = align_to<LEN>(offset);
00282         if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00283             return -L4_MSGTOOSHORT;
00284
00285         LEN l = *reinterpret_cast<LEN*>(msg + offset);
00286
00287         offset = align_to<A>(offset + sizeof(LEN));
00288         if (L4_UNLIKELY(!check_size<A>(offset, limit, l)))
00289             return -L4_MSGTOOSHORT;
00290
00291         A *data = reinterpret_cast<A*>(msg + offset);
00292
00293         if (l > a.length)
00294             l = a.length;
00295         else
00296             a.length = l;
00297
00298         for (unsigned i = 0; i < l; ++i)
00299             a.data[i] = data[i];
00300
00301         return offset + l * sizeof(A);
00302     };
00303 };
00304
00305 template<typename A, typename LEN>
00306 struct Clnt_val_ops<Array_ref<A, LEN>, Dir_out, Cls_data> :
00307     Clnt_noops<Array_ref<A, LEN> >
00308 {
00309     typedef Array_ref<A, LEN> type;
00310
00311     using Clnt_noops<type>::from_msg;
00312     static int from_msg(char *msg, unsigned offset, unsigned limit, long,
00313                        type &a, Dir_out, Cls_data)
00314     {
00315         offset = align_to<LEN>(offset);
00316         if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00317             return -L4_MSGTOOSHORT;
00318
00319         LEN l = *reinterpret_cast<LEN*>(msg + offset);
00320
00321         offset = align_to<A>(offset + sizeof(LEN));
00322         if (L4_UNLIKELY(!check_size<A>(offset, limit, l)))
00323             return -L4_MSGTOOSHORT;
00324
00325         a.data = reinterpret_cast<A*>(msg + offset);
00326         a.length = l;
00327         return offset + l * sizeof(A);
00328     };
00329 };
00330
00331 template<typename A, typename LEN, typename CLASS>
00332 struct Svr_val_ops<Array_ref<A, LEN>, Dir_out, CLASS> :
00333     Svr_noops<Array_ref<typename L4::Types::Remove_const<A>::type, LEN> &>
00334 {
00335     typedef typename L4::Types::Remove_const<A>::type elem_type;
00336     typedef Array_ref<elem_type, LEN> &svr_type;
00337
00338     using Svr_noops<svr_type>::to_svr;

```

```

00339 static int to_svr(char *msg, unsigned offset, unsigned limit,
00340                  svr_type a, Dir_out, Cls_data)
00341 {
00342     offset = align_to<LEN>(offset);
00343     if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00344         return -L4_MSGTOOLONG;
00345
00346     offset = align_to<A>(offset + sizeof(LEN));
00347     a.data = reinterpret_cast<elem_type *>(msg + offset);
00348     a.length = (limit - offset) / sizeof(A);
00349     return offset;
00350 }
00351
00352 using Svr_noops<svr_type>::from_svr;
00353 static int from_svr(char *msg, unsigned offset, unsigned limit, long,
00354                   svr_type a, Dir_out, Cls_data)
00355 {
00356     offset = align_to<LEN>(offset);
00357     if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00358         return -L4_MSGTOOLONG;
00359
00360     *reinterpret_cast<LEN *>(msg + offset) = a.length;
00361
00362     offset = align_to<A>(offset + sizeof(LEN));
00363     if (L4_UNLIKELY(!check_size<A>(offset, limit, a.length)))
00364         return -L4_MSGTOOLONG;
00365
00366     return offset + a.length * sizeof(A);
00367 }
00368 };
00369
00370 template<typename A, typename LEN>
00371 struct Svr_xmit<Array<A, LEN> &> : Svr_xmit<Array_ref<A, LEN> &> {};
00372
00373 // Pointer to array is not implemented.
00374 template<typename A, typename LEN>
00375 struct Is_valid_rpc_type<Array_ref<A, LEN> *> : L4::Types::False {};
00376
00377 // Optional input arrays are not implemented.
00378 template<typename A, typename LEN>
00379 struct Is_valid_rpc_type<Opt<Array_ref<A, LEN> > > : L4::Types::False {};
00380
00381 } // namespace Msg
00382
00383 }

```

## 16.458 ipc\_basics

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019
00020 #include "types"
00021 #include <l4/sys/utcb.h>
00022 #include <l4/sys/err.h>
00023
00024 namespace L4 {
00025
00026     namespace Ipc {
00027
00028         namespace Msg {
00029
00030             using L4::Types::True;
00031             using L4::Types::False;
00032
00033             constexpr unsigned long align_to(unsigned long bytes, unsigned long align) noexcept
00034             { return (bytes + align - 1) & ~(align - 1); }
00035
00036         }
00037     }
00038 }

```

```

00050 template<typename T>
00051 constexpr unsigned long align_to(unsigned long bytes) noexcept
00052 { return align_to(bytes, __alignof(T)); }
00053
00063 template<typename T>
00064 constexpr bool check_size(unsigned offset, unsigned limit) noexcept
00065 {
00066     return offset + sizeof(T) <= limit;
00067 }
00068
00081 template<typename T, typename CTYPE>
00082 inline bool check_size(unsigned offset, unsigned limit, CTYPE cnt) noexcept
00083 {
00084     if (L4_UNLIKELY(sizeof(CTYPE) <= sizeof(unsigned) &&
00085                     ~0U / sizeof(T) <= static_cast<unsigned>(cnt)))
00086         return false;
00087
00088     if (L4_UNLIKELY(sizeof(CTYPE) > sizeof(unsigned) &&
00089                     static_cast<CTYPE>(~0U / sizeof(T)) <= cnt))
00090         return false;
00091
00092     return sizeof(T) * cnt <= limit - offset;
00093 }
00094
00095
00096 enum
00097 {
00099     Word_bytes = sizeof(l4_umword_t),
00101     Item_words = 2,
00103     Item_bytes = Word_bytes * Item_words,
00105     Mr_words    = L4_UTCB_GENERIC_DATA_SIZE,
00107     Mr_bytes    = Word_bytes * Mr_words,
00109     Br_bytes    = Word_bytes * L4_UTCB_GENERIC_BUFFERS_SIZE,
00110 };
00111
00112
00124 template<typename T>
00125 inline int msg_add(char *msg, unsigned offs, unsigned limit, T v) noexcept
00126 {
00127     offs = align_to<T>(offs);
00128     if (L4_UNLIKELY(!check_size<T>(offs, limit)))
00129         return -L4_MSGTOOLONG;
00130     *reinterpret_cast<typename L4::Types::Remove_const<T>::type *>(msg + offs) = v;
00131     return offs + sizeof(T);
00132 }
00133
00145 template<typename T>
00146 inline int msg_get(char *msg, unsigned offs, unsigned limit, T &v) noexcept
00147 {
00148     offs = align_to<T>(offs);
00149     if (L4_UNLIKELY(!check_size<T>(offs, limit)))
00150         return -L4_MSGTOOSHORT;
00151     v = *reinterpret_cast<T *>(msg + offs);
00152     return offs + sizeof(T);
00153 }
00154
00156 struct Dir_in { typedef Dir_in type;   typedef Dir_in dir; };
00158 struct Dir_out { typedef Dir_out type; typedef Dir_out dir; };
00159
00161 struct Cls_data { typedef Cls_data type;   typedef Cls_data cls; };
00163 struct Cls_item { typedef Cls_item type;   typedef Cls_item cls; };
00165 struct Cls_buffer { typedef Cls_buffer type; typedef Cls_buffer cls; };
00166
00167 // Typical combinations
00169 struct Do_in_data : Dir_in, Cls_data {};
00171 struct Do_out_data : Dir_out, Cls_data {};
00173 struct Do_in_items : Dir_in, Cls_item {};
00175 struct Do_out_items : Dir_out, Cls_item {};
00177 struct Do_rcv_buffers : Dir_in, Cls_buffer {};
00178
00179 // implementation details
00180 namespace Detail {
00181
00182 template<typename T> struct _Plain
00183 {
00184     typedef T type;
00185     static T deref(T v) noexcept { return v; }
00186 };
00187
00188 template<typename T> struct _Plain<T *>
00189 {
00190     typedef T type;
00191     static T &deref(T *v) noexcept { return *v; }
00192 };
00193
00194 template<typename T> struct _Plain<T &>
00195 {

```

```

00196     typedef T type;
00197     static T &deref(T &v) noexcept { return v; }
00198
00199 };
00200
00201 template<typename T> struct _Plain<T const &>
00202 {
00203     typedef T type;
00204     static T const &deref(T const &v) noexcept { return v; }
00205 };
00206
00207 template<typename T> struct _Plain<T const *>
00208 {
00209     typedef T type;
00210     static T const &deref(T const *v) noexcept { return *v; }
00211 };
00212 }
00213
00221 template<typename MTYPE, typename DIR, typename CLASS> struct Clnt_val_ops;
00222
00223 template<typename T> struct Clnt_noops
00224 {
00225     template<typename A, typename B>
00226     static constexpr int to_msg(char *, unsigned offset, unsigned, T, A, B) noexcept
00227     { return offset; }
00228
00229     template<typename A, typename B>
00230     static constexpr int from_msg(char *, unsigned offset, unsigned, long, T const &, A, B) noexcept
00231     { return offset; }
00232 };
00233
00234 template<typename T> struct Svr_noops
00235 {
00236     template<typename A, typename B>
00237     static constexpr int from_svr(char *, unsigned offset, unsigned, long, T, A, B) noexcept
00238     { return offset; }
00239
00240     template<typename A, typename B>
00241     static constexpr int to_svr(char *, unsigned offset, unsigned, T, A, B) noexcept
00242     { return offset; }
00243 };
00244
00245 template<typename MTYPE, typename CLASS>
00246 struct Clnt_val_ops<MTYPE, Dir_in, CLASS> : Clnt_noops<MTYPE>
00247 {
00248     using Clnt_noops<MTYPE>::to_msg;
00249     static int to_msg(char *msg, unsigned offset, unsigned limit,
00250                     MTYPE arg, Dir_in, CLASS) noexcept
00251     { return msg_add<MTYPE>(msg, offset, limit, arg); }
00252 };
00253
00254 template<typename MTYPE, typename CLASS>
00255 struct Clnt_val_ops<MTYPE, Dir_out, CLASS> : Clnt_noops<MTYPE>
00256 {
00257     using Clnt_noops<MTYPE>::from_msg;
00258     static int from_msg(char *msg, unsigned offset, unsigned limit, long,
00259                       MTYPE &arg, Dir_out, CLASS) noexcept
00260     { return msg_get<MTYPE>(msg, offset, limit, arg); }
00261 };
00262
00263 template<typename MTYPE, typename DIR, typename CLASS> struct Svr_val_ops;
00264
00265 template<typename MTYPE, typename CLASS>
00266 struct Svr_val_ops<MTYPE, Dir_in, CLASS> : Svr_noops<MTYPE>
00267 {
00268     using Svr_noops<MTYPE>::to_svr;
00269     static int to_svr(char *msg, unsigned offset, unsigned limit,
00270                     MTYPE &arg, Dir_in, CLASS) noexcept
00271     { return msg_get<MTYPE>(msg, offset, limit, arg); }
00272 };
00273
00274 template<typename MTYPE, typename CLASS>
00275 struct Svr_val_ops<MTYPE, Dir_out, CLASS> : Svr_noops<MTYPE>
00276 {
00277     using Svr_noops<MTYPE>::to_svr;
00278     static int to_svr(char *msg, unsigned offs, unsigned limit,
00279                     MTYPE &, Dir_out, CLASS) noexcept
00280     {
00281         offs = align_to<MTYPE>(offs);
00282         if (L4_UNLIKELY(!check_size<MTYPE>(offs, limit)))
00283             return -L4_MSGTOOLONG;
00284         return offs + sizeof(MTYPE);
00285     }
00286
00287     using Svr_noops<MTYPE>::from_svr;
00288     static int from_svr(char *msg, unsigned offset, unsigned limit, long,

```



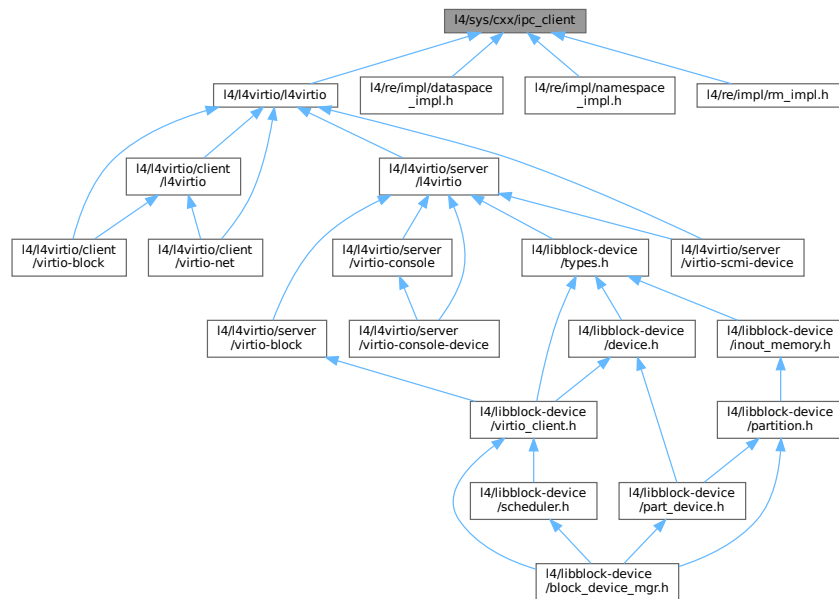
```

00303             MTYPE arg, Dir_out, CLASS) noexcept
00304     { return msg_add<MTYPE>(msg, offset, limit, arg); }
00305 };
00306
00307 template<typename T> struct Elem
00308 {
00310     typedef T arg_type;
00312     typedef T svr_type;
00314     typedef T svr_arg_type; // might by const & (depending on the size)
00315
00316     enum { Is_optional = false };
00317 };
00318 };
00319
00320 template<typename T> struct Elem<T &>
00321 {
00322     typedef T &arg_type;
00323     typedef T svr_type;
00325     typedef T &svr_arg_type;
00326     enum { Is_optional = false };
00327 };
00328
00329 template<typename T> struct Elem<T const &>
00330 {
00331     typedef T const &arg_type;
00332     typedef T svr_type;
00333     // as the RPC uses a const reference we use it here too,
00334     // we could also use pass by value depending on the size
00335     typedef T const &svr_arg_type;
00336     enum { Is_optional = false };
00337 };
00338
00339 template<typename T> struct Elem<T *> : Elem<T &>
00340 {
00341     typedef T *arg_type;
00342 };
00343
00344 template<typename T> struct Elem<T const *> : Elem<T const &>
00345 {
00346     typedef T const *arg_type;
00347 };
00348
00350 template<typename T> struct Is_valid_rpc_type : L4::Types::True {};
00351
00352 // Static assertions outside functions work only properly from C++11
00353 // onwards. On earlier version make sure the compiler fails on an ugly
00354 // undefined struct instead.
00355 template<typename T, bool B> struct Error_invalid_rpc_parameter_used;
00356 template<typename T> struct Error_invalid_rpc_parameter_used<T, true> {};
00357
00358 #if __cplusplus >= 201103L
00359 template<typename T>
00360 struct _Elem : Elem<T>
00361 {
00362     static_assert(Is_valid_rpc_type<T>::value,
00363         "L4::Rpc::Msg::_Elem<T>: type T is not a valid RPC parameter type.");
00364 };
00365 #else
00366 template<typename T>
00367 struct _Elem : Elem<T>,
00368     Error_invalid_rpc_parameter_used<T, Is_valid_rpc_type<T>::value>
00369 {};
00370 #endif
00371
00372
00373 template<typename T> struct Class : Cls_data {};
00374 template<typename T> struct Direction : Dir_in {};
00375 template<typename T> struct Direction<T const &> : Dir_in {};
00376 template<typename T> struct Direction<T const *> : Dir_in {};
00377 template<typename T> struct Direction<T &> : Dir_out {};
00378 template<typename T> struct Direction<T *> : Dir_out {};
00379
00380 template<typename T> struct _Clnt_noops :
00381     Clnt_noops<typename Detail::_Plain<typename _Elem<T>::arg_type>::type>
00382 {};
00383
00384 namespace Detail {
00385
00386     template<typename T, typename DIR, typename CLASS>
00387     struct _Clnt_val_ops :
00388         Clnt_val_ops<typename Detail::_Plain<T>::type, DIR, CLASS> {};
00389
00390     template<typename T,
00391             typename ELEM = _Elem<T>,
00392             typename CLNT_OPS = _Clnt_val_ops<typename ELEM::arg_type,
00393                 typename Direction<T>::type,
00394                 typename Class<typename Detail::_Plain<T>::type>::type>

```



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace **L4**  
*L4 low-level kernel interface.*
- namespace **L4::lpc**  
*IPC related functionality.*
- namespace **L4::lpc::Msg**  
*IPC Message related functionality.*

## Macros

- `#define L4_RPC_DEF(name)`  
*Generate the definition of an RPC stub.*

## 16.459.1 Macro Definition Documentation

### 16.459.1.1 L4\_RPC\_DEF

```
#define L4_RPC_DEF(  
    name )
```

#### Value:

```
template struct L4::Ipc::Msg::Rpc_call \  
    <name##_t, name##_t::class_type, name##_t::ipc_type, name##_t::flags_type>
```

Generate the definition of an RPC stub.

## Parameters

<i>name</i>	The fully qualified method name to be implemented, this means <code>class::method</code> .
-------------	--

This macro generates the definition (implementation) for the given RPC interface method.

Definition at line 43 of file [ipc\\_client](#).

## 16.460 ipc\_client

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019 #pragma GCC system_header
00020
00021 #include <l4/sys/cxx/ipc_basics>
00022 #include <l4/sys/cxx/ipc_types>
00023 #include <l4/sys/cxx/ipc_iface>
00024 #include <l4/sys/__typeinfo.h>
00025 #include <l4/sys/err.h>
00026
00031 namespace L4 { namespace Ipc { namespace Msg {
00032 //-----
00033
00043 #define L4_RPC_DEF(name) \
00044     template struct L4::Ipc::Msg::Rpc_call \
00045         <name##_t, name##_t::class_type, name##_t::ipc_type, name##_t::flags_type>
00046
00047
00049 //-----
00050 //Implementation of the RPC call
00051 template<typename OP, typename C, typename FLAGS, typename R, typename ...ARGS>
00052 R L4_EXPORT
00053 Rpc_call<OP, C, R (ARGS...), FLAGS>::
00054     call(L4::Cap<C> cap, typename _Elem<ARGS>::arg_type ...a, l4_utcb_t *utcb) noexcept
00055 {
00056     return Rpc_inline_call<OP, C, R (ARGS...), FLAGS>::call(cap, a..., utcb);
00057 }
00058
00059
00060 } // namespace Msg
00061 } // namespace Ipc
00062 } // namespace L4
00063

```

## 16.461 ipc\_epiface

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014-2015 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate

```

```

00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019 #pragma GCC system_header
00020
00021 #include "capability.h"
00022 #include "ipc_server"
00023 #include "ipc_string"
00024 #include <l4/sys/types.h>
00025 #include <l4/sys/utcb.h>
00026 #include <l4/sys/__typeinfo.h>
00027 #include <l4/sys/meta>
00028 #include <l4/cxx/type_traits>
00029
00030 namespace L4 {
00031
00032 // forward for Irqep_t
00033 class Irq;
00034 class Rcv_endpoint;
00035
00036 namespace Ipc_svr {
00037
00038 class Timeout;
00039
00047 class Server_iface
00048 {
00049 private:
00050     Server_iface(Server_iface const &);
00051     Server_iface const &operator = (Server_iface const &);
00052
00053 public:
00055     typedef L4::Type_info::Demand Demand;
00056
00057     Server_iface(Server_iface &&) = delete;
00058     Server_iface &operator = (Server_iface &&) = delete;
00059
00061     Server_iface() {}
00062
00063     // Destroy the server interface
00064     virtual ~Server_iface() = 0;
00065
00075     virtual int alloc_buffer_demand(Demand const &demand) = 0;
00076
00085     virtual L4::Cap<void> get_rcv_cap(int index) const = 0;
00086
00095     virtual int realloc_rcv_cap(int index) = 0;
00096
00104     virtual int add_timeout(Timeout *timeout, l4_kernel_clock_t time) = 0;
00105
00111     virtual int remove_timeout(Timeout *timeout) = 0;
00112
00124     template<typename T>
00125     L4::Cap<T> rcv_cap(int index) const
00126     { return L4::cap_cast<T>(get_rcv_cap(index)); }
00127
00137     L4::Cap<void> rcv_cap(int index) const
00138     { return get_rcv_cap(index); }
00139 };
00140
00141 inline Server_iface::~Server_iface() {}
00142
00143 } // namespace Ipc_svr
00144
00156 struct Epiface
00157 {
00158     Epiface(Epiface const &) = delete;
00159     Epiface &operator = (Epiface const &) = delete;
00160
00162     typedef Ipc_svr::Server_iface Server_iface;
00164     typedef Ipc_svr::Server_iface::Demand Demand;
00165
00166     class Stored_cap : public Cap<void>
00167     {
00168     private:
00169         enum { Managed = 0x10 };
00170
00171     public:
00172         Stored_cap() = default;
00173         Stored_cap(Cap<void> const &c, bool managed = false)
00174             : Cap<void>((c.cap() & L4_CAP_MASK) | (managed ? Managed : 0))
00175         {
00176             static_assert (!(L4_CAP_MASK & Managed), "conflicting bits used...");

```

```

00177     }
00178
00179     bool managed() const { return cap() & Managed; }
00180 };
00181
00183 Epiface() : _data(0) {}
00184
00197 virtual l4_msgtag_t dispatch(l4_msgtag_t tag, unsigned rights,
00198                             l4_utcb_t *utcb) = 0;
00199
00206 virtual Demand get_buffer_demand() const = 0; //{ return Demand(0); }
00207
00209 virtual ~Epiface() = 0;
00210
00217 Stored_cap obj_cap() const { return _cap; }
00218
00224 Server_iface *server_iface() const { return _data; }
00225
00235 int set_server(Server_iface *srv, Cap<void> cap, bool managed = false)
00236 {
00237     if ((srv && cap) || (!srv && !cap))
00238     {
00239         _data = srv;
00240         _cap = Stored_cap(cap, managed);
00241         return 0;
00242     }
00243
00244     return -L4_EINVAL;
00245 }
00246
00250 void set_obj_cap(Cap<void> const &cap) { _cap = cap; }
00251
00252 private:
00253     Server_iface *_data;
00254     Stored_cap _cap;
00255 };
00256
00257 inline Epiface::~Epiface() {}
00258
00266 template<typename RPC_IFACE, typename BASE = Epiface>
00267 struct Epiface_t0 : BASE
00268 {
00270     typedef RPC_IFACE Interface;
00271
00273     typename Type_info::Demand get_buffer_demand() const
00274     { return typename Kobject_typeid<RPC_IFACE>::Demand(); }
00275
00280     Cap<RPC_IFACE> obj_cap() const
00281     { return L4::cap_cast<RPC_IFACE>(BASE::obj_cap()); }
00282 };
00283
00291 template<typename Derived, typename BASE = Epiface,
00292         bool = cxx::is_polymorphic<BASE>::value>
00293 struct Irqep_t : Epiface_t0<void, BASE>
00294 {
00295     l4_msgtag_t dispatch(l4_msgtag_t, unsigned, l4_utcb_t *) final
00296     {
00297         static_cast<Derived*>(this)->handle_irq();
00298         return l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00299     }
00300
00305     Cap<L4::Irq> obj_cap() const
00306     { return L4::cap_cast<L4::Irq>(BASE::obj_cap()); }
00307 };
00308
00309 template<typename Derived, typename BASE>
00310 struct Irqep_t<Derived, BASE, false> : Epiface_t0<void, BASE>
00311 {
00312     l4_msgtag_t dispatch(l4_msgtag_t, unsigned, l4_utcb_t *)
00313     {
00314         static_cast<Derived*>(this)->handle_irq();
00315         return l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00316     }
00317
00322     Cap<L4::Irq> obj_cap() const
00323     { return L4::cap_cast<L4::Irq>(BASE::obj_cap()); }
00324 };
00325
00333 class Registry_iface
00334 {
00335 public:
00336     virtual ~Registry_iface() = 0;
00337
00350     virtual L4::Cap<void>
00351     register_obj(L4::Epiface *o, char const *service) = 0;
00352
00367     virtual L4::Cap<void>

```

```

00368     register_obj(L4::Epiface *o) = 0;
00369
00383     virtual L4::Cap<L4::Irq> register_irq_obj(L4::Epiface *o) = 0;
00384
00397     virtual L4::Cap<L4::Rcv_endpoint>
00398     register_obj(L4::Epiface *o, L4::Cap<L4::Rcv_endpoint> ep) = 0;
00399
00412     virtual void
00413     unregister_obj(L4::Epiface *o, bool unmap = true) = 0;
00414 };
00415
00416 inline Registry_iface::~Registry_iface() {}
00417
00418 namespace Ipc {
00419 namespace Detail {
00420
00421     using namespace L4::Typeid;
00422
00423     template<typename IFACE>
00424     struct Meta_svr
00425     {
00426         long op_num_interfaces(L4::Meta::Rights)
00427         { return 1; }
00428
00429         long op_interface(L4::Meta::Rights, l4_umword_t ifx, long &proto, L4::Ipc::String<char> &name)
00430         {
00431             if (ifx > 0)
00432                 return -L4_ERANGE;
00433             proto = L4::kobject_typeid<IFACE>()->proto();
00434             if (auto *n = L4::kobject_typeid<IFACE>()->name())
00435                 name.copy_in(n);
00436             return 0;
00437         }
00438     }
00439
00440     long op_supports(L4::Meta::Rights, l4_mword_t proto)
00441     { return L4::kobject_typeid<IFACE>()->has_proto(proto); }
00442 };
00443
00444 template<typename IFACE, typename LIST>
00445 struct _Dispatch;
00446
00447 // No match dispatcher found
00448 template<typename IFACE>
00449 struct _Dispatch<IFACE, Iface_list_end>
00450 {
00451     template< typename THIS, typename A1, typename A2 >
00452     static l4_msgtag_t f(THIS *, l4_msgtag_t, A1, A2 &)
00453     { return l4_msgtag(-L4_EBADPROTO, 0, 0, 0); }
00454 };
00455
00456 // call matching p_dispatch() function
00457 template<typename IFACE, typename I, typename LIST >
00458 struct _Dispatch<IFACE, Iface_list<I, LIST> >
00459 {
00460     // special handling for the meta protocol, to avoid 'using' murx
00461     template< typename THIS >
00462     static l4_msgtag_t _f(THIS *, l4_msgtag_t tag, unsigned r,
00463                          l4_utcb_t *utcb, True::type)
00464     {
00465         using L4::Ipc::Msg::dispatch_call;
00466         typedef L4::Meta::Rpcs Meta;
00467         typedef Meta_svr<IFACE> Msvr;
00468         return dispatch_call<Meta>(static_cast<Msvr *>(nullptr), utcb, tag, r);
00469     }
00470
00471     // normal dispatch to the op_<func> methods of \a self.
00472     template< typename THIS >
00473     static l4_msgtag_t _f(THIS *self, l4_msgtag_t t, unsigned r,
00474                          l4_utcb_t *utcb, False::type)
00475     {
00476         using L4::Ipc::Msg::dispatch_call;
00477         return dispatch_call<typename I::iface_type::Rpcs>(self, utcb, t, r);
00478     }
00479
00480     // dispatch function with switch for meta protocol
00481     template< typename THIS >
00482     static l4_msgtag_t f(THIS *self, l4_msgtag_t tag, unsigned r,
00483                         l4_utcb_t *utcb)
00484     {
00485         if (I::Proto == tag.label())
00486             return _f(self, tag, r, utcb,
00487                      Bool<I::Proto == static_cast<long>(L4_PROTO_META)>());
00488         return _Dispatch<IFACE, typename LIST::type>::f(self, tag, r, utcb);
00489     }
00490 };
00491

```

```

00492
00493 template<typename IFACE>
00494 struct Dispatch :
00495     _Dispatch<IFACE, typename L4::Kobject_typeid<IFACE>::Iface_list::type>
00496 {};
00497
00498 } // namespace Detail
00499
00500 template<typename EPIFACE>
00501 struct Dispatch : Detail::Dispatch<typename EPIFACE::Interface>
00502 {};
00503
00504 } // namespace Ipc
00505
00512 template<typename Derived, typename IFACE, typename BASE = L4::Epiface,
00513         bool = cxx::is_polymorphic<BASE>::value>
00514 struct Epiface_t : Epiface_t0<IFACE, BASE>
00515 {
00516     l4_msgtag_t
00517     dispatch(l4_msgtag_t tag, unsigned rights, l4_utcb_t *utcb) final
00518     {
00519         typedef Ipc::Dispatch<Derived> Dispatch;
00520         return Dispatch::f(static_cast<Derived*>(this), tag, rights, utcb);
00521     }
00522 };
00523
00524 template<typename Derived, typename IFACE, typename BASE>
00525 struct Epiface_t<Derived, IFACE, BASE, false> : Epiface_t0<IFACE, BASE>
00526 {
00527     l4_msgtag_t
00528     dispatch(l4_msgtag_t tag, unsigned rights, l4_utcb_t *utcb)
00529     {
00530         typedef Ipc::Dispatch<Derived> Dispatch;
00531         return Dispatch::f(static_cast<Derived*>(this), tag, rights, utcb);
00532     }
00533 };
00534
00540 class Basic_registry
00541 {
00542 public:
00543     typedef Epiface Value;
00549     static Value *find(l4_umword_t label)
00550     { return reinterpret_cast<Value*>(label & ~3UL); }
00551
00565     static l4_msgtag_t dispatch(l4_msgtag_t tag, l4_umword_t label,
00566                                l4_utcb_t *utcb)
00567     {
00568         return find(label)->dispatch(tag, label, utcb);
00569     }
00570 };
00571
00572
00573 } // namespace L4

```

## 16.462 l4/sys/cxx/ipc\_iface File Reference

Interface Definition Language.

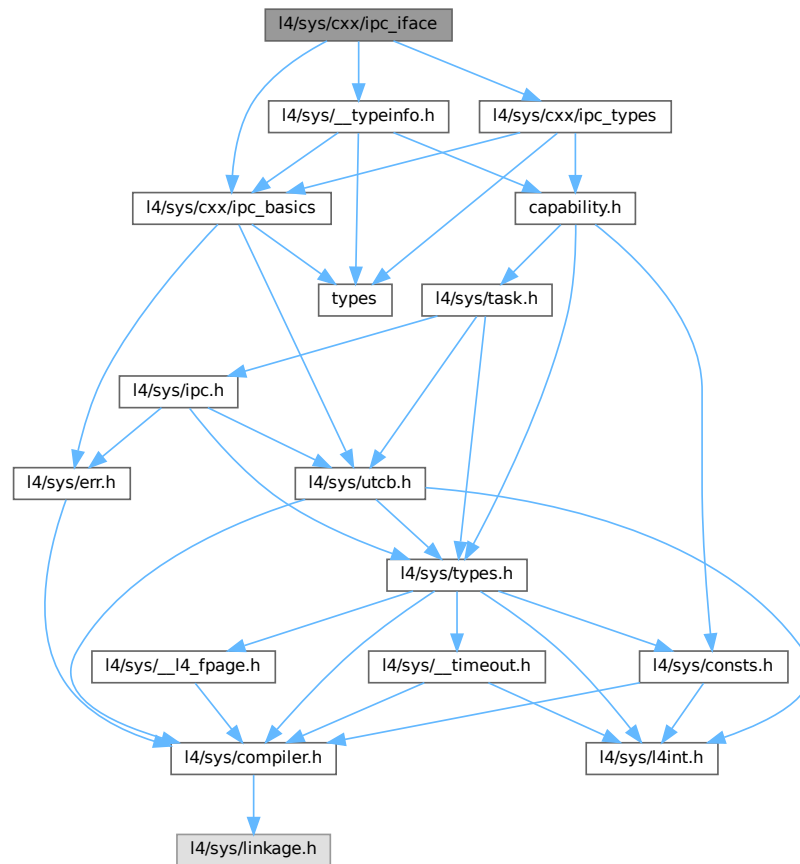
```

#include <l4/sys/cxx/ipc_basics>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/__typeinfo.h>

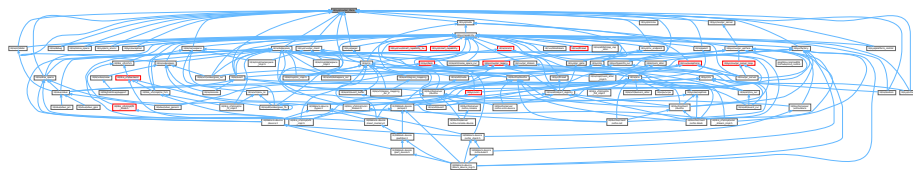
```



Include dependency graph for ipc\_iface:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [L4::lpc::Call](#)  
*RPC attribute for a standard RPC call.*
- struct [L4::lpc::Call\\_zero\\_send\\_timeout](#)  
*RPC attribute for an RPC call, with zero send timeout.*
- struct [L4::lpc::Call\\_t< RIGHTS >](#)  
*RPC attribute for an RPC call with required rights.*
- struct [L4::lpc::Send\\_only](#)  
*RPC attribute for a send-only RPC.*

## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*
- namespace [L4::lpc](#)  
*IPC related functionality.*
- namespace [L4::lpc::Msg](#)  
*IPC Message related functionality.*

## Macros

- `#define L4_INLINE_RPC_NF(res, name, args...)`  
*Define an inline RPC call type (the type only, no callable).*
- `#define L4_INLINE_RPC_NF_OP(op, res, name, args...)`  
*Define an inline RPC call type with specific opcode (the type only, no callable).*
- `#define L4_INLINE_RPC(res, name, args, attr...) res name args`  
*Define an inline RPC call (type and callable).*
- `#define L4_INLINE_RPC_OP(op, res, name, args, attr...) res name args`  
*Define an inline RPC call with specific opcode (type and callable).*
- `#define L4_RPC_NF(res, name, args...)`  
*Define an RPC call type (the type only, no callable).*
- `#define L4_RPC_NF_OP(op, res, name, args...)`  
*Define an RPC call type with specific opcode (the type only, no callable).*
- `#define L4_RPC(res, name, args, attr...) res name args`  
*Define an RPC call (type and callable).*
- `#define L4_RPC_OP(op, res, name, args, attr...) res name args`  
*Define an RPC call with specific opcode (type and callable).*

## 16.462.1 Detailed Description

Interface Definition Language.

See also

[L4\\_RPC](#), [L4\\_INLINE\\_RPC](#), [L4::lpc::Call](#) [L4::lpc::Send\\_only](#), [L4::lpc::Msg::Rpc\\_call](#), [L4::lpc::Msg::Rpc\\_call](#)  
[inline\\_call](#)

Definition in file [ipc\\_iface](#).

## 16.462.2 Macro Definition Documentation

### 16.462.2.1 L4\_INLINE\_RPC

```
#define L4_INLINE_RPC(  
    res,  
    name,  
    args,  
    attr... ) res name args
```

Define an inline RPC call (type and callable).

## Parameters

<i>res</i>	The result type of the RPC call
<i>name</i>	The name of the function ( <i>name_t</i> is used for the type.)
<i>args</i>	The argument list of the RPC function.
<i>attr</i>	Optional RPC attributes ( <a href="#">L4::ipc::Call</a> , <a href="#">L4::ipc::Call_t</a> etc.).

Definition at line 469 of file [ipc\\_iface](#).

## 16.462.2.2 L4\_INLINE\_RPC\_NF

```
#define L4_INLINE_RPC_NF(
    res,
    name,
    args... )
```

## Value:

```
struct name##_t : L4::Ip::Msg::Rpc_inline_call<name##_t, Class, res args> \
{ \
    typedef L4::Ip::Msg::Rpc_inline_call<name##_t, Class, res args> type; \
    L4_INLINE_RPC_SRV_FORWARD(name); \
}
```

Define an inline RPC call type (the type only, no callable).

## Parameters

<i>res</i>	The result type of the RPC call
<i>name</i>	The name of the function ( <i>name_t</i> is used for the type.)
<i>args</i>	The argument list of the RPC function, and RPC attributes ( <a href="#">L4::ipc::Call</a> , <a href="#">L4::ipc::Call_t</a> etc.).

Stubs generated by this macro can be used explicitly in custom wrapper methods that need to use the underlying RPC code and provide some higher level abstraction, for example with default arguments or extra argument conversion.

Definition at line 440 of file [ipc\\_iface](#).

## 16.462.2.3 L4\_INLINE\_RPC\_NF\_OP

```
#define L4_INLINE_RPC_NF_OP(
    op,
    res,
    name,
    args... )
```

## Value:

```
struct name##_t : L4::Ip::Msg::Rpc_inline_call<name##_t, Class, res args> \
{ \
    typedef L4::Ip::Msg::Rpc_inline_call<name##_t, Class, res args> type; \
    enum { Opcode = (op) }; \
    L4_INLINE_RPC_SRV_FORWARD(name); \
}
```

Define an inline RPC call type with specific opcode (the type only, no callable).

## Parameters

<i>op</i>	The opcode number for this function
<i>res</i>	The result type of the RPC call
<i>name</i>	The name of the function ( <i>name_t</i> is used for the type.)
<i>args</i>	The argument list of the RPC function, and RPC attributes ( <a href="#">L4::lpc::Call</a> , <a href="#">L4::lpc::Call_t</a> etc.).

Stubs generated by this macro can be used explicitly in custom wrapper methods that need to use the underlying RPC code and provide some higher level abstraction, for example with default arguments or extra argument conversion.

Definition at line 453 of file [ipc\\_iface](#).

**16.462.2.4 L4\_INLINE\_RPC\_OP**

```
#define L4_INLINE_RPC_OP(
    op,
    res,
    name,
    args,
    attr... ) res name args
```

Define an inline RPC call with specific opcode (type and callable).

## Parameters

<i>op</i>	The opcode number for this function
<i>res</i>	The result type of the RPC call
<i>name</i>	The name of the function ( <i>name_t</i> is used for the type.)
<i>args</i>	The argument list of the RPC function.
<i>attr</i>	Optional RPC attributes ( <a href="#">L4::lpc::Call</a> , <a href="#">L4::lpc::Call_t</a> etc.).

Definition at line 484 of file [ipc\\_iface](#).

**16.462.2.5 L4\_RPC**

```
#define L4_RPC(
    res,
    name,
    args,
    attr... ) res name args
```

Define an RPC call (type and callable).

## Parameters

<i>res</i>	The result type of the RPC call
<i>name</i>	The name of the function ( <i>name_t</i> is used for the type.)
<i>args</i>	The argument list of the RPC function.
<i>attr</i>	Optional RPC attributes ( <a href="#">L4::lpc::Call</a> , <a href="#">L4::lpc::Call_t</a> etc.).

Definition at line 528 of file [ipc\\_iface](#).

### 16.462.2.6 L4\_RPC\_NF

```
#define L4_RPC_NF(
    res,
    name,
    args... )
```

#### Value:

```
struct name##_t : L4::Ip::Msg::Rpc_call<name##_t, Class, res args>
{
    typedef L4::Ip::Msg::Rpc_call<name##_t, Class, res args> type;
    L4_INLINE_RPC_SRV_FORWARD(name);
}
```

Define an RPC call type (the type only, no callable).

#### Parameters

<i>res</i>	The result type of the RPC call
<i>name</i>	The name of the function ( <i>name_t</i> is used for the type.)
<i>args</i>	The argument list of the RPC function, and RPC attributes ( <a href="#">L4::ipc::Call</a> , <a href="#">L4::ipc::Call_t</a> etc.).

Definition at line 497 of file [ipc\\_iface](#).

### 16.462.2.7 L4\_RPC\_NF\_OP

```
#define L4_RPC_NF_OP(
    op,
    res,
    name,
    args... )
```

#### Value:

```
struct name##_t : L4::Ip::Msg::Rpc_call<name##_t, Class, res args>
{
    typedef L4::Ip::Msg::Rpc_call<name##_t, Class, res args> type;
    enum { Opcode = (op) };
    L4_INLINE_RPC_SRV_FORWARD(name);
}
```

Define an RPC call type with specific opcode (the type only, no callable).

#### Parameters

<i>op</i>	The opcode number for this function
<i>res</i>	The result type of the RPC call
<i>name</i>	The name of the function ( <i>name_t</i> is used for the type.)
<i>args</i>	The argument list of the RPC function, and RPC attributes ( <a href="#">L4::ipc::Call</a> , <a href="#">L4::ipc::Call_t</a> etc.).

Definition at line 512 of file [ipc\\_iface](#).

### 16.462.2.8 L4\_RPC\_OP

```
#define L4_RPC_OP (
    op,
    res,
    name,
    args,
    attr... ) res name args
```

Define an RPC call with specific opcode (type and callable).

#### Parameters

<i>op</i>	The opcode number for this function
<i>res</i>	The result type of the RPC call
<i>name</i>	The name of the function ( <i>name_t</i> is used for the type.)
<i>args</i>	The argument list of the RPC function.
<i>attr</i>	Optional RPC attributes ( <a href="#">L4::Ipc::Call</a> , <a href="#">L4::Ipc::Call_t</a> etc.).

Definition at line 543 of file [ipc\\_iface](#).

## 16.463 ipc\_iface

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019 #pragma GCC system_header
00020
00021 #include <l4/sys/cxx/ipc_basics>
00022 #include <l4/sys/cxx/ipc_types>
00023 #include <l4/sys/__typeinfo.h>
00024
00207 // TODO: add some more documentation
00208 namespace L4 { namespace Ipc {
00209
00226 struct L4_EXPORT Call
00227 {
00228     enum { Is_call = true };
00229     enum { Rights = 0 };
00230     static l4_timeout_t timeout() { return L4_IPC_NEVER; }
00231 };
00232
00236 struct L4_EXPORT Call_zero_send_timeout : Call
00237 {
00238     static l4_timeout_t timeout() { return L4_IPC_SEND_TIMEOUT_0; }
00239 };
00240
00256 template<unsigned RIGHTS>
00257 struct L4_EXPORT Call_t : Call
00258 {
00259     enum { Rights = RIGHTS };
00260 }
```

```

00260 };
00261
00274 struct L4_EXPORT Send_only
00275 {
00276     enum { Is_call = false };
00277     enum { Rights = 0 };
00278     static l4_timeout_t timeout() { return L4_IPC_NEVER; }
00279 };
00280
00281 namespace Msg {
00282
00293 template<typename OP, typename CLASS, typename SIG, typename FLAGS = Call>
00294 struct L4_EXPORT Rpc_inline_call;
00295
00300 template<typename OP, typename CLASS, typename FLAGS, typename R,
00301         typename ...ARGS>
00302 struct L4_EXPORT Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>
00303 {
00304     template<typename T> struct Result { typedef T result_type; };
00305     enum
00306     {
00307         Return_tag = L4::Types::Same<R, l4_msgtag_t>::value
00308     };
00309
00311     typedef Rpc_inline_call type;
00313     typedef OP op_type;
00315     typedef CLASS class_type;
00317     typedef typename Result<R>::result_type result_type;
00319     typedef R ipc_type (ARGS...);
00321     typedef result_type func_type (typename _Elem<ARGS>::arg_type...);
00322
00324     typedef FLAGS flags_type;
00325
00326     template<typename RES>
00327     static typename L4::Types::Enable_if< Return_tag, RES >::type
00328     return_err(long err) noexcept { return l4_msgtag(err, 0, 0, 0); }
00329
00330     template<typename RES>
00331     static typename L4::Types::Enable_if< Return_tag, RES >::type
00332     return_ipc_err(l4_msgtag_t tag, l4_utcb_t const *) noexcept { return tag; }
00333
00334     template<typename RES>
00335     static typename L4::Types::Enable_if< Return_tag, RES >::type
00336     return_code(l4_msgtag_t tag) noexcept { return tag; }
00337
00338     template<typename RES>
00339     static typename L4::Types::Enable_if< !Return_tag, RES >::type
00340     return_err(long err) noexcept { return err; }
00341
00342     template<typename RES>
00343     static typename L4::Types::Enable_if< !Return_tag, RES >::type
00344     return_ipc_err(l4_msgtag_t, l4_utcb_t *utcb) noexcept
00345     { return l4_ipc_to_errno(l4_ipc_error_code(utcb)); }
00346
00347     template<typename RES>
00348     static typename L4::Types::Enable_if< !Return_tag, RES >::type
00349     return_code(l4_msgtag_t tag) noexcept { return tag.label(); }
00350
00351     static R call(L4::Cap<class_type> cap,
00352                  typename _Elem<ARGS>::arg_type ...a,
00353                  l4_utcb_t *utcb = l4_utcb()) noexcept;
00354 };
00355
00360 template<typename OP, typename CLASS, typename SIG, typename FLAGS = Call>
00361 struct L4_EXPORT Rpc_call;
00362
00370 template<typename IPC, typename SIG> struct _Call;
00371
00373 template<typename IPC, typename R, typename ...ARGS>
00374 struct _Call<IPC, R (ARGS...)>
00375 {
00376 public:
00377     typedef typename IPC::class_type class_type;
00378     typedef typename IPC::result_type result_type;
00379
00380 private:
00381     L4::Cap<class_type> cap() const noexcept
00382     {
00383         return L4::Cap<class_type>(reinterpret_cast<l4_cap_idx_t>(this)
00384                                     & L4_CAP_MASK);
00385     }
00386
00387 public:
00389     result_type operator () (ARGS ...a, l4_utcb_t *utcb = l4_utcb()) const noexcept
00390     { return IPC::call(cap(), a..., utcb); }
00391 };
00392

```

```

00399 template<typename IPC> struct Call : _Call<IPC, typename IPC::func_type> {};
00400
00405 template<typename OP,
00406         typename CLASS,
00407         typename FLAGS,
00408         typename R,
00409         typename ...ARGS>
00410 struct L4_EXPORT Rpc_call<OP, CLASS, R (ARGS...), FLAGS> :
00411     Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>
00412 {
00413     static R call(L4::Cap<CLASS> cap,
00414                 typename _Elem<ARGS>::arg_type ...a,
00415                 l4_utcb_t *utcb = l4_utcb()) noexcept;
00416 };
00417
00418 #define L4_INLINE_RPC_SRV_FORWARD(name)
00419     template<typename OBJ> struct fwd
00420     {
00421         OBJ *o;
00422         fwd(OBJ *o) noexcept : o(o) {}
00423         template<typename ...ARGS> long call(ARGS ...a) noexcept(noexcept(o->op_##name(a...))) \
00424         { return o->op_##name(a...); }
00425     }
00426
00427
00440 #define L4_INLINE_RPC_NF(res, name, args...)
00441     struct name##_t : L4::IpC::Msg::Rpc_inline_call<name##_t, Class, res args>
00442     {
00443         typedef L4::IpC::Msg::Rpc_inline_call<name##_t, Class, res args> type;
00444         L4_INLINE_RPC_SRV_FORWARD(name);
00445     }
00446
00453 #define L4_INLINE_RPC_NF_OP(op, res, name, args...)
00454     struct name##_t : L4::IpC::Msg::Rpc_inline_call<name##_t, Class, res args>
00455     {
00456         typedef L4::IpC::Msg::Rpc_inline_call<name##_t, Class, res args> type;
00457         enum { Opcode = (op) };
00458         L4_INLINE_RPC_SRV_FORWARD(name);
00459     }
00460
00461 #ifndef DOXYGEN
00469 #define L4_INLINE_RPC(res, name, args, attr...) res name args
00470 #else
00471 #define L4_INLINE_RPC(res, name, args...)
00472     L4_INLINE_RPC_NF(res, name, args); L4::IpC::Msg::Call<name##_t> name
00473 #endif
00474
00475 #ifndef DOXYGEN
00484 #define L4_INLINE_RPC_OP(op, res, name, args, attr...) res name args
00485 #else
00486 #define L4_INLINE_RPC_OP(op, res, name, args...)
00487     L4_INLINE_RPC_NF_OP(op, res, name, args); L4::IpC::Msg::Call<name##_t> name
00488 #endif
00489
00497 #define L4_RPC_NF(res, name, args...)
00498     struct name##_t : L4::IpC::Msg::Rpc_call<name##_t, Class, res args>
00499     {
00500         typedef L4::IpC::Msg::Rpc_call<name##_t, Class, res args> type;
00501         L4_INLINE_RPC_SRV_FORWARD(name);
00502     }
00503
00512 #define L4_RPC_NF_OP(op, res, name, args...)
00513     struct name##_t : L4::IpC::Msg::Rpc_call<name##_t, Class, res args>
00514     {
00515         typedef L4::IpC::Msg::Rpc_call<name##_t, Class, res args> type;
00516         enum { Opcode = (op) };
00517         L4_INLINE_RPC_SRV_FORWARD(name);
00518     }
00519
00520 #ifndef DOXYGEN
00528 #define L4_RPC(res, name, args, attr...) res name args
00529 #else
00530 #define L4_RPC(res, name, args...)
00531     L4_RPC_NF(res, name, args); L4::IpC::Msg::Call<name##_t> name
00532 #endif
00533
00534 #ifndef DOXYGEN
00543 #define L4_RPC_OP(op, res, name, args, attr...) res name args
00544 #else
00545 #define L4_RPC_OP(op, res, name, args...)
00546     L4_RPC_NF_OP(op, res, name, args); L4::IpC::Msg::Call<name##_t> name
00547 #endif
00548
00549
00554 namespace Detail {
00555
00559 template<typename ...ARGS>

```



```

00560 struct Buf
00561 {
00562 public:
00563     template<typename DIR>
00564     static constexpr int write(char *, int offset, int) noexcept
00565     { return offset; }
00566
00567     template<typename DIR>
00568     static constexpr int read(char *, int offset, int, long) noexcept
00569     { return offset; }
00570
00571     typedef void Base;
00572 };
00573
00574 template<typename A, typename ...M>
00575 struct Buf<A, M...> : Buf<M...>
00576 {
00577     typedef Buf<M...> Base;
00578
00579     typedef Clnt_xmit<A> xmit;
00580     typedef typename _Elem<A>::arg_type arg_type;
00581     typedef Detail::_Plain<arg_type> plain;
00582
00583     template<typename DIR>
00584     static int
00585     write(char *base, int offset, int limit,
00586           arg_type a, typename _Elem<M>::arg_type ...m) noexcept
00587     {
00588         offset = xmit::to_msg(base, offset, limit, plain::deref(a),
00589                               typename DIR::dir(), typename DIR::cls());
00590         return Base::template write<DIR>(base, offset, limit, m...);
00591     }
00592
00593     template<typename DIR>
00594     static int
00595     read(char *base, int offset, int limit, long ret,
00596          arg_type a, typename _Elem<M>::arg_type ...m) noexcept
00597     {
00598         int r = xmit::from_msg(base, offset, limit, ret, plain::deref(a),
00599                                typename DIR::dir(), typename DIR::cls());
00600         if (L4_LIKELY(r >= 0))
00601             return Base::template read<DIR>(base, r, limit, ret, m...);
00602
00603         if (_Elem<A>::Is_optional)
00604             return Base::template read<DIR>(base, offset, limit, ret, m...);
00605
00606         return r;
00607     }
00608 };
00609
00610 template <typename ...ARGS> struct _Part
00611 {
00612     typedef Buf<ARGS...> Data;
00613
00614     template<typename DIR>
00615     static int write(void *b, int offset, int limit,
00616                     typename _Elem<ARGS>::arg_type ...m) noexcept
00617     {
00618         char *buf = static_cast<char *>(b);
00619         int r = Data::template write<DIR>(buf, offset, limit, m...);
00620         if (L4_LIKELY(r >= offset))
00621             return r - offset;
00622         return r;
00623     }
00624
00625     template<typename DIR>
00626     static int read(void *b, int offset, int limit, long ret,
00627                     typename _Elem<ARGS>::arg_type ...m) noexcept
00628     {
00629         char *buf = static_cast<char *>(b);
00630         int r = Data::template read<DIR>(buf, offset, limit, ret, m...);
00631         if (L4_LIKELY(r >= offset))
00632             return r - offset;
00633         return r;
00634     }
00635 }
00636 };
00637
00644 template<typename IPC_TYPE, typename OPCODE = void>
00645 struct Part;
00646
00647 // The version without an op-code
00648 template<typename R, typename ...ARGS>
00649 struct Part<R (ARGS...), void> : _Part<ARGS...>
00650 {
00651     typedef Buf<ARGS...> Data;
00652
00653     // write arguments, skipping the dummy opcode

```

```

00655     template<typename DIR>
00656     static int write_op(void *b, int offset, int limit,
00657         int /*placeholder for op*/,
00658         typename _Elem<ARGS>::arg_type ...m) noexcept
00659     {
00660         char *buf = static_cast<char *>(b);
00661         int r = Data::template write<DIR>(buf, offset, limit, m...);
00662         if (L4_LIKELY(r >= offset))
00663             return r - offset;
00664         return r;
00665     }
00666 };
00667
00668 // Message part with additional opcode
00669 template<typename OPCODE, typename R, typename ...ARGS>
00670 struct Part<R (ARGS...), OPCODE> : _Part<ARGS...>
00671 {
00672     typedef OPCODE opcode_type;
00673     typedef Buf<opcode_type, ARGS...> Data;
00674
00675     // write arguments, including the opcode
00676     template<typename DIR>
00677     static int write_op(void *b, int offset, int limit,
00678         opcode_type op, typename _Elem<ARGS>::arg_type ...m) noexcept
00679     {
00680         char *buf = static_cast<char *>(b);
00681         int r = Data::template write<DIR>(buf, offset, limit, op, m...);
00682         if (L4_LIKELY(r >= offset))
00683             return r - offset;
00684         return r;
00685     }
00686 };
00687 };
00688
00689 } // namespace Detail
00690
00691 //-----
00692 // Implementation of the RPC call
00693 // TODO: Add support for timeout via special RPC argument
00694 // TODO: Add support for passing the UTCB pointer as argument
00695 //
00696 template<typename OP, typename CLASS, typename FLAGS, typename R,
00697     typename ...ARGS>
00698 inline R
00699 Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>::
00700     call(L4::Cap<CLASS> cap,
00701         typename _Elem<ARGS>::arg_type ...a,
00702         l4_utcb_t *utcb) noexcept
00703 {
00704     using namespace Ipc::Msg;
00705
00706     typedef typename Kobject_typeid<CLASS>::Iface::Rpcs Rpcs;
00707     typedef typename Rpcs::template Rpc<OP> Opt;
00708     typedef Detail::Part<ipc_type, typename Rpcs::opcode_type> Args;
00709
00710     l4_msg_regs_t *mrs = l4_utcb_mr_u(utcb);
00711
00712     // handle in-data part of the arguments
00713     int send_bytes =
00714         Args::template write_op<Do_in_data>(mrs->mr, 0, Mr_bytes,
00715             Opt::Opcode, a...);
00716
00717     if (L4_UNLIKELY(send_bytes < 0))
00718         return return_err<R>(send_bytes);
00719
00720     send_bytes = align_to<l4_umword_t>(send_bytes);
00721     int const send_words = send_bytes / Word_bytes;
00722     // write the in-items part of the message if there is one
00723     int item_bytes =
00724         Args::template write<Do_in_items>(&mrs->mr[send_words], 0,
00725             Mr_bytes - send_bytes, a...);
00726
00727     if (L4_UNLIKELY(item_bytes < 0))
00728         return return_err<R>(item_bytes);
00729
00730     int send_items = item_bytes / Item_bytes;
00731
00732     {
00733         // setup the receive buffers for the RPC call
00734         l4_buf_regs_t *brs = l4_utcb_br_u(utcb);
00735         // XXX: we currently support only one type of receive buffers per call
00736         brs->bdr = 0; // we always start at br[0]
00737
00738         // the limit leaves us at least one register for the zero terminator
00739         // add the buffers given as arguments to the buffer registers
00740         int bytes =
00741             Args::template write<Do_rcv_buffers>(brs->br, 0, Br_bytes - Word_bytes,

```

```

00743                                     a...);
00744
00745         if (L4_UNLIKELY(bytes < 0))
00746             return return_err<R>(bytes);
00747
00748         brs->br[bytes / Word_bytes] = 0;
00749     }
00750
00751
00752     // here we do the actual IPC -----
00753     l4_msgtag_t t;
00754     t = l4_msgtag(CLASS::Protocol, send_words, send_items, 0);
00755     // do the call (Q: do we need support for timeouts?)
00756     if (flags_type::Is_call)
00757         t = l4_ipc_call(cap.cap(), utcb, t, flags_type::timeout());
00758     else
00759     {
00760         t = l4_ipc_send(cap.cap(), utcb, t, flags_type::timeout());
00761         if (L4_UNLIKELY(t.has_error()))
00762             return return_ipc_err<R>(t, utcb);
00763
00764         return return_code<R>(l4_msgtag(0, 0, 0, t.flags()));
00765     }
00766
00767     // unmarshalling starts here -----
00768
00769     // bail out early in the case of an IPC error
00770     if (L4_UNLIKELY(t.has_error()))
00771         return return_ipc_err<R>(t, utcb);
00772
00773     // take the label as return value
00774     long r = t.label();
00775
00776     // bail out on negative error codes too
00777     if (L4_UNLIKELY(r < 0))
00778         return return_err<R>(r);
00779
00780     int const rcv_bytes = t.words() * Word_bytes;
00781
00782     // read the static out-data values to the arguments
00783     int err = Args::template read<Do_out_data>(mrs->mr, 0, rcv_bytes, r, a...);
00784
00785     int const item_limit = t.items() * Item_bytes;
00786
00787     if (L4_UNLIKELY(err < 0 || item_limit > Mr_bytes))
00788         return return_err<R>(-L4_MSGTOOSHORT);
00789
00790     // read the static out-items to the arguments
00791     err = Args::template read<Do_out_items>(&mrs->mr[t.words()], 0, item_limit,
00792                                           r, a...);
00793
00794     if (L4_UNLIKELY(err < 0))
00795         return return_err<R>(-L4_MSGTOOSHORT);
00796
00797     return return_code<R>(t);
00798 }
00799
00800 } // namespace Msg
00801 } // namespace Ipc
00802 } // namespace L4

```

## 16.464 ipc\_legacy

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2015, 2017 Kernkonzept GmbH.
00004  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00005  *
00006  * This file is distributed under the terms of the GNU General Public
00007  * License, version 2. Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019
00020 #include <l4/sys/cxx/ipc_epiface>

```

```

00021 #ifndef L4_RPC_DISABLE_LEGACY_DISPATCH
00022 #define L4_RPC_LEGACY_DISPATCH(IFACE)
00023     template<typename IOS>
00024     int dispatch(unsigned rights, IOS &ios)
00025     {
00026         typedef ::L4::Ipc::Detail::Dispatch<IFACE> Dispatch;
00027         l4_msgtag_t r = Dispatch::f(this, ios.tag(), rights, ios.utcb());
00028         ios.set_ipc_params(r);
00029         return r.label();
00030     }
00031
00032     template<typename IOS>
00033     int p_dispatch(IFACE *, unsigned rights, IOS &ios)
00034     {
00035         using ::L4::Ipc::Msg::dispatch_call;
00036         l4_msgtag_t r;
00037         r = dispatch_call<typename IFACE::Rpc>(this, ios.utcb(),
00038                                                ios.tag(), rights);
00039         ios.set_ipc_params(r);
00040         return r.label();
00041     }
00042
00043 #define L4_RPC_LEGACY_USING(IFACE) \
00044     using IFACE::p_dispatch
00045
00046 #else
00047 #define L4_RPC_LEGACY_DISPATCH(IFACE)
00048 #define L4_RPC_LEGACY_USING(IFACE)
00049 #endif

```

## 16.465 ipc\_ret\_array

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019
00020 #include "types"
00021 #include "ipc_basics"
00022
00023 namespace L4 { namespace Ipc L4_EXPORT {
00024
00025     // -----
00034     template<typename T> struct L4_EXPORT Ret_array
00035     {
00036         typedef T const **ptr_type;
00037
00038         T *value = nullptr;
00039         unsigned max = 0;
00040         Ret_array() {}
00041         Ret_array(T *v, unsigned max) : value(v), max(max) {}
00042     };
00043
00044     namespace Msg {
00045
00046         template<typename A> struct Elem< Ret_array<A> >
00047         {
00048             enum { Is_optional = false };
00049             typedef Ret_array<A> type;
00050             typedef typename type::ptr_type arg_type;
00051             typedef type svr_type;
00052             typedef type svr_arg_type;
00053         };
00054
00055         template<typename A>
00056         struct Is_valid_rpc_type<Ret_array<A> *> : L4::Types::False {};
00057         template<typename A>
00058         struct Is_valid_rpc_type<Ret_array<A> &> : L4::Types::False {};
00059         template<typename A>

```

```

00060 struct Is_valid_rpc_type<Ret_array<A> const &> : L4::Types::False {};
00061 template<typename A>
00062 struct Is_valid_rpc_type<Ret_array<A> const *> : L4::Types::False {};
00063
00064 template<typename A> struct Class< Ret_array<A> > : Class<A>::type {};
00065 template<typename A> struct Direction< Ret_array<A> > : Dir_out {};
00066
00067 template<typename A, typename CLASS>
00068 struct Clnt_val_ops<A const *, Dir_out, CLASS> : Clnt_noops<A const *>
00069 {
00070     using Clnt_noops<A const *>::from_msg;
00071     static int from_msg(char *msg, unsigned offset, unsigned limit, long ret,
00072         A const *&arg, Dir_out, Cls_data)
00073     {
00074         offset = align_to<A>(offset);
00075         arg = reinterpret_cast<A const *>(msg + offset);
00076         if (L4_UNLIKELY(!check_size<A>(offset, limit, ret)))
00077             return -1;
00078         return offset + ret * sizeof(A);
00079     }
00080 };
00081 };
00082
00083 template<typename A, typename CLASS>
00084 struct Svr_val_ops<Ret_array<A>, Dir_out, CLASS> :
00085     Svr_noops<Ret_array<A> >
00086 {
00087     typedef Ret_array<A> ret_array;
00088     using Svr_noops<ret_array>::from_svr;
00089     static int from_svr(char *, unsigned offset, unsigned limit, long ret,
00090         ret_array const &, Dir_out, CLASS)
00091     {
00092         offset = align_to<A>(offset);
00093         if (L4_UNLIKELY(!check_size<A>(offset, limit, ret)))
00094             return -1;
00095         offset += sizeof(A) * ret;
00096         return offset;
00097     }
00098
00099     using Svr_noops<ret_array>::to_svr;
00100     static int to_svr(char *msg, unsigned offset, unsigned limit,
00101         ret_array &arg, Dir_out, CLASS)
00102     {
00103         // there can be actually no limit check here, as this
00104         // is variably sized output array
00105         // FIXME: we could somehow makesure that this is the last
00106         // output value...
00107         offset = align_to<A>(offset);
00108         arg = ret_array(reinterpret_cast<A*>(msg + offset),
00109             (limit - offset) / sizeof(A));
00110         // FIXME: we dont know the length of the array here so, cheat
00111         return offset;
00112     }
00113 };
00114 } // namespace Msg
00115
00116 }

```

## 16.466 ipc\_server\_loop

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2015, 2017, 2019, 2021-2022 Kernkonzept GmbH.
00004  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00005  *
00006  * This file is distributed under the terms of the GNU General Public
00007  * License, version 2. Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019
00020 #include "ipc_epiface"
00021
00022 namespace L4 {
00023

```

```

00035 namespace Ipc_svr {
00036
00050 enum Reply_mode
00051 {
00052     Reply_compound,
00053     Reply_separate
00054 };
00055
00061 struct Ignore_errors
00062 { static void error(l4_msgtag_t, l4_utcb_t *) {} };
00063
00069 struct Default_timeout
00070 { static l4_timeout_t timeout() { return L4_IPC_SEND_TIMEOUT_0; } };
00071
00077 struct Compound_reply
00078 {
00079     static Reply_mode before_reply(l4_msgtag_t, l4_utcb_t *)
00080     { return Reply_compound; }
00081 };
00082
00088 struct Default_setup_wait
00089 { static void setup_wait(l4_utcb_t *, Reply_mode) {} };
00090
00098 template< typename R >
00099 struct Direct_dispatch
00100 {
00102     R &r;
00103
00105     Direct_dispatch(R &r) : r(r) {}
00106
00108     l4_msgtag_t operator () (l4_msgtag_t tag, l4_umword_t obj, l4_utcb_t *utcb)
00109     { return r.dispatch(tag, obj, utcb); }
00110 };
00111
00119 template< typename R >
00120 struct Direct_dispatch<R*>
00121 {
00123     R *r;
00124
00126     Direct_dispatch(R *r) : r(r) {}
00127
00129     l4_msgtag_t operator () (l4_msgtag_t tag, l4_umword_t obj, l4_utcb_t *utcb)
00130     { return r->dispatch(tag, obj, utcb); }
00131 };
00132
00133 #ifdef __EXCEPTIONS
00143 template< typename R, typename Exc> // = L4::Runtime_error>
00144 struct Exc_dispatch : private Direct_dispatch<R>
00145 {
00147     Exc_dispatch(R r) : Direct_dispatch<R>(r) {}
00148
00152     l4_msgtag_t operator () (l4_msgtag_t tag, l4_umword_t obj, l4_utcb_t *utcb)
00153     {
00154         try
00155         {
00156             return Direct_dispatch<R>::operator () (tag, obj, utcb);
00157         }
00158         catch (Exc &e)
00159         {
00160             return l4_msgtag(e.err_no(), 0, 0, 0);
00161         }
00162         catch (int err)
00163         {
00164             return l4_msgtag(err, 0, 0, 0);
00165         }
00166         catch (long err)
00167         {
00168             return l4_msgtag(err, 0, 0, 0);
00169         }
00170     }
00171 };
00172 #endif
00173
00184 class Br_manager_no_buffers : public Server_iface
00185 {
00186 public:
00191     int alloc_buffer_demand(Demand const &demand) override
00192     {
00193         if (!demand.no_demand())
00194             return -L4_ENOMEM;
00195         return L4_EOK;
00196     }
00197
00199     L4::Cap<void> get_rcv_cap(int) const override
00200     { return L4::Cap<void>::Invalid; }
00201
00203     int realloc_rcv_cap(int) override

```

```

00204 { return -L4_ENOMEM; }
00205
00207 int add_timeout(Timeout *, l4_kernel_clock_t) override
00208 { return -L4_ENOSYS; }
00209
00211 int remove_timeout(Timeout *) override
00212 { return -L4_ENOSYS; }
00213
00214 protected:
00216 unsigned first_free_br() const
00217 { return 1; }
00218
00220 void setup_wait(l4_utcb_t *utcb, L4::Ipc_svr::Reply_mode)
00221 {
00222     l4_buf_regs_t *br = l4_utcb_br_u(utcb);
00223     br->bdr = 0;
00224     br->br[0] = 0;
00225 }
00226 };
00227
00236 struct Default_loop_hooks :
00237     public Ignore_errors, public Default_timeout, public Compound_reply,
00238     Br_manager_no_buffers
00239 {};
00240
00241 }
00242
00257 template< typename LOOP_HOOKS = Ipc_svr::Default_loop_hooks >
00258 class Server :
00259     public LOOP_HOOKS
00260 {
00261 public:
00268     /* Internal note: After all users have been converted, remove this
00269      * constructor. Also remove the constructor below then. */
00270     explicit Server(l4_utcb_t *)
00271         L4_DEPRECATED("Do not specify the UTCB with the constructor. "
00272             "Supply it on the loop function if needed.")
00273     {}
00274
00278     /* Internal note: Remove this constructor when the above deprecated
00279      * constructor with the UTCB pointer is also removed. */
00280     Server() {}
00281
00288     template< typename DISPATCH >
00289     inline L4_NORETURN void internal_loop(DISPATCH dispatch, l4_utcb_t *);
00290
00294     template< typename R >
00295     inline L4_NORETURN void loop_noexc(R r, l4_utcb_t *u = l4_utcb())
00296     { internal_loop(Ipc_svr::Direct_dispatch<R>(r), u); }
00297
00298 #ifdef __EXCEPTIONS
00305     template< typename EXC, typename R >
00306     inline L4_NORETURN void loop(R r, l4_utcb_t *u = l4_utcb())
00307     {
00308         internal_loop(Ipc_svr::Exc_dispatch<R, EXC>(r), u);
00309         // function will never return
00310     }
00311 #endif
00312 protected:
00314     inline l4_msgtag_t reply_n_wait(l4_msgtag_t reply, l4_umword_t *p, l4_utcb_t *);
00315 };
00316
00317 template< typename L >
00318 inline l4_msgtag_t
00319 Server<L>::reply_n_wait(l4_msgtag_t reply, l4_umword_t *p, l4_utcb_t *utcb)
00320 {
00321     if (reply.label() != -L4_ENOREPLY)
00322     {
00323         Ipc_svr::Reply_mode m = this->before_reply(reply, utcb);
00324         if (m == Ipc_svr::Reply_compound)
00325         {
00326             this->setup_wait(utcb, m);
00327             return l4_ipc_reply_and_wait(utcb, reply, p, this->timeout());
00328         }
00329         else
00330         {
00331             l4_msgtag_t res = l4_ipc_send(L4_INVALID_CAP | L4_SYSF_REPLY, utcb, reply, this->timeout());
00332             if (res.has_error())
00333                 return res;
00334         }
00335     }
00336     this->setup_wait(utcb, Ipc_svr::Reply_separate);
00337     return l4_ipc_wait(utcb, p, this->timeout());
00338 }
00339
00340 template< typename L >
00341 template< typename DISPATCH >

```

```

00342 inline L4_NORETURN void
00343 Server<L>::internal_loop(DISPATCH dispatch, l4_utcb_t *utcb)
00344 {
00345     l4_msgtag_t res;
00346     l4_umword_t p;
00347     l4_msgtag_t r = l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00348
00349     while (true)
00350     {
00351         res = reply_n_wait(r, &p, utcb);
00352         if (res.has_error())
00353         {
00354             this->error(res, utcb);
00355             r = l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00356             continue;
00357         }
00358
00359         r = dispatch(res, p, utcb);
00360     }
00361 }
00362
00363 } // namespace L4

```

## 16.467 ipc\_string

```

00001 // vi:set ft=c++: -- Mode: C++ --
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYINGING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019
00020 #include "types"
00021 #include "ipc_basics"
00022 #include "ipc_array"
00023
00024 namespace L4 { namespace Ipc {
00025
00026     template<typename CHAR = char const, typename LEN = unsigned long>
00027     struct String : Array<CHAR, LEN>
00028     {
00029         static LEN strlenlength(CHAR *d) { LEN l = 0; while (d[l]) ++l; return l; }
00030         String() {}
00031         String(CHAR *d) : Array<CHAR, LEN>(strlenlength(d) + 1, d) {}
00032         String(LEN len, CHAR *d) : Array<CHAR, LEN>(len, d) {}
00033         void copy_in(CHAR const *s)
00034         {
00035             if (this->length < 1)
00036                 return;
00037
00038             LEN i;
00039             for (i = 0; i < this->length - 1 && s[i]; ++i)
00040                 this->data[i] = s[i];
00041             this->length = i + 1;
00042             this->data[i] = CHAR();
00043         }
00044     };
00045
00046     #if __cplusplus >= 201103L
00047     template< typename CHAR = char, typename LEN_TYPE = unsigned long,
00048             LEN_TYPE MAX = (L4_UTCB_GENERIC_DATA_SIZE *
00049                             sizeof(l4_umword_t)) / sizeof(CHAR) >
00050     using String_in_buf = Array_in_buf<CHAR, LEN_TYPE, MAX>;
00051     #endif
00052
00053     namespace Msg {
00054         template<typename A, typename LEN>
00055         struct Clnt_xmit< String<A, LEN> > : Clnt_xmit< Array<A, LEN> > {};
00056
00057         template<typename A, typename LEN, typename CLASS>
00058         struct Svr_val_ops< String<A, LEN>, Dir_in, CLASS >

```



```

00059 : Svr_val_ops< Array_ref<A, LEN>, Dir_in, CLASS >
00060 {
00061     typedef Svr_val_ops< Array_ref<A, LEN>, Dir_in, CLASS > Base;
00062     typedef typename Base::svr_type svr_type;
00063     using Base::to_svr;
00064     static int to_svr(char *msg, unsigned offset, unsigned limit,
00065                      svr_type &a, Dir_in dir, Cls_data cls)
00066     {
00067         int r = Base::to_svr(msg, offset, limit, a, dir, cls);
00068         if (r < 0)
00069             return r;
00070
00071         // correct clients send at least the zero terminator
00072         if (a.length < 1)
00073             return -L4_MSGTOOSHORT;
00074
00075         typedef typename L4::Types::Remove_const<A>::type elem_type;
00076         const_cast<elem_type*>(a.data)[a.length - 1] = A();
00077         return r;
00078     }
00079 };
00080
00081 template<typename A, typename LEN>
00082 struct Clnt_xmit<String<A, LEN> &> : Clnt_xmit<Array<A, LEN> &>
00083 {
00084     typedef Array<A, LEN> &type;
00085
00086     using Clnt_xmit<type>::from_msg;
00087     static int from_msg(char *msg, unsigned offset, unsigned limit, long ret,
00088                        Array<A, LEN> &a, Dir_out dir, Cls_data cls)
00089     {
00090         int r = Clnt_xmit<type>::from_msg(msg, offset, limit, ret, a, dir, cls);
00091         if (r < 0)
00092             return r;
00093
00094         // check for a bad servers
00095         if (a.length < 1)
00096             return -L4_MSGTOOSHORT;
00097
00098         a.data[a.length - 1] = A();
00099         return r;
00100     };
00101 };
00102
00103 template<typename A, typename LEN>
00104 struct Clnt_xmit<String<A, LEN> *> : Clnt_xmit<String<A, LEN> &> {};
00105
00106 template<typename A, typename LEN, typename CLASS>
00107 struct Svr_val_ops<String<A, LEN>, Dir_out, CLASS>
00108 : Svr_val_ops<Array_ref<A, LEN>, Dir_out, CLASS>
00109 {};
00110
00111 template<typename A, typename LEN>
00112 struct Is_valid_rpc_type<String<A, LEN> const *> : L4::Types::False {};
00113 template<typename A, typename LEN>
00114 struct Is_valid_rpc_type<String<A, LEN> const &> : L4::Types::False {};
00115
00116 } // namespace Msg
00117
00118 }

```

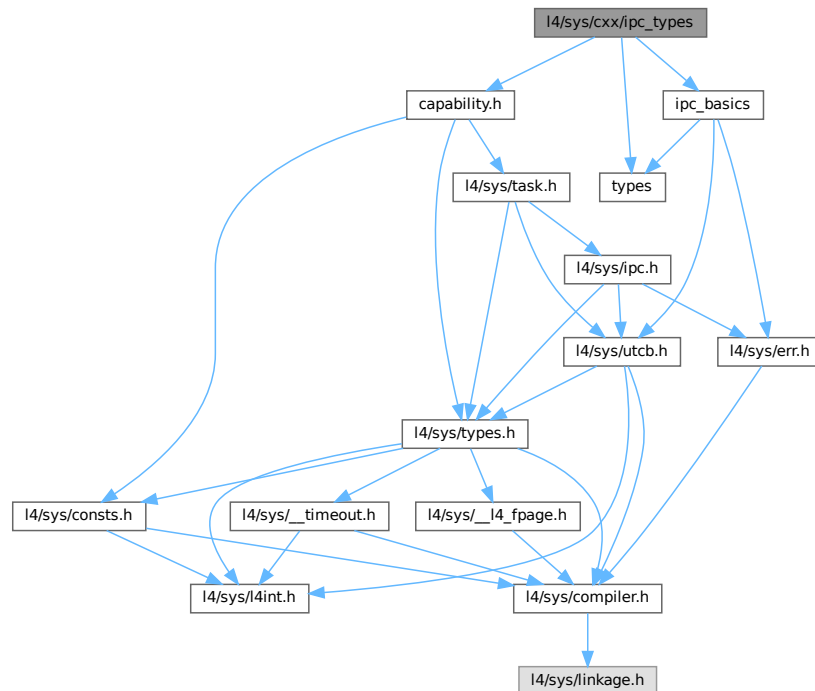
## 16.468 l4/sys/cxx/ipc\_types File Reference

```

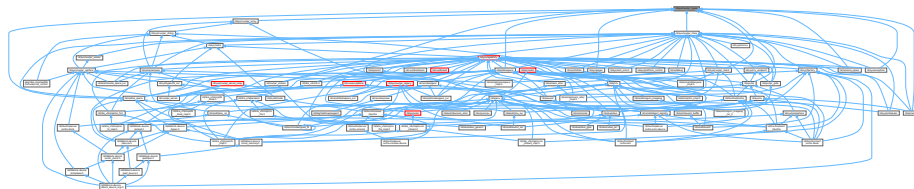
#include "capability.h"
#include "types"
#include "ipc_basics"

```

Include dependency graph for ipc\_types:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [L4::lpc::In\\_out< T >](#)  
Mark an argument as in-out argument.
- struct [L4::lpc::As\\_value< T >](#)  
Pass the argument as plain data value.
- struct [L4::lpc::Opt< T >](#)  
Attribute for defining an optional RPC argument.
- class [L4::lpc::Small\\_buf](#)  
A receive item for receiving a single object capability.
- class [L4::lpc::Gen\\_fpage](#)  
Generic RPC base for [L4](#) flex-pages.
- class [L4::lpc::Snd\\_fpage](#)  
Send flex-page.
- class [L4::lpc::Rcv\\_fpage](#)

*Rcv flex-page.*

- class `L4::lpc::Cap< T >`

*Capability type for RPC interfaces (see `L4::Cap< T >`).*

## Namespaces

- namespace `L4`  
*L4 low-level kernel interface.*
- namespace `L4::lpc`  
*IPC related functionality.*
- namespace `L4::lpc::Msg`  
*IPC Message related functionality.*

## Functions

- template<typename T >  
`Cap< T > L4::lpc::make_cap (L4::Cap< T > cap, unsigned rights) noexcept`  
*Make an `L4::lpc::Cap< T >` for the given capability and rights.*
- template<typename T >  
`Cap< T > L4::lpc::make_cap_rw (L4::Cap< T > cap) noexcept`  
*Make an `L4::lpc::Cap< T >` for the given capability with `L4_CAP_FPAGE_RW` rights.*
- template<typename T >  
`Cap< T > L4::lpc::make_cap_rws (L4::Cap< T > cap) noexcept`  
*Make an `L4::lpc::Cap< T >` for the given capability with `L4_CAP_FPAGE_RWS` rights.*
- template<typename T >  
`Cap< T > L4::lpc::make_cap_full (L4::Cap< T > cap) noexcept`  
*Make an `L4::lpc::Cap< T >` for the given capability with full fpage and object-specific rights.*

## 16.469 ipc\_types

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019
00020 #include "capability.h"
00021 #include "types"
00022 #include "ipc_basics"
00023 namespace L4 {
00024
00025     typedef int Opcode;
00026
00027     namespace Ipc {
00028
00029         template<typename T> struct L4_EXPORT Out;
```

```

00044
00052 template<typename T> struct L4_EXPORT In_out
00053 {
00054     T v;
00055     In_out() {}
00056     In_out(T v) : v(v) {}
00057     operator T () const { return v; }
00058     operator T & () { return v; }
00059 };
00060
00061 namespace Msg {
00062     template<typename A> struct Elem< In_out<A *> > : Elem<A *> {};
00063
00064     template<typename A>
00065     struct Svr_xmit< In_out<A *> > : Svr_xmit<A *>, Svr_xmit<A const *>
00066     {
00067         using Svr_xmit<A *>::from_svr;
00068         using Svr_xmit<A const *>::to_svr;
00069     };
00070
00071     template<typename A>
00072     struct Clnt_xmit< In_out<A *> > : Clnt_xmit<A *>, Clnt_xmit<A const *>
00073     {
00074         using Clnt_xmit<A *>::from_msg;
00075         using Clnt_xmit<A const *>::to_msg;
00076     };
00077
00078     template<typename A>
00079     struct Is_valid_rpc_type< In_out<A *> > : Is_valid_rpc_type<A *> {};
00080     template<typename A>
00081     struct Is_valid_rpc_type< In_out<A const *> > : L4::Types::False {};
00082
00083     #ifdef CONFIG_ALLOW_REFS
00084     template<typename A> struct Elem< In_out<A &> > : Elem<A &> {};
00085
00086     template<typename A>
00087     struct Svr_xmit< In_out<A &> > : Svr_xmit<A &>, Svr_xmit<A const &>
00088     {
00089         using Svr_xmit<A &>::from_svr;
00090         using Svr_xmit<A const &>::to_svr;
00091     };
00092
00093     template<typename A>
00094     struct Clnt_xmit< In_out<A &> > : Clnt_xmit<A &>, Clnt_xmit<A const &>
00095     {
00096         using Clnt_xmit<A &>::from_msg;
00097         using Clnt_xmit<A const &>::to_msg;
00098     };
00099
00100     template<typename A>
00101     struct Is_valid_rpc_type< In_out<A &> > : Is_valid_rpc_type<A &> {};
00102     template<typename A>
00103     struct Is_valid_rpc_type< In_out<A const &> > : L4::Types::False {};
00104
00105     #else
00106
00107     template<typename A>
00108     struct Is_valid_rpc_type< In_out<A &> > : L4::Types::False {};
00109
00110     #endif
00111
00112     // Value types don't make sense for output.
00113     template<typename A>
00114     struct Is_valid_rpc_type< In_out<A> > : L4::Types::False {};
00115
00116 }
00117
00118
00127 template<typename T> struct L4_EXPORT As_value
00128 {
00129     typedef T value_type;
00130     T v;
00131     As_value() noexcept {}
00132     As_value(T v) noexcept : v(v) {}
00133     operator T () const noexcept { return v; }
00134     operator T & () noexcept { return v; }
00135 };
00136
00137 namespace Msg {
00138     template<typename T> struct Class< As_value<T> > : Cls_data {};
00139     template<typename T> struct Elem< As_value<T> > : Elem<T> {};
00140     template<typename T> struct Elem< As_value<T> *> : Elem<T *> {};
00141 }
00142
00143
00147 template<typename T> struct L4_EXPORT Opt
00148 {

```

```

00149     T _value;
00150     bool _valid;
00151
00153     Opt() noexcept : _valid(false) {}
00154
00156     Opt(T value) noexcept : _value(value), _valid(true) {}
00157
00159     Opt &operator = (T value) noexcept
00160     {
00161         this->_value = value;
00162         this->_valid = true;
00163         return *this;
00164     }
00165
00167     void set_valid(bool valid = true) noexcept { _valid = valid; }
00168
00170     T *operator -> () noexcept { return &this->_value; }
00172     T const *operator -> () const noexcept { return &this->_value; }
00174     T value() const noexcept { return this->_value; }
00176     T &value() noexcept { return this->_value; }
00178     bool is_valid() const noexcept { return this->_valid; }
00179 };
00180
00181 namespace Msg {
00182 template<typename T> struct Elem< Opt<T &> > : Elem<T &>
00183 {
00184     enum { Is_optional = true };
00185     typedef Opt<typename Elem<T &>::svr_type> &svr_arg_type;
00186     typedef Opt<typename Elem<T &>::svr_type> svr_type;
00187 };
00188
00189 template<typename T> struct Elem< Opt<T *> > : Elem<T *>
00190 {
00191     enum { Is_optional = true };
00192     typedef Opt<typename Elem<T *>::svr_type> &svr_arg_type;
00193     typedef Opt<typename Elem<T *>::svr_type> svr_type;
00194 };
00195
00196
00197
00198 template<typename T, typename CLASS>
00199 struct Svr_val_ops<Opt<T>, Dir_out, CLASS> : Svr_noops< Opt<T> >
00200 {
00201     typedef Opt<T> svr_type;
00202     typedef Svr_val_ops<T, Dir_out, CLASS> Native;
00203
00204     using Svr_noops< Opt<T> >::to_svr;
00205     static int to_svr(char *msg, unsigned offset, unsigned limit,
00206                      Opt<T> &arg, Dir_out, CLASS) noexcept
00207     {
00208         return Native::to_svr(msg, offset, limit, arg.value(), Dir_out(), CLASS());
00209     }
00210
00211     using Svr_noops< Opt<T> >::from_svr;
00212     static int from_svr(char *msg, unsigned offset, unsigned limit, long ret,
00213                        svr_type &arg, Dir_out, CLASS) noexcept
00214     {
00215         if (arg.is_valid())
00216             return Native::from_svr(msg, offset, limit, ret, arg.value(),
00217                                     Dir_out(), CLASS());
00218         return offset;
00219     }
00220 };
00221
00222 template<typename T> struct Elem< Opt<T> > : Elem<T>
00223 {
00224     enum { Is_optional = true };
00225     typedef Opt<T> arg_type;
00226 };
00227
00228 template<typename T> struct Elem< Opt<T const *> > : Elem<T const *>
00229 {
00230     enum { Is_optional = true };
00231     typedef Opt<T const *> arg_type;
00232 };
00233
00234 template<typename T>
00235 struct Is_valid_rpc_type< Opt<T const &> > : L4::Types::False {};
00236
00237 template<typename T, typename CLASS>
00238 struct Clnt_val_ops<Opt<T>, Dir_in, CLASS> : Clnt_noops< Opt<T> >
00239 {
00240     typedef Opt<T> arg_type;
00241     typedef Detail::Clnt_val_ops<typename Elem<T>::arg_type, Dir_in, CLASS> Native;
00242
00243     using Clnt_noops< Opt<T> >::to_msg;
00244     static int to_msg(char *msg, unsigned offset, unsigned limit,

```

```

00245         arg_type arg, Dir_in, CLASS) noexcept
00246     {
00247         if (arg.is_valid())
00248             return Native::to_msg(msg, offset, limit,
00249                                     Detail::_Plain<T>::deref(arg.value()),
00250                                     Dir_in(), CLASS());
00251         return offset;
00252     }
00253 };
00254
00255 template<typename T> struct Class< Opt<T> > :
00256     Class< typename Detail::_Plain<T>::type > {};
00257 template<typename T> struct Direction< Opt<T> > : Direction<T> {};
00258 }
00259
00260 class L4_EXPORT Small_buf
00261 {
00262 public:
00263     explicit Small_buf(L4::Cap<void> cap, unsigned long flags = 0) noexcept
00264         : _data(cap.cap() | L4_RCV_ITEM_SINGLE_CAP | flags) {}
00265
00266     explicit Small_buf(l4_cap_idx_t cap, unsigned long flags = 0) noexcept
00267         : _data(cap | L4_RCV_ITEM_SINGLE_CAP | flags) {}
00268
00269     l4_umword_t raw() const noexcept { return _data; }
00270 private:
00271     l4_umword_t _data;
00272 };
00273
00274 class L4_EXPORT Gen_fpage
00275 {
00276 public:
00277     enum Type
00278     {
00279         Special = L4_FPAGE_SPECIAL « 4,
00280         Memory  = L4_FPAGE_MEMORY « 4,
00281         Io       = L4_FPAGE_IO « 4,
00282         Obj      = L4_FPAGE_OBJ « 4
00283     };
00284
00285     enum Map_type
00286     {
00287         Map    = L4_MAP_ITEM_MAP,
00288         Grant  = L4_MAP_ITEM_GRANT,
00289     };
00290
00291     enum Cacheopt
00292     {
00293         None    = 0,
00294         Cached  = L4_FPAGE_CACHEABLE « 4,
00295         Buffered = L4_FPAGE_BUFFERABLE « 4,
00296         Uncached = L4_FPAGE_UNCACHEABLE « 4
00297     };
00298
00299     enum Continue
00300     {
00301         Single   = 0,
00302         Last     = 0,
00303         More     = L4_ITEM_CONT,
00304         Compound = L4_ITEM_CONT,
00305     };
00306
00307     Gen_fpage(l4_umword_t base, l4_umword_t data) noexcept
00308         : _base(base), _data(data)
00309     {}
00310
00311     Gen_fpage(Type type, l4_addr_t base, int order,
00312               unsigned char rights,
00313               l4_addr_t snd_base,
00314               Map_type map_type,
00315               Cacheopt cache, Continue cont) noexcept
00316         : _base(L4_ITEM_MAP | (snd_base & (~0UL « 12)) | l4_umword_t(map_type)
00317               | l4_umword_t(cache) | l4_umword_t(cont)),
00318           _data(base | l4_umword_t(type) | rights | (l4_umword_t(order) « 6))
00319     {}
00320
00321     unsigned order() const noexcept { return (_data « 6) & 0x3f; }
00322     unsigned snd_order() const noexcept { return (_data « 6) & 0x3f; }
00323     unsigned rcv_order() const noexcept { return (_base « 6) & 0x3f; }
00324     l4_addr_t base() const noexcept { return _data & (~0UL « 12); }
00325     l4_addr_t snd_base() const noexcept { return _base & (~0UL « 12); }
00326     void snd_base(l4_addr_t b) noexcept { _base = (_base & ~(~0UL « 12)) | (b & (~0UL « 12)); }
00327
00328     bool is_valid() const noexcept { return _base & L4_ITEM_MAP; }
00329     bool cap_received() const noexcept { return (_base & 0x3e) == 0x38; }
00330     bool id_received() const noexcept { return (_base & 0x3e) == 0x3c; }
00331     bool local_id_received() const noexcept { return (_base & 0x3e) == 0x3e; }

```

```

00388
00395 bool is_compound() const noexcept { return _base & 1; }
00397 l4_umword_t data() const noexcept { return _data; }
00399 l4_umword_t base_x() const noexcept { return _base; }
00400
00401 protected:
00402     l4_umword_t _base;
00403     l4_umword_t _data;
00404 };
00405
00407 class Snd_fpage : public Gen_fpage
00408 {
00409 public:
00410     Snd_fpage(l4_umword_t base = 0, l4_umword_t data = 0) noexcept
00411     : Gen_fpage(base, data)
00412     {}
00413
00414     Snd_fpage(l4_fpage_t const &fp, l4_addr_t snd_base = 0,
00415               Map_type map_type = Map,
00416               Cacheopt cache = None, Continue cont = Last) noexcept
00417     : Gen_fpage(L4_ITEM_MAP | (snd_base & (~0UL << 12)) | l4_umword_t(map_type)
00418               | l4_umword_t(cache) | l4_umword_t(cont),
00419               fp.raw)
00420     {}
00421
00422     Snd_fpage(L4::Cap<void> cap, unsigned rights, Map_type map_type = Map) noexcept
00423     : Gen_fpage(L4_ITEM_MAP | l4_umword_t(map_type) | (rights & 0xf0),
00424               cap.fpage(rights).raw)
00425     {}
00426
00427     static Snd_fpage obj(l4_cap_idx_t base, int order,
00428                          unsigned char rights,
00429                          l4_addr_t snd_base = 0,
00430                          Map_type map_type = Map,
00431                          Continue cont = Last) noexcept
00432     {
00433         return Snd_fpage(l4_obj_fpage(base, order, rights), snd_base,
00434                          map_type, None, cont);
00435     }
00436
00437     static Snd_fpage mem(l4_addr_t base, int order,
00438                          unsigned char rights,
00439                          l4_addr_t snd_base = 0,
00440                          Map_type map_type = Map,
00441                          Cacheopt cache = None, Continue cont = Last) noexcept
00442     {
00443         return Snd_fpage(l4_fpage(base, order, rights), snd_base, map_type, cache,
00444                          cont);
00445     }
00446
00447     static Snd_fpage io(unsigned long base, int order,
00448                          unsigned char rights,
00449                          l4_addr_t snd_base = 0,
00450                          Map_type map_type = Map,
00451                          Continue cont = Last) noexcept
00452     {
00453         return Snd_fpage(l4_fpage_set_rights(l4_iofpage(base, order), rights),
00454                          snd_base, map_type, None, cont);
00455     }
00456 };
00457
00459 class Rcv_fpage : public Gen_fpage
00460 {
00461 public:
00462     Rcv_fpage() noexcept : Gen_fpage(0, 0), _rcv_task(L4_INVALID_CAP) {}
00463     Rcv_fpage(l4_fpage_t const &fp, l4_addr_t snd_base = 0,
00464               l4_cap_idx_t rcv_task = L4_INVALID_CAP) noexcept
00465     : Gen_fpage(L4_ITEM_MAP | (snd_base & (~0UL << 12))
00466               | (l4_is_valid_cap(rcv_task) ? Compound : 0),
00467               fp.raw),
00468       _rcv_task(rcv_task)
00469     {}
00470
00471     static Rcv_fpage obj(l4_cap_idx_t base, int order, l4_addr_t snd_base = 0,
00472                          L4::Cap<void> rcv_task = L4::Cap<void>::Invalid) noexcept
00473     {
00474         return Rcv_fpage(l4_obj_fpage(base, order, 0), snd_base,
00475                          rcv_task.cap());
00476     }
00477
00478     static Rcv_fpage mem(l4_addr_t base, int order, l4_addr_t snd_base = 0,
00479                          L4::Cap<void> rcv_task = L4::Cap<void>::Invalid) noexcept
00480     {
00481         return Rcv_fpage(l4_fpage(base, order, 0), snd_base, rcv_task.cap());
00482     }
00483
00484     static Rcv_fpage io(unsigned long base, int order, l4_addr_t snd_base = 0,

```

```

00485         L4::Cap<void> rcv_task = L4::Cap<void>::Invalid) noexcept
00486     {
00487         return Rcv_fpage(l4_iofpage(base, order), snd_base, rcv_task.cap());
00488     }
00489
00490     l4_cap_idx_t rcv_task() const { return _rcv_task; }
00491
00492 protected:
00493     l4_cap_idx_t _rcv_task;
00494 };
00495
00496
00497 namespace Msg {
00498
00499     // Snd_fpage are out items
00500     template<> struct Class<L4::Ipc::Snd_fpage> : Cls_item {};
00501
00502     // Rcv_fpage are buffer items
00503     template<> struct Class<L4::Ipc::Rcv_fpage> : Cls_buffer {};
00504
00505     template<>
00506     struct Clnt_val_ops<L4::Ipc::Rcv_fpage, Dir_in, Cls_buffer>
00507         : Clnt_noops<L4::Ipc::Rcv_fpage>
00508     {
00509         using Clnt_noops<L4::Ipc::Rcv_fpage>::to_msg;
00510
00511         static int to_msg(char *msg, unsigned offs, unsigned limit,
00512             L4::Ipc::Rcv_fpage arg, Dir_in, Cls_buffer) noexcept
00513         {
00514             offs = align_to<l4_umword_t>(offs);
00515             unsigned words = arg.is_compound() ? 3 : 2;
00516             if (L4_UNLIKELY(!check_size<l4_umword_t>(offs, limit, words)))
00517                 return -L4_EMSTOOLONG;
00518             auto *buf = reinterpret_cast<l4_umword_t*>(msg + offs);
00519             *buf++ = arg.base_x();
00520             *buf++ = arg.data();
00521             if (arg.is_compound())
00522                 *buf++ = arg.rcv_task();
00523             return offs + sizeof(l4_umword_t) * words;
00524         }
00525     };
00526
00527
00528     // Remove receive buffers from server-side arguments
00529     template<> struct Elem<L4::Ipc::Rcv_fpage>
00530     {
00531         typedef L4::Ipc::Rcv_fpage arg_type;
00532         typedef void svr_type;
00533         typedef void svr_arg_type;
00534         enum { Is_optional = false };
00535     };
00536
00537     // Small_buf are buffer items
00538     template<> struct Class<L4::Ipc::Small_buf> : Cls_buffer {};
00539
00540     // Remove receive buffers from server-side arguments
00541     template<> struct Elem<L4::Ipc::Small_buf>
00542     {
00543         typedef L4::Ipc::Small_buf arg_type;
00544         typedef void svr_type;
00545         typedef void svr_arg_type;
00546         enum { Is_optional = false };
00547     };
00548 } // namespace Msg
00549
00550 // L4::Cap<> handling
00551
00552 template<typename T> class Cap
00553 {
00554     template<typename O> friend class Cap;
00555     l4_umword_t _cap_n_rights;
00556
00557 public:
00558     enum
00559     {
00575         Rights_mask = 0xff,
00576
00581         Cap_mask    = L4_CAP_MASK
00582     };
00583
00585     template<typename O>
00586     Cap(Cap<O> const &o) noexcept : _cap_n_rights(o._cap_n_rights)
00587     {
00588         L4::Cap<T>::template check_convertible_from<O>();
00589     }
00590
00592     Cap(L4::Cap<T> cap) noexcept

```



```

00593 : _cap_n_rights((cap.cap() & Cap_mask) | (cap ? L4_CAP_FPAGE_R : 0))
00594 {}
00595
00597 template<typename O>
00598 Cap(L4::Cap<O> cap) noexcept
00599 : _cap_n_rights((cap.cap() & Cap_mask) | (cap ? L4_CAP_FPAGE_R : 0))
00600 {
00601     L4::Cap<T>::template check_convertible_from<O>();
00602 }
00603
00605 Cap() noexcept : _cap_n_rights(L4_INVALID_CAP) {}
00606
00614 Cap(L4::Cap<T> cap, unsigned char rights) noexcept
00615 : _cap_n_rights((cap.cap() & Cap_mask) | (rights & Rights_mask)) {}
00616
00622 static Cap from_ci(l4_cap_idx_t c) noexcept
00623 { return Cap(L4::Cap<T>(c & Cap_mask), c & Rights_mask); }
00624
00626 L4::Cap<T> cap() const noexcept
00627 { return L4::Cap<T>(_cap_n_rights & Cap_mask); }
00628
00630 unsigned rights() const noexcept
00631 { return _cap_n_rights & Rights_mask; }
00632
00634 L4::Ipc::Snd_fpage fpage() const noexcept
00635 { return L4::Ipc::Snd_fpage(cap(), rights()); }
00636
00638 bool is_valid() const noexcept
00639 { return !(_cap_n_rights & L4_INVALID_CAP_BIT); }
00640 };
00641
00648 template<typename T>
00649 Cap<T> make_cap(L4::Cap<T> cap, unsigned rights) noexcept
00650 { return Cap<T>(cap, rights); }
00651
00658 template<typename T>
00659 Cap<T> make_cap_rw(L4::Cap<T> cap) noexcept
00660 { return Cap<T>(cap, L4_CAP_FPAGE_RW); }
00661
00668 template<typename T>
00669 Cap<T> make_cap_rws(L4::Cap<T> cap) noexcept
00670 { return Cap<T>(cap, L4_CAP_FPAGE_RWS); }
00671
00686 template<typename T>
00687 Cap<T> make_cap_full(L4::Cap<T> cap) noexcept
00688 { return Cap<T>(cap, L4_CAP_FPAGE_RWS | L4_FPAGE_C_OBJ_RIGHTS); }
00689
00690 // caps are special the have an invalid representation
00691 template<typename T> struct L4_EXPORT Opt< Cap<T> >
00692 {
00693     Cap<T> _value;
00694     Opt() noexcept {}
00695     Opt(Cap<T> value) noexcept : _value(value) {}
00696     Opt(L4::Cap<T> value) noexcept : _value(value) {}
00697     Opt &operator = (Cap<T> value) noexcept
00698     { this->_value = value; }
00699     Opt &operator = (L4::Cap<T> value) noexcept
00700     { this->_value = value; }
00701
00702     Cap<T> value() const noexcept { return this->_value; }
00703     bool is_valid() const noexcept { return this->_value.is_valid(); }
00704 };
00705
00706
00707 namespace Msg {
00708 // prohibit L4::Cap as argument
00709 template<typename A>
00710 struct Is_valid_rpc_type< L4::Cap<A> > : L4::Types::False {};
00711
00712 template<typename A> struct Class< Cap<A> > : Cls_item {};
00713 template<typename A> struct Elem< Cap<A> >
00714 {
00715     enum { Is_optional = false };
00716     typedef Cap<A> arg_type;
00717     typedef L4::Ipc::Snd_fpage svr_type;
00718     typedef L4::Ipc::Snd_fpage svr_arg_type;
00719 };
00720
00721
00722 template<typename A, typename CLASS>
00723 struct Svr_val_ops<Cap<A>, Dir_in, CLASS> :
00724     Svr_val_ops<L4::Ipc::Snd_fpage, Dir_in, CLASS>
00725 {};
00726
00727 template<typename A, typename CLASS>
00728 struct Clnt_val_ops<Cap<A>, Dir_in, CLASS> :
00729     Clnt_noops< Cap<A> >

```

```

00730 {
00731     using Clnt_noops< Cap<A> >::to_msg;
00732
00733     static int to_msg(char *msg, unsigned offset, unsigned limit,
00734                       Cap<A> arg, Dir_in, Cls_item) noexcept
00735     {
00736         // passing an invalid cap as mandatory argument is an error
00737         // XXX: This checks for a client calling error, we could
00738         //       also just ignore this for performance reasons and
00739         //       let the client fail badly (Alex: I'd prefer this)
00740         if (L4_UNLIKELY(!arg.is_valid()))
00741             return -L4_MSGMISSARG;
00742
00743         return msg_add(msg, offset, limit, arg.fpage());
00744     }
00745 };
00746
00747 template<typename A>
00748 struct Elem<Out<L4::Cap<A> > >
00749 {
00750     enum { Is_optional = false };
00751     typedef L4::Cap<A> arg_type;
00752     typedef Ipc::Cap<A> svr_type;
00753     typedef svr_type &svr_arg_type;
00754 };
00755
00756 template<typename A> struct Direction< Out< L4::Cap<A> > > : Dir_out {};
00757 template<typename A> struct Class< Out< L4::Cap<A> > > : Cls_item {};
00758
00759 template<typename A>
00760 struct Clnt_val_ops< L4::Cap<A>, Dir_out, Cls_item > :
00761     Clnt_noops< L4::Cap<A> >
00762 {
00763     using Clnt_noops< L4::Cap<A> >::to_msg;
00764     static int to_msg(char *msg, unsigned offset, unsigned limit,
00765                       L4::Cap<A> arg, Dir_in, Cls_buffer) noexcept
00766     {
00767         if (L4_UNLIKELY(!arg.is_valid()))
00768             return -L4_MSGMISSARG; // no buffer inserted
00769         return msg_add(msg, offset, limit, Small_buf(arg));
00770     }
00771 };
00772
00773 template<typename A>
00774 struct Svr_val_ops< L4::Ipc::Cap<A>, Dir_out, Cls_item > :
00775     Svr_noops<Cap<A> &>
00776 {
00777     using Svr_noops<Cap<A> &>::from_svr;
00778     static int from_svr(char *msg, unsigned offset, unsigned limit, long,
00779                         Cap<A> arg, Dir_out, Cls_item) noexcept
00780     {
00781         if (L4_UNLIKELY(!arg.is_valid()))
00782             // do not map anything
00783             return msg_add(msg, offset, limit, L4::Ipc::Snd_fpage(arg.cap(), 0));
00784
00785         return msg_add(msg, offset, limit, arg.fpage());
00786     }
00787 };
00788
00789 // prohibit a UTCB pointer as normal RPC argument
00790 template<> struct Is_valid_rpc_type<l4_utcb_t *> : L4::Types::False {};
00791
00792 } // namespace Msg
00793 } // namespace Ipc
00794 } // namespace L4

```

## 16.470 ipc\_varg

```

00001 // vi:set ft=c++: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by

```

```

00016  * the GNU General Public License.
00017  */
00018  #pragma once
00019  #pragma GCC system_header
00020
00021  #include "types"
00022  #include "ipc_basics"
00023
00024  namespace L4 { namespace Ipc L4_EXPORT {
00025
00026  template< typename T, template <typename X> class B >
00027  struct Generic_va_type : B<T>
00028  {
00029      enum { Id = B<T>::Id };
00030      typedef B<T> ID;
00031      typedef T const &Ret_value;
00032      typedef T Value;
00033
00034      static Ret_value value(void const *d)
00035      { return *reinterpret_cast<Value const *>(d); }
00036
00037      static void const *addr_of(Value const &v) { return &v; }
00038
00039      static unsigned size(void const *) { return sizeof(T); }
00040
00041      static L4_varg_type unsigned_id()
00042      {
00043          return static_cast<L4_varg_type>(Id & ~L4_VARG_TYPE_SIGN);
00044      }
00045
00046      static L4_varg_type signed_id()
00047      {
00048          return static_cast<L4_varg_type>(Id | L4_VARG_TYPE_SIGN);
00049      }
00050
00051      static L4_varg_type id()
00052      {
00053          return static_cast<L4_varg_type>(Id);
00054      }
00055  };
00056
00057  template< typename T > struct Va_type_id;
00058  template<> struct Va_type_id<l4_umword_t> { enum { Id = L4_VARG_TYPE_UMWORD }; };
00059  template<> struct Va_type_id<l4_mword_t> { enum { Id = L4_VARG_TYPE_MWORD }; };
00060  template<> struct Va_type_id<l4_fpage_t> { enum { Id = L4_VARG_TYPE_FPAGE }; };
00061  template<> struct Va_type_id<void> { enum { Id = L4_VARG_TYPE_NIL }; };
00062  template<> struct Va_type_id<char const *> { enum { Id = L4_VARG_TYPE_STRING }; };
00063
00064  template< typename T > struct Va_type;
00065
00066  template<> struct Va_type<l4_umword_t> : Generic_va_type<l4_umword_t, Va_type_id> {};
00067  template<> struct Va_type<l4_mword_t> : Generic_va_type<l4_mword_t, Va_type_id> {};
00068  template<> struct Va_type<l4_fpage_t> : Generic_va_type<l4_fpage_t, Va_type_id> {};
00069
00070  template<> struct Va_type<void>
00071  {
00072      typedef void Ret_value;
00073      typedef void Value;
00074
00075      static void const *addr_of(void) { return 0; }
00076
00077      static void value(void const *) {}
00078      static L4_varg_type id() { return L4_VARG_TYPE_NIL; }
00079      static unsigned size(void const *) { return 0; }
00080  };
00081
00082  template<> struct Va_type<char const *>
00083  {
00084      typedef char const *Ret_value;
00085      typedef char const *Value;
00086
00087      static void const *addr_of(Value v) { return v; }
00088
00089      static L4_varg_type id() { return L4_VARG_TYPE_STRING; }
00090      static unsigned size(void const *s)
00091      {
00092          char const *_s = reinterpret_cast<char const *>(s);
00093          int l = 1;
00094          while (*_s)
00095          {
00096              ++_s; ++l;
00097          }
00098          return l;
00099      }
00100
00101      static Ret_value value(void const *d) { return static_cast<char const *>(d); }
00102  };

```

```

00103
00107 class Varg
00108 {
00109 private:
00110     enum { Direct_data = 0x8000 };
00111     l4_umword_t _tag;
00112     char const *_d;
00113
00114 public:
00115
00117     typedef l4_umword_t Tag;
00118
00120     L4_varg_type type() const { return static_cast<L4_varg_type>(_tag & 0xff); }
00125     unsigned length() const { return _tag >> 16; }
00127     Tag tag() const { return _tag & ~Direct_data; }
00129     void tag(Tag tag) { _tag = tag; }
00131     void data(char const *d) { _d = d; }
00132
00134     char const *data() const
00135     {
00136         if (_tag & Direct_data)
00137         {
00138             union T { char const *d; char v[sizeof(char const *)]; };
00139             return reinterpret_cast<T const *>(&_d)->v;
00140         }
00141         return _d;
00142     }
00143
00145     #if __cplusplus >= 201103L
00146     Varg() = default;
00147     #else
00148     Varg() {}
00149     #endif
00150
00152     Varg(L4_varg_type t, void const *v, int len)
00153     : _tag(t | (static_cast<l4_mword_t>(len) << 16)),
00154       _d(static_cast<char const *>(v))
00155     {}
00156
00157     static Varg nil() { return Varg(L4_VARG_TYPE_NIL, 0, 0); }
00158
00165     template< typename V >
00166     typename Va_type<V>::Ret_value value() const
00167     {
00168         if (_tag & Direct_data)
00169         {
00170             union X { char const *d; V v; };
00171             return reinterpret_cast<X const *>(&_d).v;
00172         }
00173
00174         return Va_type<V>::value(_d);
00175     }
00176
00177
00179     template< typename T >
00180     bool is_of() const { return Va_type<T>::id() == type(); }
00181
00183     bool is_nil() const { return is_of<void>(); }
00184
00186     bool is_of_int() const
00187     { return (type() & ~L4_VARG_TYPE_SIGN) == L4_VARG_TYPE_UMWORD; }
00188
00195     template< typename T >
00196     bool get_value(typename Va_type<T>::Value *v) const
00197     {
00198         if (!is_of<T>())
00199             return false;
00200
00201         *v = this->value<T>();
00202         return true;
00203     }
00204
00206     template< typename T >
00207     void set_value(void const *d)
00208     {
00209         typedef Va_type<T> Vt;
00210         _tag = Vt::id() | (Vt::size(d) << 16);
00211         _d = static_cast<char const *>(d);
00212     }
00213
00215     template<typename T>
00216     void set_direct_value(T val, typename L4::Types::Enable_if<sizeof(T) <= sizeof(char const *)>,
00217                           bool>::type = true)
00218     {
00219         static_assert(sizeof(T) <= sizeof(char const *), "direct Varg value too big");
00219         typedef Va_type<T> Vt;
00220         _tag = Vt::id() | (sizeof(T) << 16) | Direct_data;

```

```

00221     union X { char const *d; T v; };
00222     reinterpret_cast<X &>(_d).v = val;
00223 }
00224
00226 template<typename T> explicit
00227 Varg(T const *data) { set_value<T>(data); }
00229 Varg(char const *data) { set_value<char const *>(data); }
00230
00232 template<typename T> explicit
00233 Varg(T data, typename L4::Types::Enable_if<sizeof(T) <= sizeof(char const *)>, bool>::type = true)
00234 { set_direct_value<T>(data); }
00235 };
00236
00237
00238 template<typename T>
00239 class Varg_t : public Varg
00240 {
00241 public:
00242     typedef typename Va_type<T>::Value Value;
00243     explicit Varg_t(Value v) : Varg()
00244     { _data = v; set_value<T>(Va_type<T>::addr_of(_data)); }
00245
00246 private:
00247     Value _data;
00248 };
00249
00250 template<unsigned MAX = L4_UTCB_GENERIC_DATA_SIZE>
00251 class Varg_list;
00252
00264 class Varg_list_ref
00265 {
00266 private:
00267     template<unsigned T>
00268     friend class Varg_list;
00269
00271     class Iter_state
00272     {
00273     private:
00274         using M = l4_umword_t;
00275         using Mp = M const *;
00276         Mp _c;
00277         Mp _e;
00278
00280         Mp next_arg(Varg const &a) const
00281         {
00282             return _c + 1 + (Msg::align_to<M>(a.length()) / sizeof(M));
00283         }
00284
00285     public:
00287         Iter_state() : _c(nullptr), _e(nullptr) {}
00288
00290         Iter_state(Mp c, Mp e) : _c(c), _e(e)
00291         {}
00292
00294         bool valid() const
00295         { return _c && _c < _e; }
00296
00298         Mp begin() const { return _c; }
00299
00301         Mp end() const { return _e; }
00302
00307         Varg pop()
00308         {
00309             if (!valid())
00310                 return Varg::nil();
00311
00312             Varg a;
00313             a.tag(_c[0]);
00314             a.data(reinterpret_cast<char const *>(&_c[1]));
00315             _c = next_arg(a);
00316             if (_c > _e)
00317                 return Varg::nil();
00318
00319             return a;
00320         }
00321
00323         bool operator == (Iter_state const &o) const
00324         { return _c == o._c; }
00325
00327         bool operator != (Iter_state const &o) const
00328         { return _c != o._c; }
00329     };
00330
00331     Iter_state _s;
00332
00333 public:
00335     Varg_list_ref() = default;

```

```

00336
00343 Varg_list_ref(void const *start, void const *end)
00344 : _s(reinterpret_cast<l4_umword_t const *>(start),
00345      reinterpret_cast<l4_umword_t const *>(end))
00346 {}
00347
00349 class Iterator
00350 {
00351 private:
00352     Iter_state _s;
00353     Varg _a;
00354
00355 public:
00357     Iterator(Iter_state const &s)
00358     : _s(s)
00359     {
00360         _a = _s.pop();
00361     }
00362
00364     explicit operator bool () const
00365     { return !_a.is_nil(); }
00366
00368     Iterator &operator ++ ()
00369     {
00370         if (!_a.is_nil())
00371             _a = _s.pop();
00372
00373         return *this;
00374     }
00375
00377     Varg operator * () const
00378     { return _a; }
00379
00381     bool equals(Iterator const &o) const
00382     {
00383         if (_a.is_nil() && o._a.is_nil())
00384             return true;
00385
00386         return _s == o._s;
00387     }
00388
00389     bool operator == (Iterator const &o) const
00390     { return equals(o); }
00391
00392     bool operator != (Iterator const &o) const
00393     { return !equals(o); }
00394 };
00395
00397 Varg pop_front()
00398 { return _s.pop(); }
00399
00401 Varg next()
00402     L4_DEPRECATED("Use range for or pop_front.")
00403 { return _s.pop(); }
00404
00406 Iterator begin() const
00407 { return Iterator(_s); }
00408
00410 Iterator end() const
00411 { return Iterator(Iter_state()); }
00412 };
00413
00421 template<unsigned MAX>
00422 class Varg_list : public Varg_list_ref
00423 {
00424     l4_umword_t data[MAX];
00425     Varg_list(Varg_list const &);
00426
00427 public:
00429     Varg_list(Varg_list_ref const &r)
00430     {
00431         if (!r._s.valid())
00432             return;
00433
00434         l4_umword_t const *rs = r._s.begin();
00435         unsigned c = r._s.end() - rs;
00436         for (unsigned i = 0; i < c; ++i)
00437             data[i] = rs[i];
00438
00439         this->_s = Iter_state(data, data + c);
00440     }
00441 };
00442
00443 namespace Msg {
00444     template<> struct Elem<Varg const *>
00445     {

```

```

00447     typedef Varg const *arg_type;
00448     typedef Varg_list_ref svr_type;
00449     typedef Varg_list_ref svr_arg_type;
00450     enum { Is_optional = false };
00451 };
00452
00453 template<> struct Is_valid_rpc_type<Varg> : L4::Types::False {};
00454 template<> struct Is_valid_rpc_type<Varg*> : L4::Types::False {};
00455 template<> struct Is_valid_rpc_type<Varg&> : L4::Types::False {};
00456 template<> struct Is_valid_rpc_type<Varg const&> : L4::Types::False {};
00457
00458 template<> struct Direction<Varg const*> : Dir_in {};
00459 template<> struct Class<Varg const*> : Cls_data {};
00460
00461 template<typename DIR, typename CLASS>
00462 struct Clnt_val_ops<Varg, DIR, CLASS>;
00463
00464 template<>
00465 struct Clnt_val_ops<Varg, Dir_in, Cls_data> :
00466     Clnt_noops<Varg const&>
00467 {
00468     using Clnt_noops<Varg const&>::to_msg;
00469     static int to_msg(char *msg, unsigned offs, unsigned limit,
00470                     Varg const &a, Dir_in, Cls_data)
00471     {
00472         for (Varg const *i = &a; i->tag(); ++i)
00473         {
00474             offs = align_to<l4_umword_t>(offs);
00475             if (L4_UNLIKELY(!check_size<l4_umword_t>(offs, limit)))
00476                 return -L4_MSGTOOLONG;
00477             *reinterpret_cast<l4_umword_t*>(msg + offs) = i->tag();
00478             offs += sizeof(l4_umword_t);
00479             if (L4_UNLIKELY(!check_size<char>(offs, limit, i->length())))
00480                 return -L4_MSGTOOLONG;
00481             char const *d = i->data();
00482             for (unsigned x = 0; x < i->length(); ++x)
00483                 msg[offs++] = *d++;
00484         }
00485
00486         return offs;
00487     }
00488 };
00489
00490 template<>
00491 struct Svr_val_ops<Varg_list_ref, Dir_in, Cls_data> :
00492     Svr_noops<Varg_list_ref>
00493 {
00494     using Svr_noops<Varg_list_ref>::to_svr;
00495     static int to_svr(char *msg, unsigned offset, unsigned limit,
00496                     Varg_list_ref &a, Dir_in, Cls_data)
00497     {
00498         unsigned start = align_to<l4_umword_t>(offset);
00499         unsigned offs;
00500         for (offs = start; offs < limit;)
00501         {
00502             unsigned noffs = align_to<l4_umword_t>(offs);
00503             if (L4_UNLIKELY(!check_size<l4_umword_t>(noffs, limit)))
00504                 break;
00505
00506             offs = noffs;
00507             Varg arg;
00508             arg.tag(*reinterpret_cast<l4_umword_t*>(msg + offs));
00509
00510             if (!arg.tag())
00511                 break;
00512
00513             offs += sizeof(l4_umword_t);
00514
00515             if (L4_UNLIKELY(!check_size<char>(offs, limit, arg.length())))
00516                 return -L4_MSGTOOLONG;
00517             offs += arg.length();
00518         }
00519
00520         a = Varg_list_ref(msg + start, msg + align_to<l4_umword_t>(offs));
00521         return offs;
00522     }
00523 };
00524 }
00525 }

```





## 16.472 smart\_capability\_1x

[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00025 #include <l4/sys/capability>
00026
00027 namespace L4 { namespace Detail {
00028
00029 template< typename T, typename IMPL >
00030 class Smart_cap_base : public Cap_base, protected IMPL
00031 {
00032 protected:
00033     template<typename X>
00034     static IMPL &impl(Smart_cap_base<X, IMPL> &o) { return o; }
00035
00036     template<typename X>
00037     static IMPL const &impl(Smart_cap_base<X, IMPL> const &o) { return o; }
00038
00039 public:
00040     template<typename X, typename I>
00041     friend class ::L4::Detail::Smart_cap_base;
00042
00043     Smart_cap_base(Smart_cap_base const &) = delete;
00044     Smart_cap_base &operator = (Smart_cap_base const &) = delete;
00045
00046     Smart_cap_base() noexcept : Cap_base(Invalid) {}
00047
00048     explicit Smart_cap_base(Cap_base::Cap_type t) noexcept
00049     : Cap_base(t)
00050     {}
00051
00052     template<typename O>
00053     explicit constexpr Smart_cap_base(Cap<O> c) noexcept
00054     : Cap_base(c.cap())
00055     {}
00056
00057     template<typename O>
00058     explicit constexpr Smart_cap_base(Cap<O> c, IMPL const &impl) noexcept
00059     : Cap_base(c.cap()), IMPL(impl)
00060     {}
00061
00062     Cap<T> release() noexcept
00063     {
00064         l4_cap_idx_t c = this->cap();
00065         IMPL::invalidate(*this);
00066         return Cap<T>(c);
00067     }
00068
00069     void reset()
00070     { IMPL::free(*this); }
00071
00072     Cap<T> operator -> () const noexcept { return Cap<T>(this->cap()); }
00073     Cap<T> get() const noexcept { return Cap<T>(this->cap()); }
00074     ~Smart_cap_base() noexcept { IMPL::free(*this); }
00075 };
00076
00077
00078 template< typename T, typename IMPL >
00079 class Unique_cap_impl final : public Smart_cap_base<T, IMPL>
00080 {
00081 private:
00082     typedef Smart_cap_base<T, IMPL> Base;
00083
00084 public:
00085     using Base::Base;
00086     Unique_cap_impl() noexcept = default;
```

```

00087
00088 Unique_cap_impl(Unique_cap_impl &&o) noexcept
00089 : Base(o.release(), Base::impl(o))
00090 {}
00091
00092 template<typename O>
00093 Unique_cap_impl(Unique_cap_impl<O, IMPL> &&o) noexcept
00094 : Base(o.release(), Base::impl(o))
00095 { Cap<T>::template check_convertible_from<O>(); }
00096
00097 Unique_cap_impl &operator = (Unique_cap_impl &&o) noexcept
00098 {
00099     if (&o == this)
00100         return *this;
00101
00102     IMPL::free(*this);
00103     this->_c = o.release().cap();
00104     this->IMPL::operator = (Base::impl(o));
00105     return *this;
00106 }
00107
00108 template<typename O>
00109 Unique_cap_impl &operator = (Unique_cap_impl<O, IMPL> &&o) noexcept
00110 {
00111     Cap<T>::template check_convertible_from<O>();
00112
00113     IMPL::free(*this);
00114     this->_c = o.release().cap();
00115     this->IMPL::operator = (Base::impl(o));
00116     return *this;
00117 }
00118 };
00119
00120 template<typename T, typename IMPL>
00121 class Shared_cap_impl final : public Smart_cap_base<T, IMPL>
00122 {
00123 private:
00124     typedef Smart_cap_base<T, IMPL> Base;
00125
00126 public:
00127     using Base::Base;
00128     Shared_cap_impl() noexcept = default;
00129
00130     Shared_cap_impl(Shared_cap_impl &&o) noexcept
00131     : Base(o.release())
00132     {}
00133
00134     template<typename O>
00135     Shared_cap_impl(Shared_cap_impl<O, IMPL> &&o) noexcept
00136     : Base(o.release())
00137     { Cap<T>::template check_convertible_from<O>(); }
00138
00139     Shared_cap_impl &operator = (Shared_cap_impl &&o) noexcept
00140     {
00141         if (&o == this)
00142             return *this;
00143
00144         IMPL::free(*this);
00145         this->_c = o.release().cap();
00146         this->IMPL::operator = (Base::impl(o));
00147         return *this;
00148     }
00149
00150     template<typename O>
00151     Shared_cap_impl &operator = (Shared_cap_impl<O, IMPL> &&o) noexcept
00152     {
00153         Cap<T>::template check_convertible_from<O>();
00154
00155         IMPL::free(*this);
00156         this->_c = o.release().cap();
00157         this->IMPL::operator = (Base::impl(o));
00158         return *this;
00159     }
00160
00161     Shared_cap_impl(Shared_cap_impl const &o) noexcept
00162     : Base(L4::Cap<T>(IMPL::copy(o).cap()))
00163     {}
00164
00165     template<typename O>
00166     Shared_cap_impl(Shared_cap_impl<O, IMPL> const &o) noexcept
00167     : Base(IMPL::copy(o))
00168     { Cap<T>::template check_convertible_from<O>(); }
00169
00170     Shared_cap_impl &operator = (Shared_cap_impl const &o) noexcept
00171     {
00172         if (&o == this)
00173             return *this;

```

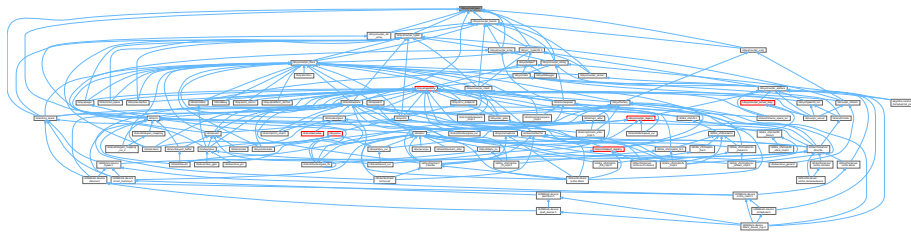
```

00174
00175     IMPL::free(*this);
00176     this->IMPL::operator = (static_cast<IMPL const &>(o));
00177     this->_c = this->IMPL::copy(o).cap();
00178     return *this;
00179 }
00180
00181 template<typename O>
00182 Shared_cap_impl &operator = (Shared_cap_impl<O, IMPL> const &o) noexcept
00183 {
00184     Cap<T>::template check_convertible_from<O>();
00185     IMPL::free(*this);
00186     this->IMPL::operator = (static_cast<IMPL const &>(o));
00187     this->_c = this->IMPL::copy(o).cap();
00188     return *this;
00189 }
00190 };
00191
00192 }} // L4::Detail

```

## 16.473 l4/sys/cxx/types File Reference

This graph shows which files directly or indirectly include this file:



### Data Structures

- class [L4::Types::Flags< BITS\\_ENUM, UNDERLYING >](#)  
*Template for defining typical [Flags](#) bitmaps.*
- struct [L4::Types::Int\\_for\\_type< T >](#)  
*Metafunction to get an integral type of the same size as *T*.*
- struct [L4::Types::Flags\\_ops\\_t< DT >](#)  
*Mixin class to define a set of friend bitwise operators on *DT*.*
- struct [L4::Types::Flags\\_t< DT, T >](#)  
*Template type to define a flags type with bitwise operations.*
- struct [L4::Types::Bool< V >](#)  
*Boolean meta type.*
- struct [L4::Types::False](#)  
*[False](#) meta value.*
- struct [L4::Types::True](#)  
*[True](#) meta value.*
- struct [L4::Types::Same< A, B >](#)  
*Compare two data types for equality.*

### Namespaces

- namespace [L4](#)  
*[L4](#) low-level kernel interface.*
- namespace [L4::Types](#)  
*[L4](#) basic type helpers for C++.*

## Macros

- `#define L4_TYPES_FLAGS_OPS_DEF(T)`  
*Helper macro to define a set of bitwise operators on an enum type.*

## 16.473.1 Macro Definition Documentation

### 16.473.1.1 L4\_TYPES\_FLAGS\_OPS\_DEF

```
#define L4_TYPES_FLAGS_OPS_DEF(  
    T )
```

#### Value:

```
friend constexpr T operator ~ (T f)
{
    return T(~static_cast<typename L4::Types::Int_for_type<T>::type>(f));
}

friend constexpr T operator | (T l, T r)
{
    return T(static_cast<typename L4::Types::Int_for_type<T>::type>(l)
        | static_cast<typename L4::Types::Int_for_type<T>::type>(r));
}

friend constexpr T operator & (T l, T r)
{
    return T(static_cast<typename L4::Types::Int_for_type<T>::type>(l)
        & static_cast<typename L4::Types::Int_for_type<T>::type>(r));
}
```

Helper macro to define a set of bitwise operators on an enum type.

This allows to use the enum type as bitmask type with '&', '|', and '~' operators that keep the enum type as result.

Definition at line 206 of file [types](#).

## 16.474 types

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=c++: -- Mode: C++ --
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018
00020
00021 #pragma once
00022
00023 // very simple type traits for basic L4 functions, for a more complete set
00024 // use <l4/cxx/type_traits> or the standard <type_traits>.
00025
00026 namespace L4 {
00027
00031 namespace Types {
00032
00062     template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
```

```

00063 class Flags
00064 {
00065 public:
00067     typedef UNDERLYING value_type;
00069     typedef BITS_ENUM bits_enum_type;
00071     typedef Flags<BITS_ENUM, UNDERLYING> type;
00072
00073 private:
00074     value_type _v;
00075     explicit Flags(value_type v) : _v(v) {}
00076
00077 public:
00079     enum None_type { None };
00080
00089     Flags(None_type) : _v(0) {}
00090
00092     Flags() : _v(0) {}
00093
00102     Flags(BITS_ENUM e) : _v((value_type{1}) « e) {}
00103
00109     static type from_raw(value_type v) { return type(v); }
00110
00112     explicit operator bool () const
00113     { return _v != 0; }
00114
00116     bool operator ! () const { return _v == 0; }
00117
00119     friend type operator | (type lhs, type rhs)
00120     { return type(lhs._v | rhs._v); }
00121
00123     friend type operator | (type lhs, bits_enum_type rhs)
00124     { return lhs | type(rhs); }
00125
00127     friend type operator & (type lhs, type rhs)
00128     { return type(lhs._v & rhs._v); }
00129
00131     friend type operator & (type lhs, bits_enum_type rhs)
00132     { return lhs & type(rhs); }
00133
00135     type &operator |= (type rhs) { _v |= rhs._v; return *this; }
00137     type &operator |= (bits_enum_type rhs) { return operator |= (type(rhs)); }
00138
00140     type &operator &= (type rhs) { _v &= rhs._v; return *this; }
00142     type &operator &= (bits_enum_type rhs) { return operator &= (type(rhs)); }
00143
00145     type operator ~ () const { return type(~_v); }
00146
00153     type &clear(bits_enum_type flag) { return operator &= (~type(flag)); }
00154
00156     value_type as_value() const { return _v; }
00157 };
00158
00164 template<unsigned SIZE, bool = true> struct Int_for_size;
00165
00166 template<> struct Int_for_size<sizeof(unsigned char), true>
00167 { typedef unsigned char type; };
00168
00169 template<> struct Int_for_size<sizeof(unsigned short),
00170                             (sizeof(unsigned short) > sizeof(unsigned char))>
00171 { typedef unsigned short type; };
00172
00173 template<> struct Int_for_size<sizeof(unsigned),
00174                             (sizeof(unsigned) > sizeof(unsigned short))>
00175 { typedef unsigned type; };
00176
00177 template<> struct Int_for_size<sizeof(unsigned long),
00178                             (sizeof(unsigned long) > sizeof(unsigned))>
00179 { typedef unsigned long type; };
00180
00181 template<> struct Int_for_size<sizeof(unsigned long long),
00182                             (sizeof(unsigned long long) > sizeof(unsigned long))>
00183 { typedef unsigned long long type; };
00184
00191 template<typename T> struct Int_for_type
00192 {
00196     typedef typename Int_for_size<sizeof(T)>::type type;
00197 };
00198
00206 #define L4_TYPES_FLAGS_OPS_DEF(T)
00207     friend constexpr T operator ~ (T f)
00208     {
00209         return T(~static_cast<typename L4::Types::Int_for_type<T>::type>(f));
00210     }
00211
00212     friend constexpr T operator | (T l, T r)
00213     {
00214         return T(static_cast<typename L4::Types::Int_for_type<T>::type>(l)

```

```

00215         | static_cast<typename L4::Types::Int_for_type<T>::type>(r)); \
00216     } \
00217 \
00218     friend constexpr T operator & (T l, T r) \
00219     { \
00220         return T(static_cast<typename L4::Types::Int_for_type<T>::type>(l) \
00221             & static_cast<typename L4::Types::Int_for_type<T>::type>(r)); \
00222     } \
00223 \
00230 template<typename DT>
00231 struct Flags_ops_t
00232 {
00234     friend constexpr DT operator | (DT l, DT r)
00235     { return DT(l.raw | r.raw); }
00236 \
00238     friend constexpr DT operator & (DT l, DT r)
00239     { return DT(l.raw & r.raw); }
00240 \
00242     friend constexpr bool operator == (DT l, DT r)
00243     { return l.raw == r.raw; }
00244 \
00246     friend constexpr bool operator != (DT l, DT r)
00247     { return l.raw != r.raw; }
00248 \
00250     DT operator |= (DT r)
00251     {
00252         static_cast<DT *>(this)->raw |= r.raw;
00253         return *static_cast<DT *>(this);
00254     }
00255 \
00257     DT operator &= (DT r)
00258     {
00259         static_cast<DT *>(this)->raw &= r.raw;
00260         return *static_cast<DT *>(this);
00261     }
00262 \
00264     explicit constexpr operator bool () const
00265     {
00266         return static_cast<DT const *>(this)->raw != 0;
00267     }
00268 \
00270     constexpr DT operator ~ () const
00271     { return DT(~static_cast<DT const *>(this)->raw); }
00272 };
00273 \
00282 template<typename DT, typename T>
00283 struct Flags_t : Flags_ops_t<Flags_t<DT, T>
00284 {
00286     T raw;
00288     Flags_t() = default;
00290     explicit constexpr Flags_t(T f) : raw(f) {}
00291 };
00292 \
00293 \
00299 template< bool V > struct Bool
00300 {
00301     typedef Bool<V> type;
00302     enum { value = V };
00303 };
00304 \
00307 struct False : Bool<false> {};
00308 \
00311 struct True : Bool<true> {};
00312 \
00313 /*****/
00322 template<typename A, typename B>
00323 struct Same : False {};
00324 \
00325 template<typename A>
00326 struct Same<A, A> : True {};
00327 \
00328 template<bool EXP, typename T = void> struct Enable_if {};
00329 template<typename T> struct Enable_if<true, T> { typedef T type; };
00330 \
00331 template<typename T1, typename T2, typename T = void>
00332 struct Enable_if_same : Enable_if<Same<T1, T2>::value, T> {};
00333 \
00334 template<typename T> struct Remove_const { typedef T type; };
00335 template<typename T> struct Remove_const<T const> { typedef T type; };
00336 template<typename T> struct Remove_volatile { typedef T type; };
00337 template<typename T> struct Remove_volatile<T volatile> { typedef T type; };
00338 template<typename T> struct Remove_cv
00339 { typedef typename Remove_const<typename Remove_volatile<T>::type>::type type; };
00340 \
00341 template<typename T> struct Remove_pointer { typedef T type; };
00342 template<typename T> struct Remove_pointer<T*> { typedef T type; };
00343 template<typename T> struct Remove_reference { typedef T type; };

```

```

00344  template<typename T> struct Remove_reference<T&> { typedef T type; };
00345  template<typename T> struct Remove_pr { typedef T type; };
00346  template<typename T> struct Remove_pr<T&> { typedef T type; };
00347  template<typename T> struct Remove_pr<T*> { typedef T type; };
00348 } // Types
00349 } // L4

```

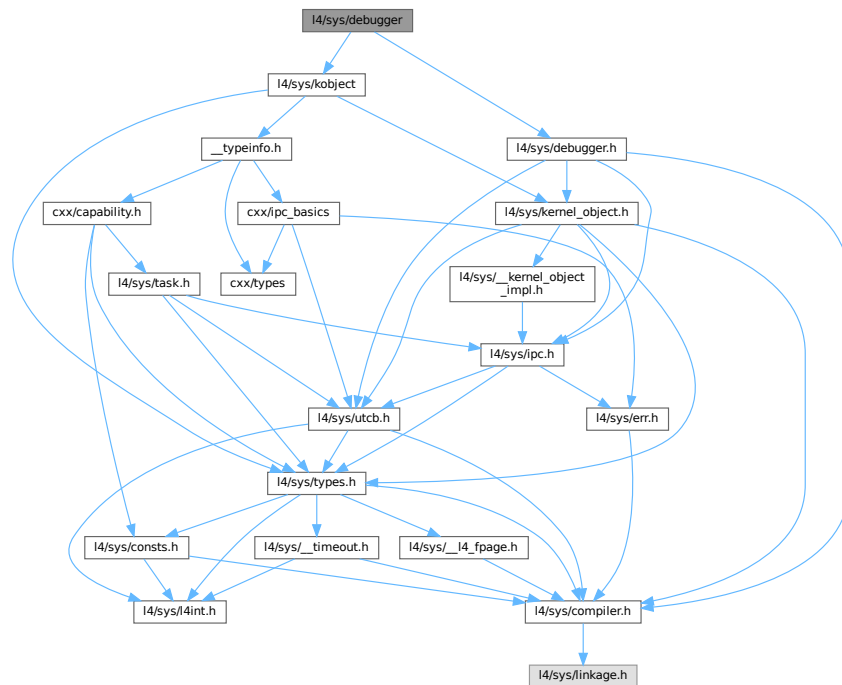
## 16.475 I4/sys/debugger File Reference

The debugger interface specifies common debugging related definitions.

```
#include <l4/sys/debugger.h>
```

```
#include <l4/sys/kobject>
```

Include dependency graph for debugger:



### Data Structures

- class [L4::Debugger](#)  
*C++ kernel debugger API.*

### Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

## 16.475.1 Detailed Description

The debugger interface specifies common debugging related definitions.

Definition in file [debugger](#).

## 16.476 debugger

[Go to the documentation of this file.](#)

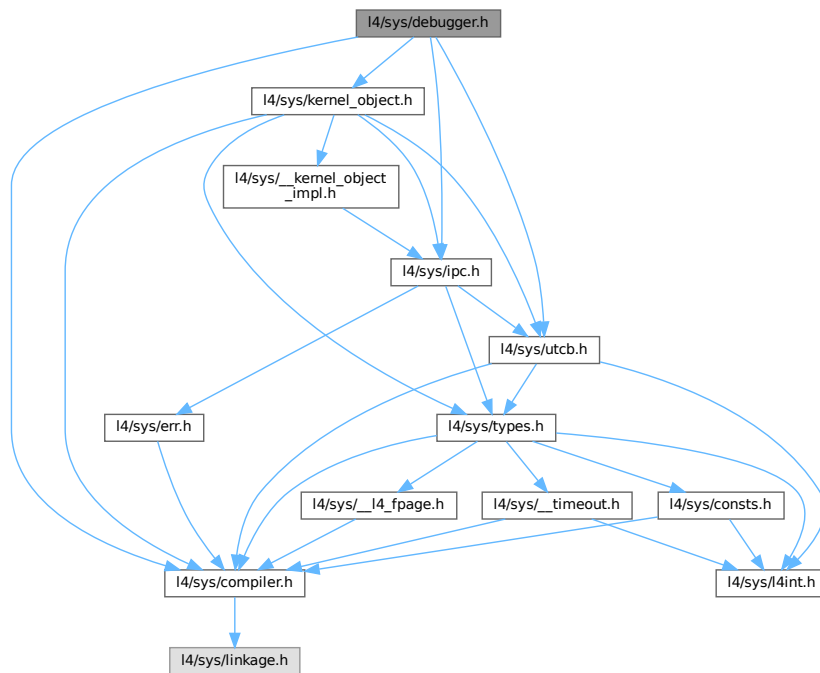
```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2010-2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/sys/debugger.h>
00027 #include <l4/sys/kobject>
00028
00029 namespace L4 {
00030
00053 class Debugger : public Kobject_t<Debugger, Kobject, L4_PROTO_DEBUGGER>
00054 {
00055 public:
00056     enum
00057     {
00058         Switch_log_on  = L4_DEBUGGER_SWITCH_LOG_ON,
00059         Switch_log_off = L4_DEBUGGER_SWITCH_LOG_OFF,
00060     };
00061
00070     l4_msgtag_t set_object_name(const char *name,
00071                                l4_utcb_t *utcb = l4_utcb()) noexcept
00072     { return l4_debugger_set_object_name_u(cap(), name, utcb); }
00073
00082     unsigned long global_id(l4_utcb_t *utcb = l4_utcb()) noexcept
00083     { return l4_debugger_global_id_u(cap(), utcb); }
00084
00094     unsigned long kobj_to_id(l4_addr_t kobjp,
00095                              l4_utcb_t *utcb = l4_utcb()) noexcept
00096     { return l4_debugger_kobj_to_id_u(cap(), kobjp, utcb); }
00097
00108     long query_log_typeid(const char *name, unsigned idx,
00109                           l4_utcb_t *utcb = l4_utcb()) noexcept
00110     { return l4_debugger_query_log_typeid_u(cap(), name, idx, utcb); }
00111
00127     long query_log_name(unsigned idx,
00128                          char *name, unsigned namelen,
00129                          char *shortname, unsigned shortnamelen,
00130                          l4_utcb_t *utcb = l4_utcb()) noexcept
00131     {
00132         return l4_debugger_query_log_name_u(cap(), idx, name, namelen,
00133                                             shortname, shortnamelen, utcb);
00134     }
00135
00144     l4_msgtag_t switch_log(const char *name, unsigned on_off,
00145                             l4_utcb_t *utcb = l4_utcb()) noexcept
00146     { return l4_debugger_switch_log_u(cap(), name, on_off, utcb); }
00147
00159     l4_msgtag_t get_object_name(unsigned id, char *name, unsigned size,
00160                                 l4_utcb_t *utcb = l4_utcb()) noexcept
00161     { return l4_debugger_get_object_name_u(cap(), id, name, size, utcb); }
00162
00172     l4_msgtag_t add_image_info(l4_addr_t base, const char *name,
00173                                l4_utcb_t *utcb = l4_utcb()) noexcept
00174     { return l4_debugger_add_image_info_u(cap(), base, name, utcb); }
00175 };
00176 }
```



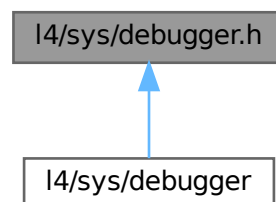
## 16.477 l4/sys/debugger.h File Reference

Debugger related definitions.

```
#include <l4/sys/compiler.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
#include <l4/sys/kernel_object.h>
Include dependency graph for debugger.h:
```



This graph shows which files directly or indirectly include this file:



### Functions

- [l4\\_msgtag\\_t l4\\_debugger\\_set\\_object\\_name](#) ([l4\\_cap\\_idx\\_t](#) cap, const char \*name) [L4\\_NOTHROW](#)

*Set the name of a kernel object.*

- [l4\\_msgtag\\_t l4\\_debugger\\_get\\_object\\_name](#) ([l4\\_cap\\_idx\\_t](#) cap, unsigned id, char \*name, unsigned size) [L4\\_NOTHROW](#)

*Get name of the kernel object with id id.*

- unsigned long [l4\\_debugger\\_global\\_id](#) ([l4\\_cap\\_idx\\_t](#) cap) [L4\\_NOTHROW](#)

*Get the globally unique ID of the object behind a capability.*

- unsigned long [l4\\_debugger\\_kobj\\_to\\_id](#) ([l4\\_cap\\_idx\\_t](#) cap, [l4\\_addr\\_t](#) kobjp) [L4\\_NOTHROW](#)

*Get the globally unique ID of the object behind the kobject pointer.*

- long [l4\\_debugger\\_query\\_log\\_typeid](#) ([l4\\_cap\\_idx\\_t](#) cap, const char \*name, unsigned idx) [L4\\_NOTHROW](#)

*Query the log-id for a log type.*

- long [l4\\_debugger\\_query\\_log\\_name](#) ([l4\\_cap\\_idx\\_t](#) cap, unsigned idx, char \*name, unsigned namelen, char \*shortname, unsigned shortnamelen) [L4\\_NOTHROW](#)

*Query the name of a log type given the ID.*

- [l4\\_msgtag\\_t l4\\_debugger\\_switch\\_log](#) ([l4\\_cap\\_idx\\_t](#) cap, const char \*name, int on\_off) [L4\\_NOTHROW](#)

*Set or unset log.*

- [l4\\_msgtag\\_t l4\\_debugger\\_add\\_image\\_info](#) ([l4\\_cap\\_idx\\_t](#) cap, [l4\\_addr\\_t](#) base, const char \*name) [L4\\_NOTHROW](#)

*Add loaded image information for a task.*

## 16.477.1 Detailed Description

Debugger related definitions.

Definition in file [debugger.h](#).

## 16.478 debugger.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00007 /*
00008  * (c) 2008-2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025
00026 #include <l4/sys/compiler.h>
00027 #include <l4/sys/utcb.h>
00028 #include <l4/sys/ipc.h>
00029
00053 L4_INLINE l4_msgtag_t
00054 l4_debugger_set_object_name(l4_cap_idx_t cap, const char *name) L4_NOTHROW;
00055
00059 L4_INLINE l4_msgtag_t
00060 l4_debugger_set_object_name_u(l4_cap_idx_t cap, const char *name, l4_utcb_t *utcb) L4_NOTHROW;
00061
00074 L4_INLINE l4_msgtag_t
00075 l4_debugger_get_object_name(l4_cap_idx_t cap, unsigned id,
00076                             char *name, unsigned size) L4_NOTHROW;
00077
00081 L4_INLINE l4_msgtag_t
00082 l4_debugger_get_object_name_u(l4_cap_idx_t cap, unsigned id,
```

```

00083         char *name, unsigned size,
00084         l4_utcb_t *utcb) L4_NOTHROW;
00085
00097 L4_INLINE unsigned long
00098 l4_debugger_global_id(l4_cap_idx_t cap) L4_NOTHROW;
00099
00103 L4_INLINE unsigned long
00104 l4_debugger_global_id_u(l4_cap_idx_t cap, l4_utcb_t *utcb) L4_NOTHROW;
00105
00118 L4_INLINE unsigned long
00119 l4_debugger_kobj_to_id(l4_cap_idx_t cap, l4_addr_t kobjp) L4_NOTHROW;
00120
00124 L4_INLINE unsigned long
00125 l4_debugger_kobj_to_id_u(l4_cap_idx_t cap, l4_addr_t kobjp, l4_utcb_t *utcb) L4_NOTHROW;
00126
00139 L4_INLINE long
00140 l4_debugger_query_log_typeid(l4_cap_idx_t cap, const char *name,
00141                             unsigned idx) L4_NOTHROW;
00142
00146 L4_INLINE long
00147 l4_debugger_query_log_typeid_u(l4_cap_idx_t cap, const char *name,
00148                               unsigned idx, l4_utcb_t *utcb) L4_NOTHROW;
00149
00166 L4_INLINE long
00167 l4_debugger_query_log_name(l4_cap_idx_t cap, unsigned idx,
00168                           char *name, unsigned namelen,
00169                           char *shortname, unsigned shortnamelen) L4_NOTHROW;
00170
00174 L4_INLINE long
00175 l4_debugger_query_log_name_u(l4_cap_idx_t cap, unsigned idx,
00176                             char *name, unsigned namelen,
00177                             char *shortname, unsigned shortnamelen,
00178                             l4_utcb_t *utcb) L4_NOTHROW;
00179
00190 L4_INLINE l4_msgtag_t
00191 l4_debugger_switch_log(l4_cap_idx_t cap, const char *name,
00192                       int on_off) L4_NOTHROW;
00193
00197 L4_INLINE l4_msgtag_t
00198 l4_debugger_switch_log_u(l4_cap_idx_t cap, const char *name, int on_off,
00199                          l4_utcb_t *utcb) L4_NOTHROW;
00200
00211 L4_INLINE l4_msgtag_t
00212 l4_debugger_add_image_info(l4_cap_idx_t cap, l4_addr_t base,
00213                           const char *name) L4_NOTHROW;
00214
00218 L4_INLINE l4_msgtag_t
00219 l4_debugger_add_image_info_u(l4_cap_idx_t cap, l4_addr_t base, const char *name,
00220                             l4_utcb_t *utcb) L4_NOTHROW;
00221
00222 enum
00223 {
00224     L4_DEBUGGER_NAME_SET_OP          = 0UL,
00225     L4_DEBUGGER_GLOBAL_ID_OP        = 1UL,
00226     L4_DEBUGGER_KOBJ_TO_ID_OP       = 2UL,
00227     L4_DEBUGGER_QUERY_LOG_TYPEID_OP = 3UL,
00228     L4_DEBUGGER_SWITCH_LOG_OP       = 4UL,
00229     L4_DEBUGGER_NAME_GET_OP         = 5UL,
00230     L4_DEBUGGER_QUERY_LOG_NAME_OP   = 6UL,
00231     L4_DEBUGGER_ADD_IMAGE_INFO_OP   = 7UL,
00232 };
00233
00234 enum
00235 {
00236     L4_DEBUGGER_SWITCH_LOG_ON  = 1,
00237     L4_DEBUGGER_SWITCH_LOG_OFF = 0,
00238 };
00239
00240 /* IMPLEMENTATION ----- */
00241
00242 #include <l4/sys/kernel_object.h>
00243
00257 L4_INLINE unsigned
00258 __strcpy_maxlen(char *dst, char const *src, unsigned maxlen)
00259 {
00260     unsigned i;
00261     if (!maxlen)
00262         return 0;
00263
00264     for (i = 0; i < maxlen - 1 && src[i]; ++i)
00265         dst[i] = src[i];
00266     dst[i] = '\0';
00267
00268     return i + 1;
00269 }
00270
00271 L4_INLINE l4_msgtag_t

```

```

00272 l4_debugger_set_object_name_u(l4_cap_idx_t cap,
00273                               const char *name, l4_utcb_t *utcb) L4_NOTHROW
00274 {
00275     unsigned i;
00276     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_NAME_SET_OP;
00277     i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[1], name,
00278                        (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(l4_umword_t));
00279     i = l4_bytes_to_mwords(i);
00280     return l4_invoke_debugger(cap, l4_msgtag(0, 1 + i, 0, 0), utcb);
00281 }
00282
00283 L4_INLINE unsigned long
00284 l4_debugger_global_id_u(l4_cap_idx_t cap, l4_utcb_t *utcb) L4_NOTHROW
00285 {
00286     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_GLOBAL_ID_OP;
00287     if (l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 1, 0, 0), utcb), utcb))
00288         return ~0UL;
00289     return l4_utcb_mr_u(utcb)->mr[0];
00290 }
00291
00292 L4_INLINE unsigned long
00293 l4_debugger_kobj_to_id_u(l4_cap_idx_t cap, l4_addr_t kobjp, l4_utcb_t *utcb) L4_NOTHROW
00294 {
00295     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_KOBJ_TO_ID_OP;
00296     l4_utcb_mr_u(utcb)->mr[1] = kobjp;
00297     if (l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb), utcb))
00298         return ~0UL;
00299     return l4_utcb_mr_u(utcb)->mr[0];
00300 }
00301
00302 L4_INLINE long
00303 l4_debugger_query_log_typeid_u(l4_cap_idx_t cap, const char *name,
00304                               unsigned idx,
00305                               l4_utcb_t *utcb) L4_NOTHROW
00306 {
00307     unsigned i;
00308     long e;
00309     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_QUERY_LOG_TYPEID_OP;
00310     l4_utcb_mr_u(utcb)->mr[1] = idx;
00311     i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[2], name, 32);
00312     i = l4_bytes_to_mwords(i);
00313     e = l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2 + i, 0, 0), utcb), utcb);
00314     if (e < 0)
00315         return e;
00316     return l4_utcb_mr_u(utcb)->mr[0];
00317 }
00318
00319 L4_INLINE long
00320 l4_debugger_query_log_name_u(l4_cap_idx_t cap, unsigned idx,
00321                             char *name, unsigned namelen,
00322                             char *shortname, unsigned shortnamelen,
00323                             l4_utcb_t *utcb) L4_NOTHROW
00324 {
00325     long e;
00326     char const *n;
00327     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_QUERY_LOG_NAME_OP;
00328     l4_utcb_mr_u(utcb)->mr[1] = idx;
00329     e = l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb), utcb);
00330     if (e < 0)
00331         return e;
00332     n = (char const *)&l4_utcb_mr_u(utcb)->mr[0];
00333     __strcpy_maxlen(name, n, namelen);
00334     __strcpy_maxlen(shortname, n + __builtin_strlen(n) + 1, shortnamelen);
00335     return 0;
00336 }
00337
00338
00339 L4_INLINE l4_msgtag_t
00340 l4_debugger_switch_log_u(l4_cap_idx_t cap, const char *name, int on_off,
00341                          l4_utcb_t *utcb) L4_NOTHROW
00342 {
00343     unsigned i;
00344     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_SWITCH_LOG_OP;
00345     l4_utcb_mr_u(utcb)->mr[1] = on_off;
00346     i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[2], name, 32);
00347     i = l4_bytes_to_mwords(i);
00348     return l4_invoke_debugger(cap, l4_msgtag(0, 2 + i, 0, 0), utcb);
00349 }
00350
00351 L4_INLINE l4_msgtag_t
00352 l4_debugger_get_object_name_u(l4_cap_idx_t cap, unsigned id,
00353                              char *name, unsigned size,
00354                              l4_utcb_t *utcb) L4_NOTHROW
00355 {
00356     l4_msgtag_t t;
00357     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_NAME_GET_OP;
00358     l4_utcb_mr_u(utcb)->mr[1] = id;

```

```

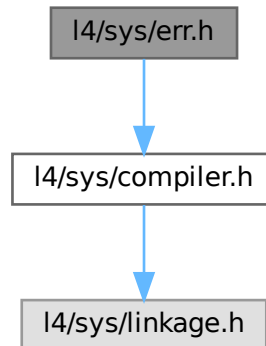
00359     t = l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb);
00360     __strcpy_maxlen(name, (char const *)&l4_utcb_mr_u(utcb)->mr[0], size);
00361     return t;
00362 }
00363
00364 L4_INLINE l4_msgtag_t
00365 l4_debugger_add_image_info_u(l4_cap_idx_t cap, l4_addr_t base,
00366                             const char *name, l4_utcb_t *utcb) L4_NOTHROW
00367 {
00368     unsigned i;
00369     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_ADD_IMAGE_INFO_OP;
00370     l4_utcb_mr_u(utcb)->mr[1] = base;
00371     i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[2], name,
00372                        (L4_UTCB_GENERIC_DATA_SIZE - 3) * sizeof(l4_umword_t));
00373     i = l4_bytes_to_mwords(i);
00374     return l4_invoke_debugger(cap, l4_msgtag(0, 2 + i, 0, 0), utcb);
00375 }
00376
00377
00378 L4_INLINE l4_msgtag_t
00379 l4_debugger_set_object_name(l4_cap_idx_t cap,
00380                             const char *name) L4_NOTHROW
00381 {
00382     return l4_debugger_set_object_name_u(cap, name, l4_utcb());
00383 }
00384
00385 L4_INLINE unsigned long
00386 l4_debugger_global_id(l4_cap_idx_t cap) L4_NOTHROW
00387 {
00388     return l4_debugger_global_id_u(cap, l4_utcb());
00389 }
00390
00391 L4_INLINE unsigned long
00392 l4_debugger_kobj_to_id(l4_cap_idx_t cap, l4_addr_t kobjp) L4_NOTHROW
00393 {
00394     return l4_debugger_kobj_to_id_u(cap, kobjp, l4_utcb());
00395 }
00396
00397 L4_INLINE long
00398 l4_debugger_query_log_typeid(l4_cap_idx_t cap, const char *name,
00399                             unsigned idx) L4_NOTHROW
00400 {
00401     return l4_debugger_query_log_typeid_u(cap, name, idx, l4_utcb());
00402 }
00403
00404 L4_INLINE long
00405 l4_debugger_query_log_name(l4_cap_idx_t cap, unsigned idx,
00406                           char *name, unsigned namelen,
00407                           char *shortname, unsigned shortnamelen) L4_NOTHROW
00408 {
00409     return l4_debugger_query_log_name_u(cap, idx, name, namelen,
00410                                         shortname, shortnamelen, l4_utcb());
00411 }
00412
00413 L4_INLINE l4_msgtag_t
00414 l4_debugger_switch_log(l4_cap_idx_t cap, const char *name,
00415                        int on_off) L4_NOTHROW
00416 {
00417     return l4_debugger_switch_log_u(cap, name, on_off, l4_utcb());
00418 }
00419
00420 L4_INLINE l4_msgtag_t
00421 l4_debugger_get_object_name(l4_cap_idx_t cap, unsigned id,
00422                             char *name, unsigned size) L4_NOTHROW
00423 {
00424     return l4_debugger_get_object_name_u(cap, id, name, size, l4_utcb());
00425 }
00426
00427 L4_INLINE l4_msgtag_t
00428 l4_debugger_add_image_info(l4_cap_idx_t cap, l4_addr_t base,
00429                             const char *name) L4_NOTHROW
00430 {
00431     return l4_debugger_add_image_info_u(cap, base, name, l4_utcb());
00432 }

```

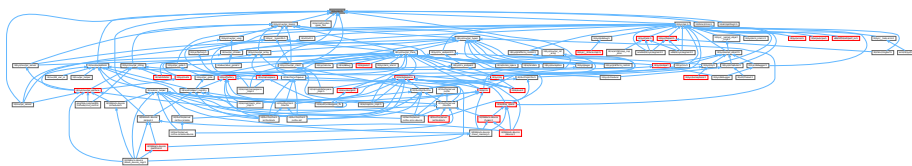
## 16.479 l4/sys/err.h File Reference

Error codes.

```
#include <l4/sys/compiler.h>
Include dependency graph for err.h:
```



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum [l4\\_error\\_code\\_t](#) {  
[L4\\_EOK](#) = 0 , [L4\\_EPERM](#) = 1 , [L4\\_ENOENT](#) = 2 , [L4\\_EIO](#) = 5 ,  
[L4\\_ENXIO](#) = 6 , [L4\\_E2BIG](#) = 7 , [L4\\_EAGAIN](#) = 11 , [L4\\_ENOMEM](#) = 12 ,  
[L4\\_EACCESS](#) = 13 , [L4\\_EFAULT](#) = 14 , [L4\\_EBUSY](#) = 16 , [L4\\_EEXIST](#) = 17 ,  
[L4\\_ENODEV](#) = 19 , [L4\\_ENOTDIR](#) = 20 , [L4\\_EINVAL](#) = 22 , [L4\\_ENOSPC](#) = 28 ,  
[L4\\_ERANGE](#) = 34 , [L4\\_ENAMETOOLONG](#) = 36 , [L4\\_ENOSYS](#) = 38 , [L4\\_EBADPROTO](#) = 39 ,  
[L4\\_EADDRNOTAVAIL](#) = 99 , [L4\\_ERRNOMAX](#) = 100 , [L4\\_ENOREPLY](#) = 1000 , [L4\\_MSGTOOSHORT](#) =  
1001 ,  
[L4\\_MSGTOOLONG](#) = 1002 , [L4\\_MSGMISSARG](#) = 1003 , [L4\\_EIPC\\_LO](#) = 2000 , [L4\\_EIPC\\_HI](#) = 2000 +  
0x1f }  
[L4](#) error codes.

### 16.479.1 Detailed Description

Error codes.

Definition in file [err.h](#).

## 16.480 err.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 #include <l4/sys/compiler.h>
00026
00041 enum l4_error_code_t
00042 {
00043     L4_EOK                = 0,
00044     L4_EPERM              = 1,
00045     L4_ENOENT              = 2,
00046     L4_EIO                 = 5,
00047     L4_ENXIO               = 6,
00048     L4_E2BIG               = 7,
00049     L4_EAGAIN              = 11,
00050     L4_ENOMEM              = 12,
00051     L4_EACCESS             = 13,
00052     L4_EFAULT              = 14,
00053     L4_EBUSY               = 16,
00054     L4_EEXIST              = 17,
00055     L4_ENODEV              = 19,
00056     L4_ENOTDIR             = 20,
00057     L4_EINVAL              = 22,
00058     L4_ENOSPC              = 28,
00059     L4_ERANGE              = 34,
00060     L4_ENAMETOOLONG        = 36,
00061     L4_ENOSYS              = 38,
00062     L4_EBADPROTO           = 39,
00063     L4_EADDRNOTAVAIL       = 99,
00064     L4_ERRNOMAX            = 100,
00066     L4_ENOREPLY            = 1000,
00067     L4_MSGTOOSHORT         = 1001,
00068     L4_MSGTOOLONG          = 1002,
00069     L4_MSGMISSARG          = 1003,
00071     L4_EIPC_LO              = 2000,
00072     L4_EIPC_HI             = 2000 + 0x1f,
00073 };
00074
00075 __BEGIN_DECLS
00076 L4_CV char const *l4sys_errtostr(long err) L4_NOTHROW;
00077 __END_DECLS
00078
00079

```

## 16.481 l4/sys/exception File Reference

Exception C++ interface.

```

#include <l4/sys/capability>
#include <l4/sys/types.h>
#include <l4/sys/cxx/ipc_types>

```





## 16.482 exception

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00025 #include <l4/sys/capability>
00026 #include <l4/sys/types.h>
00027 #include <l4/sys/cxx/ipc_types>
00028 #include <l4/sys/cxx/ipc_iface>
00029
00030 namespace L4 {
00031
00042 class L4_EXPORT Exception :
00043     public Kobject_0t<Exception, L4_PROTO_EXCEPTION>
00044 {
00045 public:
00046     // TODO: pass a reference/pointer to the UTCB not copy the regs
00047     L4_INLINE_RPC(
00048         l4_msgtag_t, exception, (L4::Ipc::In_out<l4_exc_regs_t *> regs,
00049                                 L4::Ipc::Rcv_fpage rwin,
00050                                 L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp));
00051
00052     typedef L4::Typeid::Rpc_nocode<exception_t> Rpcs;
00053 };
00054
00055 }
```

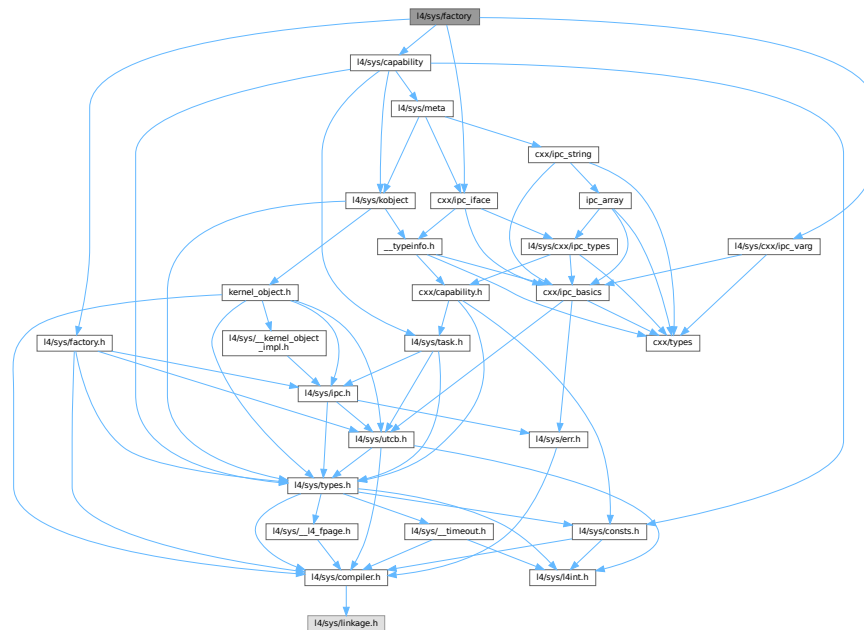
## 16.483 l4/sys/factory File Reference

Common factory related definitions.

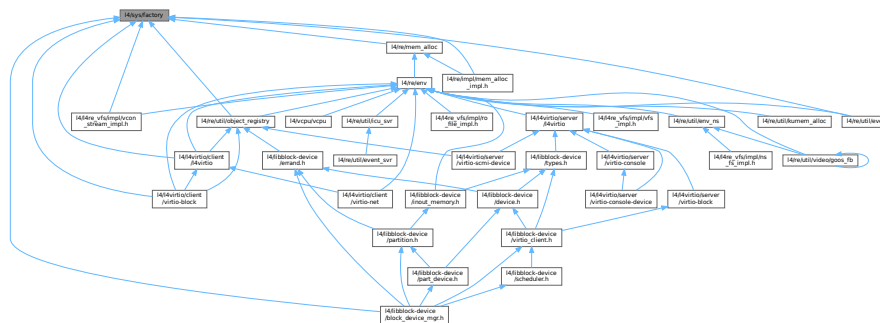
```

#include <l4/sys/factory.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_varg>
```

Include dependency graph for factory:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4::Factory`  
*C++ Factory interface, see `Factory` for the C interface.*
- struct `L4::Factory::Nil`  
*Special type to add a void argument into the factory create stream.*
- struct `L4::Factory::Lstr`  
*Special type to add a pascal string into the factory create stream.*
- class `L4::Factory::S`  
*Stream class for the `create()` argument stream.*

## Namespaces

- namespace L4  
*L4 low-level kernel interface.*

## 16.483.1 Detailed Description

Common factory related definitions.

Definition in file [factory](#).

## 16.484 factory

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include <l4/sys/factory.h>
00028 #include <l4/sys/capability>
00029 #include <l4/sys/cxx/ipc_iface>
00030 #include <l4/sys/cxx/ipc_varg>
00031
00032 namespace L4 {
00033
00048 class Factory : public Kobject_t<Factory, Kobject, L4_PROTO_FACTORY>
00049 {
00050 public:
00051
00052     typedef l4_mword_t Proto;
00053
00057     struct Nil {};
00058
00064     struct Lstr
00065     {
00069         char const *s;
00070
00074         unsigned len;
00075
00080         Lstr(char const *s, unsigned len) noexcept : s(s), len(len) {}
00081     };
00082
00089     class S
00090     {
00091 private:
00092         l4_utcb_t *u;
00093         l4_msgtag_t t;
00094         l4_cap_idx_t f;
00095
00096         template<typename T>
00097         static T &&_move(T &c) { return static_cast<T &&>(c); }
00098
00099 public:
00100         S(S const &) = delete;
00101         S &operator = (S const &) &= delete;
00102
00108         S(S &&o) noexcept
00109         : u(o.u), t(o.t), f(o.f)
00110         { o.t.raw = 0; }
00111
00112         S &operator = (S &&o) & noexcept
00113         {
00114             u = o.u;
00115             t = o.t;
00116             f = o.f;
```

```

00117         o.t.raw = 0;
00118         return *this;
00119     }
00120
00132     S(l4_cap_idx_t f, long obj, L4::Cap<void> target,
00133        l4_utcb_t *utcb) noexcept
00134     : u(utcb), t(l4_factory_create_start_u(obj, target.cap(), u)), f(f)
00135     {}
00136
00141     ~S() noexcept
00142     {
00143         if (t.raw)
00144             l4_factory_create_commit_u(f, t, u);
00145     }
00146
00158     operator l4_msgtag_t () noexcept
00159     {
00160         l4_msgtag_t r = l4_factory_create_commit_u(f, t, u);
00161         t.raw = 0;
00162         return r;
00163     }
00164
00170     void put(l4_mword_t i) noexcept
00171     {
00172         l4_factory_create_add_int_u(i, &t, u);
00173     }
00174
00180     void put(l4_umword_t i) noexcept
00181     {
00182         l4_factory_create_add_uint_u(i, &t, u);
00183     }
00184
00192     void put(char const *s) & noexcept
00193     {
00194         l4_factory_create_add_str_u(s, &t, u);
00195     }
00196
00206     void put(Lstr const &s) & noexcept
00207     {
00208         l4_factory_create_add_lstr_u(s.s, s.len, &t, u);
00209     }
00210
00214     void put(Null) & noexcept
00215     {
00216         l4_factory_create_add_nil_u(&t, u);
00217     }
00218
00224     void put(l4_fpage_t d) & noexcept
00225     {
00226         l4_factory_create_add_fpage_u(d, &t, u);
00227     }
00228
00229     template<typename T>
00230     S &operator « (T const &d) & noexcept
00231     {
00232         put(d);
00233         return *this;
00234     }
00235
00236     template<typename T>
00237     S &&operator « (T const &d) && noexcept
00238     {
00239         put(d);
00240         return _move(*this);
00241     }
00242 };
00243
00244
00245 public:
00246
00274     S create(Cap<void> target, long obj, l4_utcb_t *utcb = l4_utcb()) noexcept
00275     {
00276         return S(cap(), obj, target, utcb);
00277     }
00278
00307     template<typename OBJ>
00308     S create(Cap<OBJ> target, l4_utcb_t *utcb = l4_utcb()) noexcept
00309     {
00310         return S(cap(), OBJ::Protocol, target, utcb);
00311     }
00312
00313     L4_INLINE_RPC_NF(
00314         l4_msgtag_t, create, (L4::Ipc::Out<L4::Cap<void> > target, l4_mword_t obj,
00315                               L4::Ipc::Varg const *args),
00316         L4::Ipc::Call_t<L4_CAP_FPAGE_S>);
00317
00348     l4_msgtag_t create_task(Cap<Task> const & target_cap,

```

```

00349         l4_fpage_t *utcb_area,
00350         l4_utcb_t *utcb = l4_utcb()) noexcept
00351 { return l4_factory_create_task_u(cap(), target_cap.cap(), utcb_area, utcb); }
00352
00381 l4_msgtag_t create_factory(Cap<Factory> const &target_cap,
00382                          unsigned long limit,
00383                          l4_utcb_t *utcb = l4_utcb()) noexcept
00384 { return l4_factory_create_factory_u(cap(), target_cap.cap(), limit, utcb); }
00385
00414 l4_msgtag_t create_gate(Cap<void> const &target_cap,
00415                       Cap<Thread> const &thread_cap, l4_umword_t label,
00416                       l4_utcb_t *utcb = l4_utcb()) noexcept
00417 { return l4_factory_create_gate_u(cap(), target_cap.cap(), thread_cap.cap(), label, utcb); }
00418
00419 typedef L4::Typeid::Rpc_nocode<create_t> Rpc;
00420 };
00421
00422 }

```

## 16.485 l4/sys/factory.h File Reference

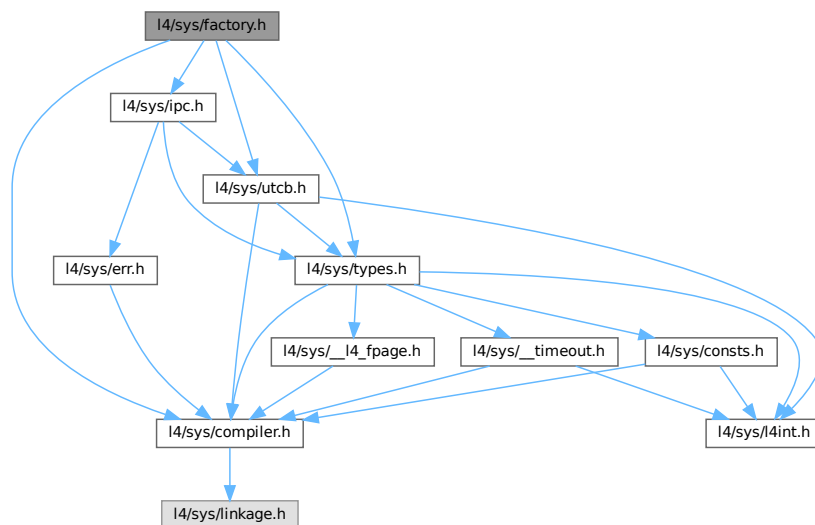
Common factory related definitions.

```

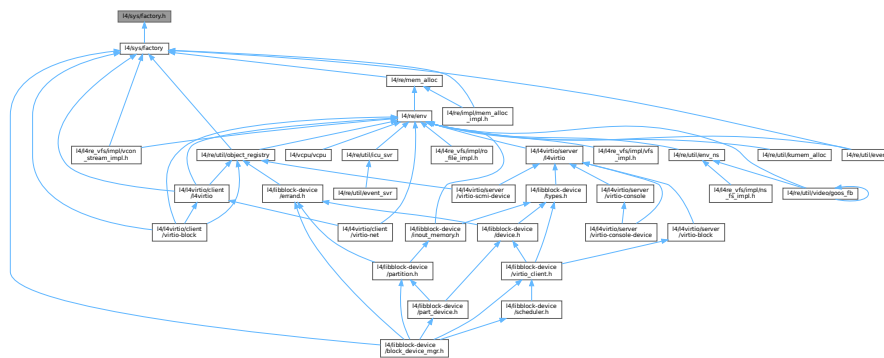
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for factory.h:



This graph shows which files directly or indirectly include this file:



## Functions

- `l4_msgtag_t l4_factory_create_task (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_fpage_t *utcb_area)`  
L4\_NOTHROW  
*Create a new task.*
- `l4_msgtag_t l4_factory_create_thread (l4_cap_idx_t factory, l4_cap_idx_t target_cap)` L4\_NOTHROW  
*Create a new thread.*
- `l4_msgtag_t l4_factory_create_factory (l4_cap_idx_t factory, l4_cap_idx_t target_cap, unsigned long limit)`  
L4\_NOTHROW  
*Create a new factory.*
- `l4_msgtag_t l4_factory_create_gate (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_cap_idx_t thread_cap, l4_umword_t label)` L4\_NOTHROW  
*Create a new IPC gate.*
- `l4_msgtag_t l4_factory_create_irq (l4_cap_idx_t factory, l4_cap_idx_t target_cap)` L4\_NOTHROW  
*Create a new IRQ sender.*
- `l4_msgtag_t l4_factory_create_vm (l4_cap_idx_t factory, l4_cap_idx_t target_cap)` L4\_NOTHROW  
*Create a new virtual machine.*
- `l4_msgtag_t l4_factory_create_vcpu_context (l4_cap_idx_t factory, l4_cap_idx_t target_cap)` L4\_NOTHROW  
*Create a new hardware vCPU context.*
- `l4_msgtag_t l4_factory_create (l4_cap_idx_t factory, long obj, l4_cap_idx_t target)` L4\_NOTHROW  
*Create a new object.*

## 16.485.1 Detailed Description

Common factory related definitions.

Definition in file [factory.h](#).

## 16.486 factory.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00010  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>,
00011  *      Henning Schild <hschild@os.inf.tu-dresden.de>
00012  *      economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * This file is part of TUD:OS and distributed under the terms of the
00015  * GNU General Public License 2.
00016  * Please see the COPYING-GPL-2 file for details.
00017  *
00018  * As a special exception, you may use this file as part of a free software
00019  * library without restriction. Specifically, if other files instantiate
00020  * templates or use macros or inline functions from this file, or you compile
00021  * this file and link it with other files to produce an executable, this
00022  * file does not by itself cause the resulting executable to be covered by
00023  * the GNU General Public License. This exception does not however
00024  * invalidate any other reasons why the executable file might be covered by
00025  * the GNU General Public License.
00026  */
00027 #pragma once
00028
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/types.h>
00031 #include <l4/sys/utcb.h>
00032
00096 L4_INLINE l4_msgtag_t
00097 l4_factory_create_task(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
00098                      l4_fpage_t *utcb_area) L4_NOTHROW;
00099
00104 L4_INLINE l4_msgtag_t
00105 l4_factory_create_task_u(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
00106                       l4_fpage_t *utcb_area, l4_utcb_t *utcb) L4_NOTHROW;
00107
00125 L4_INLINE l4_msgtag_t
00126 l4_factory_create_thread(l4_cap_idx_t factory,
00127                        l4_cap_idx_t target_cap) L4_NOTHROW;
00128
00133 L4_INLINE l4_msgtag_t
00134 l4_factory_create_thread_u(l4_cap_idx_t factory,
00135                          l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW;
00136
00161 L4_INLINE l4_msgtag_t
00162 l4_factory_create_factory(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
00163                        unsigned long limit) L4_NOTHROW;
00164
00169 L4_INLINE l4_msgtag_t
00170 l4_factory_create_factory_u(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
00171                          unsigned long limit, l4_utcb_t *utcb) L4_NOTHROW;
00172
00198 L4_INLINE l4_msgtag_t
00199 l4_factory_create_gate(l4_cap_idx_t factory,
00200                      l4_cap_idx_t target_cap,
00201                      l4_cap_idx_t thread_cap, l4_umword_t label) L4_NOTHROW;
00202
00207 L4_INLINE l4_msgtag_t
00208 l4_factory_create_gate_u(l4_cap_idx_t factory,
00209                       l4_cap_idx_t target_cap,
00210                       l4_cap_idx_t thread_cap, l4_umword_t label,
00211                       l4_utcb_t *utcb) L4_NOTHROW;
00212
00228 L4_INLINE l4_msgtag_t
00229 l4_factory_create_irq(l4_cap_idx_t factory,
00230                    l4_cap_idx_t target_cap) L4_NOTHROW;
00231
00236 L4_INLINE l4_msgtag_t
00237 l4_factory_create_irq_u(l4_cap_idx_t factory,
00238                      l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW;
00239
00257 L4_INLINE l4_msgtag_t
00258 l4_factory_create_vm(l4_cap_idx_t factory,
00259                   l4_cap_idx_t target_cap) L4_NOTHROW;
00260
00279 L4_INLINE l4_msgtag_t
00280 l4_factory_create_vcpu_context(l4_cap_idx_t factory,
00281                             l4_cap_idx_t target_cap) L4_NOTHROW;
00282
00287 L4_INLINE l4_msgtag_t
00288 l4_factory_create_vm_u(l4_cap_idx_t factory,
00289                     l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW;

```

```

00290
00295 L4_INLINE l4_msgtag_t
00296 l4_factory_create_vcpu_context_u(l4_cap_idx_t factory,
00297                                   l4_cap_idx_t target_cap,
00298                                   l4_utcb_t *utcb) L4_NOTHROW;
00299
00304 L4_INLINE l4_msgtag_t
00305 l4_factory_create_start_u(long obj, l4_cap_idx_t target,
00306                           l4_utcb_t *utcb) L4_NOTHROW;
00307
00312 L4_INLINE int
00313 l4_factory_create_add_fpage_u(l4_fpage_t d, l4_msgtag_t *tag,
00314                               l4_utcb_t *utcb) L4_NOTHROW;
00315
00320 L4_INLINE int
00321 l4_factory_create_add_int_u(l4_mword_t d, l4_msgtag_t *tag,
00322                             l4_utcb_t *utcb) L4_NOTHROW;
00323
00328 L4_INLINE int
00329 l4_factory_create_add_uint_u(l4_umword_t d, l4_msgtag_t *tag,
00330                              l4_utcb_t *utcb) L4_NOTHROW;
00331
00336 L4_INLINE int
00337 l4_factory_create_add_str_u(char const *s, l4_msgtag_t *tag,
00338                             l4_utcb_t *utcb) L4_NOTHROW;
00339
00344 L4_INLINE int
00345 l4_factory_create_add_lstr_u(char const *s, unsigned len, l4_msgtag_t *tag,
00346                              l4_utcb_t *utcb) L4_NOTHROW;
00347
00352 L4_INLINE int
00353 l4_factory_create_add_nil_u(l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW;
00354
00359 L4_INLINE l4_msgtag_t
00360 l4_factory_create_commit_u(l4_cap_idx_t factory, l4_msgtag_t tag,
00361                             l4_utcb_t *utcb) L4_NOTHROW;
00362
00367 L4_INLINE l4_msgtag_t
00368 l4_factory_create_u(l4_cap_idx_t factory, long obj, l4_cap_idx_t target,
00369                     l4_utcb_t *utcb) L4_NOTHROW;
00370
00371
00388 L4_INLINE l4_msgtag_t
00389 l4_factory_create(l4_cap_idx_t factory, long obj,
00390                  l4_cap_idx_t target) L4_NOTHROW;
00391
00392 /* IMPLEMENTATION -----*/
00393
00394 #include <l4/sys/ipc.h>
00395
00396 L4_INLINE l4_msgtag_t
00397 l4_factory_create_task_u(l4_cap_idx_t factory,
00398                          l4_cap_idx_t target_cap, l4_fpage_t *utcb_area,
00399                          l4_utcb_t *u) L4_NOTHROW
00400 {
00401     l4_msgtag_t t;
00402     t = l4_factory_create_start_u(L4_PROTO_TASK, target_cap, u);
00403     l4_factory_create_add_fpage_u(*utcb_area, &t, u);
00404     t = l4_factory_create_commit_u(factory, t, u);
00405     if (!l4_msgtag_has_error(t))
00406     {
00407         l4_msg_regs_t *v = l4_utcb_mr_u(u);
00408         utcb_area->raw = v->mr[0];
00409     }
00410     return t;
00411 }
00412
00413 L4_INLINE l4_msgtag_t
00414 l4_factory_create_thread_u(l4_cap_idx_t factory,
00415                            l4_cap_idx_t target_cap, l4_utcb_t *u) L4_NOTHROW
00416 {
00417     return l4_factory_create_u(factory, L4_PROTO_THREAD, target_cap, u);
00418 }
00419
00420 L4_INLINE l4_msgtag_t
00421 l4_factory_create_factory_u(l4_cap_idx_t factory,
00422                             l4_cap_idx_t target_cap, unsigned long limit,
00423                             l4_utcb_t *u) L4_NOTHROW
00424 {
00425     l4_msgtag_t t;
00426     t = l4_factory_create_start_u(L4_PROTO_FACTORY, target_cap, u);
00427     l4_factory_create_add_uint_u(limit, &t, u);
00428     return l4_factory_create_commit_u(factory, t, u);
00429 }
00430
00431 L4_INLINE l4_msgtag_t
00432 l4_factory_create_gate_u(l4_cap_idx_t factory,

```



```

00433         l4_cap_idx_t target_cap,
00434         l4_cap_idx_t thread_cap, l4_umword_t label,
00435         l4_utcb_t *u) L4_NOTHROW
00436 {
00437     l4_msgtag_t t;
00438     l4_msg_regst_t *v;
00439     int items = 0;
00440     t = l4_factory_create_start_u(0, target_cap, u);
00441     l4_factory_create_add_uint_u(label, &t, u);
00442     v = l4_utcb_mr_u(u);
00443     if (!(thread_cap & L4_INVALID_CAP_BIT))
00444     {
00445         items = 1;
00446         v->mr[3] = l4_map_obj_control(0,0);
00447         v->mr[4] = l4_obj_fpage(thread_cap, 0, L4_CAP_FPAGE_RWS).raw;
00448     }
00449     t = l4_msgtag(l4_msgtag_label(t), l4_msgtag_words(t), items, l4_msgtag_flags(t));
00450     return l4_factory_create_commit_u(factory, t, u);
00451 }
00452
00453 L4_INLINE l4_msgtag_t
00454 l4_factory_create_irq_u(l4_cap_idx_t factory,
00455                        l4_cap_idx_t target_cap, l4_utcb_t *u) L4_NOTHROW
00456 {
00457     return l4_factory_create_u(factory, L4_PROTO_IRQ_SENDER, target_cap, u);
00458 }
00459
00460 L4_INLINE l4_msgtag_t
00461 l4_factory_create_vm_u(l4_cap_idx_t factory,
00462                      l4_cap_idx_t target_cap,
00463                      l4_utcb_t *u) L4_NOTHROW
00464 {
00465     return l4_factory_create_u(factory, L4_PROTO_VM, target_cap, u);
00466 }
00467
00468 L4_INLINE l4_msgtag_t
00469 l4_factory_create_vcpu_context_u(l4_cap_idx_t factory,
00470                                l4_cap_idx_t target_cap,
00471                                l4_utcb_t *u) L4_NOTHROW
00472 {
00473     return l4_factory_create_u(factory, L4_PROTO_VCPU_CONTEXT, target_cap, u);
00474 }
00475
00476
00477
00478
00479
00480 L4_INLINE l4_msgtag_t
00481 l4_factory_create_task(l4_cap_idx_t factory,
00482                      l4_cap_idx_t target_cap, l4_fpage_t *utcb_area) L4_NOTHROW
00483 {
00484     return l4_factory_create_task_u(factory, target_cap, utcb_area, l4_utcb());
00485 }
00486
00487 L4_INLINE l4_msgtag_t
00488 l4_factory_create_thread(l4_cap_idx_t factory,
00489                        l4_cap_idx_t target_cap) L4_NOTHROW
00490 {
00491     return l4_factory_create_thread_u(factory, target_cap, l4_utcb());
00492 }
00493
00494 L4_INLINE l4_msgtag_t
00495 l4_factory_create_factory(l4_cap_idx_t factory,
00496                        l4_cap_idx_t target_cap, unsigned long limit) L4_NOTHROW
00497 {
00498     return l4_factory_create_factory_u(factory, target_cap, limit, l4_utcb());
00499 }
00500
00501
00502 L4_INLINE l4_msgtag_t
00503 l4_factory_create_gate(l4_cap_idx_t factory,
00504                      l4_cap_idx_t target_cap,
00505                      l4_cap_idx_t thread_cap, l4_umword_t label) L4_NOTHROW
00506 {
00507     return l4_factory_create_gate_u(factory, target_cap, thread_cap, label, l4_utcb());
00508 }
00509
00510 L4_INLINE l4_msgtag_t
00511 l4_factory_create_irq(l4_cap_idx_t factory,
00512                      l4_cap_idx_t target_cap) L4_NOTHROW
00513 {
00514     return l4_factory_create_irq_u(factory, target_cap, l4_utcb());
00515 }
00516
00517 L4_INLINE l4_msgtag_t
00518 l4_factory_create_vm(l4_cap_idx_t factory,
00519                    l4_cap_idx_t target_cap) L4_NOTHROW

```

```

00520 {
00521     return l4_factory_create_vm_u(factory, target_cap, l4_utcb());
00522 }
00523
00524 L4_INLINE l4_msgtag_t
00525 l4_factory_create_vcpu_context(l4_cap_idx_t factory,
00526                                l4_cap_idx_t target_cap) L4_NOTHROW
00527 {
00528     return l4_factory_create_vcpu_context_u(factory, target_cap, l4_utcb());
00529 }
00530
00531 L4_INLINE l4_msgtag_t
00532 l4_factory_create_start_u(long obj, l4_cap_idx_t target_cap,
00533                           l4_utcb_t *u) L4_NOTHROW
00534 {
00535     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00536     l4_buf_regs_t *b = l4_utcb_br_u(u);
00537     v->mr[0] = obj;
00538     b->bdr = 0;
00539     b->br[0] = target_cap | L4_RCV_ITEM_SINGLE_CAP;
00540     return l4_msgtag(L4_PROTO_FACTORY, 1, 0, 0);
00541 }
00542
00543 L4_INLINE int
00544 l4_factory_create_add_fpage_u(l4_fpage_t d, l4_msgtag_t *tag,
00545                               l4_utcb_t *u) L4_NOTHROW
00546 {
00547     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00548     int w = l4_msgtag_words(*tag);
00549     if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00550         return 0;
00551     v->mr[w] = L4_VARG_TYPE_FPAGE | (sizeof(l4_fpage_t) << 16);
00552     v->mr[w + 1] = d.raw;
00553     w += 2;
00554     tag->raw = (tag->raw & ~0x3FUL) | (w & 0x3f);
00555     return 1;
00556 }
00557
00558 L4_INLINE int
00559 l4_factory_create_add_int_u(l4_mword_t d, l4_msgtag_t *tag,
00560                             l4_utcb_t *u) L4_NOTHROW
00561 {
00562     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00563     int w = l4_msgtag_words(*tag);
00564     if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00565         return 0;
00566     v->mr[w] = L4_VARG_TYPE_MWORD | (sizeof(l4_mword_t) << 16);
00567     v->mr[w + 1] = d;
00568     w += 2;
00569     tag->raw = (tag->raw & ~0x3FUL) | (w & 0x3f);
00570     return 1;
00571 }
00572
00573 L4_INLINE int
00574 l4_factory_create_add_uint_u(l4_umword_t d, l4_msgtag_t *tag,
00575                               l4_utcb_t *u) L4_NOTHROW
00576 {
00577     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00578     int w = l4_msgtag_words(*tag);
00579     if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00580         return 0;
00581     v->mr[w] = L4_VARG_TYPE_UMWORD | (sizeof(l4_umword_t) << 16);
00582     v->mr[w + 1] = d;
00583     w += 2;
00584     tag->raw = (tag->raw & ~0x3FUL) | (w & 0x3f);
00585     return 1;
00586 }
00587
00588 L4_INLINE int
00589 l4_factory_create_add_str_u(char const *s, l4_msgtag_t *tag,
00590                             l4_utcb_t *u) L4_NOTHROW
00591 {
00592     return l4_factory_create_add_lstr_u(s, __builtin_strlen(s) + 1, tag, u);
00593 }
00594
00595 L4_INLINE int
00596 l4_factory_create_add_lstr_u(char const *s, unsigned len, l4_msgtag_t *tag,
00597                               l4_utcb_t *u) L4_NOTHROW
00598 {
00599     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00600     unsigned w = l4_msgtag_words(*tag);
00601     char *c;
00602     unsigned i;
00603     if (w + 1 + l4_bytes_to_mwords(len) > L4_UTCB_GENERIC_DATA_SIZE)
00604         return 0;

```

```

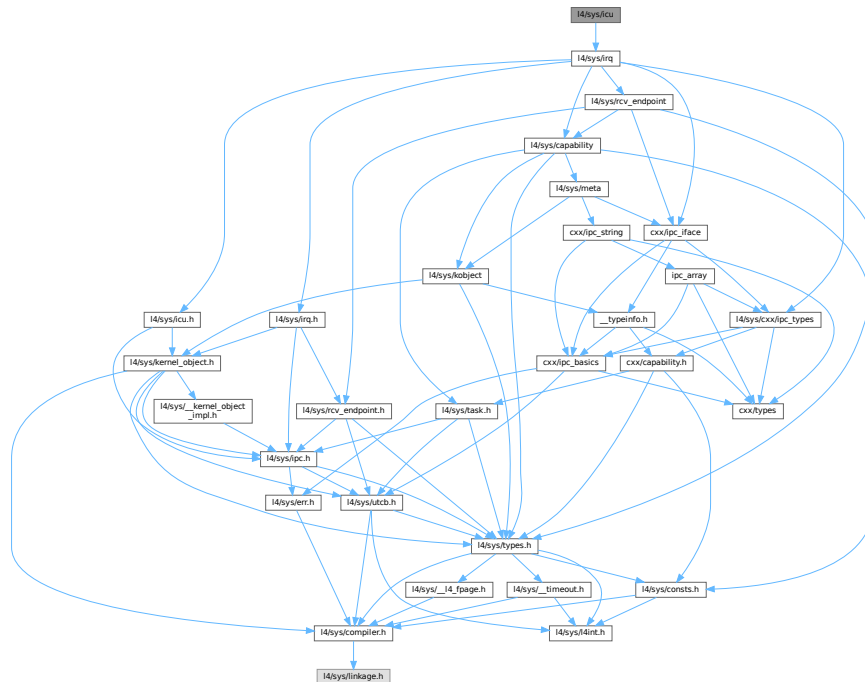
00607
00608     v->mr[w] = L4_VARG_TYPE_STRING | (len << 16);
00609     c = (char*)&v->mr[w + 1];
00610     for (i = 0; i < len; ++i)
00611         *c++ = *s++;
00612
00613     w = w + 1 + l4_bytes_to_mwords(len);
00614
00615     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00616     return 1;
00617 }
00618
00619 L4_INLINE int
00620 l4_factory_create_add_nil_u(l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW
00621 {
00622     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00623     int w = l4_msgtag_words(*tag);
00624     v->mr[w] = L4_VARG_TYPE_NIL;
00625     ++w;
00626     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00627     return 1;
00628 }
00629
00630
00631 L4_INLINE l4_msgtag_t
00632 l4_factory_create_commit_u(l4_cap_idx_t factory, l4_msgtag_t tag,
00633                           l4_utcb_t *u) L4_NOTHROW
00634 {
00635     return l4_ipc_call(factory, u, tag, L4_IPC_NEVER);
00636 }
00637
00638 L4_INLINE l4_msgtag_t
00639 l4_factory_create_u(l4_cap_idx_t factory, long obj, l4_cap_idx_t target,
00640                   l4_utcb_t *utcb) L4_NOTHROW
00641 {
00642     l4_msgtag_t t = l4_factory_create_start_u(obj, target, utcb);
00643     return l4_factory_create_commit_u(factory, t, utcb);
00644 }
00645
00646
00647 L4_INLINE l4_msgtag_t
00648 l4_factory_create(l4_cap_idx_t factory, long obj,
00649                  l4_cap_idx_t target) L4_NOTHROW
00650 {
00651     return l4_factory_create_u(factory, obj, target, l4_utcb());
00652 }

```

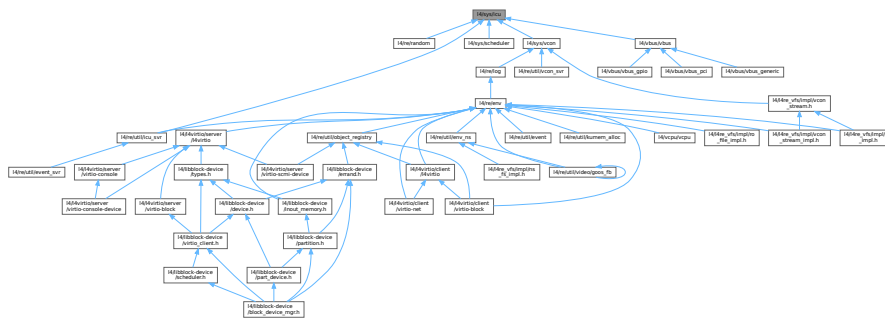
## 16.487 I4/sys/icu File Reference

Interrupt controller.

```
#include <l4/sys/irq>
Include dependency graph for icu:
```



This graph shows which files directly or indirectly include this file:



## 16.487.1 Detailed Description

Interrupt controller.

Definition in file [icu](#).

## 16.488 icu

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
```

```

00007  /*
00008  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024  #pragma once
00025
00026  #include <l4/sys/irq>

```

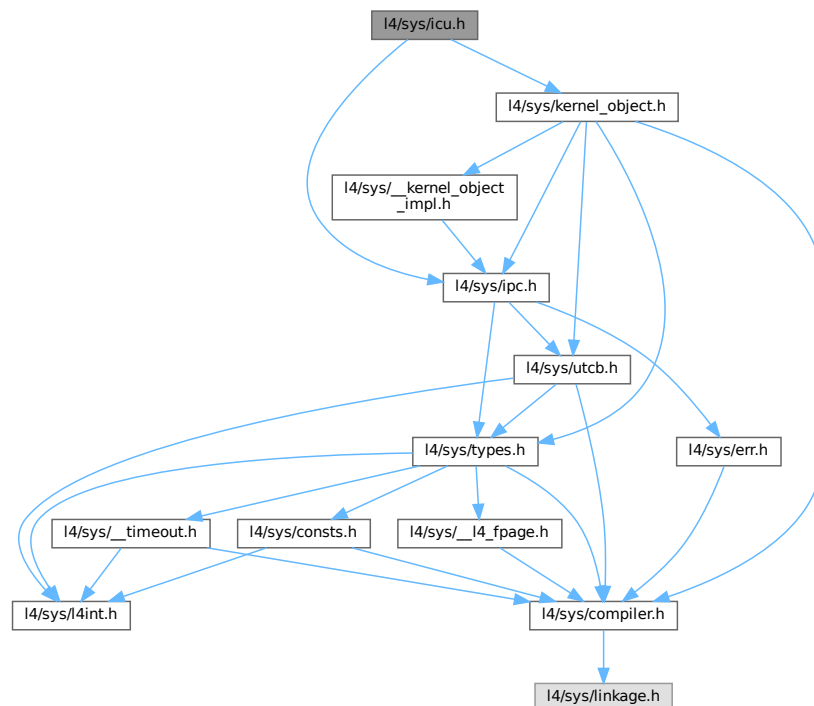
## 16.489 l4/sys/icu.h File Reference

Interrupt controller.

```
#include <l4/sys/kernel_object.h>
```

```
#include <l4/sys/ipc.h>
```

Include dependency graph for icu.h:





## Functions

- [l4\\_msgtag\\_t l4\\_icu\\_bind](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_cap\\_idx\\_t](#) irq) [L4\\_NOTHROW](#)  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- [l4\\_msgtag\\_t l4\\_icu\\_bind\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_cap\\_idx\\_t](#) irq, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- [l4\\_msgtag\\_t l4\\_icu\\_unbind](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_cap\\_idx\\_t](#) irq) [L4\\_NOTHROW](#)  
*Remove binding of an interrupt line from the interrupt controller object.*
- [l4\\_msgtag\\_t l4\\_icu\\_unbind\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_cap\\_idx\\_t](#) irq, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Remove binding of an interrupt line from the interrupt controller object.*
- [l4\\_msgtag\\_t l4\\_icu\\_set\\_mode](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) mode) [L4\\_NOTHROW](#)  
*Set interrupt mode.*
- [l4\\_msgtag\\_t l4\\_icu\\_set\\_mode\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) mode, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Set interrupt mode.*
- [l4\\_msgtag\\_t l4\\_icu\\_info](#) ([l4\\_cap\\_idx\\_t](#) icu, [l4\\_icu\\_info\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Get information about the ICU features.*
- [l4\\_msgtag\\_t l4\\_icu\\_info\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, [l4\\_icu\\_info\\_t](#) \*info, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Get information about the ICU features.*
- [l4\\_msgtag\\_t l4\\_icu\\_msi\\_info](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_uint64\\_t](#) source, [l4\\_icu\\_msi\\_info\\_t](#) \*msi\_info) [L4\\_NOTHROW](#)  
*Get MSI info about IRQ.*
- [l4\\_msgtag\\_t l4\\_icu\\_msi\\_info\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_uint64\\_t](#) source, [l4\\_icu\\_msi\\_info\\_t](#) \*msi\_info, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Get MSI info about IRQ.*
- [l4\\_msgtag\\_t l4\\_icu\\_unmask](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) to) [L4\\_NOTHROW](#)  
*Unmask an IRQ line.*
- [l4\\_msgtag\\_t l4\\_icu\\_unmask\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) to, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Unmask the given interrupt line.*
- [l4\\_msgtag\\_t l4\\_icu\\_mask](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) to) [L4\\_NOTHROW](#)  
*Mask an IRQ line.*
- [l4\\_msgtag\\_t l4\\_icu\\_mask\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) to, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Mask an IRQ line.*

### 16.489.1 Detailed Description

Interrupt controller.

Definition in file [icu.h](#).

## 16.490 icu.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/sys/kernel_object.h>
00028 #include <l4/sys/ipc.h>
00029
00063 enum L4_icu_flags
00064 {
00072     L4_ICU_FLAG_MSI = 0x80000000,
00073 };
00074
00075
00080 enum L4_irq_mode
00081 {
00083     L4_IRQ_F_NONE           = 0,
00084     L4_IRQ_F_LEVEL         = 0x2,
00085     L4_IRQ_F_EDGE          = 0x0,
00086     L4_IRQ_F_POS           = 0x0,
00087     L4_IRQ_F_NEG           = 0x4,
00088     L4_IRQ_F_BOTH          = 0x8,
00089     L4_IRQ_F_LEVEL_HIGH    = 0x3,
00090     L4_IRQ_F_LEVEL_LOW     = 0x7,
00091     L4_IRQ_F_POS_EDGE      = 0x1,
00092     L4_IRQ_F_NEG_EDGE      = 0x5,
00093     L4_IRQ_F_BOTH_EDGE     = 0x9,
00094     L4_IRQ_F_MASK          = 0xf,
00097     L4_IRQ_F_SET_WAKEUP    = 0x10,
00098     L4_IRQ_F_CLEAR_WAKEUP  = 0x20,
00099 };
00100
00101
00106 enum L4_icu_opcode
00107 {
00113     L4_ICU_OP_BIND = 0,
00114
00120     L4_ICU_OP_UNBIND = 1,
00121
00127     L4_ICU_OP_INFO = 2,
00128
00134     L4_ICU_OP_MSI_INFO = 3,
00135
00141     L4_ICU_OP_UNMASK = 4,
00142
00148     L4_ICU_OP_MASK = 5,
00149
00155     L4_ICU_OP_SET_MODE = 6,
00156 };
00157
00158 enum L4_icu_ctl_op
00159 {
00160     L4_ICU_CTL_UNMASK = 0,
00161     L4_ICU_CTL_MASK   = 1
00162 };
00163
00164
00172 typedef struct l4_icu_info_t
00173 {
00179     unsigned features;
00180
00184     unsigned nr_irqs;
00185
00189     unsigned nr_msis;
00190 } l4_icu_info_t;

```



```

00191
00193 typedef struct l4_icu_msi_info_t
00194 {
00196     l4_uint64_t msi_addr;
00198     l4_uint32_t msi_data;
00199 } l4_icu_msi_info_t;
00200
00227 L4_INLINE l4_msgtag_t
00228 l4_icu_bind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW;
00229
00236 L4_INLINE l4_msgtag_t
00237 l4_icu_bind_u(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq,
00238               l4_utcb_t *utcb) L4_NOTHROW;
00239
00250 L4_INLINE l4_msgtag_t
00251 l4_icu_unbind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW;
00252
00259 L4_INLINE l4_msgtag_t
00260 l4_icu_unbind_u(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq,
00261                 l4_utcb_t *utcb) L4_NOTHROW;
00262
00273 L4_INLINE l4_msgtag_t
00274 l4_icu_set_mode(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode) L4_NOTHROW;
00275
00282 L4_INLINE l4_msgtag_t
00283 l4_icu_set_mode_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode,
00284                  l4_utcb_t *utcb) L4_NOTHROW;
00285
00294 L4_INLINE l4_msgtag_t
00295 l4_icu_info(l4_cap_idx_t icu, l4_icu_info_t *info) L4_NOTHROW;
00296
00303 L4_INLINE l4_msgtag_t
00304 l4_icu_info_u(l4_cap_idx_t icu, l4_icu_info_t *info,
00305               l4_utcb_t *utcb) L4_NOTHROW;
00306
00313 L4_INLINE l4_msgtag_t
00314 l4_icu_msi_info(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
00315                 l4_icu_msi_info_t *msi_info) L4_NOTHROW;
00316
00323 L4_INLINE l4_msgtag_t
00324 l4_icu_msi_info_u(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
00325                  l4_icu_msi_info_t *msi_info, l4_utcb_t *utcb) L4_NOTHROW;
00326
00327
00345 L4_INLINE l4_msgtag_t
00346 l4_icu_unmask(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00347               l4_timeout_t to) L4_NOTHROW;
00348
00355 L4_INLINE l4_msgtag_t
00356 l4_icu_unmask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00357                 l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW;
00358
00376 L4_INLINE l4_msgtag_t
00377 l4_icu_mask(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00378             l4_timeout_t to) L4_NOTHROW;
00379
00386 L4_INLINE l4_msgtag_t
00387 l4_icu_mask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00388               l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW;
00389
00393 L4_INLINE l4_msgtag_t
00394 l4_icu_control_u(l4_cap_idx_t icu, unsigned irqnum, unsigned op,
00395                  l4_umword_t *label, l4_timeout_t to,
00396                  l4_utcb_t *utcb) L4_NOTHROW;
00397
00398
00399 /*****
00400  * Implementations
00401  */
00402
00403 L4_INLINE l4_msgtag_t
00404 l4_icu_bind_u(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq,
00405               l4_utcb_t *utcb) L4_NOTHROW
00406 {
00407     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00408     m->mr[0] = L4_ICU_OP_BIND;
00409     m->mr[1] = irqnum;
00410     m->mr[2] = l4_map_obj_control(0, 0);
00411     m->mr[3] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).raw;
00412     return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 1, 0), L4_IPC_NEVER);
00413 }
00414
00415 L4_INLINE l4_msgtag_t
00416 l4_icu_unbind_u(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq,
00417                 l4_utcb_t *utcb) L4_NOTHROW
00418 {
00419     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);

```

```

00420     m->mr[0] = L4_ICU_OP_UNBIND;
00421     m->mr[1] = irqnum;
00422     m->mr[2] = l4_map_obj_control(0, 0);
00423     m->mr[3] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).raw;
00424     return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 1, 0), L4_IPC_NEVER);
00425 }
00426
00427 L4_INLINE l4_msgtag_t
00428 l4_icu_info_u(l4_cap_idx_t icu, l4_icu_info_t *info,
00429              l4_utcb_t *utcb) L4_NOTHROW
00430 {
00431     l4_msgtag_t res;
00432     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00433     m->mr[0] = L4_ICU_OP_INFO;
00434     res = l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0), L4_IPC_NEVER);
00435     info->features = m->mr[0];
00436     info->nr_irqs = m->mr[1];
00437     info->nr_msis = m->mr[2];
00438     return res;
00439 }
00440
00441 L4_INLINE l4_msgtag_t
00442 l4_icu_msi_info_u(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
00443                  l4_icu_msi_info_t *msi_info, l4_utcb_t *utcb) L4_NOTHROW
00444 {
00445     l4_msgtag_t res;
00446     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00447     m->mr[0] = L4_ICU_OP_MSI_INFO;
00448     m->mr[1] = irqnum;
00449     m->mr64[l4_utcb_mr64_idx(2)] = source;
00450     res = l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ,
00451                                           2 + 1 * sizeof(l4_uint64_t)
00452                                           / sizeof(l4_umword_t),
00453                                           0, 0), L4_IPC_NEVER);
00454     if (L4_UNLIKELY(l4_msgtag_has_error(res)))
00455         return res;
00456
00457     if (L4_UNLIKELY(l4_msgtag_words(res) * sizeof(l4_umword_t) < sizeof(*msi_info)))
00458         return res;
00459
00460     __builtin_memcpy(msi_info, &m->mr[0], sizeof(*msi_info));
00461     return res;
00462 }
00463
00464 L4_INLINE l4_msgtag_t
00465 l4_icu_set_mode_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode,
00466                  l4_utcb_t *utcb) L4_NOTHROW
00467 {
00468     l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00469     mr->mr[0] = L4_ICU_OP_SET_MODE;
00470     mr->mr[1] = irqnum;
00471     mr->mr[2] = mode;
00472     return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 3, 0, 0), L4_IPC_NEVER);
00473 }
00474
00475 L4_INLINE l4_msgtag_t
00476 l4_icu_control_u(l4_cap_idx_t icu, unsigned irqnum, unsigned op,
00477                  l4_umword_t *label, l4_timeout_t to,
00478                  l4_utcb_t *utcb) L4_NOTHROW
00479 {
00480     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00481     m->mr[0] = L4_ICU_OP_UNMASK + op;
00482     m->mr[1] = irqnum;
00483     if (label)
00484         return l4_ipc_send_and_wait(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 0, 0),
00485                                     label, to);
00486     else
00487         return l4_ipc_send(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 0, 0), to);
00488 }
00489
00490 L4_INLINE l4_msgtag_t
00491 l4_icu_mask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00492               l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
00493 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_MASK, label, to, utcb); }
00494
00495 L4_INLINE l4_msgtag_t
00496 l4_icu_unmask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00497                 l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
00498 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_UNMASK, label, to, utcb); }
00499
00500
00501
00502
00503 L4_INLINE l4_msgtag_t
00504 l4_icu_bind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW
00505 { return l4_icu_bind_u(icu, irqnum, irq, l4_utcb()); }
00506

```

```

00507 L4_INLINE l4_msgtag_t
00508 l4_icu_unbind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW
00509 { return l4_icu_unbind_u(icu, irqnum, irq, l4_utcb()); }
00510
00511 L4_INLINE l4_msgtag_t
00512 l4_icu_info(l4_cap_idx_t icu, l4_icu_info_t *info) L4_NOTHROW
00513 { return l4_icu_info_u(icu, info, l4_utcb()); }
00514
00515 L4_INLINE l4_msgtag_t
00516 l4_icu_msi_info(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
00517                l4_icu_msi_info_t *msi_info) L4_NOTHROW
00518 { return l4_icu_msi_info_u(icu, irqnum, source, msi_info, l4_utcb()); }
00519
00520 L4_INLINE l4_msgtag_t
00521 l4_icu_unmask(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00522              l4_timeout_t to) L4_NOTHROW
00523 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_UNMASK, label, to, l4_utcb()); }
00524
00525 L4_INLINE l4_msgtag_t
00526 l4_icu_mask(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00527            l4_timeout_t to) L4_NOTHROW
00528 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_MASK, label, to, l4_utcb()); }
00529
00530 L4_INLINE l4_msgtag_t
00531 l4_icu_set_mode(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode) L4_NOTHROW
00532 {
00533     return l4_icu_set_mode_u(icu, irqnum, mode, l4_utcb());
00534 }

```

## 16.491 iommu

```

00001 // vi:set ft=c++: -*- Mode: C++ -*-
00002 /* \file
00003  * IO-MMU interface description.
00004  */
00005 #pragma once
00006
00007 #include <l4/sys/cxx/ipc_iface>
00008
00009 namespace L4 {
00021 class Iommu :
00022     public Kobject_x<Iommu, Proto_t<L4_PROTO_IOMMU>, Type_info::Demand_t<1> >
00023 {
00024 public:
00037 L4_INLINE_RPC(
00038     l4_msgtag_t, bind, (l4_uint64_t src_id, Ipc::Cap<Task> dma_space));
00039
00050 L4_INLINE_RPC(
00051     l4_msgtag_t, unbind, (l4_uint64_t src_id, Ipc::Cap<Task> dma_space));
00052
00053 typedef Typeid::Rpcs_code<l4_umword_t>::F<bind_t, unbind_t> Rpcs;
00054 };
00055
00056 }

```

## 16.492 ipc.h

```

00001
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  * Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  * Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00011  * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00012  * economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * This file is part of TUD:OS and distributed under the terms of the
00015  * GNU General Public License 2.
00016  * Please see the COPYING-GPL-2 file for details.
00017  *
00018  * As a special exception, you may use this file as part of a free software
00019  * library without restriction. Specifically, if other files instantiate
00020  * templates or use macros or inline functions from this file, or you compile
00021  * this file and link it with other files to produce an executable, this
00022  * file does not by itself cause the resulting executable to be covered by
00023  * the GNU General Public License. This exception does not however
00024  * invalidate any other reasons why the executable file might be covered by
00025  * the GNU General Public License.
00026  */
00027 #ifndef __L4SYS__INCLUDE__ARCH_AMD64__L4API_L4F__IPC_H__

```

```

00028 #define __L4SYS__INCLUDE__ARCH_AMD64__L4API_L4F__IPC_H__
00029
00030 #include_next <l4/sys/ipc.h>
00031
00032 L4_INLINE l4_msgtag_t
00033 l4_ipc(l4_cap_idx_t dest, l4_utcb_t *utcb,
00034        l4_umword_t flags,
00035        l4_umword_t slabel,
00036        l4_msgtag_t tag,
00037        l4_umword_t *rlabel,
00038        l4_timeout_t timeout) L4_NOTHROW
00039 {
00040     l4_umword_t dummy, dummy2;
00041     register l4_umword_t to __asm__("r8") = timeout.raw;
00042
00043     (void)utcb;
00044
00045     __asm__ __volatile__
00046     ("syscall"
00047      :
00048      "=d" (dummy2),
00049      "=S" (slabel),
00050      "=D" (dummy),
00051      "=a" (tag.raw)
00052      :
00053      "S" (slabel),
00054      "r" (to),
00055      "a" (tag.raw),
00056      "d" (dest | flags)
00057      :
00058      "memory", "cc", "rcx", "r11", "r15"
00059      );
00060
00061     if (rlabel)
00062         *rlabel = slabel;
00063
00064     return tag;
00065 }
00066
00067 #endif /* ! __L4SYS__INCLUDE__ARCH_AMD64__L4API_L4F__IPC_H__ */

```

## 16.493 arm/l4f/l4/sys/ipc.h File Reference

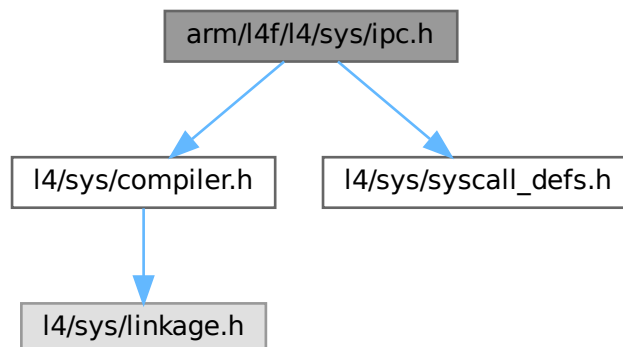
L4 IPC System Calls, ARM.

```

#include <l4/sys/compiler.h>
#include <l4/sys/syscall_defs.h>

```

Include dependency graph for ipc.h:



## Functions

- `l4_msgtag_t l4_ipc(l4_cap_idx_t dest, l4_utcb_t *utcb, l4_umword_t flags, l4_umword_t slabel, l4_msgtag_t tag, l4_umword_t *rlabel, l4_timeout_t timeout) L4_NOTHROW`

Generic *L4* object invocation.

### 16.493.1 Detailed Description

*L4* IPC System Calls, ARM.

Definition in file `ipc.h`.

## 16.494 ipc.h

[Go to the documentation of this file.](#)

```

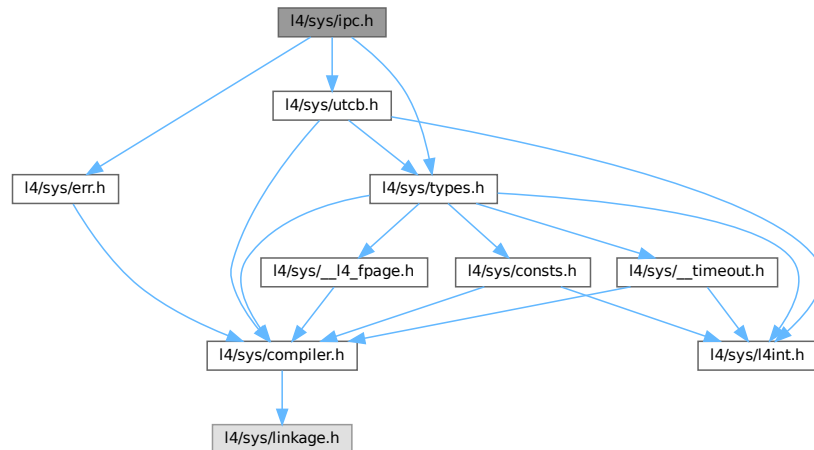
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include_next <l4/sys/ipc.h>
00027
00028 #ifdef __GNUC__
00029
00030 #include <l4/sys/compiler.h>
00031 #include <l4/sys/syscall_defs.h>
00032
00033 L4_INLINE l4_msgtag_t
00034 l4_ipc(l4_cap_idx_t dest, l4_utcb_t *utcb,
00035        l4_umword_t flags,
00036        l4_umword_t slabel,
00037        l4_msgtag_t tag,
00038        l4_umword_t *rlabel,
00039        l4_timeout_t timeout) L4_NOTHROW
00040 {
00041     register l4_umword_t _dest    __asm__("r2") = dest | flags;
00042     register l4_umword_t _timeout __asm__("r3") = timeout.raw;
00043     register l4_umword_t _tag     __asm__("r0") = tag.raw;
00044     register l4_umword_t _label   __asm__("r4") = slabel;
00045     (void)utcb;
00046
00047     __asm__ __volatile__
00048     ("mov r5, %[sc]          \n"
00049      "blx __l4_sys_syscall   \n"
00050      :
00051      "+r" (_dest),
00052      "+r" (_timeout),
00053      "+r" (_label),
00054      "+r" (_tag)
00055      :
00056      [sc] "i" (L4_SYSCALL_INVOKE)
00057      :
00058      "cc", "memory", "r5", "ip", "lr");
00059
00060     if (rlabel)
00061         *rlabel = _label;
00062     tag.raw = _tag;
00063
00064     return tag;
00065 }
00066
00067 #endif //__GNUC__

```

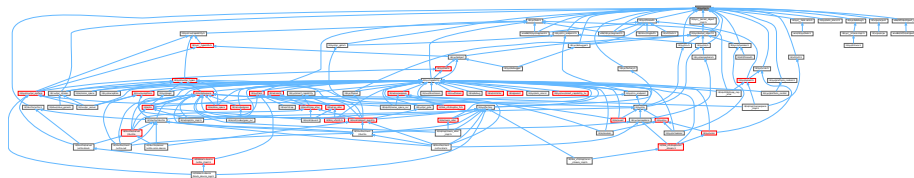
## 16.495 l4/sys/ipc.h File Reference

Common IPC interface.

```
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/err.h>
Include dependency graph for ipc.h:
```



This graph shows which files directly or indirectly include this file:



### Enumerations

```
enum l4_ipc_tcr_error_t {
    L4_IPC_ERROR_MASK = 0x1F , L4_IPC_SND_ERR_MASK = 0x01 , L4_IPC_ENOT_EXISTENT = 0x04 ,
    L4_IPC_RETIMEOUT = 0x03 ,
    L4_IPC_SETIMEOUT = 0x02 , L4_IPC_RECANCELED = 0x07 , L4_IPC_SECANCELED = 0x06 ,
    L4_IPC_REMAPFAILED = 0x11 ,
    L4_IPC_SEMAPFAILED = 0x10 , L4_IPC_RESNDPFTO = 0x0b , L4_IPC_SESNDPFTO = 0x0a ,
    L4_IPC_RERCVPFTO = 0x0d ,
    L4_IPC_SERCVPFTO = 0x0c , L4_IPC_REABORTED = 0x0f , L4_IPC_SEABORTED = 0x0e ,
    L4_IPC_REMSGCUT = 0x09 ,
    L4_IPC_SEMSGCUT = 0x08 }

```

*Error codes in the error TCR.*

## Functions

- [l4\\_umword\\_t l4\\_ipc\\_error](#) ([l4\\_msgtag\\_t](#) tag, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Get the IPC error code for an IPC operation.*
- [long l4\\_error](#) ([l4\\_msgtag\\_t](#) tag) [L4\\_NOTHROW](#)  
*Get IPC error code if any or message tag label otherwise for an IPC call.*
- [int l4\\_ipc\\_is\\_snd\\_error](#) ([l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Returns whether an error occurred in send phase of an invocation.*
- [int l4\\_ipc\\_is\\_rcv\\_error](#) ([l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Returns whether an error occurred in receive phase of an invocation.*
- [int l4\\_ipc\\_error\\_code](#) ([l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Get the error condition of the last invocation from the TCR.*
- [long l4\\_ipc\\_to\\_errno](#) (unsigned long ipc\_error\_code) [L4\\_NOTHROW](#)  
*Get a negative error code for the given IPC error code.*
- [l4\\_msgtag\\_t l4\\_ipc\\_send](#) ([l4\\_cap\\_idx\\_t](#) dest, [l4\\_utcb\\_t](#) \*utcb, [l4\\_msgtag\\_t](#) tag, [l4\\_timeout\\_t](#) timeout) [L4\\_NOTHROW](#)  
*Send a message to an object (do **not** wait for a reply).*
- [l4\\_msgtag\\_t l4\\_ipc\\_wait](#) ([l4\\_utcb\\_t](#) \*utcb, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) timeout) [L4\\_NOTHROW](#)  
*Wait for an incoming message from any possible sender.*
- [l4\\_msgtag\\_t l4\\_ipc\\_receive](#) ([l4\\_cap\\_idx\\_t](#) object, [l4\\_utcb\\_t](#) \*utcb, [l4\\_timeout\\_t](#) timeout) [L4\\_NOTHROW](#)  
*Wait for a message from a specific source.*
- [l4\\_msgtag\\_t l4\\_ipc\\_call](#) ([l4\\_cap\\_idx\\_t](#) object, [l4\\_utcb\\_t](#) \*utcb, [l4\\_msgtag\\_t](#) tag, [l4\\_timeout\\_t](#) timeout) [L4\\_NOTHROW](#)  
*Object call (usual invocation).*
- [l4\\_msgtag\\_t l4\\_ipc\\_reply\\_and\\_wait](#) ([l4\\_utcb\\_t](#) \*utcb, [l4\\_msgtag\\_t](#) tag, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) timeout) [L4\\_NOTHROW](#)  
*Reply and wait operation (uses the reply capability).*
- [l4\\_msgtag\\_t l4\\_ipc\\_send\\_and\\_wait](#) ([l4\\_cap\\_idx\\_t](#) dest, [l4\\_utcb\\_t](#) \*utcb, [l4\\_msgtag\\_t](#) tag, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) timeout) [L4\\_NOTHROW](#)  
*Send a message and do an open wait.*
- [l4\\_msgtag\\_t l4\\_ipc](#) ([l4\\_cap\\_idx\\_t](#) dest, [l4\\_utcb\\_t](#) \*utcb, [l4\\_umword\\_t](#) flags, [l4\\_umword\\_t](#) slabel, [l4\\_msgtag\\_t](#) tag, [l4\\_umword\\_t](#) \*rlabel, [l4\\_timeout\\_t](#) timeout) [L4\\_NOTHROW](#)  
*Generic L4 object invocation.*
- [l4\\_msgtag\\_t l4\\_ipc\\_sleep](#) ([l4\\_timeout\\_t](#) timeout) [L4\\_NOTHROW](#)  
*Sleep for an amount of time.*
- [l4\\_msgtag\\_t l4\\_ipc\\_sleep\\_ms](#) ([l4\\_uint32\\_t](#) ms) [L4\\_NOTHROW](#)  
*Sleep for a certain amount of milliseconds.*
- [l4\\_msgtag\\_t l4\\_ipc\\_sleep\\_us](#) ([l4\\_uint64\\_t](#) us) [L4\\_NOTHROW](#)  
*Sleep for a certain amount of microseconds.*
- [int l4\\_sndfpage\\_add](#) ([l4\\_fpage\\_t](#) const snd\_fpage, unsigned long snd\_base, [l4\\_msgtag\\_t](#) \*tag) [L4\\_NOTHROW](#)  
*Add a flex-page to be sent to the UTCB.*

### 16.495.1 Detailed Description

Common IPC interface.

Definition in file [ipc.h](#).

## 16.495.2 Function Documentation

### 16.495.2.1 l4\_ipc\_to\_errno()

```
long l4_ipc_to_errno (
    unsigned long ipc_error_code )    [inline]
```

Get a negative error code for the given IPC error code.

#### Parameters

<code>ipc_error_code</code>	IPC error code as delivered by the kernel. (or returned by the <a href="#">l4_ipc_error_code()</a> function).
-----------------------------	---

#### Returns

negative error code in the range of [L4\\_EIPC\\_LO](#) to [L4\\_EIPC\\_HI](#).

Definition at line 572 of file [ipc.h](#).

References [L4\\_EIPC\\_LO](#).

## 16.496 ipc.h

[Go to the documentation of this file.](#)

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00010  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction. Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License. This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 #ifndef __L4SYS__INCLUDE__L4API_FIASCO__IPC_H__
00027 #define __L4SYS__INCLUDE__L4API_FIASCO__IPC_H__
00028
00029 #include <l4/sys/types.h>
00030 #include <l4/sys/utcb.h>
00031 #include <l4/sys/err.h>
00032
00073 /*****
00074  *** IPC result checking
00075  *****/
00076
00092 enum l4_ipc_tcr_error_t
00093 {
00094     L4_IPC_ERROR_MASK           = 0x1F,
00095     L4_IPC_SND_ERR_MASK        = 0x01,
00097     L4_IPC_ENOT_EXISTENT       = 0x04,
00100     L4_IPC_RETIMEOUT           = 0x03,
00103     L4_IPC_SETTIMEOUT          = 0x02,
00106     L4_IPC_RECANCELED          = 0x07,
00109     L4_IPC_SECANCELED          = 0x06,
00112     L4_IPC_REMAPFAILED         = 0x11,
00116     L4_IPC_SEMAPFAILED         = 0x10,
```



```

00119     L4_IPC_RESNDPFTO           = 0x0b,
00123     L4_IPC_SESNDFPFTO        = 0x0a,
00127     L4_IPC_RERCVPFTO       = 0x0d,
00131     L4_IPC_SERCVPFTO        = 0x0c,
00135     L4_IPC_REABORTED         = 0x0f,
00138     L4_IPC_SEABORTED        = 0x0e,
00147     L4_IPC_REMSGCUT         = 0x09,
00148
00154     L4_IPC_SEMSGCUT           = 0x08,
00155 };
00156
00157
00168 L4_INLINE l4_umword_t
00169 l4_ipc_error(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;
00170
00171
00188 L4_INLINE long
00189 l4_error(l4_msgtag_t tag) L4_NOTHROW;
00190
00191 L4_INLINE long
00192 l4_error_u(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;
00193
00194 /*****
00195  *** IPC results
00196  *****/
00197
00207 L4_INLINE int l4_ipc_is_snd_error(l4_utcb_t *utcb) L4_NOTHROW;
00208
00218 L4_INLINE int l4_ipc_is_rcv_error(l4_utcb_t *utcb) L4_NOTHROW;
00219
00229 L4_INLINE int l4_ipc_error_code(l4_utcb_t *utcb) L4_NOTHROW;
00230
00237 L4_INLINE long l4_ipc_to_errno(unsigned long ipc_error_code) L4_NOTHROW;
00238
00239
00240 /*****
00241  *** IPC calls
00242  *****/
00243
00272 L4_INLINE l4_msgtag_t
00273 l4_ipc_send(l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag,
00274             l4_timeout_t timeout) L4_NOTHROW;
00275
00276
00303 L4_INLINE l4_msgtag_t
00304 l4_ipc_wait(l4_utcb_t *utcb, l4_umword_t *label,
00305             l4_timeout_t timeout) L4_NOTHROW;
00306
00307
00332 L4_INLINE l4_msgtag_t
00333 l4_ipc_receive(l4_cap_idx_t object, l4_utcb_t *utcb,
00334               l4_timeout_t timeout) L4_NOTHROW;
00335
00366 L4_INLINE l4_msgtag_t
00367 l4_ipc_call(l4_cap_idx_t object, l4_utcb_t *utcb, l4_msgtag_t tag,
00368             l4_timeout_t timeout) L4_NOTHROW;
00369
00370
00396 L4_INLINE l4_msgtag_t
00397 l4_ipc_reply_and_wait(l4_utcb_t *utcb, l4_msgtag_t tag,
00398                      l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW;
00399
00428 L4_INLINE l4_msgtag_t
00429 l4_ipc_send_and_wait(l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag,
00430                      l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW;
00431
00438 #if 0
00449 L4_INLINE l4_msgtag_t
00450 l4_ipc_wait_next_period(l4_utcb_t *utcb,
00451                         l4_umword_t *label,
00452                         l4_timeout_t timeout);
00453
00454 #endif
00455
00475 L4_ALWAYS_INLINE l4_msgtag_t
00476 l4_ipc(l4_cap_idx_t dest,
00477        l4_utcb_t *utcb,
00478        l4_umword_t flags,
00479        l4_umword_t slabel,
00480        l4_msgtag_t tag,
00481        l4_umword_t *rlabel,
00482        l4_timeout_t timeout) L4_NOTHROW;
00483
00500 L4_INLINE l4_msgtag_t
00501 l4_ipc_sleep(l4_timeout_t timeout) L4_NOTHROW;
00502
00519 L4_INLINE l4_msgtag_t

```

```

00520 l4_ipc_sleep_ms(l4_uint32_t ms) L4_NOTHROW;
00521
00538 L4_INLINE l4_msgtag_t
00539 l4_ipc_sleep_us(l4_uint64_t us) L4_NOTHROW;
00540
00555 L4_INLINE int
00556 l4_sndfpage_add(l4_fpage_t const snd_fpage, unsigned long snd_base,
00557                 l4_msgtag_t *tag) L4_NOTHROW;
00558
00559 /*
00560  * \internal
00561  * \ingroup l4_ipc_api
00562  */
00563 L4_INLINE int
00564 l4_sndfpage_add_u(l4_fpage_t const snd_fpage, unsigned long snd_base,
00565                  l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW;
00566
00567
00568 /*****
00569  * Implementations
00570  *****/
00571
00572 L4_INLINE long l4_ipc_to_errno(unsigned long ipc_error_code) L4_NOTHROW
00573 { return -(L4_EIPC_LO + ipc_error_code); }
00574
00575 L4_INLINE l4_msgtag_t
00576 l4_ipc_call(l4_cap_idx_t object, l4_utcb_t *utcb, l4_msgtag_t tag,
00577             l4_timeout_t timeout) L4_NOTHROW
00578 {
00579     return l4_ipc(object, utcb, L4_SYSF_CALL, 0, tag, 0, timeout);
00580 }
00581
00582 L4_INLINE l4_msgtag_t
00583 l4_ipc_reply_and_wait(l4_utcb_t *utcb, l4_msgtag_t tag,
00584                      l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW
00585 {
00586     return l4_ipc(L4_INVALID_CAP, utcb, L4_SYSF_REPLY_AND_WAIT, 0, tag, label, timeout);
00587 }
00588
00589 L4_INLINE l4_msgtag_t
00590 l4_ipc_send_and_wait(l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag,
00591                     l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW
00592 {
00593     return l4_ipc(dest, utcb, L4_SYSF_SEND_AND_WAIT, 0, tag, label, timeout);
00594 }
00595
00596 L4_INLINE l4_msgtag_t
00597 l4_ipc_send(l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag,
00598            l4_timeout_t timeout) L4_NOTHROW
00599 {
00600     return l4_ipc(dest, utcb, L4_SYSF_SEND, 0, tag, 0, timeout);
00601 }
00602
00603 L4_INLINE l4_msgtag_t
00604 l4_ipc_wait(l4_utcb_t *utcb, l4_umword_t *label,
00605            l4_timeout_t timeout) L4_NOTHROW
00606 {
00607     l4_msgtag_t t;
00608     t.raw = 0;
00609     return l4_ipc(L4_INVALID_CAP, utcb, L4_SYSF_WAIT, 0, t, label, timeout);
00610 }
00611
00612 L4_INLINE l4_msgtag_t
00613 l4_ipc_receive(l4_cap_idx_t object, l4_utcb_t *utcb,
00614              l4_timeout_t timeout) L4_NOTHROW
00615 {
00616     l4_msgtag_t t;
00617     t.raw = 0;
00618     return l4_ipc(object, utcb, L4_SYSF_RECV, 0, t, 0, timeout);
00619 }
00620
00621 L4_INLINE l4_msgtag_t
00622 l4_ipc_sleep(l4_timeout_t timeout) L4_NOTHROW
00623 { return l4_ipc_receive(L4_INVALID_CAP, NULL, timeout); }
00624
00625 L4_INLINE l4_msgtag_t
00626 l4_ipc_sleep_ms(l4_uint32_t ms) L4_NOTHROW
00627 {
00628     l4_uint64_t us = ms * 1000ULL; // cannot overflow because ms < 2^32
00629     return l4_ipc_sleep(l4_timeout(L4_IPC_TIMEOUT_NEVER, l4_timeout_from_us(us)));
00630 }
00631
00632 L4_INLINE l4_msgtag_t
00633 l4_ipc_sleep_us(l4_uint64_t us) L4_NOTHROW
00634 {
00635     return l4_ipc_sleep(l4_timeout(L4_IPC_TIMEOUT_NEVER,
00636                                   l4_timeout_from_us(us)));
00636 }

```

```

00637 }
00638
00639 L4_INLINE l4_umword_t
00640 l4_ipc_error(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW
00641 {
00642     if (L4_LIKELY(!l4_msgtag_has_error(tag)))
00643         return 0;
00644     return l4_utcb_tcr_u(utcb)->error & L4_IPC_ERROR_MASK;
00645 }
00646
00647 L4_INLINE long
00648 l4_error_u(l4_msgtag_t tag, l4_utcb_t *u) L4_NOTHROW
00649 {
00650     if (L4_UNLIKELY(l4_msgtag_has_error(tag)))
00651         return l4_ipc_to_errno(l4_utcb_tcr_u(u)->error & L4_IPC_ERROR_MASK);
00652     return l4_msgtag_label(tag);
00653 }
00654
00655 L4_INLINE long
00656 l4_error(l4_msgtag_t tag) L4_NOTHROW
00657 {
00658     return l4_error_u(tag, l4_utcb());
00659 }
00660
00661
00662
00663 L4_INLINE int l4_ipc_is_snd_error(l4_utcb_t *u) L4_NOTHROW
00664 { return (l4_utcb_tcr_u(u)->error & 1) == 0; }
00665
00666 L4_INLINE int l4_ipc_is_rcv_error(l4_utcb_t *u) L4_NOTHROW
00667 { return l4_utcb_tcr_u(u)->error & 1; }
00668
00669 L4_INLINE int l4_ipc_error_code(l4_utcb_t *u) L4_NOTHROW
00670 { return l4_utcb_tcr_u(u)->error & L4_IPC_ERROR_MASK; }
00671
00672
00673 /*
00674  * \internal
00675  * \ingroup l4_ipc_api
00676  */
00677 L4_INLINE int
00678 l4_sndfpage_add_u(l4_fpage_t const snd_fpage, unsigned long snd_base,
00679                  l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW
00680 {
00681     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00682     int i = l4_msgtag_words(*tag) + 2 * l4_msgtag_items(*tag);
00683
00684     if (i >= L4_UTCB_GENERIC_DATA_SIZE - 1)
00685         return -L4_ENOMEM;
00686
00687     v->mr[i] = snd_base | L4_ITEM_MAP | L4_ITEM_CONT;
00688     v->mr[i + 1] = snd_fpage.raw;
00689
00690     *tag = l4_msgtag(l4_msgtag_label(*tag), l4_msgtag_words(*tag),
00691                     l4_msgtag_items(*tag) + 1, l4_msgtag_flags(*tag));
00692     return 0;
00693 }
00694
00695 L4_INLINE int
00696 l4_sndfpage_add(l4_fpage_t const snd_fpage, unsigned long snd_base,
00697                l4_msgtag_t *tag) L4_NOTHROW
00698 {
00699     return l4_sndfpage_add_u(snd_fpage, snd_base, tag, l4_utcb());
00700 }
00701
00702
00703 #endif /* ! __L4SYS__INCLUDE__L4API_FIASCO__IPC_H__ */

```

## 16.497 x86/l4f/l4/sys/ipc.h File Reference

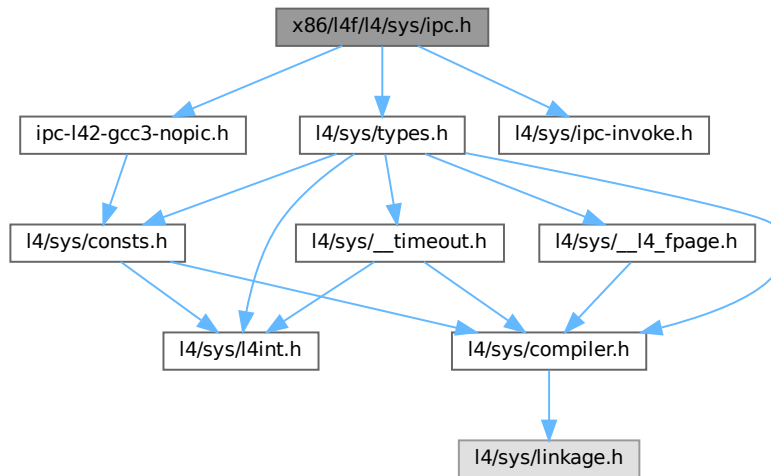
L4 IPC System Calls, x86.

```

#include <l4/sys/types.h>
#include <l4/sys/ipc-invoke.h>
#include "ipc-l42-gcc3-nopic.h"

```

Include dependency graph for ipc.h:



## 16.497.1 Detailed Description

[L4 IPC System Calls, x86.](#)

Definition in file [ipc.h](#).

## 16.498 ipc.h

[Go to the documentation of this file.](#)

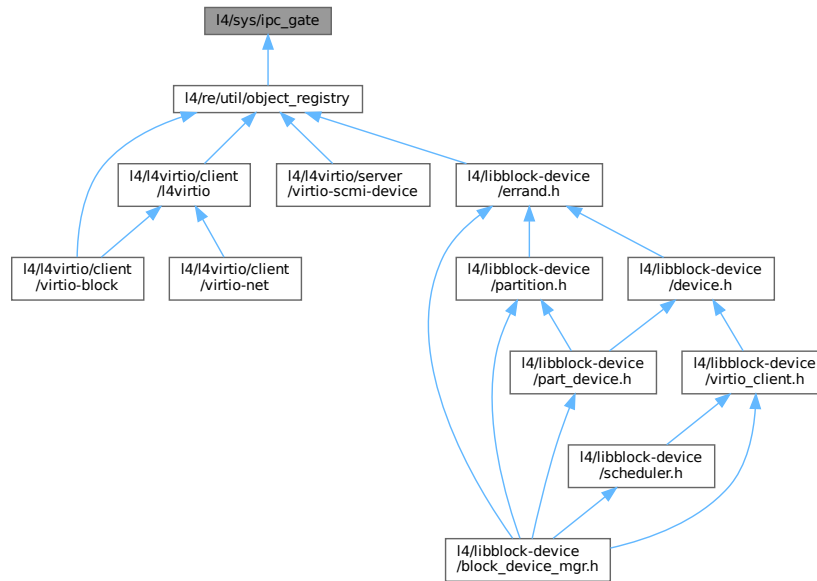
```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Lars Reuther <reuther@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #ifndef __L4_IPC_H__
00026 #define __L4_IPC_H__
00027
00028 #include <l4/sys/types.h>
00029
00030 #include_next <l4/sys/ipc.h>
00031
00032 /*****
00033  *** Implementation
00034  *****/
00035
00036 #include <l4/sys/ipc-invoke.h>
00037 #include "ipc-l42-gcc3-nopic.h"
00038
00039 #endif /* !__L4_IPC_H__ */

```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::ipc\\_gate](#)

The C++ IPC gate interface, see [IPC-Gate API](#) for the C interface.

## Namespaces

- namespace [L4](#)

[L4](#) low-level kernel interface.

## 16.499.1 Detailed Description

The C++ IPC gate interface.

Definition in file [ipc\\_gate](#).

## 16.500 ipc\_gate

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=c++: -- Mode: C++ --
00002 /*
00003  * (c) 2009-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  */

```

```

00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/sys/ipc_gate.h>
00027 #include <l4/sys/capability>
00028 #include <l4/sys/rcv_endpoint>
00029 #include <l4/sys/cxx/ipc_iface>
00030
00031 namespace L4 {
00032
00033 class Thread;
00034
00094 class L4_EXPORT Ipc_gate :
00095     public Kobject_t<Ipc_gate, Rcv_endpoint, L4_PROTO_KOBJECT,
00096         Type_info::Demand_t<1> >
00097 {
00098 public:
00112     L4_INLINE_RPC_OP(L4_IPC_GATE_GET_INFO_OP,
00113         l4_msgtag_t, get_infos, (l4_umword_t *label));
00114
00115     typedef L4::Typeid::Rpcsys<bind_thread_t, get_infos_t> Rpcsys;
00116 };
00117
00118 }

```

## 16.501 l4/sys/ipc\_gate.h File Reference

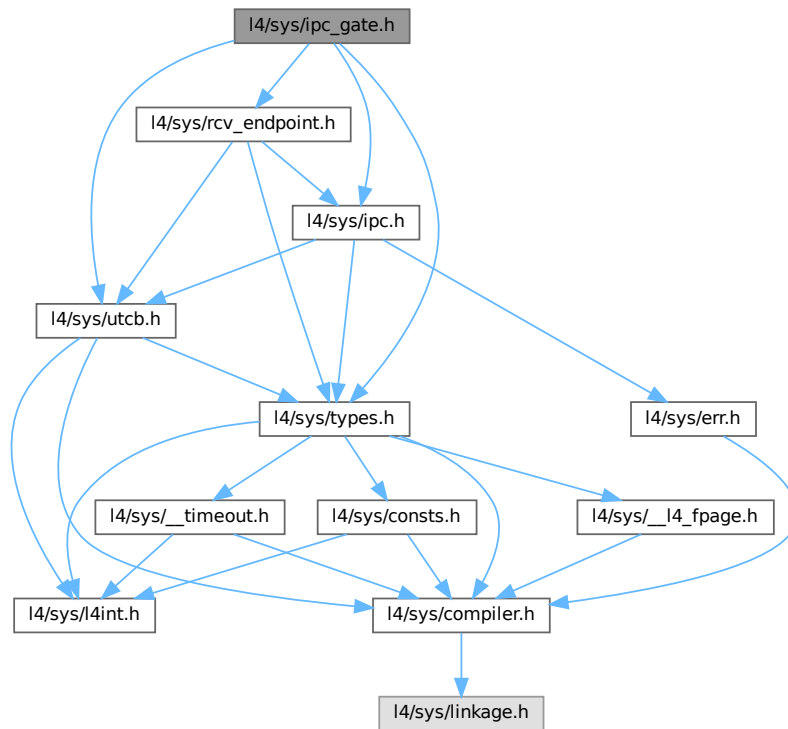
The C IPC gate interface, see [L4::lpc\\_gate](#) for the C++ interface.

```

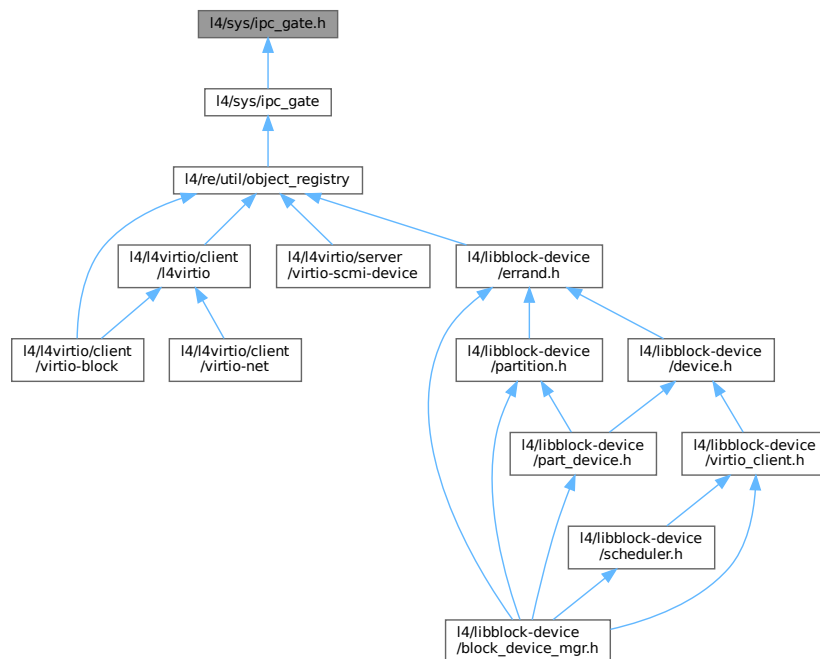
#include <l4/sys/utcb.h>
#include <l4/sys/types.h>
#include <l4/sys/rcv_endpoint.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for ipc\_gate.h:



This graph shows which files directly or indirectly include this file:





## Enumerations

- enum [L4\\_ipc\\_gate\\_ops](#) { [L4\\_IPC\\_GATE\\_BIND\\_OP](#) = 0x10 , [L4\\_IPC\\_GATE\\_GET\\_INFO\\_OP](#) = 0x11 }
- Operations on the IPC-gate.*

## Functions

- [l4\\_msgtag\\_t l4\\_ipc\\_gate\\_get\\_infos](#) ([l4\\_cap\\_idx\\_t](#) gate, [l4\\_umword\\_t](#) \*label)
- Get information about the IPC-gate.*

### 16.501.1 Detailed Description

The C IPC gate interface, see [L4::ipc\\_gate](#) for the C++ interface.

IPC gates are used to create secure communication channels between protection domains. An IPC gate can be created using the [Factory](#) interface.

Depending on the permissions of the capability used, an IPC gate forwards IPC to the [Thread](#) that is *bound* to the IPC gate (cf. [l4\\_rcv\\_ep\\_bind\\_thread\(\)](#)). If the capability has the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission, only IPC using a protocol different from the [L4\\_PROTO\\_KOBJECT](#) protocol is forwarded. Without the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission, all IPC is forwarded. The latter is the usual case for a client in a client/server scenario. When no thread is bound yet, the forwarded IPC blocks until a thread is bound or the IPC times out.

Forwarded IPC is always forwarded to the userland of the bound thread. That means, the [Thread](#) interface of the bound thread is not accessible via an IPC gate. The [IPC-Gate API](#) of an IPC gate is only accessible if the capability used has the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission (cf. previous paragraph). Conversely that means, if the capability used lacks the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission, [IPC-Gate API](#) calls are forwarded to the bound thread instead of being processed by the IPC gate itself. In a client/server scenario, a client should only get IPC gate capabilities without [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission so the client cannot tamper with the IPC gate.

When binding a thread to an IPC gate, a user-defined, kernel protected, machine-word sized payload called the IPC gate's *label* is assigned to the IPC gate (cf. [l4\\_rcv\\_ep\\_bind\\_thread\(\)](#)). When a send-only IPC or call IPC is forwarded via an IPC gate, the label provided by the sender is ignored and replaced by the IPC gate's label where the two least significant bits are the result of bitwise disjunction of the corresponding label bits with the [L4\\_CAP\\_FPAGE\\_S](#) and [L4\\_CAP\\_FPAGE\\_W](#) permissions of the capability used. Hence, the label provided via [l4\\_rcv\\_ep\\_bind\\_thread\(\)](#) should usually have its two least significant bits set to zero. The replaced label is only visible to the bound thread upon receive. However, the configured label of an IPC gate can also be queried via [l4\\_ipc\\_gate\\_get\\_infos\(\)](#) if the capability used has the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission.

When deleting an IPC gate or when unbinding it from a thread, the label of IPC already in flight won't be changed. To ensure that no IPC from this IPC gate is received by a thread with an unexpected label, [l4\\_thread\\_modify\\_sender\\_start\(\)](#) shall be used to change the labels of every pending IPC to that gate. This is also required if the label of an already bound IPC gate is changed. It is not necessary after binding the IPC gate to a thread for the first time.

When binding a new thread to an IPC gate that is currently bound, the same label should be used that was used with the old thread. Otherwise the old and the new thread need to synchronize to avoid IPC messages with unexpected labels.

#### Include File

```
#include <l4/sys/ipc_gate.h>
```

For the C++ interface refer to the [L4::ipc\\_gate](#) documentation.

#### See also

[Object Invocation](#)

Definition in file [ipc\\_gate.h](#).

## 16.502 ipc\_gate.h

[Go to the documentation of this file.](#)

```

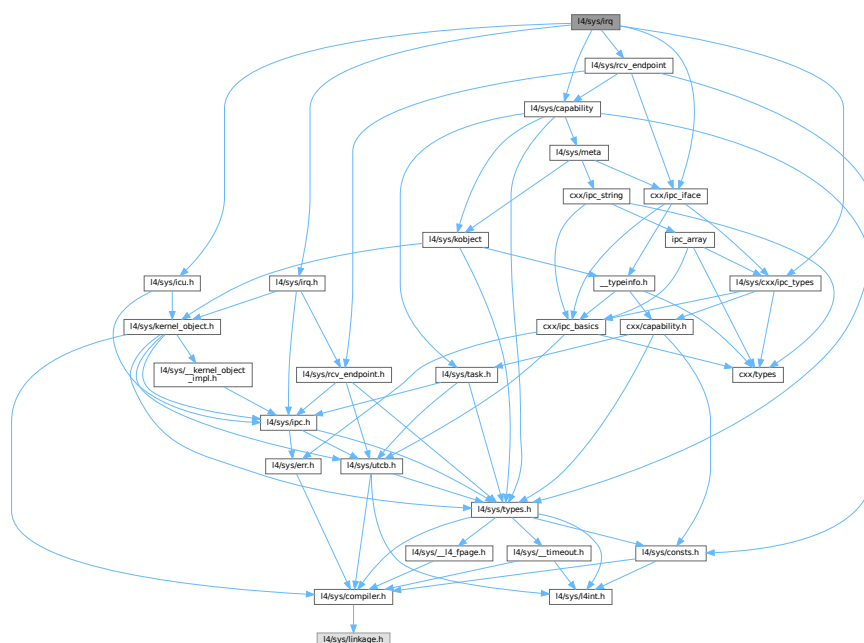
00001 /*
00002  * (c) 2009-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019
00080 #pragma once
00081
00082 #include <l4/sys/utcb.h>
00083 #include <l4/sys/types.h>
00084 #include <l4/sys/rcv_endpoint.h>
00085
00100 L4_INLINE l4_msgtag_t
00101 l4_ipc_gate_get_infos(l4_cap_idx_t gate, l4_umword_t *label);
00102
00107 L4_INLINE l4_msgtag_t
00108 l4_ipc_gate_get_infos_u(l4_cap_idx_t gate, l4_umword_t *label, l4_utcb_t *utcb);
00109
00116 enum l4_ipc_gate_ops
00117 {
00118     L4_IPC_GATE_BIND_OP      = 0x10,
00119     L4_IPC_GATE_GET_INFO_OP = 0x11,
00120 };
00121
00122
00123 /* IMPLEMENTATION -----*/
00124
00125 #include <l4/sys/ipc.h>
00126
00127 L4_INLINE l4_msgtag_t
00128 l4_ipc_gate_bind_thread_u(l4_cap_idx_t gate,
00129                          l4_cap_idx_t thread, l4_umword_t label,
00130                          l4_utcb_t *utcb)
00131 {
00132     return l4_rcv_ep_bind_thread_u(gate, thread, label, utcb);
00133 }
00134
00135 L4_INLINE l4_msgtag_t
00136 l4_ipc_gate_get_infos_u(l4_cap_idx_t gate, l4_umword_t *label, l4_utcb_t *utcb)
00137 {
00138     l4_msgtag_t tag;
00139     l4_msg_regst_t *m = l4_utcb_mr_u(utcb);
00140     m->mr[0] = L4_IPC_GATE_GET_INFO_OP;
00141     tag = l4_ipc_call(gate, utcb, l4_msgtag(L4_PROTO_KOBJECT, 1, 0, 0),
00142                     L4_IPC_NEVER);
00143     if (!l4_msgtag_has_error(tag) && l4_msgtag_label(tag) >= 0)
00144         *label = m->mr[0];
00145     return tag;
00146 }
00147
00148
00149
00150
00151 L4_INLINE l4_msgtag_t
00152 l4_ipc_gate_bind_thread(l4_cap_idx_t gate, l4_cap_idx_t thread,
00153                        l4_umword_t label)
00154 {
00155     return l4_rcv_ep_bind_thread_u(gate, thread, label, l4_utcb());
00156 }
00157
00158 L4_INLINE l4_msgtag_t
00159 l4_ipc_gate_get_infos(l4_cap_idx_t gate, l4_umword_t *label)
00160 {
00161     return l4_ipc_gate_get_infos_u(gate, label, l4_utcb());
00162 }

```

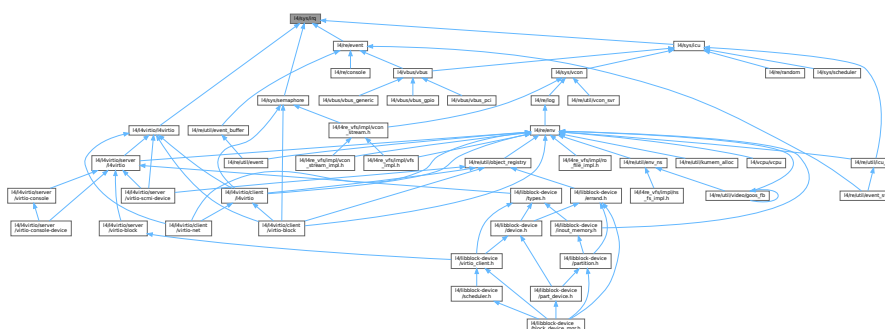
## 16.503 I4/sys/irq File Reference

## C++ Irq interface.

```
#include <l4/sys/icu.h>
#include <l4/sys/irq.h>
#include <l4/sys/capability>
#include <l4/sys/rcv_endpoint>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_types>
Include dependency graph for irq:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4::lrq_eoi`  
*Interface for sending an unmask message to an object.*

- struct [L4::Triggerable](#)  
*Interface that allows an object to be triggered by some source.*
- class [L4::Irq](#)  
*C++ [Irq](#) interface, see [IRQs](#) for the C interface.*
- struct [L4::Irq\\_mux](#)  
*IRQ multiplexer for shared IRQs.*
- class [L4::Icu](#)  
*C++ [Icu](#) interface, see [Interrupt controller](#) for the C interface.*
- class [L4::Icu::Info](#)  
*This class encapsulates information about an ICU.*

## Namespaces

- namespace [L4](#)  
[L4](#) low-level kernel interface.

## 16.503.1 Detailed Description

C++ Irq interface.

Definition in file [irq](#).

## 16.504 irq

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include <l4/sys/icu.h>
00028 #include <l4/sys/irq.h>
00029 #include <l4/sys/capability>
00030 #include <l4/sys/rcv_endpoint>
00031 #include <l4/sys/cxx/ipc_iface>
00032 #include <l4/sys/cxx/ipc_types>
00033
00034 namespace L4 {
00035
00048 class Irq_eoi : public Kobject_0t<Irq_eoi, L4::PROTO_EMPTY>
00049 {
00050 public:
00075     l4_msgtag_t unmask(unsigned irqnum, l4_umword_t *label = 0,
00076                       l4_timeout_t to = L4_IPC_NEVER,
00077                       l4_utcb_t *utcb = l4_utcb()) noexcept
00078     {
00079         return l4_icu_control_u(cap(), irqnum, L4_ICU_CTL_UNMASK, label, to, utcb);
00079     }
00079 }
```

```

00080     }
00081 };
00082
00090 struct Triggerable : Kobject_t<Triggerable, Irq_eoi, L4_PROTO_IRQ>
00091 {
00102     l4_msgtag_t trigger(l4_utcb_t *utcb = l4_utcb()) noexcept
00103     { return l4_irq_trigger_u(cap(), utcb); }
00104 };
00105
00131 class Irq : public Kobject_2t<Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER>
00132 {
00133 public:
00134     using Triggerable::unmask;
00135
00148     l4_msgtag_t detach(l4_utcb_t *utcb = l4_utcb()) noexcept
00149     { return l4_irq_detach_u(cap(), utcb); }
00150
00151
00163     l4_msgtag_t receive(l4_timeout_t timeout = L4_IPC_NEVER,
00164                        l4_utcb_t *utcb = l4_utcb()) noexcept
00165     { return l4_irq_receive_u(cap(), timeout, utcb); }
00166
00176     l4_msgtag_t wait(l4_umword_t *label, l4_timeout_t timeout = L4_IPC_NEVER,
00177                    l4_utcb_t *utcb = l4_utcb()) noexcept
00178     { return unmask(-1, label, timeout, utcb); }
00179
00193     l4_msgtag_t unmask(l4_utcb_t *utcb = l4_utcb()) noexcept
00194     { return unmask(-1, 0, L4_IPC_NEVER, utcb); }
00195 };
00196
00212 struct Irq_mux : Kobject_t<Irq_mux, Triggerable, L4_PROTO_IRQ_MUX>
00213 {
00227     l4_msgtag_t chain(Cap<Triggerable> const &slave,
00228                     l4_utcb_t *utcb = l4_utcb()) noexcept
00229     { return l4_irq_mux_chain_u(cap(), slave.cap(), utcb); }
00230 };
00231
00232
00257 class Icu :
00258     public Kobject_t<Icu, Irq_eoi, L4_PROTO_IRQ,
00259                     Type_info::Demand_t<1> >
00260 {
00261 public:
00262     enum Mode
00263     {
00264         F_none           = L4_IRQ_F_NONE,
00265         F_level_high     = L4_IRQ_F_LEVEL_HIGH,
00266         F_level_low      = L4_IRQ_F_LEVEL_LOW,
00267         F_pos_edge       = L4_IRQ_F_POS_EDGE,
00268         F_neg_edge       = L4_IRQ_F_NEG_EDGE,
00269         F_both_edge      = L4_IRQ_F_BOTH_EDGE,
00270         F_mask           = L4_IRQ_F_MASK,
00271
00272         F_set_wakeup     = L4_IRQ_F_SET_WAKEUP,
00273         F_clear_wakeup   = L4_IRQ_F_CLEAR_WAKEUP,
00274     };
00275
00276     enum Flags
00277     {
00278         F_msi = L4_ICU_FLAG_MSI
00279     };
00280
00284     class Info : public l4_icu_info_t
00285     {
00286     public:
00288         bool supports_msi() const noexcept { return features & F_msi; }
00289     };
00290
00316     l4_msgtag_t bind(unsigned irqnum, L4::Cap<Triggerable> irq,
00317                    l4_utcb_t *utcb = l4_utcb()) noexcept
00318     { return l4_icu_bind_u(cap(), irqnum, irq.cap(), utcb); }
00319
00320     L4_RPC_NF_OP(
00321         L4_ICU_OP_BIND,
00322         l4_msgtag_t, bind, (l4_umword_t irqnum, Ipc::Cap<Irq> irq)
00323     );
00324
00334     l4_msgtag_t unbind(unsigned irqnum, L4::Cap<Triggerable> irq,
00335                      l4_utcb_t *utcb = l4_utcb()) noexcept
00336     { return l4_icu_unbind_u(cap(), irqnum, irq.cap(), utcb); }
00337
00338     L4_RPC_NF_OP(
00339         L4_ICU_OP_UNBIND,
00340         l4_msgtag_t, unbind, (l4_umword_t irqnum, Ipc::Cap<Irq> irq)
00341     );
00342
00351     l4_msgtag_t info(l4_icu_info_t *info, l4_utcb_t *utcb = l4_utcb()) noexcept

```

```

00352 { return l4_icu_info_u(cap(), info, utcb); }
00353
00354 struct _Info { l4_umword_t features, nr_irqs, nr_msis; };
00355 L4_RPC_NF_OP(L4_ICU_OP_INFO, l4_msgtag_t, info, (_Info *info));
00356
00369 L4_INLINE_RPC_OP(L4_ICU_OP_MSI_INFO,
00370     l4_msgtag_t, msi_info, (l4_umword_t irqnum, l4_uint64_t source,
00371     l4_icu_msi_info_t *msi_info));
00372
00376 l4_msgtag_t control(unsigned irqnum, unsigned op, l4_umword_t *label,
00377     l4_timeout_t to, l4_utcb_t *utcb = l4_utcb()) noexcept
00378 { return l4_icu_control_u(cap(), irqnum, op, label, to, utcb); }
00379
00399 l4_msgtag_t mask(unsigned irqnum,
00400     l4_umword_t *label = 0,
00401     l4_timeout_t to = L4_IPC_NEVER,
00402     l4_utcb_t *utcb = l4_utcb()) noexcept
00403 { return l4_icu_mask_u(cap(), irqnum, label, to, utcb); }
00404
00405 L4_RPC_NF_OP(
00406     L4_ICU_OP_MASK,
00407     l4_msgtag_t, mask, (l4_umword_t irqnum),
00408     L4::lpc::Send_only
00409 );
00410
00411
00412 L4_RPC_NF_OP(
00413     L4_ICU_OP_UNMASK,
00414     l4_msgtag_t, unmask, (l4_umword_t irqnum),
00415     L4::lpc::Send_only
00416 );
00417
00427 l4_msgtag_t set_mode(unsigned irqnum, l4_umword_t mode,
00428     l4_utcb_t *utcb = l4_utcb()) noexcept
00429 { return l4_icu_set_mode_u(cap(), irqnum, mode, utcb); }
00430
00431 L4_RPC_NF_OP(
00432     L4_ICU_OP_SET_MODE,
00433     l4_msgtag_t, set_mode, (l4_umword_t irqnum, l4_umword_t mode)
00434 );
00435
00436 typedef L4::Typeid::Rpcsys<
00437     bind_t, unbind_t, info_t, msi_info_t, unmask_t, mask_t, set_mode_t
00438 > Rpcsys;
00439 };
00440
00441 }

```

## 16.505 amd64/l4/util/irq.h File Reference

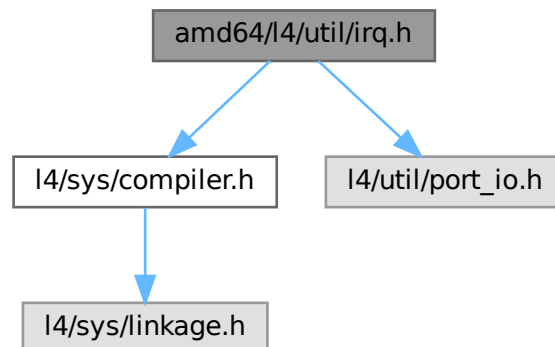
some PIC and hardware interrupt related functions

```

#include <l4/sys/compiler.h>
#include <l4/util/port_io.h>

```

Include dependency graph for irq.h:



## Functions

- void [l4util\\_irq\\_acknowledge](#) (unsigned int irq)  
*Acknowledge IRQ at PIC in fully special nested mode.*

## 16.505.1 Detailed Description

some PIC and hardware interrupt related functions

### Date

2003

### Author

Jork Loeser [jork.loeser@inf.tu-dresden.de](mailto:jork.loeser@inf.tu-dresden.de) Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [irq.h](#).

## 16.505.2 Function Documentation

### 16.505.2.1 l4util\_irq\_acknowledge()

```
void l4util_irq_acknowledge (  
    unsigned int irq ) [inline]
```

Acknowledge IRQ at PIC in fully special nested mode.

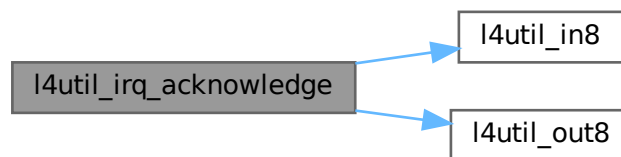
## Parameters

<i>irq</i>	number of interrupt to acknowledge
------------	------------------------------------

Definition at line 66 of file [irq.h](#).

References [l4util\\_in8\(\)](#), and [l4util\\_out8\(\)](#).

Here is the call graph for this function:



## 16.506 irq.h

[Go to the documentation of this file.](#)

```

00001
00010 /*
00011  * (c) 2003-2009 Author(s)
00012  *     economic rights: Technische Universität Dresden (Germany)
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU Lesser General Public License 2.1.
00015  * Please see the COPYING-LGPL-2.1 file for details.
00016  */
00017
00018 #ifndef __L4_IRQ_H__
00019 #define __L4_IRQ_H__
00020
00021 #include <l4/sys/compiler.h>
00022 #include <l4/util/port_io.h>
00023
00024 EXTERN_C_BEGIN
00025
00029 L4_INLINE void
00030 l4util_irq_acknowledge(unsigned int irq);
00031
00034 static inline void
00035 l4util_cli(void)
00036 {
00037     __asm__ __volatile__ ("cli" : : : "memory");
00038 }
00039
00042 static inline void
00043 l4util_sti(void)
00044 {
00045     __asm__ __volatile__ ("sti" : : : "memory");
00046 }
00047
00051 static inline void
00052 l4util_flags_save(l4_umword_t *flags)
00053 {
00054     __asm__ __volatile__ ("pushf ; popq %0" : "=g" (*flags) : : "memory");
00055 }
00056
00059 static inline void
00060 l4util_flags_restore(l4_umword_t *flags)
00061 {
00062     __asm__ __volatile__ ("pushq %0 ; popf" : : "g" (*flags) : "memory");

```

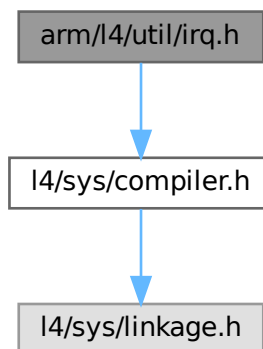


```
00063 }
00064
00065 L4_INLINE void
00066 l4util_irq_acknowledge(unsigned int irq)
00067 {
00068     if (irq > 7)
00069     {
00070         l4util_out8(0x60+(irq & 7), 0xA0);
00071         l4util_out8(0x0B, 0xA0);
00072         if (l4util_in8(0xA0) == 0)
00073             l4util_out8(0x60 + 2, 0x20);
00074     }
00075     else
00076         l4util_out8(0x60+irq, 0x20);    /* acknowledge the irq */
00077 };
00078
00079 EXTERN_C_END
00080
00081 #endif
```

## 16.507 arm/l4/util/irq.h File Reference

ARM specific implementation of irq functions.

#include <l4/sys/compiler.h>  
Include dependency graph for irq.h:



### 16.507.1 Detailed Description

ARM specific implementation of irq functions.

Do not use.

Definition in file [irq.h](#).

## 16.508 irq.h

[Go to the documentation of this file.](#)

```

00001
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU Lesser General Public License 2.1.
00014  * Please see the COPYING-LGPL-2.1 file for details.
00015  */
00016 #ifndef __L4UTIL__ARCH_ARCH__IRQ_H__
00017 #define __L4UTIL__ARCH_ARCH__IRQ_H__
00018
00019 #ifdef __GNUC__
00020
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00025 L4_INLINE void l4util_cli (void);
00026 L4_INLINE void l4util_sti (void);
00027 L4_INLINE void l4util_flags_save(l4_umword_t *flags);
00028 L4_INLINE void l4util_flags_restore(l4_umword_t *flags);
00029
00030 L4_INLINE
00031 void
00032 l4util_cli(void)
00033 {
00034     extern void __do_not_use_l4util_cli(void);
00035     __do_not_use_l4util_cli();
00036 }
00037
00038
00039 L4_INLINE
00040 void
00041 l4util_sti(void)
00042 {
00043     extern void __do_not_use_l4util_sti(void);
00044     __do_not_use_l4util_sti();
00045 }
00046
00047
00048 L4_INLINE
00049 void
00050 l4util_flags_save(l4_umword_t *flags)
00051 {
00052     (void) flags;
00053     extern void __do_not_use_l4util_flags_save(void);
00054     __do_not_use_l4util_flags_save();
00055 }
00056
00057 L4_INLINE
00058 void
00059 l4util_flags_restore(l4_umword_t *flags)
00060 {
00061     (void) flags;
00062     extern void __do_not_use_l4util_flags_restore(void);
00063     __do_not_use_l4util_flags_restore();
00064 }
00065
00066 EXTERN_C_END
00067
00068 #endif // __GNUC__
00069
00070 #endif /* ! __L4UTIL__ARCH_ARCH__IRQ_H__ */

```

## 16.509 I4/irq/irq.h File Reference

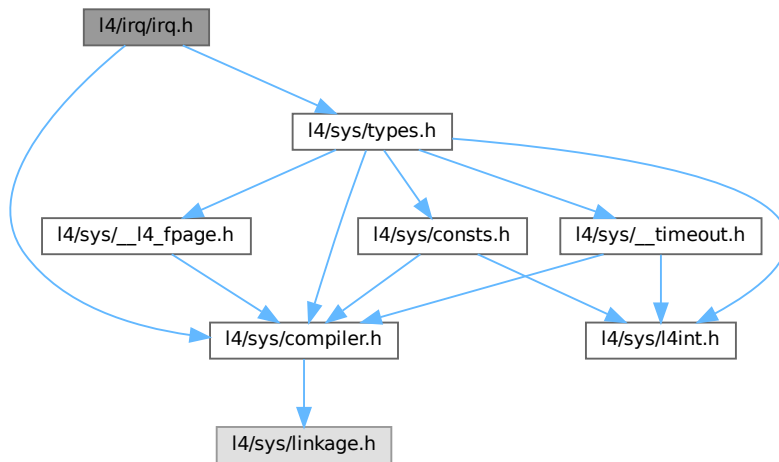
IRQ handling routines.

```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>

```

Include dependency graph for irq.h:



## Functions

- `l4irq_t * l4irq_attach` (int irqnum)  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq_attach_ft` (int irqnum, unsigned mode)  
*Attach/connect to IRQ using given type.*
- `l4irq_t * l4irq_attach_thread` (int irqnum, `l4_cap_idx_t` to\_thread)  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq_attach_thread_ft` (int irqnum, `l4_cap_idx_t` to\_thread, unsigned mode)  
*Attach/connect to IRQ using given type.*
- `long l4irq_wait` (`l4irq_t *`irq)  
*Wait for specified IRQ.*
- `long l4irq_unmask_and_wait_any` (`l4irq_t *`unmask\_irq, `l4irq_t **`ret\_irq)  
*Unmask a specific IRQ and wait for any attached IRQ.*
- `long l4irq_wait_any` (`l4irq_t **`irq)  
*Wait for any attached IRQ.*
- `long l4irq_unmask` (`l4irq_t *`irq)  
*Unmask a specific IRQ.*
- `long l4irq_detach` (`l4irq_t *`irq)  
*Detach from IRQ.*
- `l4irq_t * l4irq_request` (int irqnum, `void(*isr_handler)(void *)`, `void *isr_data`, int irq\_thread\_prio, unsigned mode)  
*Attach asynchronous ISR handler to IRQ.*
- `long l4irq_release` (`l4irq_t *`irq)  
*Release asynchronous ISR handler and free resources.*
- `l4irq_t * l4irq_attach_cap` (`l4_cap_idx_t` irqcap)  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq_attach_cap_ft` (`l4_cap_idx_t` irqcap, unsigned mode)  
*Attach/connect to IRQ using given type.*
- `l4irq_t * l4irq_attach_thread_cap` (`l4_cap_idx_t` irqcap, `l4_cap_idx_t` to\_thread)

*Attach/connect to IRQ.*

- `l4irq_t * l4irq_attach_thread_cap_ft (l4_cap_idx_t irqcap, l4_cap_idx_t to_thread, unsigned mode)`

*Attach/connect to IRQ using given type.*

- `l4irq_t * l4irq_request_cap (l4_cap_idx_t irqcap, void(*isr_handler)(void *), void *isr_data, int irq_thread_↵prio, unsigned mode)`

*Attach asynchronous ISR handler to IRQ.*

### 16.509.1 Detailed Description

IRQ handling routines.

Definition in file [irq.h](#).

## 16.510 irq.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Henning Schild <hschild@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/compiler.h>
00017 #include <l4/sys/types.h>
00018
00019 __BEGIN_DECLS
00020
00028 struct l4irq_t;
00029 typedef struct l4irq_t l4irq_t;
00030
00041 L4_CV l4irq_t *
00042 l4irq_attach(int irqnum);
00043
00055 L4_CV l4irq_t *
00056 l4irq_attach_ft(int irqnum, unsigned mode);
00057
00068 L4_CV l4irq_t *
00069 l4irq_attach_thread(int irqnum, l4_cap_idx_t to_thread);
00070
00082 L4_CV l4irq_t *
00083 l4irq_attach_thread_ft(int irqnum, l4_cap_idx_t to_thread,
00084                        unsigned mode);
00085
00093 L4_CV long
00094 l4irq_wait(l4irq_t *irq);
00095
00105 L4_CV long
00106 l4irq_unmask_and_wait_any(l4irq_t *unmask_irq, l4irq_t **ret_irq);
00107
00115 L4_CV long
00116 l4irq_wait_any(l4irq_t **irq);
00117
00128 L4_CV long
00129 l4irq_unmask(l4irq_t *irq);
00130
00138 L4_CV long
00139 l4irq_detach(l4irq_t *irq);
00140
00141
00142
00143 /*****
00165 L4_CV l4irq_t *
00166 l4irq_request(int irqnum, void (*isr_handler)(void *), void *isr_data,
00167              int irq_thread_prio, unsigned mode);
00168
00176 L4_CV long

```

```

00177 l4irq_release(l4irq_t *irq);
00178
00179
00180
00181 /*****
00182 /*****
00183
00199 L4_CV l4irq_t *
00200 l4irq_attach_cap(l4_cap_idx_t irqcap);
00201
00213 L4_CV l4irq_t *
00214 l4irq_attach_cap_ft(l4_cap_idx_t irqcap, unsigned mode);
00215
00226 L4_CV l4irq_t *
00227 l4irq_attach_thread_cap(l4_cap_idx_t irqcap, l4_cap_idx_t to_thread);
00228
00240 L4_CV l4irq_t *
00241 l4irq_attach_thread_cap_ft(l4_cap_idx_t irqcap, l4_cap_idx_t to_thread,
00242                             unsigned mode);
00243
00244 /*****
00265 L4_CV l4irq_t *
00266 l4irq_request_cap(l4_cap_idx_t irqcap,
00267                  void (*isr_handler)(void *), void *isr_data,
00268                  int irq_thread_prio, unsigned mode);
00269
00270 __END_DECLS

```

## 16.511 l4/sys/irq.h File Reference

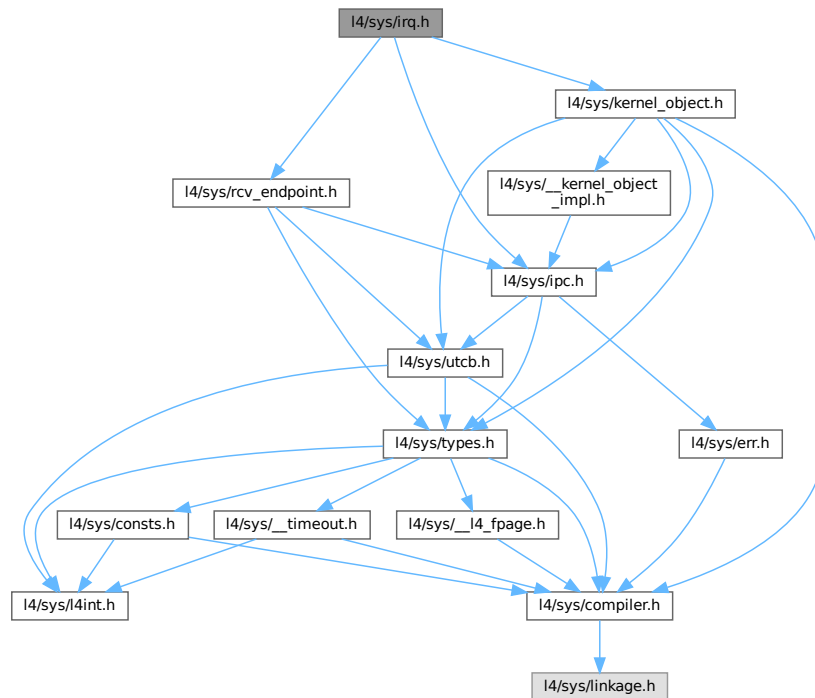
C Irq interface.

```
#include <l4/sys/kernel_object.h>
```

```
#include <l4/sys/ipc.h>
```

```
#include <l4/sys/rcv_endpoint.h>
```

Include dependency graph for irq.h:





## 16.512 irq.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/sys/kernel_object.h>
00028 #include <l4/sys/ipc.h>
00029 #include <l4/sys/rcv_endpoint.h>
00030
00078 L4_INLINE l4_msgtag_t
00079 l4_irq_mux_chain(l4_cap_idx_t irq, l4_cap_idx_t slave) L4_NOTHROW;
00080
00087 L4_INLINE l4_msgtag_t
00088 l4_irq_mux_chain_u(l4_cap_idx_t irq, l4_cap_idx_t slave,
00089                  l4_utcb_t *utcb) L4_NOTHROW;
00090
00104 L4_INLINE l4_msgtag_t
00105 l4_irq_detach(l4_cap_idx_t irq) L4_NOTHROW;
00106
00113 L4_INLINE l4_msgtag_t
00114 l4_irq_detach_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW;
00115
00116
00132 L4_INLINE l4_msgtag_t
00133 l4_irq_trigger(l4_cap_idx_t irq) L4_NOTHROW;
00134
00141 L4_INLINE l4_msgtag_t
00142 l4_irq_trigger_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW;
00143
00153 L4_INLINE l4_msgtag_t
00154 l4_irq_receive(l4_cap_idx_t irq, l4_timeout_t to) L4_NOTHROW;
00155
00162 L4_INLINE l4_msgtag_t
00163 l4_irq_receive_u(l4_cap_idx_t irq, l4_timeout_t timeout, l4_utcb_t *utcb) L4_NOTHROW;
00164
00175 L4_INLINE l4_msgtag_t
00176 l4_irq_wait(l4_cap_idx_t irq, l4_umword_t *label,
00177            l4_timeout_t to) L4_NOTHROW;
00178
00185 L4_INLINE l4_msgtag_t
00186 l4_irq_wait_u(l4_cap_idx_t irq, l4_umword_t *label,
00187             l4_timeout_t timeout, l4_utcb_t *utcb) L4_NOTHROW;
00188
00199 L4_INLINE l4_msgtag_t
00200 l4_irq_unmask(l4_cap_idx_t irq) L4_NOTHROW;
00201
00208 L4_INLINE l4_msgtag_t
00209 l4_irq_unmask_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW;
00210
00214 enum L4_irq_sender_op
00215 {
00216     L4_IRQ_SENDER_OP_RESERVED1 = 0, // Ex ATTACH
00217     L4_IRQ_SENDER_OP_DETACH    = 1
00218 };
00219
00223 enum L4_irq_mux_op
00224 {
00225     L4_IRQ_MUX_OP_CHAIN = 0
00226 };
00227
00231 enum L4_irq_op
00232 {
00233     L4_IRQ_OP_TRIGGER    = 2,
00234     L4_IRQ_OP_EOI        = 4
00235 };

```

```

00236
00237 /*****
00238  * Implementations
00239  */
00240
00241 L4_INLINE l4_msgtag_t
00242 l4_irq_mux_chain_u(l4_cap_idx_t irq, l4_cap_idx_t slave,
00243                   l4_utcb_t *utcb) L4_NOTHROW
00244 {
00245     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00246     m->mr[0] = L4_IRQ_MUX_OP_CHAIN;
00247     m->mr[1] = l4_map_obj_control(0, 0);
00248     m->mr[2] = l4_obj_fpage(slave, 0, L4_CAP_FPAGE_RWS).raw;
00249     return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ_MUX, 1, 1, 0),
00250                       L4_IPC_NEVER);
00251 }
00252
00253 L4_INLINE l4_msgtag_t
00254 l4_irq_detach_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW
00255 {
00256     l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_SENDER_OP_DETACH;
00257     return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ_SENDER, 1, 0, 0),
00258                       L4_IPC_NEVER);
00259 }
00260
00261 L4_INLINE l4_msgtag_t
00262 l4_irq_trigger_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW
00263 {
00264     return l4_ipc_send(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 0, 0, 0),
00265                       L4_IPC_BOTH_TIMEOUT_0);
00266 }
00267
00268 L4_INLINE l4_msgtag_t
00269 l4_irq_receive_u(l4_cap_idx_t irq, l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
00270 {
00271     l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
00272     return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0), to);
00273 }
00274
00275 L4_INLINE l4_msgtag_t
00276 l4_irq_wait_u(l4_cap_idx_t irq, l4_umword_t *label,
00277              l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
00278 {
00279     l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
00280     return l4_ipc_send_and_wait(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0),
00281                                label, to);
00282 }
00283
00284 L4_INLINE l4_msgtag_t
00285 l4_irq_unmask_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW
00286 {
00287     l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
00288     return l4_ipc_send(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0), L4_IPC_NEVER);
00289 }
00290
00291
00292 L4_INLINE l4_msgtag_t
00293 l4_irq_mux_chain(l4_cap_idx_t irq, l4_cap_idx_t slave) L4_NOTHROW
00294 {
00295     return l4_irq_mux_chain_u(irq, slave, l4_utcb());
00296 }
00297
00298 L4_INLINE l4_msgtag_t
00299 l4_irq_detach(l4_cap_idx_t irq) L4_NOTHROW
00300 {
00301     return l4_irq_detach_u(irq, l4_utcb());
00302 }
00303
00304 L4_INLINE l4_msgtag_t
00305 l4_irq_trigger(l4_cap_idx_t irq) L4_NOTHROW
00306 {
00307     return l4_irq_trigger_u(irq, l4_utcb());
00308 }
00309
00310 L4_INLINE l4_msgtag_t
00311 l4_irq_receive(l4_cap_idx_t irq, l4_timeout_t to) L4_NOTHROW
00312 {
00313     return l4_irq_receive_u(irq, to, l4_utcb());
00314 }
00315
00316 L4_INLINE l4_msgtag_t
00317 l4_irq_wait(l4_cap_idx_t irq, l4_umword_t *label,
00318            l4_timeout_t to) L4_NOTHROW
00319 {
00320     return l4_irq_wait_u(irq, label, to, l4_utcb());
00321 }
00322

```

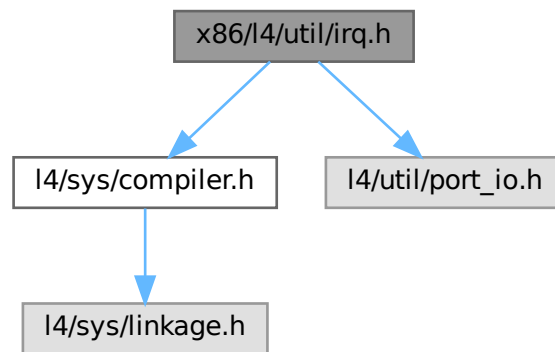


```
00323 L4_INLINE l4_msgtag_t
00324 l4_irq_unmask(l4_cap_idx_t irq) L4_NOTHROW
00325 {
00326     return l4_irq_unmask_u(irq, l4_uctb());
00327 }
00328
```

## 16.513 x86/l4/util/irq.h File Reference

some PIC and hardware interrupt related functions

```
#include <l4/sys/compiler.h>
#include <l4/util/port_io.h>
Include dependency graph for irq.h:
```



### Functions

- void `l4util_irq_acknowledge` (unsigned int irq)  
*Acknowledge IRQ at PIC in fully special nested mode.*

### 16.513.1 Detailed Description

some PIC and hardware interrupt related functions

#### Date

2003

#### Author

Jork Loeser [jork.loeser@inf.tu-dresden.de](mailto:jork.loeser@inf.tu-dresden.de) Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [irq.h](#).

## 16.513.2 Function Documentation

### 16.513.2.1 l4util\_irq\_acknowledge()

```
void l4util_irq_acknowledge (
    unsigned int irq ) [inline]
```

Acknowledge IRQ at PIC in fully special nested mode.

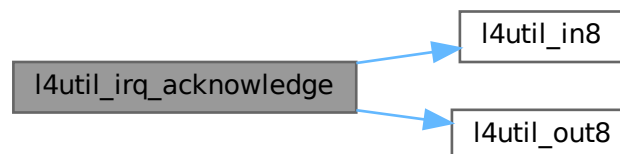
#### Parameters

<i>irq</i>	number of interrupt to acknowledge
------------	------------------------------------

Definition at line 66 of file [irq.h](#).

References [l4util\\_in8\(\)](#), and [l4util\\_out8\(\)](#).

Here is the call graph for this function:



## 16.514 irq.h

[Go to the documentation of this file.](#)

```
00001
00010 /*
00011  * (c) 2003-2009 Author(s)
00012  *     economic rights: Technische Universität Dresden (Germany)
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU Lesser General Public License 2.1.
00015  * Please see the COPYING-LGPL-2.1 file for details.
00016  */
00017
00018 #ifndef __L4_IRQ_H__
00019 #define __L4_IRQ_H__
00020
00021 #include <l4/sys/compiler.h>
00022 #include <l4/util/port_io.h>
00023
00024 EXTERN_C_BEGIN
00025
00029 L4_INLINE void
00030 l4util_irq_acknowledge(unsigned int irq);
00031
00034 static inline void
00035 l4util_cli (void)
00036 {
00037     __asm__ __volatile__ ("cli" : : : "memory");
00038 }
00039
00042 static inline void
```

```

00043 l4util_sti (void)
00044 {
00045     __asm__ __volatile__ ("sti" : : : "memory");
00046 }
00047
00051 static inline void
00052 l4util_flags_save (l4_umword_t *flags)
00053 {
00054     __asm__ __volatile__ ("pushfl ; popl %0" : "=g" (*flags) : : "memory");
00055 }
00056
00059 static inline void
00060 l4util_flags_restore (l4_umword_t *flags)
00061 {
00062     __asm__ __volatile__ ("pushl %0 ; popfl" : : "g" (*flags) : "memory");
00063 }
00064
00065 L4_INLINE void
00066 l4util_irq_acknowledge(unsigned int irq)
00067 {
00068     if (irq > 7)
00069     {
00070         l4util_out8(0x60+(irq & 7), 0xA0);
00071         l4util_out8(0x0B, 0xA0);
00072         if (l4util_in8(0xA0) == 0)
00073             l4util_out8(0x60 + 2, 0x20);
00074     }
00075     else
00076         l4util_out8(0x60+irq, 0x20);    /* acknowledge the irq */
00077 };
00078
00079 EXTERN_C_END
00080
00081 #endif

```

## 16.515 l4/sys/kdebug.h File Reference

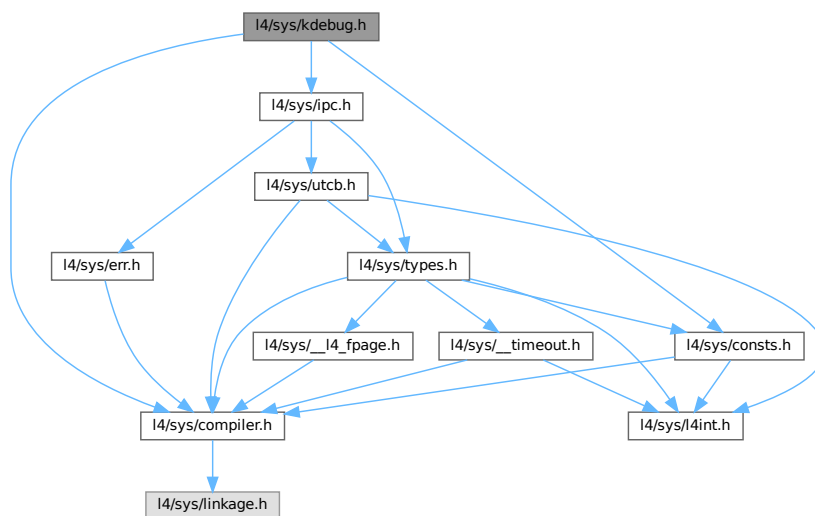
Functionality for invoking the kernel debugger.

```

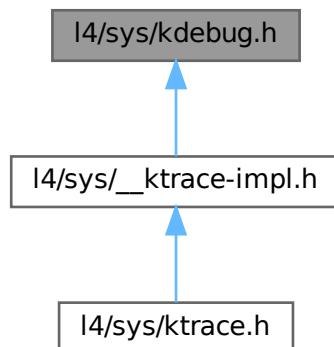
#include <l4/sys/compiler.h>
#include <l4/sys/consts.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for kdebug.h:



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum [l4\\_kdebug\\_ops\\_t](#)  
*Op-codes for operations that can be invoked on the base debugger capability.*

## Functions

- void [enter\\_kdebug](#) (char const \*text) [L4\\_NOTHROW](#)  
*Enter the kernel debugger.*
- [l4\\_msgtag\\_t \\_\\_kdebug\\_op](#) (unsigned op) [L4\\_NOTHROW](#)  
*Invoke a nullary operation on the base debugger capability.*
- [l4\\_msgtag\\_t \\_\\_kdebug\\_text](#) (unsigned op, char const \*text, unsigned len) [L4\\_NOTHROW](#)  
*Invoke a text output operation on the base debugger capability.*
- [l4\\_msgtag\\_t \\_\\_kdebug\\_3\\_text](#) (unsigned op, char const \*text, unsigned len, [l4\\_umword\\_t](#) v1, [l4\\_umword\\_t](#) v2, [l4\\_umword\\_t](#) v3) [L4\\_NOTHROW](#)  
*Invoke a text output operation with 3 additional machine word arguments on the base debugger capability.*
- [l4\\_msgtag\\_t \\_\\_kdebug\\_op\\_1](#) (unsigned op, [l4\\_mword\\_t](#) val) [L4\\_NOTHROW](#)  
*Invoke an unary operation on the base debugger capability.*
- void [outnstring](#) (char const \*text, unsigned len)  
*Output a fixed-length string via the kernel debugger.*
- void [outstring](#) (char const \*text)  
*Output a string via the kernel debugger.*
- void [outchar](#) (char c)  
*Output a single character via the kernel debugger.*
- void [outumword](#) ([l4\\_umword\\_t](#) number)  
*Output a hexadecimal unsigned machine word via the kernel debugger.*
- void [outhex64](#) ([l4\\_uint64\\_t](#) number)  
*Output a 64-bit unsigned hexadecimal number via the kernel debugger.*
- void [outhex32](#) ([l4\\_uint32\\_t](#) number)  
*Output a 32-bit unsigned hexadecimal number via the kernel debugger.*
- void [outhex20](#) ([l4\\_uint32\\_t](#) number)

- Output a 20-bit unsigned hexadecimal number via the kernel debugger.*
- void [outhex16](#) ([l4\\_uint16\\_t](#) number)
- Output a 16-bit unsigned hexadecimal number via the kernel debugger.*
- void [outhex12](#) ([l4\\_uint16\\_t](#) number)
- Output a 12-bit unsigned hexadecimal number via the kernel debugger.*
- void [outhex8](#) ([l4\\_uint8\\_t](#) number)
- Output an 8-bit unsigned hexadecimal number via the kernel debugger.*
- void [outdec](#) ([l4\\_mword\\_t](#) number)
- Output a decimal unsigned machine word via the kernel debugger.*

## 16.515.1 Detailed Description

Functionality for invoking the kernel debugger.

Definition in file [kdebug.h](#).

## 16.515.2 Enumeration Type Documentation

### 16.515.2.1 l4\_kdebug\_ops\_t

```
enum l4\_kdebug\_ops\_t
```

Op-codes for operations that can be invoked on the base debugger capability.

See also [\\_\\_ktrace-impl.h](#) for additional op-codes.

Definition at line 42 of file [kdebug.h](#).

## 16.515.3 Function Documentation

### 16.515.3.1 \_\_kdebug\_3\_text()

```
l4\_msgtag\_t __kdebug_3_text (
    unsigned op,
    char const * text,
    unsigned len,
    l4\_umword\_t v1,
    l4\_umword\_t v2,
    l4\_umword\_t v3 ) [inline]
```

Invoke a text output operation with 3 additional machine word arguments on the base debugger capability.

#### Parameters

<i>op</i>	Text output operation code from <a href="#">l4_kdebug_ops_t</a> or a value above 0x200 used by the kernel trace buffer implementation ( <a href="#">__ktrace-impl.h</a> ).
<i>text</i>	Output string.
<i>len</i>	Length of the output string. The maximum length is limited to L4_UTCB_GENERIC_DATA_SIZE - 5 machine words. Output strings longer than this limit will be cropped.
<i>v1</i>	First machine word argument.
<i>v2</i>	Second machine word argument.
<i>v3</i>	Third machine word argument.

## Return values

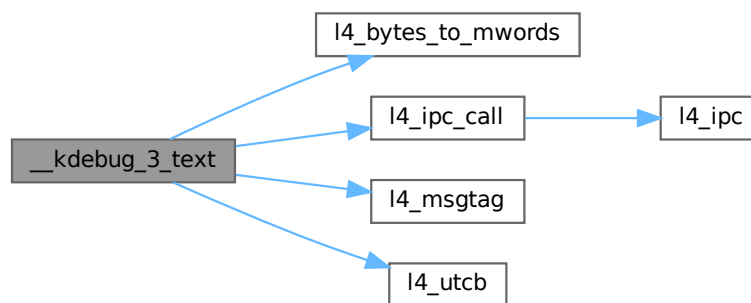
<i>Message</i>	tag returned from the IPC on the base debugger capability.
----------------	--

Definition at line 137 of file [kdebug.h](#).

References [L4\\_BASE\\_DEBUGGER\\_CAP](#), [l4\\_bytes\\_to\\_mwords\(\)](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_DEBUGGER](#), [l4\\_utcb\(\)](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [fiasco\\_tbuf\\_log\\_3val\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.515.3.2 \_\_kdebug\_op()

```
l4_msgtag_t __kdebug_op (
    unsigned op ) [inline]
```

Invoke a nullary operation on the base debugger capability.

## Parameters

<i>op</i>	Nullary operation code from <a href="#">l4_kdebug_ops_t</a> or a value above 0x200 used by the kernel trace buffer implementation ( <a href="#">__ktrace-impl.h</a> ).
-----------	--

## Return values

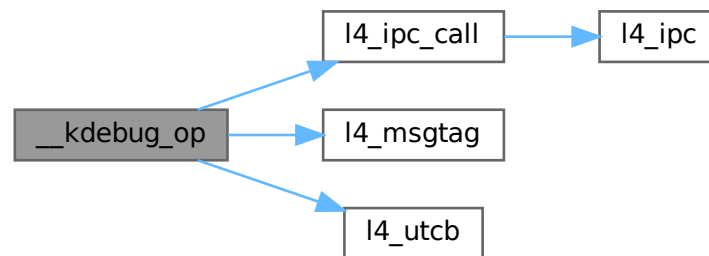
<i>Message</i>	tag returned from the IPC on the base debugger capability.
----------------	--

Definition at line 66 of file [kdebug.h](#).

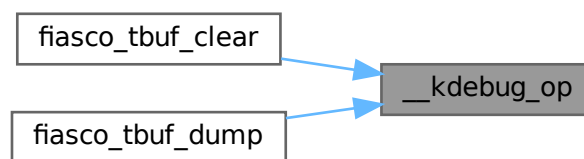
References [L4\\_BASE\\_DEBUGGER\\_CAP](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_DEBUGGER](#), [l4\\_utcb\(\)](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [fiasco\\_tbuf\\_clear\(\)](#), and [fiasco\\_tbuf\\_dump\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.515.3.3 \_\_kdebug\_op\_1()

```

l4_msgtag_t __kdebug_op_1 (
    unsigned op,
    l4_mword_t val ) [inline]
  
```

Invoke an unary operation on the base debugger capability.

## Parameters

<i>op</i>	Unary operation code from <a href="#">l4_kdebug_ops_t</a> or a value above 0x200 used by the kernel trace buffer implementation ( <a href="#">__ktrace-impl.h</a> ).
<i>val</i>	Machine word argument to the unary operation.

## Return values

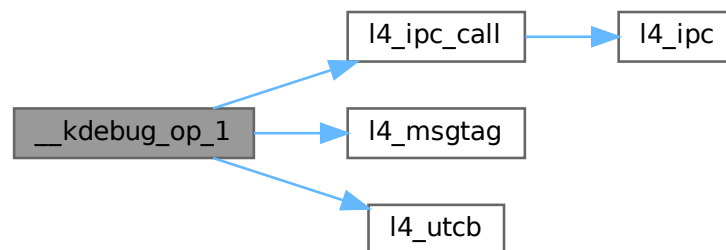
<i>Message</i>	tag returned from the IPC on the base debugger capability.
----------------	--

Definition at line 174 of file [kdebug.h](#).

References [L4\\_BASE\\_DEBUGGER\\_CAP](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_DEBUGGER](#), [l4\\_utcb\(\)](#), and [l4\\_msg\\_regs\\_t::mr](#).

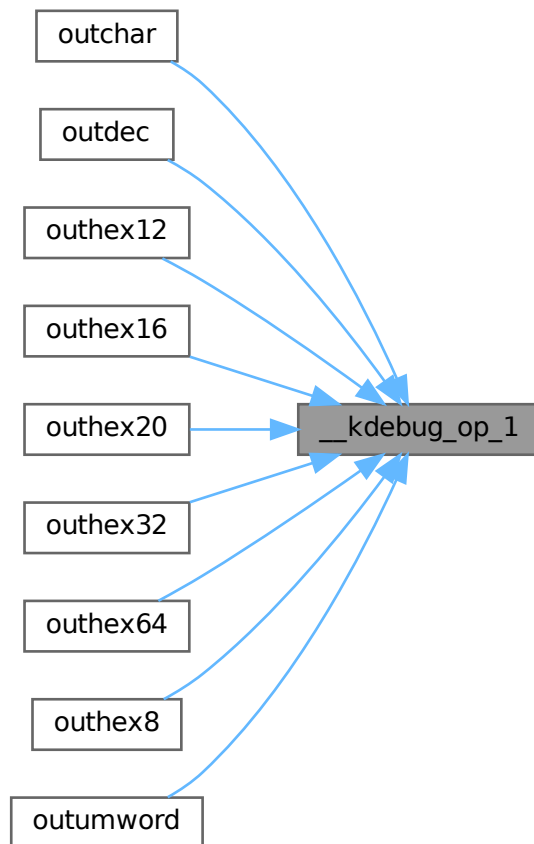
Referenced by [outchar\(\)](#), [outdec\(\)](#), [outhex12\(\)](#), [outhex16\(\)](#), [outhex20\(\)](#), [outhex32\(\)](#), [outhex64\(\)](#), [outhex8\(\)](#), and [outumword\(\)](#).

Here is the call graph for this function:





Here is the caller graph for this function:



#### 16.515.3.4 \_\_kdebug\_text()

```

l4_msgtag_t __kdebug_text (
    unsigned op,
    char const * text,
    unsigned len ) [inline]
  
```

Invoke a text output operation on the base debugger capability.

##### Parameters

<i>op</i>	Text output operation code from <a href="#">l4_kdebug_ops_t</a> or a value above 0x200 used by the kernel trace buffer implementation ( <a href="#">__ktrace-impl.h</a> ).
<i>text</i>	Output string.
<i>len</i>	Length of the output string. The maximum length is limited to <code>L4_UTCB_GENERIC_DATA_SIZE - 2</code> machine words. Output strings longer than this limit will be cropped.

## Return values

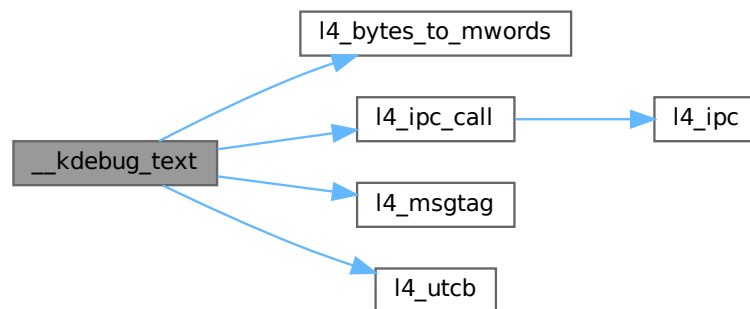
<i>Message</i>	tag returned from the IPC on the base debugger capability.
----------------	--

Definition at line 96 of file [kdebug.h](#).

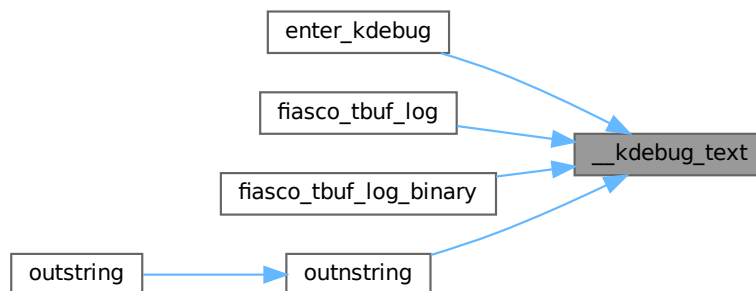
References [L4\\_BASE\\_DEBUGGER\\_CAP](#), [l4\\_bytes\\_to\\_mwords\(\)](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_DEBUGGER](#), [l4\\_utcb\(\)](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [enter\\_kdebug\(\)](#), [fiasco\\_tbuf\\_log\(\)](#), [fiasco\\_tbuf\\_log\\_binary\(\)](#), and [outnstring\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.515.3.5 enter\_kdebug()

```
void enter_kdebug (
    char const * text ) [inline]
```

Enter the kernel debugger.

## Parameters

<i>text</i>	Optional message displayed by the kernel debugger when entered.
-------------	---

Enter the kernel debugger, if configured. An optional message can be passed to the kernel debugger which is printed upon the entering of the debugger.

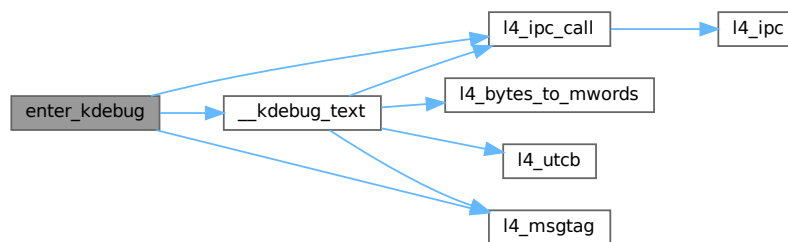
## Examples

[examples/sys/singlestep/main.c](#).

Definition at line 202 of file [kdebug.h](#).

References [\\_\\_kdebug\\_text\(\)](#), [L4\\_BASE\\_DEBUGGER\\_CAP](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), and [L4\\_PROTO\\_DEBUGGER](#).

Here is the call graph for this function:



### 16.515.3.6 outchar()

```
void outchar (  
    char c ) [inline]
```

Output a single character via the kernel debugger.

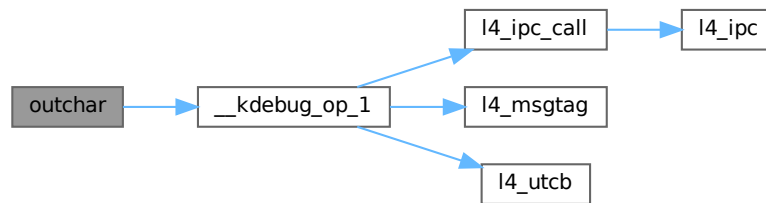
## Parameters

<i>c</i>	Output character.
----------	-------------------

Definition at line 243 of file [kdebug.h](#).

References [\\_\\_kdebug\\_op\\_1\(\)](#).

Here is the call graph for this function:



### 16.515.3.7 outdec()

```
void outdec (
    l4_mword_t number ) [inline]
```

Output a decimal unsigned machine word via the kernel debugger.

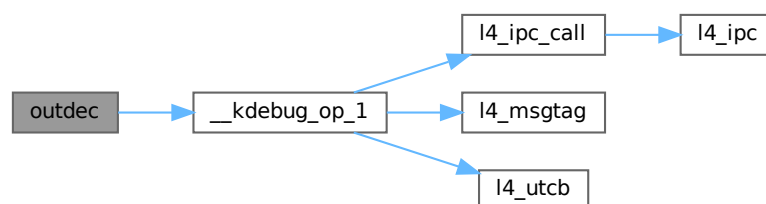
#### Parameters

<i>number</i>	Output machine word.
---------------	----------------------

Definition at line 332 of file `kdebug.h`.

References `__kdebug_op_1()`.

Here is the call graph for this function:



### 16.515.3.8 outhex12()

```
void outhex12 (
    l4_uint16_t number ) [inline]
```

Output a 12-bit unsigned hexadecimal number via the kernel debugger.

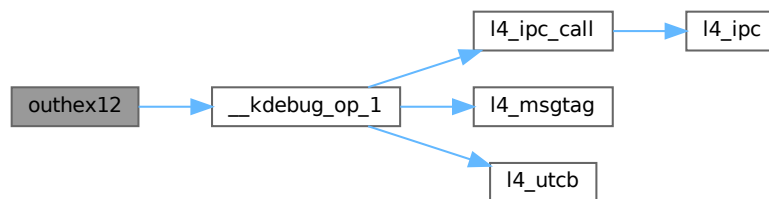
## Parameters

<i>number</i>	Output 12-bit number. Only the 12 LSB bits are used.
---------------	--

Definition at line 312 of file [kdebug.h](#).

References [\\_\\_kdebug\\_op\\_1\(\)](#).

Here is the call graph for this function:



### 16.515.3.9 outhex16()

```
void outhex16 (  
    l4_uint16_t number ) [inline]
```

Output a 16-bit unsigned hexadecimal number via the kernel debugger.

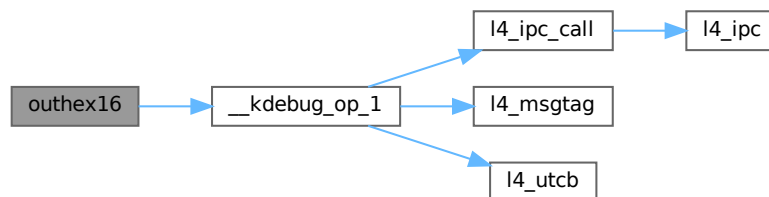
## Parameters

<i>number</i>	Output 16-bit number.
---------------	-----------------------

Definition at line 302 of file [kdebug.h](#).

References [\\_\\_kdebug\\_op\\_1\(\)](#).

Here is the call graph for this function:



### 16.515.3.10 outhex20()

```
void outhex20 (
    14_uint32_t number ) [inline]
```

Output a 20-bit unsigned hexadecimal number via the kernel debugger.

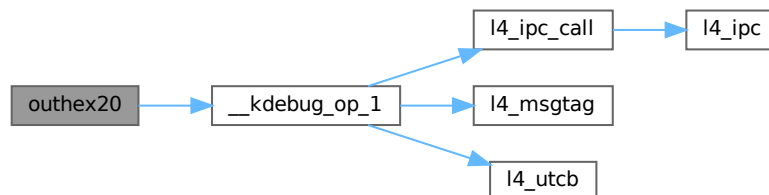
#### Parameters

<i>number</i>	Output 20-bit number. Only the 20 LSB bits are used.
---------------	--

Definition at line 292 of file [kdebug.h](#).

References [\\_\\_kdebug\\_op\\_1\(\)](#).

Here is the call graph for this function:



### 16.515.3.11 outhex32()

```
void outhex32 (
    14_uint32_t number ) [inline]
```

Output a 32-bit unsigned hexadecimal number via the kernel debugger.

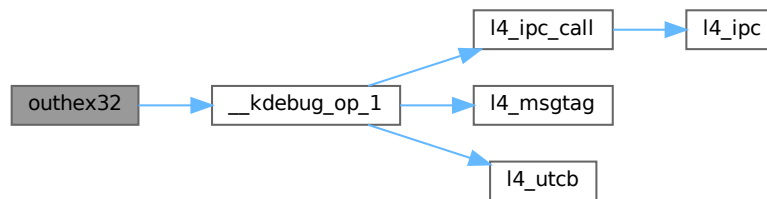
#### Parameters

<i>number</i>	Output 32-bit number.
---------------	-----------------------

Definition at line 282 of file [kdebug.h](#).

References [\\_\\_kdebug\\_op\\_1\(\)](#).

Here is the call graph for this function:



### 16.515.3.12 outhex64()

```
void outhex64 (
    l4_uint64_t number ) [inline]
```

Output a 64-bit unsigned hexadecimal number via the kernel debugger.

#### Parameters

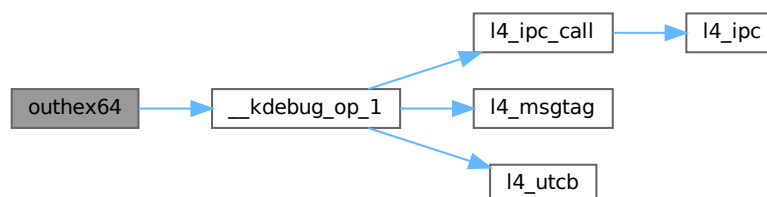
<i>number</i>	Output 64-bit number.
---------------	-----------------------

The two 32-bit halves are printed non-atomically.

Definition at line 271 of file [kdebug.h](#).

References [\\_\\_kdebug\\_op\\_1\(\)](#).

Here is the call graph for this function:



### 16.515.3.13 outhex8()

```
void outhex8 (
    l4_uint8_t number ) [inline]
```

Output an 8-bit unsigned hexadecimal number via the kernel debugger.

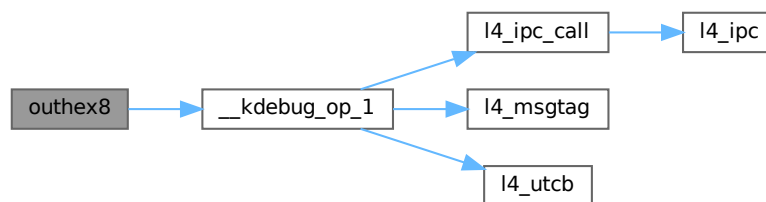
## Parameters

<i>number</i>	Output 8-bit number.
---------------	----------------------

Definition at line 322 of file [kdebug.h](#).

References [\\_\\_kdebug\\_op\\_1\(\)](#).

Here is the call graph for this function:



#### 16.515.3.14 outnstring()

```
void outnstring (
    char const * text,
    unsigned len ) [inline]
```

Output a fixed-length string via the kernel debugger.

## Parameters

<i>text</i>	Beginning of the output string.
<i>len</i>	Length of the output string. The maximum length is limited to <code>L4_UTCB_GENERIC_DATA_SIZE - 2</code> machine words. Output strings longer than this limit will be cropped.

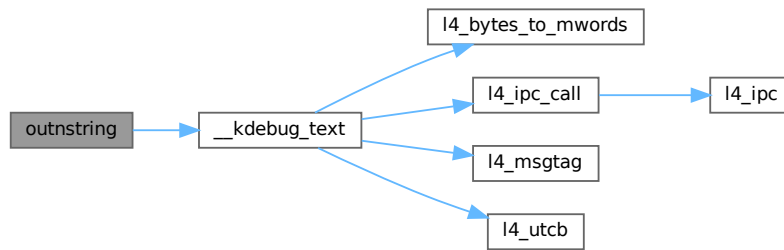
Definition at line 224 of file [kdebug.h](#).

References [\\_\\_kdebug\\_text\(\)](#).

Referenced by [outstring\(\)](#).



Here is the call graph for this function:



Here is the caller graph for this function:



### 16.515.3.15 outstring()

```
void outstring (
    char const * text ) [inline]
```

Output a string via the kernel debugger.

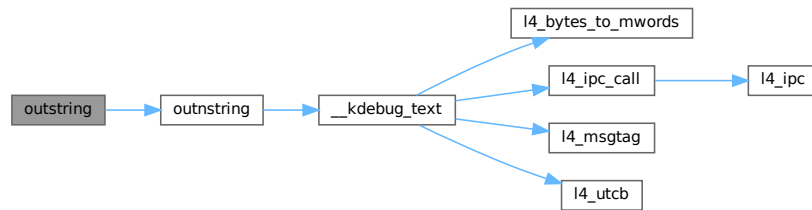
#### Parameters

<i>text</i>	Beginning of the output string. The maximum length of the output string is limited to <code>L4_UTCB_GENERIC_DATA_SIZE - 2</code> machine words. Output strings longer than this limit will be cropped.
-------------	--

Definition at line 235 of file [kdebug.h](#).

References [outnstring\(\)](#).

Here is the call graph for this function:



### 16.515.3.16 outumword()

```
void outumword (
    l4_umword_t number ) [inline]
```

Output a hexadecimal unsigned machine word via the kernel debugger.

#### Parameters

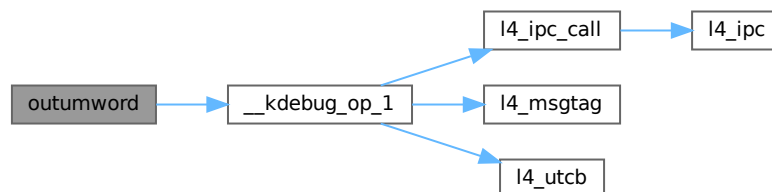
<i>number</i>	Output machine word.
---------------	----------------------

If the machine word is 64 bits long, it is printed non-atomically as two 32-bit numbers.

Definition at line 256 of file [kdebug.h](#).

References [\\_\\_kdebug\\_op\\_1\(\)](#).

Here is the call graph for this function:



## 16.516 kdebug.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
```

```

00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018
00019 #pragma once
00020
00027 #ifndef __KDEBUG_H__
00028 #define __KDEBUG_H__
00029
00030 #include <l4/sys/compiler.h>
00031 #include <l4/sys/consts.h>
00032 #include <l4/sys/ipc.h>
00033
00034
00035 L4_INLINE void
00036 enter_kdebug(char const *text) L4_NOTHROW;
00037
00042 enum l4_kdebug_ops_t
00043 {
00044     L4_KDEBUG_ENTER      = 0,
00045     L4_KDEBUG_OUTCHAR    = 1,
00046     L4_KDEBUG_OUTNSTRING = 2,
00047     L4_KDEBUG_OUTHEX32   = 3,
00048     L4_KDEBUG_OUTHEX20   = 4,
00049     L4_KDEBUG_OUTHEX16   = 5,
00050     L4_KDEBUG_OUTHEX12   = 6,
00051     L4_KDEBUG_OUTHEX8    = 7,
00052     L4_KDEBUG_OUTDEC     = 8,
00053 };
00054
00055
00065 L4_INLINE l4_msgtag_t
00066 __kdebug_op(unsigned op) L4_NOTHROW
00067 {
00068     l4_msgtag_t res;
00069     l4_uctb_t *u = l4_uctb();
00070     l4_msg_regs_t *mr = l4_uctb_mr_u(u);
00071     l4_umword_t mr0 = mr->mr[0];
00072
00073     mr->mr[0] = op;
00074     res = l4_ipc_call(L4_BASE_DEBUGGER_CAP, u,
00075                     l4_msgtag(L4_PROTO_DEBUGGER, 1, 0, 0),
00076                     L4_IPC_NEVER);
00077     mr->mr[0] = mr0;
00078     return res;
00079 }
00080
00095 L4_INLINE l4_msgtag_t
00096 __kdebug_text(unsigned op, char const *text, unsigned len) L4_NOTHROW
00097 {
00098     l4_msg_regs_t store;
00099     l4_msgtag_t res;
00100     l4_uctb_t *u = l4_uctb();
00101     l4_msg_regs_t *mr = l4_uctb_mr_u(u);
00102
00103     if (len > (sizeof(store) - (2 * sizeof(l4_umword_t))))
00104         len = sizeof(store) - (2 * sizeof(l4_umword_t));
00105
00106     __builtin_memcpy(&store, mr, sizeof(store));
00107     mr->mr[0] = op;
00108     mr->mr[1] = len;
00109     __builtin_memcpy(&mr->mr[2], text, len);
00110     res = l4_ipc_call(L4_BASE_DEBUGGER_CAP, u,
00111                     l4_msgtag(L4_PROTO_DEBUGGER,
00112                             l4_bytes_to_mwords(len) + 2, 0, 0),
00113                     L4_IPC_NEVER);
00114     __builtin_memcpy(mr, &store, sizeof(*mr));
00115     return res;
00116 }
00117
00136 L4_INLINE l4_msgtag_t
00137 __kdebug_3_text(unsigned op, char const *text, unsigned len,
00138                l4_umword_t v1, l4_umword_t v2, l4_umword_t v3) L4_NOTHROW
00139 {
00140     l4_msg_regs_t store;
00141     l4_msgtag_t res;

```

```

00142     l4_utcb_t *u = l4_utcb();
00143     l4_msg_regs_t *mr = l4_utcb_mr_u(u);
00144
00145     if (len > (sizeof(store) - (5 * sizeof(l4_umword_t))))
00146         len = sizeof(store) - (5 * sizeof(l4_umword_t));
00147
00148     __builtin_memcpy(&store, mr, sizeof(store));
00149     mr->mr[0] = op;
00150     mr->mr[1] = v1;
00151     mr->mr[2] = v2;
00152     mr->mr[3] = v3;
00153     mr->mr[4] = len;
00154     __builtin_memcpy(&mr->mr[5], text, len);
00155     res = l4_ipc_call(L4_BASE_DEBUGGER_CAP, u,
00156                     l4_msgtag(L4_PROTO_DEBUGGER,
00157                               l4_bytes_to_mwords(len) + 5, 0, 0),
00158                     L4_IPC_NEVER);
00159     __builtin_memcpy(mr, &store, sizeof(*mr));
00160     return res;
00161 }
00162
00173 L4_INLINE l4_msgtag_t
00174 __kdebug_op_1(unsigned op, l4_mword_t val) L4_NOTHROW
00175 {
00176     l4_umword_t m[2];
00177     l4_msgtag_t res;
00178     l4_utcb_t *u = l4_utcb();
00179     l4_msg_regs_t *mr = l4_utcb_mr_u(u);
00180
00181     m[0] = mr->mr[0];
00182     m[1] = mr->mr[1];
00183     mr->mr[0] = op;
00184     mr->mr[1] = val;
00185     res = l4_ipc_call(L4_BASE_DEBUGGER_CAP, u,
00186                     l4_msgtag(L4_PROTO_DEBUGGER, 2, 0, 0),
00187                     L4_IPC_NEVER);
00188     mr->mr[0] = m[0];
00189     mr->mr[1] = m[1];
00190     return res;
00191 }
00192
00202 L4_INLINE void enter_kdebug(char const *text) L4_NOTHROW
00203 {
00204     /* special case, enter without any text and use of the UTCB */
00205     if (!text)
00206     {
00207         l4_ipc_call(L4_BASE_DEBUGGER_CAP, 0,
00208                     l4_msgtag(L4_PROTO_DEBUGGER, 0, 0, 0),
00209                     L4_IPC_NEVER);
00210         return;
00211     }
00212
00213     __kdebug_text(L4_KDEBUG_ENTER, text, __builtin_strlen(text));
00214 }
00215
00224 L4_INLINE void outnstring(char const *text, unsigned len)
00225 { __kdebug_text(L4_KDEBUG_OUTNSTRING, text, len); }
00226
00235 L4_INLINE void outstring(char const *text)
00236 { outnstring(text, __builtin_strlen(text)); }
00237
00243 L4_INLINE void outchar(char c)
00244 {
00245     __kdebug_op_1(L4_KDEBUG_OUTCHAR, c);
00246 }
00247
00256 L4_INLINE void outumword(l4_umword_t number)
00257 {
00258     if (sizeof(l4_umword_t) == sizeof(l4_uint64_t))
00259         __kdebug_op_1(L4_KDEBUG_OUTHEX32, (l4_uint64_t)number >> 32);
00260
00261     __kdebug_op_1(L4_KDEBUG_OUTHEX32, number);
00262 }
00263
00271 L4_INLINE void outhex64(l4_uint64_t number)
00272 {
00273     __kdebug_op_1(L4_KDEBUG_OUTHEX32, number >> 32);
00274     __kdebug_op_1(L4_KDEBUG_OUTHEX32, number);
00275 }
00276
00282 L4_INLINE void outhex32(l4_uint32_t number)
00283 {
00284     __kdebug_op_1(L4_KDEBUG_OUTHEX32, number);
00285 }
00286
00292 L4_INLINE void outhex20(l4_uint32_t number)
00293 {

```

```

00294  __kdebug_op_1(L4_KDEBUG_OUTHEX20, number);
00295 }
00296
00302 L4_INLINE void outhex16(l4_uint16_t number)
00303 {
00304  __kdebug_op_1(L4_KDEBUG_OUTHEX16, number);
00305 }
00306
00312 L4_INLINE void outhex12(l4_uint16_t number)
00313 {
00314  __kdebug_op_1(L4_KDEBUG_OUTHEX12, number);
00315 }
00316
00322 L4_INLINE void outhex8(l4_uint8_t number)
00323 {
00324  __kdebug_op_1(L4_KDEBUG_OUTHEX8, number);
00325 }
00326
00332 L4_INLINE void outdec(l4_mword_t number)
00333 {
00334  __kdebug_op_1(L4_KDEBUG_OUTDEC, number);
00335 }
00336
00337 #endif // __KDEBUG_H__

```

## 16.517 l4/sys/kernel\_object.h File Reference

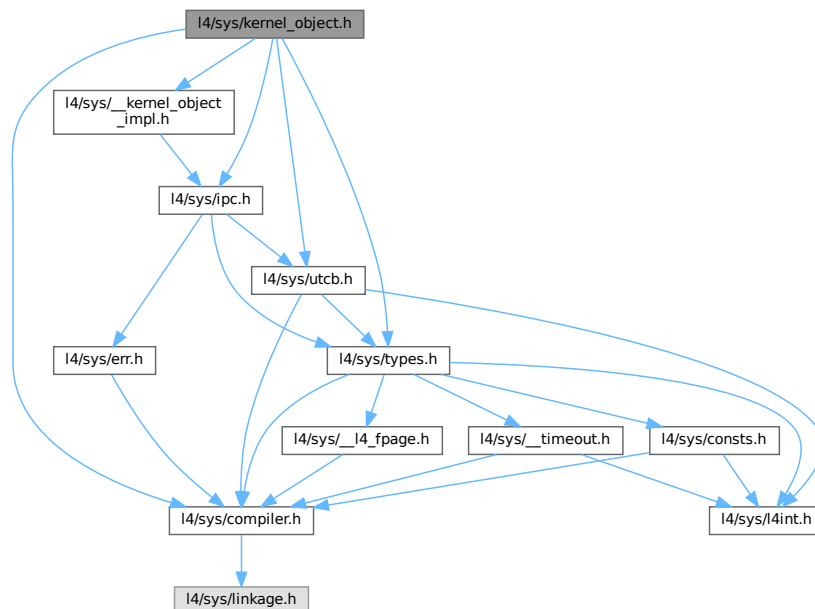
Kernel object system calls.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/sys/utcb.h>
#include <l4/sys/__kernel_object_impl.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for kernel\_object.h:



This graph shows which files directly or indirectly include this file:



### 16.517.1 Detailed Description

Kernel object system calls.

Definition in file [kernel\\_object.h](#).

## 16.518 kernel\_object.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #ifndef __L4SYS__KERNEL_OBJECT_H__
00025 #define __L4SYS__KERNEL_OBJECT_H__
00026
00027 #include <l4/sys/types.h>
00028 #include <l4/sys/compiler.h>
00029 #include <l4/sys/utcb.h>
00030
00049 L4_INLINE l4_msgtag_t
00050 l4_invoke_debugger(l4_cap_idx_t obj, l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;
00051
00052
00053 /*****
00054  * Implementation
00055  *****/
00056
00057 #include <l4/sys/__kernel_object_impl.h>
00058 #include <l4/sys/ipc.h>
00059
00060 enum L4_kobject_op {
00061     L4_KOBJECT_OP_DEC_REFCNT = 0,
00062     L4_KOBJECT_OP_REGISTER_IRQ,
00063 };
00064
00065 L4_INLINE l4_msgtag_t
00066 l4_kobject_dec_refcnt_u(l4_cap_idx_t obj, l4_mword_t diff, l4_utcb_t *u) L4_NOTHROW;
00067
00068 L4_INLINE l4_msgtag_t
00069 l4_kobject_dec_refcnt(l4_cap_idx_t obj, l4_mword_t diff) L4_NOTHROW;
00070
00071 L4_INLINE l4_msgtag_t
00072 l4_kobject_dec_refcnt_u(l4_cap_idx_t obj, l4_mword_t diff, l4_utcb_t *u) L4_NOTHROW
00073 {
00074     l4_msg_regs_t *m = l4_utcb_mr_u(u);
00075     m->mr[0] = L4_KOBJECT_OP_DEC_REFCNT;
00076     m->mr[1] = diff;

```

```

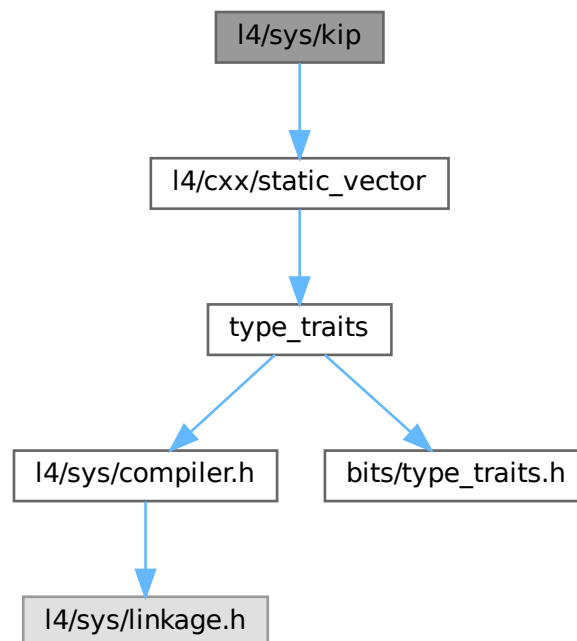
00077     return l4_ipc_call(obj, u, l4_msgtag(L4_PROTO_KOBJECT, 2, 0, 0), L4_IPC_NEVER);
00078 }
00079
00080 L4_INLINE l4_msgtag_t
00081 l4_kobject_dec_refcnt(l4_cap_idx_t obj, l4_mword_t diff) L4_NOTHROW
00082 {
00083     return l4_kobject_dec_refcnt_u(obj, diff, l4_utcb());
00084 }
00085
00086 #endif /* ! __L4SYS__KERNEL_OBJECT_H__ */

```

## 16.519 l4/sys/kip File Reference

```
#include <l4/cxx/static_vector>
```

Include dependency graph for kip:



### Data Structures

- class [L4::Kip::Mem\\_desc](#)  
*Memory descriptors stored in the kernel interface page.*

### Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

## 16.519.1 Detailed Description

L4::Kip class, memory descriptors.

### Author

Alexander Warg [alexander.warg@os.inf.tu-dresden.de](mailto:alexander.warg@os.inf.tu-dresden.de)

Definition in file [kip](#).

## 16.520 kip

[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00010 /*
00011  * (c) 2008-2009 Author(s)
00012  *      economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * This file is part of TUD:OS and distributed under the terms of the
00015  * GNU General Public License 2.
00016  * Please see the COPYING-GPL-2 file for details.
00017  *
00018  * As a special exception, you may use this file as part of a free software
00019  * library without restriction. Specifically, if other files instantiate
00020  * templates or use macros or inline functions from this file, or you compile
00021  * this file and link it with other files to produce an executable, this
00022  * file does not by itself cause the resulting executable to be covered by
00023  * the GNU General Public License. This exception does not however
00024  * invalidate any other reasons why the executable file might be covered by
00025  * the GNU General Public License.
00026  */
00027 #ifndef L4_SYS_KIP_H__
00028 #define L4_SYS_KIP_H__
00029
00030 #include <l4/cxx/static_vector>
00031
00032 /* C++ version of memory descriptors */
00033
00043 namespace L4
00044 {
00045     namespace Kip
00046     {
00053         class Mem_desc
00054         {
00055         public:
00059             enum Mem_type
00060             {
00061                 Undefined      = 0x0,
00062                 Conventional   = 0x1,
00063                 Reserved       = 0x2,
00064                 Dedicated      = 0x3,
00065                 Shared         = 0x4,
00066
00067                 Info           = 0xd,
00068                 Bootloader     = 0xe,
00069                 Arch           = 0xf
00070             };
00071
00075             enum Info_sub_type
00076             {
00077                 Info_acpi_rsdp = 0
00078             };
00079
00083             enum Arch_sub_type_common
00084             {
00085                 Arch_acpi_tables = 3,
00086                 Arch_acpi_nvs    = 4,
00087             };
00088
00089         private:
00090             unsigned long _l, _h;
00091
00092             static unsigned long &memory_info(void *kip) noexcept
00093             { return *(reinterpret_cast<unsigned long *>(kip) + 21); }
00094         }
```



```

00095     static unsigned long memory_info(void const *kip) noexcept
00096     { return *(reinterpret_cast<unsigned long const *>(kip) + 21); }
00097
00098 public:
00106     static Mem_desc *first(void *kip) noexcept
00107     {
00108         return reinterpret_cast<Mem_desc *>(reinterpret_cast<char *>(kip)
00109             + (memory_info(kip) » ((sizeof(unsigned long) / 2) * 8)));
00110     }
00111
00112     static Mem_desc const *first(void const *kip) noexcept
00113     {
00114         char const *addr = reinterpret_cast<char const *>(kip)
00115             + (memory_info(kip) » ((sizeof(unsigned long) / 2) * 8));
00116         return reinterpret_cast<Mem_desc const *>(addr);
00117     }
00118
00126     static unsigned long count(void const *kip) noexcept
00127     {
00128         return memory_info(kip)
00129             & ((1UL « ((sizeof(unsigned long) / 2) * 8)) - 1);
00130     }
00131
00138     static void count(void *kip, unsigned count) noexcept
00139     {
00140         unsigned long &mi = memory_info(kip);
00141         mi = (mi & ~(1UL « ((sizeof(unsigned long) / 2) * 8)) - 1) | count;
00142     }
00143
00149     static inline cxx::static_vector<Mem_desc const> all(void const *kip)
00150     {
00151         return cxx::static_vector<Mem_desc const>(Mem_desc::first(kip),
00152             Mem_desc::count(kip));
00153     }
00154
00160     static inline cxx::static_vector<Mem_desc> all(void *kip)
00161     {
00162         return cxx::static_vector<Mem_desc>(Mem_desc::first(kip),
00163             Mem_desc::count(kip));
00164     }
00165
00179     Mem_desc(unsigned long start, unsigned long end,
00180         Mem_type t, unsigned char st = 0, bool virt = false,
00181         bool eager = false) noexcept
00182     : _l((start & ~0x3ffUL) | (t & 0x0f) | ((st < 4) & 0x0f0)
00183         | (virt ? 0x0200 : 0x0) | (eager ? 0x100 : 0x0)), _h(end | 0x3ffUL)
00184     {}
00185
00191     unsigned long start() const noexcept { return _l & ~0x3ffUL; }
00192
00198     unsigned long end() const noexcept { return _h | 0x3ffUL; }
00199
00205     unsigned long size() const noexcept { return end() + 1 - start(); }
00206
00212     Mem_type type() const noexcept
00213     {
00214         return static_cast<Mem_type>(_l & 0x0f);
00215     }
00216
00222     unsigned char sub_type() const noexcept { return (_l » 4) & 0x0f; }
00223
00230     unsigned is_virtual() const noexcept { return _l & 0x200; }
00231
00236     unsigned eager_map() const noexcept { return _l & 0x100; }
00237
00250     void set(unsigned long start, unsigned long end,
00251         Mem_type t, unsigned char st = 0, bool virt = false,
00252         bool eager = false) noexcept
00253     {
00254         _l = (start & ~0x3ffUL) | (t & 0x0f) | ((st < 4) & 0x0f0)
00255             | (virt?0x0200:0x0) | (eager ? 0x0100 : 0x0);
00256
00257         _h = end | 0x3ffUL;
00258     }
00259
00260 };
00261 };
00262 };
00263
00264 #endif

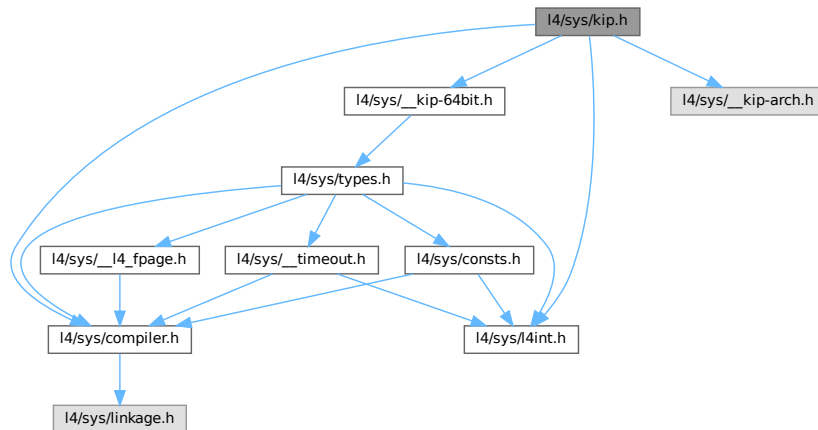
```

## 16.521 l4/sys/kip.h File Reference

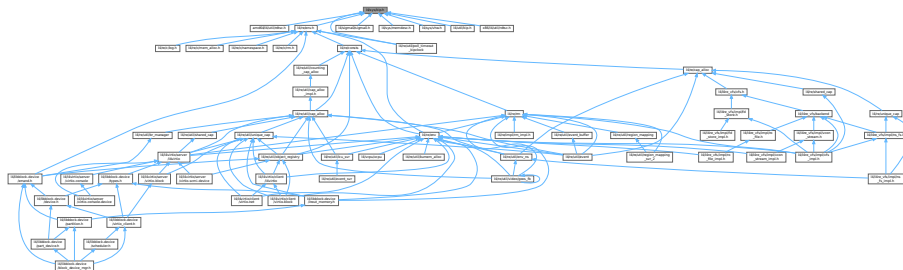
Kernel Info Page access functions.

```
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/__kip-arch.h>
#include <l4/sys/__kip-64bit.h>
```

Include dependency graph for kip.h:



This graph shows which files directly or indirectly include this file:



### Macros

- `#define L4_KERNEL_INFO_MAGIC (0x4BE6344CL) /* "L4μK" */`  
Kernel Info Page identifier ("L4μK").
- `#define l4_kip_for_each_feature(s) for (s += __builtin_strlen(s) + 1; *s; s += __builtin_strlen(s) + 1)`  
Cycle through kernel features given in the KIP.

### Enumerations

- `enum { L4_KIP_OFFS_READ_US = 0x900 , L4_KIP_OFFS_READ_NS = 0x980 }`

## Functions

- [l4\\_kernel\\_info\\_t](#) const \* [l4\\_kip](#) (void) [L4\\_NOTHROW](#)  
*Get Kernel Info Page.*
- [l4\\_umword\\_t](#) [l4\\_kip\\_version](#) ([l4\\_kernel\\_info\\_t](#) const \*kip) [L4\\_NOTHROW](#)  
*Get the kernel version.*
- const char \* [l4\\_kip\\_version\\_string](#) ([l4\\_kernel\\_info\\_t](#) const \*kip) [L4\\_NOTHROW](#)  
*Get the kernel version string.*
- int [l4\\_kernel\\_info\\_version\\_offset](#) ([l4\\_kernel\\_info\\_t](#) const \*kip) [L4\\_NOTHROW](#)  
*Return offset in bytes of version\_strings relative to the KIP base.*
- [l4\\_cpu\\_time\\_t](#) [l4\\_kip\\_clock](#) ([l4\\_kernel\\_info\\_t](#) const \*kip) [L4\\_NOTHROW](#)  
*Return clock value from the KIP.*
- [l4\\_umword\\_t](#) [l4\\_kip\\_clock\\_lw](#) ([l4\\_kernel\\_info\\_t](#) const \*kip) [L4\\_NOTHROW](#)  
*Return least significant machine word of clock value from the KIP.*
- [l4\\_uint64\\_t](#) [l4\\_kip\\_clock\\_ns](#) ([l4\\_kernel\\_info\\_t](#) const \*kip) [L4\\_NOTHROW](#)  
*Return current clock using the KIP in nanoseconds.*
- int [l4\\_kip\\_kernel\\_has\\_feature](#) ([l4\\_kernel\\_info\\_t](#) const \*kip, char const \*str)  
*Check if kernel supports a feature.*

### 16.521.1 Detailed Description

Kernel Info Page access functions.

Definition in file [kip.h](#).

### 16.521.2 Macro Definition Documentation

#### 16.521.2.1 l4\_kip\_for\_each\_feature

```
#define l4_kip_for_each_feature(  
    s )    for (s += __builtin_strlen(s) + 1; *s; s += __builtin_strlen(s) + 1)
```

Cycle through kernel features given in the KIP.

Cycles through all KIP kernel feature strings. *s* must be a character pointer (`char const *`) initialized with [l4\\_kip\\_version\\_string\(\)](#).

Definition at line 241 of file [kip.h](#).

### 16.521.3 Function Documentation

#### 16.521.3.1 l4\_kip\_kernel\_has\_feature()

```
int l4_kip_kernel_has_feature (  
    l4\_kernel\_info\_t const * kip,  
    char const * str ) [inline]
```

Check if kernel supports a feature.

## Parameters

<i>kip</i>	Pointer to the kernel info page (KIP).
<i>str</i>	Feature name to check.

## Returns

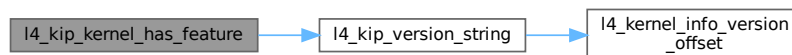
1 if the kernel supports the feature, 0 if not.

Checks the feature field in the KIP for the given string.

Definition at line 255 of file [kip.h](#).

References [l4\\_kip\\_for\\_each\\_feature](#), and [l4\\_kip\\_version\\_string\(\)](#).

Here is the call graph for this function:



## 16.522 kip.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/sys/compiler.h>
00027 #include <l4/sys/l4int.h>
00028
00029 #include <l4/sys/__kip-arch.h>
00030
00031 struct l4_kip_platform_info
00032 {
00033     char                name[16];
00034     l4_uint32_t         is_mp;
00035     struct l4_kip_platform_info_arch arch;
00036 };
00037
00038 #if L4_MWORD_BITS == 32
00039 #include <l4/sys/__kip-32bit.h>
00040 #else
00041 #include <l4/sys/__kip-64bit.h>
00042 #endif
00043
00044

```

```

00058 enum l4_kernel_info_consts_t
00059 {
00060     L4_KIP_VERSION_FIASCO      = 0x87004444,
00061     L4_KIP_VERSION_FIASCO_MASK = 0xff00ffff,
00062 };
00063
00064 enum
00065 {
00074     L4_KIP_OFFSETS_READ_US      = 0x900,
00075
00085     L4_KIP_OFFSETS_READ_NS      = 0x980,
00086 };
00087
00091 extern l4_kernel_info_t const *l4_global_kip;
00092
00096 #define L4_KERNEL_INFO_MAGIC (0x4BE6344CL) /* "L4µK" */
00097
00098
00104 L4_INLINE l4_kernel_info_t const *l4_kip(void) L4_NOTHROW;
00105
00106
00114 L4_INLINE l4_umword_t l4_kip_version(l4_kernel_info_t const *kip) L4_NOTHROW;
00115
00123 L4_INLINE const char *l4_kip_version_string(l4_kernel_info_t const *kip) L4_NOTHROW;
00124
00133 L4_INLINE int
00134 l4_kernel_info_version_offset(l4_kernel_info_t const *kip) L4_NOTHROW;
00135
00153 L4_INLINE l4_cpu_time_t
00154 l4_kip_clock(l4_kernel_info_t const *kip) L4_NOTHROW;
00155
00168 L4_INLINE l4_umword_t
00169 l4_kip_clock_lw(l4_kernel_info_t const *kip) L4_NOTHROW
00170     L4_DEPRECATED("Use l4_kip_clock() instead");
00171
00185 L4_INLINE l4_uint64_t
00186 l4_kip_clock_ns(l4_kernel_info_t const *kip) L4_NOTHROW;
00187
00190 /*****
00191  * Implementations
00192  *****/
00193
00194 L4_INLINE l4_kernel_info_t const*
00195 l4_kip(void) L4_NOTHROW
00196 { return l4_global_kip; }
00197
00198 L4_INLINE l4_umword_t
00199 l4_kip_version(l4_kernel_info_t const *kip) L4_NOTHROW
00200 { return kip->version & L4_KIP_VERSION_FIASCO_MASK; }
00201
00202 L4_INLINE const char*
00203 l4_kip_version_string(l4_kernel_info_t const *k) L4_NOTHROW
00204 { return (const char *)k + l4_kernel_info_version_offset(k); }
00205
00206 L4_INLINE int
00207 l4_kernel_info_version_offset(l4_kernel_info_t const *kip) L4_NOTHROW
00208 { return kip->offset_version_strings << 4; }
00209
00210 L4_INLINE l4_cpu_time_t
00211 l4_kip_clock(l4_kernel_info_t const *kip) L4_NOTHROW
00212 {
00213     // Use kernel-provided code to determine the current clock.
00214     typedef l4_uint64_t (*kip_time_fn_read_us)(void);
00215     kip_time_fn_read_us read_us =
00216         (kip_time_fn_read_us)((l4_uint8_t const*)kip + L4_KIP_OFFSETS_READ_US);
00217     return read_us();
00218 }
00219
00220 L4_INLINE l4_cpu_time_t
00221 l4_kip_clock_ns(l4_kernel_info_t const *kip) L4_NOTHROW
00222 {
00223     typedef l4_uint64_t (*kip_time_fn_read_ns)(void);
00224     kip_time_fn_read_ns read_ns =
00225         (kip_time_fn_read_ns)((l4_uint8_t const*)kip + L4_KIP_OFFSETS_READ_NS);
00226     return read_ns();
00227 }
00228
00229 L4_INLINE l4_umword_t
00230 l4_kip_clock_lw(l4_kernel_info_t const *kip) L4_NOTHROW
00231 {
00232     return l4_kip_clock(kip);
00233 }
00234
00241 #define l4_kip_for_each_feature(s) \
00242     for (s += __builtin_strlen(s) + 1; *s; s += __builtin_strlen(s) + 1)
00243
00254 L4_INLINE int

```

```

00255 l4_kip_kernel_has_feature(l4_kernel_info_t const *kip, char const *str)
00256 {
00257     const char *s = l4_kip_version_string(kip);
00258     if (!s)
00259         return 0;
00260
00261     l4_kip_for_each_feature(s)
00262     {
00263         if (__builtin_strcmp(s, str) == 0)
00264             return 1;
00265     }
00266
00267     return 0;
00268 }

```

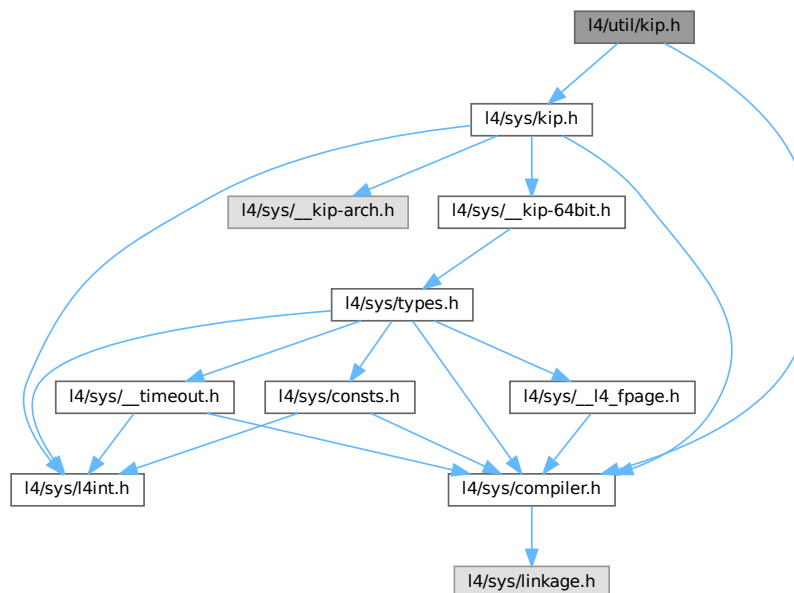
## 16.523 l4/util/kip.h File Reference

```

#include <l4/sys/kip.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for kip.h:



### Macros

- `#define l4util_kip_for_each_feature(s) l4_kip_for_each_feature(s)`  
Cycle through kernel features given in the KIP.

### Functions

- `int l4util_kip_kernel_is_ux (l4_kernel_info_t const *k)`  
Return whether the kernel is running natively or under UX.
- `int l4util_kip_kernel_has_feature (l4_kernel_info_t const *k, char const *str)`  
Check if kernel supports a feature.
- `unsigned long l4util_kip_kernel_abi_version (l4_kernel_info_t const *k)`  
Return kernel ABI version.

## 16.524 kip.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU Lesser General Public License 2.1.
00011  * Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013
00014 #pragma once
00015
00016 #include <l4/sys/kip.h>
00017 #include <l4/sys/compiler.h>
00018
00026 EXTERN_C_BEGIN
00027
00034 L4_CV int l4util_kip_kernel_is_ux(l4_kernel_info_t const *k);
00035
00048 L4_CV int l4util_kip_kernel_has_feature(l4_kernel_info_t const *k, char const *str);
00049
00056 L4_CV unsigned long l4util_kip_kernel_abi_version(l4_kernel_info_t const *k);
00057
00058 EXTERN_C_END
00059
00068 #define l4util_kip_for_each_feature(s) l4_kip_for_each_feature(s)
00069

```

## 16.525 kobject

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /* \file
00003  * Kobject C++ interface.
00004  */
00005 /*
00006  * Copyright (C) 2015-2017, 2019, 2021 Kernkonzept GmbH.
00007  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is distributed under the terms of the GNU General Public
00010  * License, version 2. Please see the COPYING-GPL-2 file for details.
00011  *
00012  * As a special exception, you may use this file as part of a free software
00013  * library without restriction. Specifically, if other files instantiate
00014  * templates or use macros or inline functions from this file, or you compile
00015  * this file and link it with other files to produce an executable, this
00016  * file does not by itself cause the resulting executable to be covered by
00017  * the GNU General Public License. This exception does not however
00018  * invalidate any other reasons why the executable file might be covered by
00019  * the GNU General Public License.
00020  */
00021 #pragma once
00022
00023 #include "kernel_object.h"
00024 #include "types.h"
00025 #include "__typeinfo.h"
00026
00027 namespace L4 {
00028
00046 class L4_EXPORT Kobject
00047 {
00048 private:
00049     Kobject();
00050     Kobject(Kobject const &);
00051     Kobject &operator = (Kobject const &);
00052
00053     template<typename T> friend struct Kobject_typeid;
00054
00055 protected:
00056     typedef Typeid::Iface<L4_PROTO_META, Kobject> __Iface;
00057     typedef Typeid::Iface_list<__Iface> __Iface_list;
00058
00065     struct __Kobject_typeid
00066     {
00067         typedef Type_info::Demand_t<> Demand;
00068         static Type_info const _m;
00069     };
00070
00079     l4_cap_idx_t cap() const noexcept { return _c(); }

```

```

00080
00081 private:
00082
00087     l4_cap_idx_t _c() const noexcept
00088     { return reinterpret_cast<l4_cap_idx_t>(this) & L4_CAP_MASK; }
00089
00090 public:
00110     l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
00111     { return l4_kobject_dec_refcnt_u(cap(), diff, utcb); }
00112 };
00113
00114 template<typename Derived, long PROTO = L4::PROTO_ANY,
00115         typename S_DEMAND = Type_info::Demand_t<> >
00116 struct Kobject_0t : Kobject_t<Derived, L4::Kobject, PROTO, S_DEMAND> {};
00117
00118 }
00119

```

## 16.526 l4/sys/ktrace.h File Reference

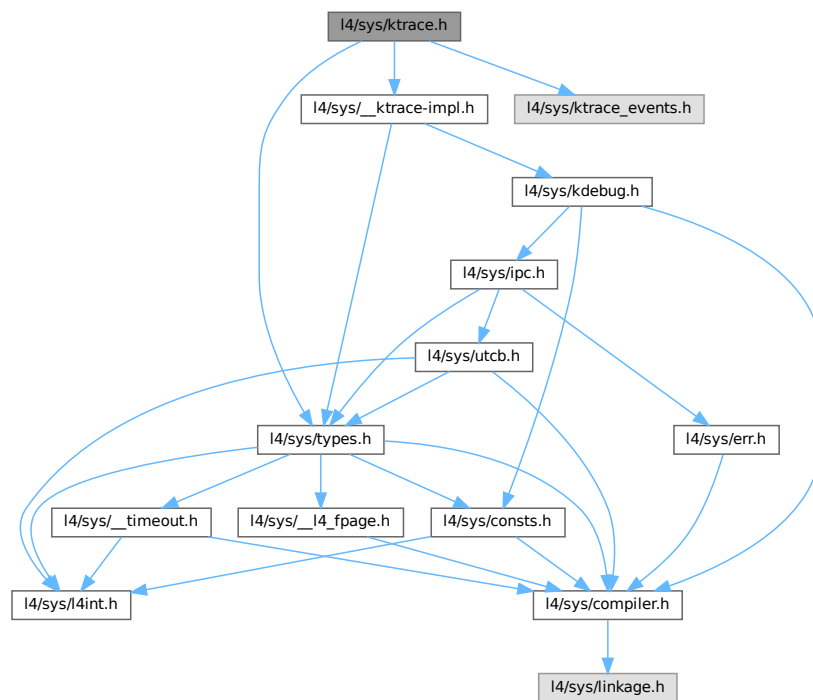
[L4](#) kernel event tracing.

```

#include <l4/sys/types.h>
#include <l4/sys/ktrace_events.h>
#include <l4/sys/__ktrace-impl.h>

```

Include dependency graph for ktrace.h:



### Functions

- [l4\\_umword\\_t fiasco\\_tbuf\\_log](#) (const char \*text)  
Create new trace-buffer entry with describing <text>.
- [l4\\_umword\\_t fiasco\\_tbuf\\_log\\_3val](#) (const char \*text, [l4\\_umword\\_t](#) v1, [l4\\_umword\\_t](#) v2, [l4\\_umword\\_t](#) v3)



Create new trace-buffer entry with describing <text> and three additional values.

- [l4\\_umword\\_t fiasco\\_tbuf\\_log\\_binary](#) (const unsigned char \*data)

Create new trace-buffer entry with binary data.

- void [fiasco\\_tbuf\\_clear](#) (void)

Clear trace-buffer.

- void [fiasco\\_tbuf\\_dump](#) (void)

Dump trace-buffer to kernel console.

## 16.526.1 Detailed Description

[L4](#) kernel event tracing.

Definition in file [ktrace.h](#).

## 16.527 ktrace.h

[Go to the documentation of this file.](#)

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *          Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009  *          Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *          2015 Adam Lackorzynski <adam@l4re.org>
00011  *          economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction. Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License. This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 /*****
00027 #ifndef __L4_KTRACE_H__
00028 #define __L4_KTRACE_H__
00029
00030 #include <l4/sys/types.h>
00031 #include <l4/sys/ktrace_events.h>
00032
00052 L4_INLINE l4_umword_t
00053 fiasco_tbuf_log(const char *text);
00054
00066 L4_INLINE l4_umword_t
00067 fiasco_tbuf_log_3val(const char *text, l4_umword_t v1, l4_umword_t v2, l4_umword_t v3);
00068
00076 L4_INLINE l4_umword_t
00077 fiasco_tbuf_log_binary(const unsigned char *data);
00078
00083 L4_INLINE void
00084 fiasco_tbuf_clear(void);
00085
00090 L4_INLINE void
00091 fiasco_tbuf_dump(void);
00092
00093 #include <l4/sys/__ktrace-impl.h>
00094
00095 #endif
```

## 16.528 amd64/l4/sys/l4int.h File Reference

Fixed sized integer types, amd64 version.

## Macros

- `#define L4_MWORD_BITS 64`  
*Size of machine words in bits.*

## Typedefs

- `typedef unsigned long l4_size_t`  
*Unsigned size type.*
- `typedef signed long l4_ssize_t`  
*Signed size type.*

## 16.528.1 Detailed Description

Fixed sized integer types, amd64 version.

Definition in file [l4int.h](#).

## 16.529 l4int.h

[Go to the documentation of this file.](#)

```
00001 /*****
00002  */
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  *
00011  * As a special exception, you may use this file as part of a free software
00012  * library without restriction. Specifically, if other files instantiate
00013  * templates or use macros or inline functions from this file, or you compile
00014  * this file and link it with other files to produce an executable, this
00015  * file does not by itself cause the resulting executable to be covered by
00016  * the GNU General Public License. This exception does not however
00017  * invalidate any other reasons why the executable file might be covered by
00018  * the GNU General Public License.
00019  */
00020 #pragma once
00021 #include_next <l4/sys/l4int.h>
00022
00023 #define L4_MWORD_BITS 64
00024 typedef unsigned long l4_size_t;
00025 typedef signed long l4_ssize_t;
```

## 16.530 arm/l4/sys/l4int.h File Reference

Fixed sized integer types, arm version.

## Macros

- `#define L4_MWORD_BITS 32`  
*Size of machine words in bits.*

## Typedefs

- typedef unsigned int **l4\_size\_t**  
*Unsigned size type.*
- typedef signed int **l4\_ssize\_t**  
*Signed size type.*

### 16.530.1 Detailed Description

Fixed sized integer types, arm version.

Definition in file [l4int.h](#).

## 16.531 l4int.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *           Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *           economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include_next <l4/sys/l4int.h>
00027
00033 #define L4_MWORD_BITS      32
00035 typedef unsigned int      l4_size_t;
00036 typedef signed int        l4_ssize_t;

```

### 16.532 l4/sys/l4int.h File Reference

Fixed sized integer types, generic version.

This graph shows which files directly or indirectly include this file:



## Typedefs

- typedef signed char **I4\_int8\_t**  
*Signed 8bit value.*
- typedef unsigned char **I4\_uint8\_t**  
*Unsigned 8bit value.*
- typedef signed short int **I4\_int16\_t**  
*Signed 16bit value.*
- typedef unsigned short int **I4\_uint16\_t**  
*Unsigned 16bit value.*
- typedef signed int **I4\_int32\_t**  
*Signed 32bit value.*
- typedef unsigned int **I4\_uint32\_t**  
*Unsigned 32bit value.*
- typedef signed long long **I4\_int64\_t**  
*Signed 64bit value.*
- typedef unsigned long long **I4\_uint64\_t**  
*Unsigned 64bit value.*
- typedef unsigned long **I4\_addr\_t**  
*Address type.*
- typedef signed long **I4\_mword\_t**  
*Signed machine word.*
- typedef unsigned long **I4\_umword\_t**  
*Unsigned machine word.*
- typedef [I4\\_uint64\\_t](#) **I4\_cpu\_time\_t**  
*CPU clock type.*
- typedef [I4\\_uint64\\_t](#) **I4\_kernel\_clock\_t**  
*Kernel clock type.*

## 16.532.1 Detailed Description

Fixed sized integer types, generic version.

Definition in file [I4int.h](#).

## 16.533 I4int.h

[Go to the documentation of this file.](#)

```

00001
00013 /*
00014  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00015  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00016  *      economic rights: Technische Universität Dresden (Germany)
00017  *
00018  * This file is part of TUD:OS and distributed under the terms of the
00019  * GNU General Public License 2.
00020  * Please see the COPYING-GPL-2 file for details.
00021  *
00022  * As a special exception, you may use this file as part of a free software
00023  * library without restriction. Specifically, if other files instantiate
00024  * templates or use macros or inline functions from this file, or you compile
00025  * this file and link it with other files to produce an executable, this
00026  * file does not by itself cause the resulting executable to be covered by
00027  * the GNU General Public License. This exception does not however
00028  * invalidate any other reasons why the executable file might be covered by
00029  * the GNU General Public License.

```

```

00030  */
00031  #ifndef __L4_SYS_L4INT_H__
00032  #define __L4_SYS_L4INT_H__
00033
00034  /* fixed sized data types */
00035  typedef signed char      l4_int8_t;
00036  typedef unsigned char    l4_uint8_t;
00037  typedef signed short int l4_int16_t;
00038  typedef unsigned short int l4_uint16_t;
00039  typedef signed int       l4_int32_t;
00040  typedef unsigned int     l4_uint32_t;
00041  typedef signed long long l4_int64_t;
00042  typedef unsigned long long l4_uint64_t;
00044  /* some common data types */
00045  typedef unsigned long     l4_addr_t;
00048  typedef signed long       l4_mword_t;
00051  typedef unsigned long     l4_umword_t;
00058  typedef l4_uint64_t l4_cpu_time_t;
00059
00064  typedef l4_uint64_t l4_kernel_clock_t;
00065
00066  #endif /* !__L4_SYS_L4INT_H__ */

```

## 16.534 x86/l4/sys/l4int.h File Reference

Fixed sized integer types, x86 version.

### Macros

- `#define L4_MWORD_BITS 32`  
*Size of machine words in bits.*

### Typedefs

- `typedef unsigned int l4_size_t`  
*Unsigned size type.*
- `typedef signed int l4_ssize_t`  
*Signed size type.*

### 16.534.1 Detailed Description

Fixed sized integer types, x86 version.

Definition in file [l4int.h](#).

## 16.535 l4int.h

[Go to the documentation of this file.](#)

```

00001
00006  /*
00007   * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008   *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009   *      economic rights: Technische Universität Dresden (Germany)
00010   *
00011   * This file is part of TUD:OS and distributed under the terms of the
00012   * GNU General Public License 2.
00013   * Please see the COPYING-GPL-2 file for details.
00014   *
00015   * As a special exception, you may use this file as part of a free software

```

```

00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024  #pragma once
00025
00026  #include_next <l4/sys/l4int.h>
00027
00033  #define L4_MWORD_BITS          32
00035  typedef unsigned int           l4_size_t;
00036  typedef signed int             l4_ssize_t;

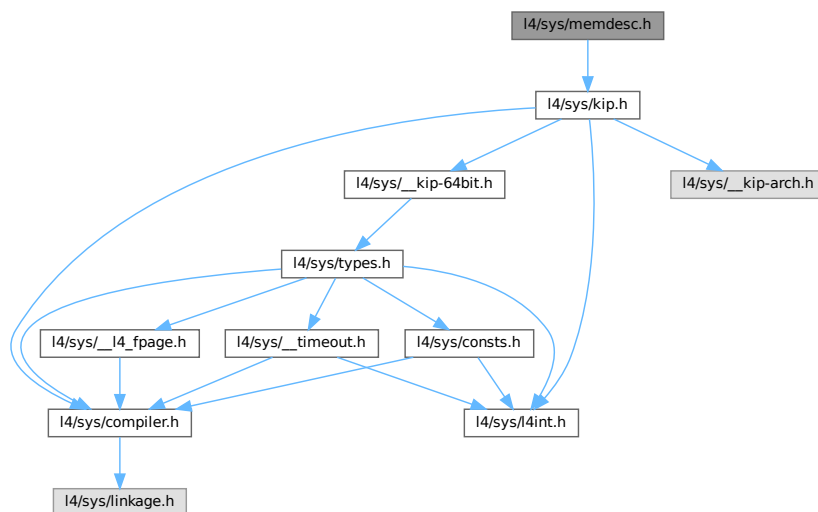
```

## 16.536 l4/sys/memdesc.h File Reference

Memory description functions.

```
#include <l4/sys/kip.h>
```

Include dependency graph for memdesc.h:



### Data Structures

- struct [l4\\_kernel\\_info\\_mem\\_desc\\_t](#)  
*Memory descriptor data structure.*

### Typedefs

- typedef struct [l4\\_kernel\\_info\\_mem\\_desc\\_t](#) [l4\\_kernel\\_info\\_mem\\_desc\\_t](#)  
*Memory descriptor data structure.*

## Enumerations

- enum `l4_mem_type_t` {  
`l4_mem_type_undefined` = 0x0 , `l4_mem_type_conventional` = 0x1 , `l4_mem_type_reserved` = 0x2 ,  
`l4_mem_type_dedicated` = 0x3 ,  
`l4_mem_type_shared` = 0x4 , `l4_mem_type_info` = 0xd , `l4_mem_type_bootloader` = 0xe , `l4_mem_type_archspecific`  
= 0xf }  
*Type of a memory descriptor.*
- enum `l4_mem_info_sub_type_t` { `l4_mem_info_acpi_rsdp` = 0 }  
*Memory sub types for l4\_mem\_type\_info descriptors.*
- enum `l4_mem_archspecific_sub_type_common_t` { `l4_mem_archspecific_acpi_tables` = 3 , `l4_mem_archspecific_acpi_nvs`  
= 4 }  
*Memory sub types for l4\_mem\_type\_archspecific descriptors.*

## Functions

- `l4_kernel_info_mem_desc_t * l4_kernel_info_get_mem_descs (l4_kernel_info_t *kip)` `L4_NOTHROW`  
*Get pointer to memory descriptors from KIP.*
- unsigned `l4_kernel_info_get_num_mem_descs (l4_kernel_info_t *kip)` `L4_NOTHROW`  
*Get number of memory descriptors in KIP.*
- void `l4_kernel_info_set_mem_desc (l4_kernel_info_mem_desc_t *md, l4_addr_t start, l4_addr_t end, unsigned type, unsigned virt, unsigned sub_type)` `L4_NOTHROW`  
*Populate a memory descriptor.*
- `l4_umword_t l4_kernel_info_get_mem_desc_start (l4_kernel_info_mem_desc_t *md)` `L4_NOTHROW`  
*Get start address of the region described by the memory descriptor.*
- `l4_umword_t l4_kernel_info_get_mem_desc_end (l4_kernel_info_mem_desc_t *md)` `L4_NOTHROW`  
*Get end address of the region described by the memory descriptor.*
- `l4_umword_t l4_kernel_info_get_mem_desc_type (l4_kernel_info_mem_desc_t *md)` `L4_NOTHROW`  
*Get type of the memory region.*
- `l4_umword_t l4_kernel_info_get_mem_desc_subtype (l4_kernel_info_mem_desc_t *md)` `L4_NOTHROW`  
*Get sub-type of memory region.*
- `l4_umword_t l4_kernel_info_get_mem_desc_is_virtual (l4_kernel_info_mem_desc_t *md)` `L4_NOTHROW`  
*Get virtual flag of the memory descriptor.*

### 16.536.1 Detailed Description

Memory description functions.

Definition in file [memdesc.h](#).

## 16.537 memdesc.h

[Go to the documentation of this file.](#)

```
00001
00006 /*
00007  * (c) 2007-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
```

```

00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #ifndef __L4SYS__MEMDESC_H__
00025 #define __L4SYS__MEMDESC_H__
00026
00027 #include <l4/sys/kip.h>
00028
00044 enum l4_mem_type_t
00045 {
00046     l4_mem_type_undefined = 0x0,
00047     l4_mem_type_conventional = 0x1,
00048     l4_mem_type_reserved = 0x2,
00049     l4_mem_type_dedicated = 0x3,
00050     l4_mem_type_shared = 0x4,
00051
00052     l4_mem_type_info = 0xd,
00053     l4_mem_type_bootloader = 0xe,
00054     l4_mem_type_archspecific = 0xf,
00055 };
00056
00061 enum l4_mem_info_sub_type_t
00062 {
00063     l4_mem_info_acpi_rsdp = 0
00064 };
00065
00070 enum l4_mem_archspecific_sub_type_common_t
00071 {
00072     l4_mem_archspecific_acpi_tables = 3,
00073     l4_mem_archspecific_acpi_nvs = 4,
00074 };
00075
00076
00084 typedef struct l4_kernel_info_mem_desc_t
00085 {
00086     l4_umword_t l;
00087     l4_umword_t h;
00088 } l4_kernel_info_mem_desc_t;
00089
00090
00097 L4_INLINE
00098 l4_kernel_info_mem_desc_t *
00099 l4_kernel_info_get_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW;
00100
00107 L4_INLINE
00108 unsigned
00109 l4_kernel_info_get_num_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW;
00110
00122 L4_INLINE
00123 void
00124 l4_kernel_info_set_mem_desc(l4_kernel_info_mem_desc_t *md,
00125                             l4_addr_t start,
00126                             l4_addr_t end,
00127                             unsigned type,
00128                             unsigned virt,
00129                             unsigned sub_type) L4_NOTHROW;
00130
00137 L4_INLINE
00138 l4_umword_t
00139 l4_kernel_info_get_mem_desc_start(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00140
00147 L4_INLINE
00148 l4_umword_t
00149 l4_kernel_info_get_mem_desc_end(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00150
00157 L4_INLINE
00158 l4_umword_t
00159 l4_kernel_info_get_mem_desc_type(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00160
00170 L4_INLINE
00171 l4_umword_t
00172 l4_kernel_info_get_mem_desc_subtype(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00173
00180 L4_INLINE
00181 l4_umword_t
00182 l4_kernel_info_get_mem_desc_is_virtual(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00183
00184 /*****
00185  * Implementations
00186  *****/
00187

```



```

00188 L4_INLINE
00189 l4_kernel_info_mem_desc_t *
00190 l4_kernel_info_get_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW
00191 {
00192     return (l4_kernel_info_mem_desc_t *)(((l4_addr_t)kip)
00193         + (kip->mem_info » (sizeof(l4_umword_t) * 4)));
00194 }
00195
00196 L4_INLINE
00197 unsigned
00198 l4_kernel_info_get_num_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW
00199 {
00200     return kip->mem_info & ((1UL « (sizeof(l4_umword_t)*4)) -1);
00201 }
00202
00203 L4_INLINE
00204 void
00205 l4_kernel_info_set_mem_desc(l4_kernel_info_mem_desc_t *md,
00206                             l4_addr_t start,
00207                             l4_addr_t end,
00208                             unsigned type,
00209                             unsigned virt,
00210                             unsigned sub_type) L4_NOTHROW
00211 {
00212     md->l = (start & ~0x3ffUL) | (type & 0x0f) | ((sub_type < 4) & 0xf0)
00213         | (virt ? 0x200 : 0x0);
00214     md->h = end;
00215 }
00216
00217
00218 L4_INLINE
00219 l4_umword_t
00220 l4_kernel_info_get_mem_desc_start(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00221 {
00222     return md->l & ~0x3ffUL;
00223 }
00224
00225 L4_INLINE
00226 l4_umword_t
00227 l4_kernel_info_get_mem_desc_end(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00228 {
00229     return md->h | 0x3ffUL;
00230 }
00231
00232 L4_INLINE
00233 l4_umword_t
00234 l4_kernel_info_get_mem_desc_type(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00235 {
00236     return md->l & 0xf;
00237 }
00238
00239 L4_INLINE
00240 l4_umword_t
00241 l4_kernel_info_get_mem_desc_subtype(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00242 {
00243     return (md->l & 0xf0) » 4;
00244 }
00245
00246 L4_INLINE
00247 l4_umword_t
00248 l4_kernel_info_get_mem_desc_is_virtual(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00249 {
00250     return md->l & 0x200;
00251 }
00252
00253 #endif /* ! __L4SYS__MEMDESC_H__ */

```

## 16.538 l4/sys/pager File Reference

Pager and lo\_pager C++ interface.

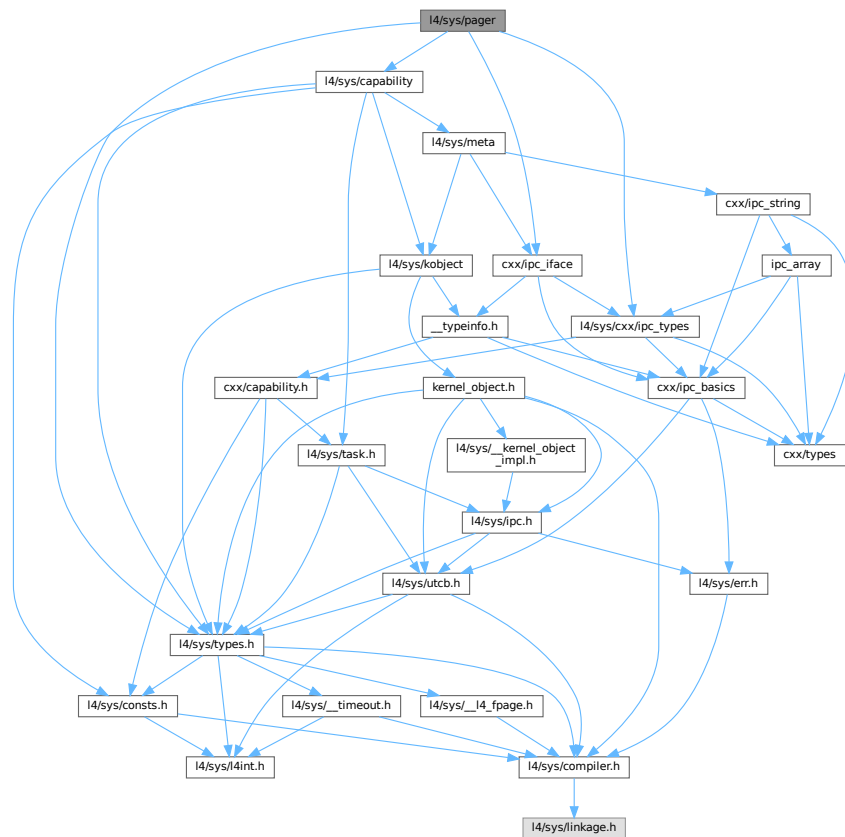
```

#include <l4/sys/capability>
#include <l4/sys/types.h>
#include <l4/sys/cxx/ipc_types>

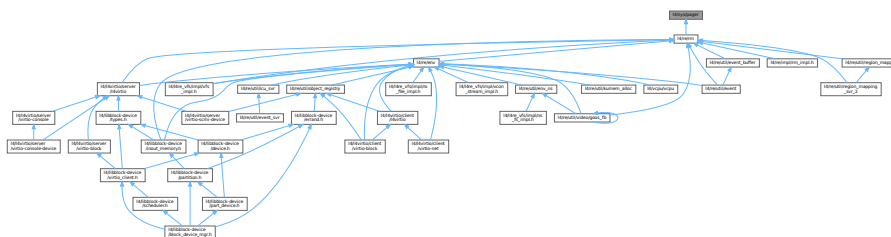
```

```
#include <l4/sys/cxx/ipc_iface>
```

Include dependency graph for pager:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::lo\\_pager](#)  
[lo\\_pager](#) interface.
- class [L4::Pager](#)  
[Pager](#) interface including the [lo\\_pager](#) interface.

## Namespaces

- namespace [L4](#)  
[L4](#) low-level kernel interface.

## 16.538.1 Detailed Description

Pager and Io\_pager C++ interface.

Definition in file [pager](#).

## 16.539 pager

[Go to the documentation of this file.](#)

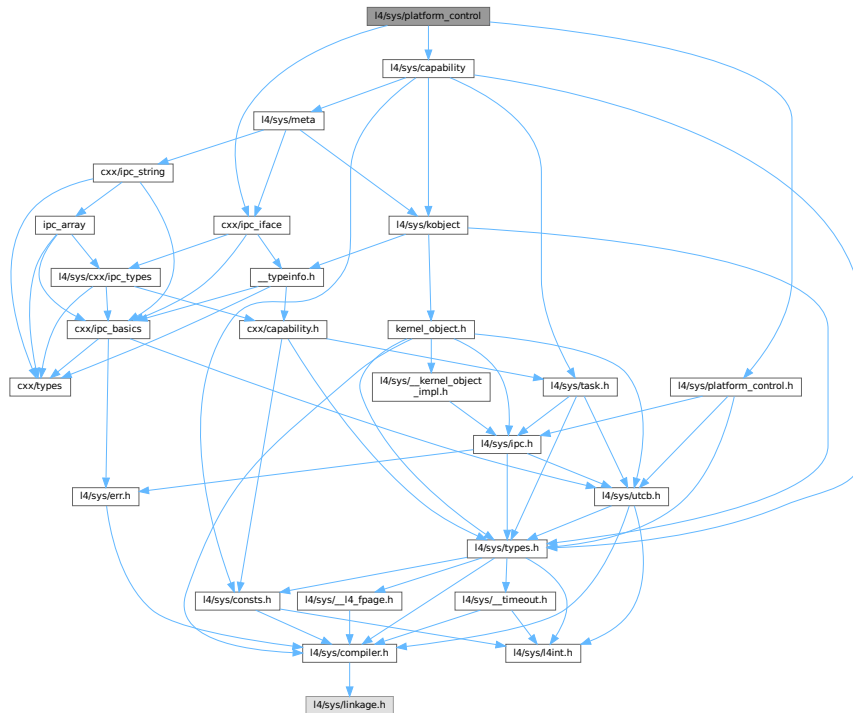
```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00025 #include <l4/sys/capability>
00026 #include <l4/sys/types.h>
00027 #include <l4/sys/cxx/ipc_types>
00028 #include <l4/sys/cxx/ipc_iface>
00029
00030 namespace L4 {
00031
00061 class L4_EXPORT Io_pager :
00062     public Kobject_0t<Io_pager, L4_PROTO_IO_PAGE_FAULT>
00063 {
00064 public:
00080     L4_INLINE_RPC(
00081         l4_msgtag_t, io_page_fault, (l4_fpage_t io_pfa, l4_umword_t pc,
00082                                     L4::Ipc::Rcv_fpage rwin,
00083                                     L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp));
00084
00085     typedef L4::Typeid::Rpc_nocode<io_page_fault_t> Rpcs;
00086 };
00087
00098 class L4_EXPORT Pager :
00099     public Kobject_t<Pager, Io_pager, L4_PROTO_PAGE_FAULT>
00100 {
00101 public:
00134     L4_INLINE_RPC(
00135         l4_msgtag_t, page_fault, (l4_umword_t pfa, l4_umword_t pc,
00136                                  L4::Ipc::Rcv_fpage rwin,
00137                                  L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp));
00138
00139     typedef L4::Typeid::Rpc_nocode<page_fault_t> Rpcs;
00140 };
00141
00142 }
```

## 16.540 l4/sys/platform\_control File Reference

Platform control object.

```
#include <l4/sys/capability>
#include <l4/sys/platform_control.h>
```

```
#include <l4/sys/cxx/ipc_iface>
Include dependency graph for platform_control:
```



## Data Structures

- class [L4::Platform\\_control](#)  
*L4 C++ interface for controlling platform-wide properties, see [Platform Control C API](#) for the C interface.*

## Namespaces

- namespace **L4**  
*L4 low-level kernel interface.*

### 16.540.1 Detailed Description

Platform control object.

Definition in file [platform\\_control](#).

## 16.541 platform\_control

Go to the documentation of this file.

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00008  *      Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025
00026 #pragma once
00027
00028 #include <l4/sys/capability>
00029 #include <l4/sys/platform_control.h>
00030 #include <l4/sys/cxx/ipc_iface>
00031
00032 namespace L4 {
00033
00047 class L4_EXPORT Platform_control
00048 : public Kobject_t<Platform_control, Kobject, L4_PROTO_PLATFORM_CTL>
00049 {
00050 public:
00065     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_SYS_SUSPEND_OP,
00066                     l4_msgtag_t, system_suspend, (l4_umword_t extras));
00067
00073     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_SYS_SHUTDOWN_OP,
00074                     l4_msgtag_t, system_shutdown, (l4_umword_t reboot));
00075
00085     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP,
00086                     l4_msgtag_t, cpu_allow_shutdown,
00087                     (l4_umword_t phys_id, l4_umword_t enable));
00088
00098     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_ENABLE_OP,
00099                     l4_msgtag_t, cpu_enable, (l4_umword_t phys_id));
00100
00110     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_DISABLE_OP,
00111                     l4_msgtag_t, cpu_disable, (l4_umword_t phys_id));
00112
00113     typedef L4::Typeid::Rpcsys<system_suspend_t, system_shutdown_t,
00114                               cpu_allow_shutdown_t, cpu_enable_t,
00115                               cpu_disable_t> Rpcsys;
00116 };
00117
00118 }
```

## 16.542 platform\_control.h

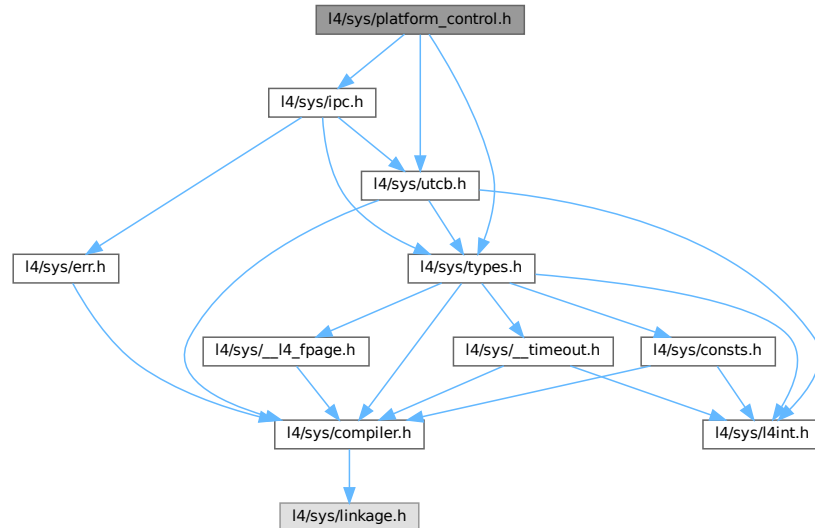
```
00001 /*
00002  * Copyright (C) 2024 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include_next <l4/sys/platform_control.h>
00010 #include <l4/sys/__platform_control-arm.h>
```

## 16.543 l4/sys/platform\_control.h File Reference

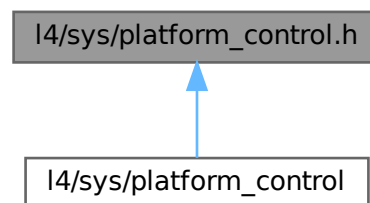
Platform control object.

```
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
```

Include dependency graph for platform\_control.h:



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum `L4_platform_ctl_ops` {  
`L4_PLATFORM_CTL_SYS_SUSPEND_OP` = 0UL , `L4_PLATFORM_CTL_SYS_SHUTDOWN_OP` = 1UL ,  
`L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP` = 2UL , `L4_PLATFORM_CTL_CPU_ENABLE_OP` =  
3UL ,  
`L4_PLATFORM_CTL_CPU_DISABLE_OP` = 4UL , `L4_PLATFORM_CTL_SET_TASK_ASID_OP` = 0x10UL }  
*Operations on platform-control objects.*
- enum `L4_platform_ctl_proto` { `L4_PROTO_PLATFORM_CTL` = 0 }  
*Predefined protocol type for messages to platform-control objects.*

## Functions

- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_system\\_suspend \(l4\\_cap\\_idx\\_t pfc, l4\\_umword\\_t extras\) L4\\_NOTHROW](#)  
*Enter suspend to RAM.*
- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_system\\_shutdown \(l4\\_cap\\_idx\\_t pfc, l4\\_umword\\_t reboot\) L4\\_NOTHROW](#)  
*Shutdown or reboot the system.*
- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_cpu\\_allow\\_shutdown \(l4\\_cap\\_idx\\_t pfc, l4\\_umword\\_t phys\\_id, l4\\_umword\\_t enable\) L4\\_NOTHROW](#)  
*Allow a CPU to be shut down.*
- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_cpu\\_enable \(l4\\_cap\\_idx\\_t pfc, l4\\_umword\\_t phys\\_id\) L4\\_NOTHROW](#)  
*Enable an offline CPU.*
- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_cpu\\_disable \(l4\\_cap\\_idx\\_t pfc, l4\\_umword\\_t phys\\_id\) L4\\_NOTHROW](#)  
*Disable an online CPU.*

### 16.543.1 Detailed Description

Platform control object.

Definition in file [platform\\_control.h](#).

## 16.544 platform\_control.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00007  *
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU General Public License 2.
00010  * Please see the COPYING-GPL-2 file for details.
00011  *
00012  * As a special exception, you may use this file as part of a free software
00013  * library without restriction. Specifically, if other files instantiate
00014  * templates or use macros or inline functions from this file, or you compile
00015  * this file and link it with other files to produce an executable, this
00016  * file does not by itself cause the resulting executable to be covered by
00017  * the GNU General Public License. This exception does not however
00018  * invalidate any other reasons why the executable file might be covered by
00019  * the GNU General Public License.
00020  */
00021
00022 #pragma once
00023
00024 #include <l4/sys/types.h>
00025 #include <l4/sys/utcb.h>
00026
00060 L4_INLINE l4_msgtag_t
00061 l4_platform_ctl_system_suspend(l4_cap_idx_t pfc,
00062                                l4_umword_t extras) L4_NOTHROW;
00063
00067 L4_INLINE l4_msgtag_t
00068 l4_platform_ctl_system_suspend_u(l4_cap_idx_t pfc,
00069                                  l4_umword_t extras,
00070                                  l4_utcb_t *utcb) L4_NOTHROW;
00071
00072
00081 L4_INLINE l4_msgtag_t
00082 l4_platform_ctl_system_shutdown(l4_cap_idx_t pfc,
00083                                 l4_umword_t reboot) L4_NOTHROW;
00084
00088 L4_INLINE l4_msgtag_t
00089 l4_platform_ctl_system_shutdown_u(l4_cap_idx_t pfc,
00090                                   l4_umword_t reboot,
00091                                   l4_utcb_t *utcb) L4_NOTHROW;
00092
00102 L4_INLINE l4_msgtag_t
00103 l4_platform_ctl_cpu_allow_shutdown(l4_cap_idx_t pfc,
```

```

00104         l4_umword_t phys_id,
00105         l4_umword_t enable) L4_NOTHROW;
00106
00110 L4_INLINE l4_msgtag_t
00111 l4_platform_ctl_cpu_allow_shutdown_u(l4_cap_idx_t pfc,
00112         l4_umword_t phys_id,
00113         l4_umword_t enable,
00114         l4_utcb_t *utcb) L4_NOTHROW;
00125 L4_INLINE l4_msgtag_t
00126 l4_platform_ctl_cpu_enable(l4_cap_idx_t pfc,
00127         l4_umword_t phys_id) L4_NOTHROW;
00128
00132 L4_INLINE l4_msgtag_t
00133 l4_platform_ctl_cpu_enable_u(l4_cap_idx_t pfc,
00134         l4_umword_t phys_id,
00135         l4_utcb_t *utcb) L4_NOTHROW;
00136
00147 L4_INLINE l4_msgtag_t
00148 l4_platform_ctl_cpu_disable(l4_cap_idx_t pfc,
00149         l4_umword_t phys_id) L4_NOTHROW;
00150
00154 L4_INLINE l4_msgtag_t
00155 l4_platform_ctl_cpu_disable_u(l4_cap_idx_t pfc,
00156         l4_umword_t phys_id,
00157         l4_utcb_t *utcb) L4_NOTHROW;
00158
00160 /* ends l4_platform_control_api group */
00161
00170 enum L4_platform_ctl_ops
00171 {
00172     L4_PLATFORM_CTL_SYS_SUSPEND_OP      = 0UL,
00173     L4_PLATFORM_CTL_SYS_SHUTDOWN_OP     = 1UL,
00174     L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP = 2UL,
00175     L4_PLATFORM_CTL_CPU_ENABLE_OP       = 3UL,
00176     L4_PLATFORM_CTL_CPU_DISABLE_OP      = 4UL,
00177     L4_PLATFORM_CTL_SET_TASK_ASID_OP    = 0x10UL,
00178 };
00180
00185 enum L4_platform_ctl_proto
00186 {
00192     L4_PROTO_PLATFORM_CTL = 0
00193 };
00194
00195 /* IMPLEMENTATION -----*/
00196
00197 #include <l4/sys/ipc.h>
00198
00199 L4_INLINE l4_msgtag_t
00200 l4_platform_ctl_system_suspend_u(l4_cap_idx_t pfc,
00201         l4_umword_t extras,
00202         l4_utcb_t *utcb) L4_NOTHROW
00203 {
00204     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00205     v->mr[0] = L4_PLATFORM_CTL_SYS_SUSPEND_OP;
00206     v->mr[1] = extras;
00207     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00208         L4_IPC_NEVER);
00209 }
00210
00211 L4_INLINE l4_msgtag_t
00212 l4_platform_ctl_system_shutdown_u(l4_cap_idx_t pfc,
00213         l4_umword_t reboot,
00214         l4_utcb_t *utcb) L4_NOTHROW
00215 {
00216     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00217     v->mr[0] = L4_PLATFORM_CTL_SYS_SHUTDOWN_OP;
00218     v->mr[1] = reboot;
00219     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00220         L4_IPC_NEVER);
00221 }
00222
00223
00224 L4_INLINE l4_msgtag_t
00225 l4_platform_ctl_system_suspend(l4_cap_idx_t pfc,
00226         l4_umword_t extras) L4_NOTHROW
00227 {
00228     return l4_platform_ctl_system_suspend_u(pfc, extras, l4_utcb());
00229 }
00230
00231 L4_INLINE l4_msgtag_t
00232 l4_platform_ctl_system_shutdown(l4_cap_idx_t pfc,
00233         l4_umword_t reboot) L4_NOTHROW
00234 {
00235     return l4_platform_ctl_system_shutdown_u(pfc, reboot, l4_utcb());
00236 }
00237
00238 L4_INLINE l4_msgtag_t

```



```

00239 l4_platform_ctl_cpu_allow_shutdown_u(l4_cap_idx_t pfc,
00240                                     l4_umword_t phys_id,
00241                                     l4_umword_t enable,
00242                                     l4_utcb_t *utcb) L4_NOTHROW
00243 {
00244     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00245     v->mr[0] = L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP;
00246     v->mr[1] = phys_id;
00247     v->mr[2] = enable;
00248     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 3, 0, 0),
00249                       L4_IPC_NEVER);
00250 }
00251
00252 L4_INLINE l4_msgtag_t
00253 l4_platform_ctl_cpu_allow_shutdown(l4_cap_idx_t pfc,
00254                                   l4_umword_t phys_id,
00255                                   l4_umword_t enable) L4_NOTHROW
00256 {
00257     return l4_platform_ctl_cpu_allow_shutdown_u(pfc, phys_id, enable, l4_utcb());
00258 }
00259
00260 L4_INLINE l4_msgtag_t
00261 l4_platform_ctl_cpu_enable_u(l4_cap_idx_t pfc,
00262                             l4_umword_t phys_id,
00263                             l4_utcb_t *utcb) L4_NOTHROW
00264 {
00265     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00266     v->mr[0] = L4_PLATFORM_CTL_CPU_ENABLE_OP;
00267     v->mr[1] = phys_id;
00268     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00269                       L4_IPC_NEVER);
00270 }
00271
00272 L4_INLINE l4_msgtag_t
00273 l4_platform_ctl_cpu_disable_u(l4_cap_idx_t pfc,
00274                              l4_umword_t phys_id,
00275                              l4_utcb_t *utcb) L4_NOTHROW
00276 {
00277     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00278     v->mr[0] = L4_PLATFORM_CTL_CPU_DISABLE_OP;
00279     v->mr[1] = phys_id;
00280     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00281                       L4_IPC_NEVER);
00282 }
00283
00284 L4_INLINE l4_msgtag_t
00285 l4_platform_ctl_cpu_enable(l4_cap_idx_t pfc,
00286                           l4_umword_t phys_id) L4_NOTHROW
00287 {
00288     return l4_platform_ctl_cpu_enable_u(pfc, phys_id, l4_utcb());
00289 }
00290
00291 L4_INLINE l4_msgtag_t
00292 l4_platform_ctl_cpu_disable(l4_cap_idx_t pfc,
00293                             l4_umword_t phys_id) L4_NOTHROW
00294 {
00295     return l4_platform_ctl_cpu_disable_u(pfc, phys_id, l4_utcb());
00296 }

```

## 16.545 l4/sys/rcv\_endpoint File Reference

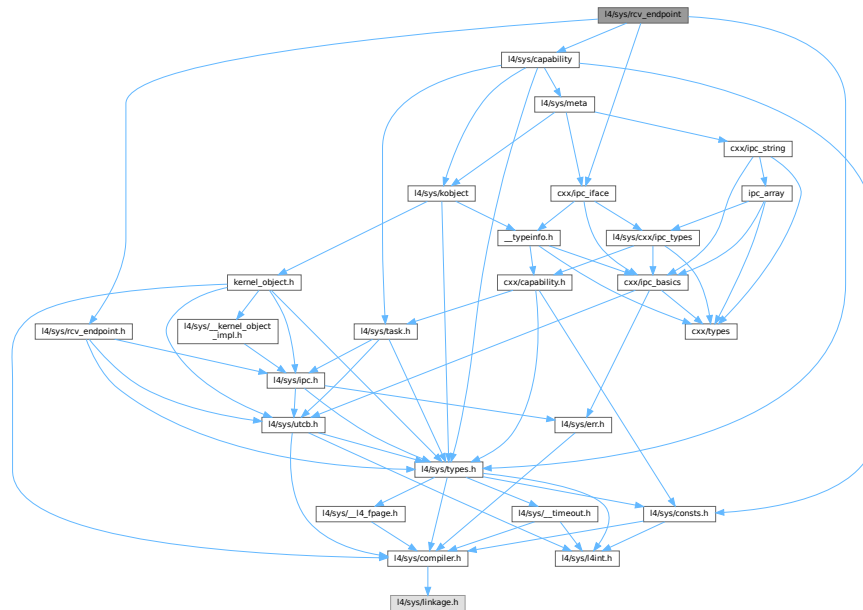
The C++ Receive endpoint interface.

```

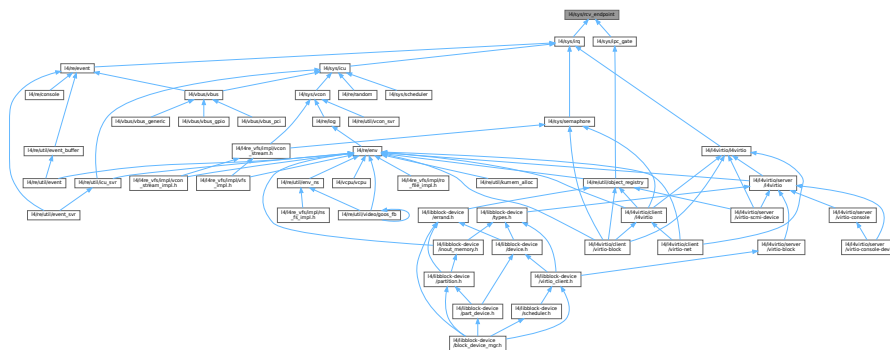
#include <l4/sys/rcv_endpoint.h>
#include <l4/sys/types.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>

```

Include dependency graph for `rcv_endpoint`:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::Rcv\\_endpoint](#)  
*Interface for kernel objects that allow to receive IPC from them.*

## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

## 16.545.1 Detailed Description

The C++ Receive endpoint interface.

Definition in file [rcv\\_endpoint](#).

## 16.546 rcv\_endpoint

[Go to the documentation of this file.](#)

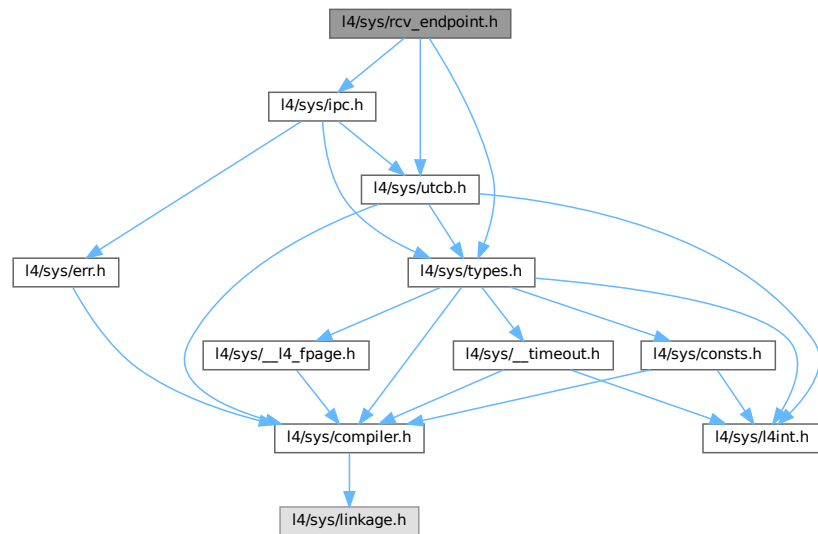
```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #pragma once
00023
00024 #include <l4/sys/rcv_endpoint.h>
00025 #include <l4/sys/types.h>
00026 #include <l4/sys/capability>
00027 #include <l4/sys/cxx/ipc_iface>
00028
00029 namespace L4 {
00030
00031 class Thread;
00032
00040 class L4_EXPORT Rcv_endpoint :
00041     public Kobject_t<Rcv_endpoint, Kobject, L4_PROTO_KOBJECT,
00042         Type_info::Demand_t<1> >
00043 {
00044 public:
00072     L4_INLINE_RPC_OP(L4_RCV_EP_BIND_OP,
00073         l4_msgtag_t, bind_thread, (Ipc::Cap<Thread> t, l4_umword_t label));
00074
00075     typedef L4::Typeid::Rpcsys<bind_thread_t> Rpcsys;
00076 };
00077
00078 }
```

## 16.547 l4/sys/rcv\_endpoint.h File Reference

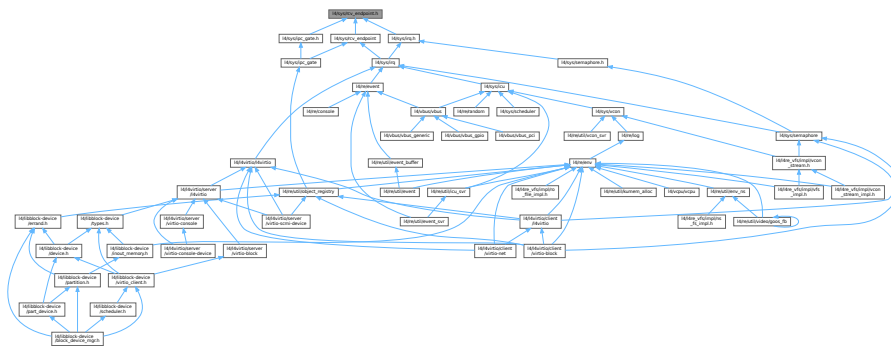
Receive endpoint C interface.

```
#include <l4/sys/utcb.h>
#include <l4/sys/types.h>
#include <l4/sys/ipc.h>
```

Include dependency graph for `rcv_endpoint.h`:



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum `L4_rcv_ep_ops` { `L4_RCV_EP_BIND_OP` = 0x10 }  
Receive endpoint operations.

## Functions

- `l4_msgtag_t l4_rcv_ep_bind_thread(l4_cap_idx_t ep, l4_cap_idx_t thread, l4_umword_t label)`  
Bind the IPC receive endpoint to a thread.

## 16.547.1 Detailed Description

Receive endpoint C interface.

Definition in file `rcv_endpoint.h`.

## 16.547.2 Enumeration Type Documentation

### 16.547.2.1 L4\_rcv\_ep\_ops

enum [L4\\_rcv\\_ep\\_ops](#)

Receive endpoint operations.

Enumerator

L4_RCV_EP_BIND_OP	Bind operation.
-------------------	-----------------

Definition at line 67 of file [rcv\\_endpoint.h](#).

## 16.548 rcv\_endpoint.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00007  *
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU General Public License 2.
00010  * Please see the COPYING-GPL-2 file for details.
00011  *
00012  * As a special exception, you may use this file as part of a free software
00013  * library without restriction. Specifically, if other files instantiate
00014  * templates or use macros or inline functions from this file, or you compile
00015  * this file and link it with other files to produce an executable, this
00016  * file does not by itself cause the resulting executable to be covered by
00017  * the GNU General Public License. This exception does not however
00018  * invalidate any other reasons why the executable file might be covered by
00019  * the GNU General Public License.
00020  */
00021 #pragma once
00022
00023 #include <l4/sys/utcb.h>
00024 #include <l4/sys/types.h>
00025
00054 L4_INLINE l4_msgtag_t
00055 l4_rcv_ep_bind_thread(l4_cap_idx_t ep, l4_cap_idx_t thread,
00056                      l4_umword_t label);
00057
00062 L4_INLINE l4_msgtag_t
00063 l4_rcv_ep_bind_thread_u(l4_cap_idx_t ep, l4_cap_idx_t thread,
00064                        l4_umword_t label, l4_utcb_t *utcb);
00065
00067 enum L4_rcv_ep_ops
00068 {
00069     L4_RCV_EP_BIND_OP      = 0x10,
00070 };
00071
00072 /* IMPLEMENTATION -----*/
00073
00074 #include <l4/sys/ipc.h>
00075
00076 L4_INLINE l4_msgtag_t
00077 l4_rcv_ep_bind_thread_u(l4_cap_idx_t ep,
00078                        l4_cap_idx_t thread, l4_umword_t label,
00079                        l4_utcb_t *utcb)
00080 {
00081     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00082     m->mr[0] = L4_RCV_EP_BIND_OP;
00083     m->mr[1] = label;
00084     m->mr[2] = l4_map_obj_control(0, 0);
00085     m->mr[3] = l4_obj_fpage(thread, 0, L4_CAP_FPAGE_RWS).raw;
00086     return l4_ipc_call(ep, utcb, l4_msgtag(L4_PROTO_KOBJECT, 2, 1, 0),
00087                       L4_IPC_NEVER);
00088 }
00089
00090 L4_INLINE l4_msgtag_t

```

```

00091 l4_rcv_ep_bind_thread(l4_cap_idx_t ep, l4_cap_idx_t thread,
00092                        l4_umword_t label)
00093 {
00094     return l4_rcv_ep_bind_thread_u(ep, thread, label, l4_utcb());
00095 }
00096
00097

```

## 16.549 l4/sys/scheduler File Reference

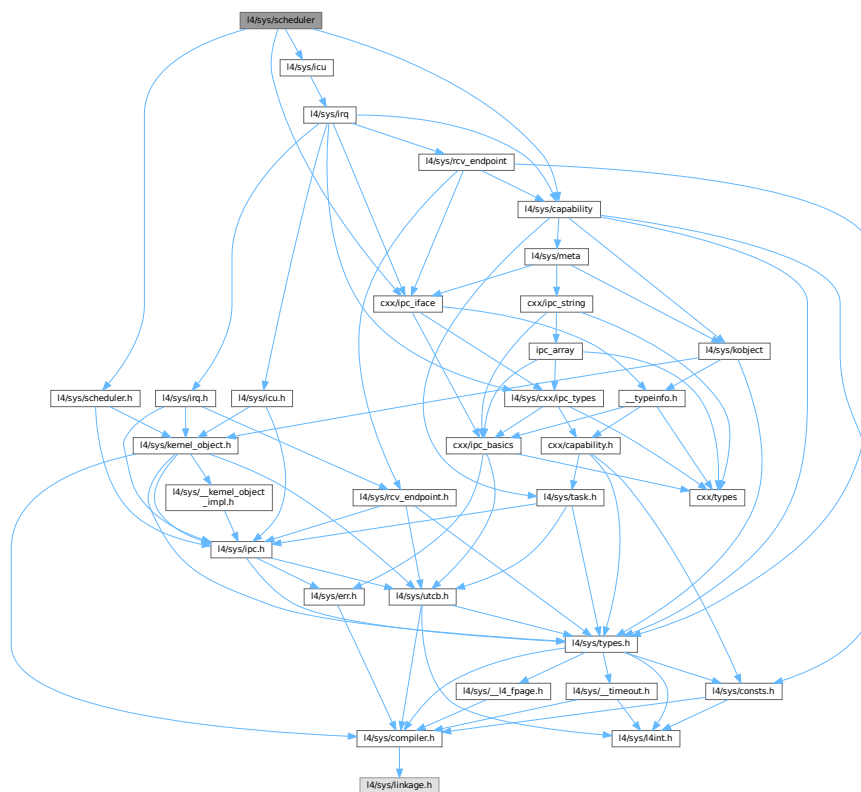
Scheduler object functions.

```

#include <l4/sys/icu>
#include <l4/sys/scheduler.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>

```

Include dependency graph for scheduler:



### Data Structures

- class [L4::Scheduler](#)  
C++ interface of the [Scheduler](#) kernel object, see [Scheduler](#) for the C interface.

### Namespaces

- namespace [L4](#)  
[L4](#) low-level kernel interface.

## 16.549.1 Detailed Description

Scheduler object functions.

Definition in file [scheduler](#).

## 16.550 scheduler

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/sys/icu>
00027 #include <l4/sys/scheduler.h>
00028 #include <l4/sys/capability>
00029 #include <l4/sys/cxx/ipc_iface>
00030
00031 namespace L4 {
00032
00057 class L4_EXPORT Scheduler :
00058     public Kobject_t<Scheduler, Icu, L4_PROTO_SCHEDULER,
00059         Type_info::Demand_t<l> >
00060 {
00061 public:
00062     // ABI function for 'info' call
00063     L4_INLINE_RPC_NF_OP(L4_SCHEDULER_INFO_OP,
00064         l4_msgtag_t, info, (l4_umword_t gran_offset, l4_umword_t *map,
00065             l4_umword_t *cpu_max, l4_umword_t *sched_classes));
00066
00085     l4_msgtag_t info(l4_umword_t *cpu_max, l4_sched_cpu_set_t *cpus,
00086         l4_umword_t *sched_classes = nullptr,
00087         l4_utcb_t *utcb = l4_utcb()) const noexcept
00088     {
00089         l4_umword_t max = 0;
00090         l4_umword_t sc = 0;
00091         l4_msgtag_t t =
00092             info_t::call(c(), cpus->gran_offset, &cpus->map, &max, &sc, utcb);
00093         if (cpu_max)
00094             *cpu_max = max;
00095         if (sched_classes)
00096             *sched_classes = sc;
00097         return t;
00098     }
00099
00123     L4_INLINE_RPC_OP(L4_SCHEDULER_RUN_THREAD_OP,
00124         l4_msgtag_t, run_thread, (Ipc::Cap<Thread> thread, l4_sched_param_t const &sp));
00125
00152     L4_INLINE_RPC_OP(L4_SCHEDULER_IDLE_TIME_OP,
00153         l4_msgtag_t, idle_time, (l4_sched_cpu_set_t const &cpus,
00154             l4_kernel_clock_t *us));
00155
00165     bool is_online(l4_umword_t cpu, l4_utcb_t *utcb = l4_utcb()) const noexcept
00166     { return l4_scheduler_is_online_u(cap(), cpu, utcb); }
00167
00168     typedef L4::Typeid::Rpc_sys<info_t, run_thread_t, idle_time_t> Rpc;
00169 };
00170 }
```

## 16.551 scheduler.h

```

00001 /*
00002  * Copyright (C) 2024 Kernkonzept GmbH.
00003  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <vector>
00011
00012 #include <l4/cxx/unique_ptr>
00013 #include <l4/re/error_helper>
00014 #include <l4/sys/cxx/ipc_epiface>
00015
00016 #include <l4/libblock-device/debug.h>
00017 #include <l4/libblock-device/virtio_client.h>
00018
00019 namespace Block_device {
00020
00021 template <typename DEV>
00022 class Scheduler_base
00023 {
00024 protected:
00025     using Device_type = DEV;
00026     using Client_type = Virtio_client<Device_type>;
00027
00028 private:
00029     class Irq_object : public L4::Irqep_t<Irq_object>
00030     {
00031     public:
00032         Irq_object(Scheduler_base *parent) : _parent(parent) {}
00033
00034         void handle_irq() { _parent->schedule(); }
00035
00036     private:
00037         Scheduler_base *_parent;
00038     };
00039     Irq_object _irq_handler;
00040
00041     struct Context
00042     {
00043         cxx::unique_ptr<Pending_request> pending;
00044         Client_type *client;
00045
00046         bool device_busy;
00047         l4_size_t cost;
00048
00049         Context(Client_type *client) : client(client), device_busy(false), cost(0)
00050         {}
00051
00052         bool same_notification_domain(Client_type const *c) const
00053         { return c->notification_domain() == client->notification_domain(); }
00054     };
00055
00056     using Queue_type = std::vector<cxx::unique_ptr<Context>;
00057     using Iterator_type = typename Queue_type::const_iterator;
00058
00059 public:
00060     Scheduler_base(L4::Registry_iface *registry)
00061     : _irq_handler(this), _registry(registry), _next(_clients.cend())
00062     {
00063         L4Re::chkcap(registry->register_irq_obj(&_irq_handler),
00064             "Registering device notify IRQ object.");
00065     }
00066
00067     virtual ~Scheduler_base()
00068     {
00069         // We need to explicitly delete the IRQ object created in register_irq_obj()
00070         // ourselves. Even though unregister_obj() will unmap the cap, it might stay
00071         // alive because it was given out to the client. Hence it might be
00072         // dispatched even after unregister_obj() returned!
00073         L4::Cap<L4::Task>(L4Re::This_task)
00074         ->unmap(_irq_handler.obj_cap().fpage(),
00075             L4_FP_ALL_SPACES | L4_FP_DELETE_OBJ);
00076         _registry->unregister_obj(&_irq_handler);
00077     }
00078
00079     virtual l4_size_t get_weight(Client_type const *) = 0;
00080
00081     virtual l4_size_t get_cost(Pending_request const &) = 0;
00082
00083     void add_client(Client_type *client)
00084     {
00085         Dbg::trace().printf("Adding client %p to request scheduler.\n", client);
00086     }

```



```

00105
00106 // make sure the client uses the request scheduler's device_notify_irq
00107 client->set_device_notify_irq{
00108     L4::cap_cast<L4::Irq>(_irq_handler.obj_cap());
00109
00110 client->set_client_invalidate_cb([this, client](bool fail_pending) {
00111     client_invalidate(client, fail_pending);
00112 });
00113
00114 client->set_client_idle_cb([this, client]() { client_idle(client); });
00115
00116 _clients.push_back(cxx::make_unique<Context>(client));
00117 _next = _clients.cend();
00118 }
00119
00120 void remove_client(Client_type *client)
00121 {
00122     Dbg::trace().printf("Removing client %p from request scheduler.\n", client);
00123     _clients.erase(std::remove_if(_clients.begin(), _clients.end(),
00124                                   [client](cxx::unique_ptr<Context> &c) {
00125                                     return c->client == client;
00126                                   }));
00127     _next = _clients.cend();
00128 }
00129
00130 Queue_type const &clients()
00131 { return _clients; }
00132
00133 private:
00134 void client_invalidate(Client_type *client, bool fail_pending)
00135 {
00136     for (auto &c : _clients)
00137         if (c->client == client)
00138         {
00139             c->device_busy = false;
00140             c->cost = 0;
00141             if (c->pending)
00142             {
00143                 if (fail_pending)
00144                     c->pending->fail_request();
00145                 c->pending.reset();
00146             }
00147         }
00148 }
00149
00150 void client_idle(Client_type *client)
00151 {
00152     bool resched = false;
00153     for (auto &c : _clients)
00154         if (c->device_busy && c->same_notification_domain(client))
00155         {
00156             c->device_busy = false;
00157             resched = true;
00158         }
00159     if (resched)
00160     {
00161         // By triggering the scheduler asynchronously we make synchronous
00162         // request processing in the device implementation possible. In
00163         // any case we need to be careful not to start scheduling the
00164         // pending request which is being currently handled.
00165         L4::cap_cast<L4::Irq>(this->_irq_handler.obj_cap())->trigger();
00166     }
00167 }
00168
00169 bool handle_pending(Context *c)
00170 {
00171     auto cost = get_cost(*(c->pending));
00172     if (c->cost + cost > get_weight(c->client))
00173     {
00174         Dbg::trace().printf("Preempting client %p (cost=%zu+%zu, weight=%zu)\n",
00175                             c->client, c->cost, cost, get_weight(c->client));
00176
00177         // Charge client's entire weight to force schedule() to give another
00178         // client a chance.
00179         c->cost = get_weight(c->client);
00180         return true;
00181     }
00182
00183     // Keep the pending request in its place while handling the request.
00184     // This helps to make sure that the scheduler will not try to schedule
00185     // new requests while handling the pending one.
00186     int ret = c->pending->handle_request();
00187     if (ret == -L4_EBUSY)
00188     {
00189         c->device_busy = true;
00190     }
00191 }

```

```

00203         return false;
00204     }
00205
00206     c->cost += cost;
00207
00208     if (ret < 0)
00209         c->pending.reset();
00210     else
00211         c->pending.release();
00212     return true;
00213 }
00214
00231 bool schedule_client(Context *c)
00232 {
00233     if (c->pending)
00234     {
00235         if (c->device_busy)
00236         {
00237             Dbg::trace().printf(
00238                 "Skipping pending request of client %p (busy).\n", c->client);
00239             return false;
00240         }
00241
00242         Dbg::trace().printf("Handling pending request of client %p.\n",
00243             c->client);
00244         // The client has a pending request, we need to handle it first
00245         // before new requests can be processed. If we manage to handle it,
00246         // we need to check again in the next round for new requests.
00247         return handle_pending(c);
00248     }
00249
00250     if (c->client->check_for_new_requests())
00251     {
00252         auto req = c->client->get_request();
00253         if (req)
00254         {
00255             Dbg::trace().printf("Scheduling request from client %p.\n",
00256                 c->client);
00257             c->pending = c->client->start_request(cxx::move(req));
00258             if (c->pending)
00259             {
00260                 // We processed one new request by turning it into a pending
00261                 // one and possibly sending it to the device (or not). We
00262                 // need to recheck only if the request was successfully sent
00263                 // to the device.
00264                 return handle_pending(c);
00265             }
00266             // We processed one new request immediately (e.g. failed
00267             // sanity check, runtime error or client state).
00268             return true;
00269         }
00270     }
00271
00272     return false;
00273 }
00274
00287 void schedule()
00288 {
00289     if (_clients.empty())
00290         return;
00291
00292     if (_next == _clients.cend())
00293         _next = _clients.cbegin();
00294
00295     (*_next)->cost = 0;
00296
00297     Iterator_type start(_next);
00298     bool recheck = false;
00299     for (;;)
00300     {
00301         bool progress = schedule_client(_next->get());
00302         // Move onto the next client only after the current client has depleted
00303         // its chances to process its queue or if it didn't make any forward
00304         // progress
00305         if (!progress || ((*_next)->cost >= get_weight((*_next)->client)))
00306         {
00307             ++_next;
00308             if (_next == _clients.cend())
00309                 _next = _clients.cbegin();
00310             (*_next)->cost = 0;
00311         }
00312         recheck |= progress;
00313         if (_next == start)
00314         {
00315             if (!recheck)
00316             {
00317                 // already processed all clients and requests, start with

```

```

00318             // the next client next time
00319             ++_next;
00320             break;
00321         }
00322         else
00323             recheck = false;
00324     }
00325 }
00326 }
00327
00328 L4::Registry_iface *_registry;
00329 Queue_type _clients;
00330 Iterator_type _next;
00331 };
00332
00333 template <typename DEV>
00334 struct Rr_scheduler : Scheduler_base<DEV>
00335 {
00336     using Scheduler_base<DEV>::Scheduler_base;
00337
00338     l4_size_t
00339     get_weight(typename Scheduler_base<DEV>::Client_type const *) override
00340     { return 1; }
00341
00342     l4_size_t get_cost(Pending_request const &) override
00343     { return 1; }
00344 };
00345
00346 // name space

```

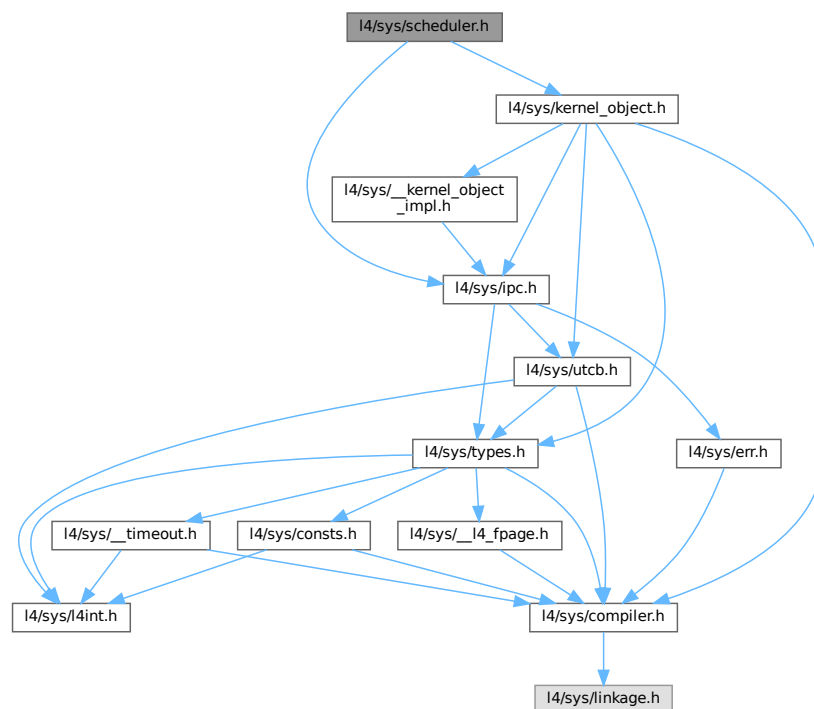
## 16.552 l4/sys/scheduler.h File Reference

Scheduler object functions.

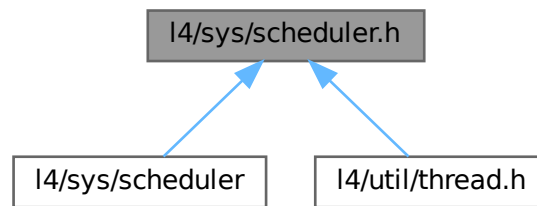
```
#include <l4/sys/kernel_object.h>
```

```
#include <l4/sys/ipc.h>
```

Include dependency graph for scheduler.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4\\_sched\\_cpu\\_set\\_t](#)  
*CPU sets.*
- struct [l4\\_sched\\_param\\_t](#)  
*Scheduler parameter set.*

## Typedefs

- typedef struct [l4\\_sched\\_cpu\\_set\\_t](#) [l4\\_sched\\_cpu\\_set\\_t](#)  
*CPU sets.*
- typedef struct [l4\\_sched\\_param\\_t](#) [l4\\_sched\\_param\\_t](#)  
*Scheduler parameter set.*

## Enumerations

- enum [L4\\_scheduler\\_classes](#) { [L4\\_SCHEDULER\\_CLASS\\_FIXED\\_PRIO](#) = 1UL << 1 , [L4\\_SCHEDULER\\_CLASS\\_WFQ](#) = 1UL << 2 }
  - enum [L4\\_scheduler\\_ops](#) { [L4\\_SCHEDULER\\_INFO\\_OP](#) = 0UL , [L4\\_SCHEDULER\\_RUN\\_THREAD\\_OP](#) = 1UL , [L4\\_SCHEDULER\\_IDLE\\_TIME\\_OP](#) = 2UL }
- Supported scheduler classes.*
- Operations on the Scheduler object.*

## Functions

- [l4\\_sched\\_cpu\\_set\\_t l4\\_sched\\_cpu\\_set](#) ([l4\\_umword\\_t](#) offset, unsigned char granularity, [l4\\_umword\\_t](#) map=1) [L4\\_NOTHROW](#)
  - [l4\\_msgtag\\_t l4\\_scheduler\\_info](#) ([l4\\_cap\\_idx\\_t](#) scheduler, [l4\\_umword\\_t](#) \*cpu\_max, [l4\\_sched\\_cpu\\_set\\_t](#) \*cpus) [L4\\_NOTHROW](#))
  - [l4\\_msgtag\\_t l4\\_scheduler\\_info\\_with\\_classes](#) ([l4\\_cap\\_idx\\_t](#) scheduler, [l4\\_umword\\_t](#) \*cpu\_max, [l4\\_sched\\_cpu\\_set\\_t](#) \*cpus, [l4\\_umword\\_t](#) \*sched\_classes) [L4\\_NOTHROW](#))
  - [l4\\_sched\\_param\\_t l4\\_sched\\_param](#) (unsigned prio, [l4\\_umword\\_t](#) quantum=0) [L4\\_NOTHROW](#)
- Get scheduler information.*
- Get scheduler information.*

*Construct scheduler parameter.*

- `l4_msgtag_t l4_scheduler_run_thread (l4_cap_idx_t scheduler, l4_cap_idx_t thread, l4_sched_param_t const *sp) L4_NOTHROW`

*Run a thread on a Scheduler.*

- `l4_msgtag_t l4_scheduler_idle_time (l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus, l4_kernel_clock_t *us) L4_NOTHROW`

*Query the idle time (in  $\mu$ s) of a CPU.*

- `int l4_scheduler_is_online (l4_cap_idx_t scheduler, l4_umword_t cpu) L4_NOTHROW`

*Query if a CPU is online.*

## 16.552.1 Detailed Description

Scheduler object functions.

Definition in file [scheduler.h](#).

## 16.553 scheduler.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 #include <l4/sys/kernel_object.h>
00026 #include <l4/sys/ipc.h>
00027
00057 enum l4_scheduler_classes
00058 {
00060     L4_SCHEDULER_CLASS_FIXED_PRIO = 1UL << 1,
00062     L4_SCHEDULER_CLASS_WFQ        = 1UL << 2,
00063 };
00064
00069 typedef struct l4_sched_cpu_set_t
00070 {
00083     l4_umword_t gran_offset;
00084
00088     l4_umword_t map;
00089
00090 #ifdef __cplusplus
00092     unsigned char granularity() const { return gran_offset >> 24; }
00094     unsigned offset() const { return gran_offset & 0x00ffffff; }
00101     void set(unsigned char granularity, unsigned offset)
00102     {
00103         gran_offset = (static_cast<l4_umword_t>(granularity) << 24)
00104             | (offset & 0x00ffffff);
00105     }
00106 #endif
00107 } l4_sched_cpu_set_t;
00108
00119 L4_INLINE l4_sched_cpu_set_t
00120 l4_sched_cpu_set(l4_umword_t offset, unsigned char granularity,
00121                 l4_umword_t map L4_DEFAULT_PARAM(1)) L4_NOTHROW;
00122
```

```

00139 L4_INLINE l4_msgtag_t
00140 l4_scheduler_info(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00141                  l4_sched_cpu_set_t *cpus)
00142     L4_NOTHROW __attribute__((nonnull (3)));
00143
00165 L4_INLINE l4_msgtag_t
00166 l4_scheduler_info_with_classes(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00167                               l4_sched_cpu_set_t *cpus,
00168                               l4_umword_t *sched_classes)
00169     L4_NOTHROW __attribute__((nonnull (3)));
00170
00174 L4_INLINE l4_msgtag_t
00175 l4_scheduler_info_u(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00176                    l4_sched_cpu_set_t *cpus, l4_umword_t *sched_classes,
00177                    l4_utcb_t *utcb) L4_NOTHROW __attribute__((nonnull (3, 5)));
00178
00179
00184 typedef struct l4_sched_param_t
00185 {
00186     l4_sched_cpu_set_t affinity;
00187     l4_umword_t prio;
00188     l4_umword_t quantum;
00189 } l4_sched_param_t;
00190
00197 L4_INLINE l4_sched_param_t
00198 l4_sched_param(unsigned prio,
00199                l4_umword_t quantum L4_DEFAULT_PARAM(0)) L4_NOTHROW;
00200
00208 L4_INLINE l4_msgtag_t
00209 l4_scheduler_run_thread(l4_cap_idx_t scheduler,
00210                        l4_cap_idx_t thread, l4_sched_param_t const *sp)
00211     L4_NOTHROW __attribute__((nonnull));
00212
00216 L4_INLINE l4_msgtag_t
00217 l4_scheduler_run_thread_u(l4_cap_idx_t scheduler, l4_cap_idx_t thread,
00218                          l4_sched_param_t const *sp, l4_utcb_t *utcb)
00219     L4_NOTHROW __attribute__((nonnull));
00220
00228 L4_INLINE l4_msgtag_t
00229 l4_scheduler_idle_time(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00230                       l4_kernel_clock_t *us)
00231     L4_NOTHROW __attribute__((nonnull));
00232
00236 L4_INLINE l4_msgtag_t
00237 l4_scheduler_idle_time_u(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00238                         l4_kernel_clock_t *us, l4_utcb_t *utcb)
00239     L4_NOTHROW __attribute__((nonnull));
00240
00241
00242
00253 L4_INLINE int
00254 l4_scheduler_is_online(l4_cap_idx_t scheduler, l4_umword_t cpu) L4_NOTHROW;
00255
00259 L4_INLINE int
00260 l4_scheduler_is_online_u(l4_cap_idx_t scheduler, l4_umword_t cpu,
00261                         l4_utcb_t *utcb) L4_NOTHROW __attribute__((nonnull));
00262
00263
00264
00271 enum l4_scheduler_ops
00272 {
00273     L4_SCHEDULER_INFO_OP = 0UL,
00274     L4_SCHEDULER_RUN_THREAD_OP = 1UL,
00275     L4_SCHEDULER_IDLE_TIME_OP = 2UL,
00276 };
00277
00278 /***** Implementations *****/
00279
00280 L4_INLINE l4_sched_cpu_set_t
00281 l4_sched_cpu_set(l4_umword_t offset, unsigned char granularity,
00282                 l4_umword_t map) L4_NOTHROW
00283 {
00284     l4_sched_cpu_set_t cs;
00285     cs.gran_offset = ((l4_umword_t)granularity << 24) | (offset & 0x00ffffff);
00286     cs.map = map;
00287     return cs;
00288 }
00289
00290 L4_INLINE l4_sched_param_t
00291 l4_sched_param(unsigned prio, l4_umword_t quantum) L4_NOTHROW
00292 {
00293     l4_sched_param_t sp;
00294     sp.prio = prio;
00295     sp.quantum = quantum;
00296     sp.affinity = l4_sched_cpu_set(0, ~0, 1);
00297     return sp;
00298 }

```

```

00299
00300
00301 L4_INLINE l4_msgtag_t
00302 l4_scheduler_info_u(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00303                    l4_sched_cpu_set_t *cpus, l4_umword_t *sched_classes,
00304                    l4_utcb_t *utcb) L4_NOTHROW
00305 {
00306     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00307     l4_msgtag_t res;
00308
00309     m->mr[0] = L4_SCHEDULER_INFO_OP;
00310     m->mr[1] = cpus->gran_offset;
00311
00312     res = l4_ipc_call(scheduler, utcb, l4_msgtag(L4_PROTO_SCHEDULER, 2, 0, 0), L4_IPC_NEVER);
00313
00314     if (l4_msgtag_has_error(res))
00315         return res;
00316
00317     cpus->map = m->mr[0];
00318
00319     if (cpu_max)
00320         *cpu_max = m->mr[1];
00321
00322     if (sched_classes)
00323         *sched_classes = m->mr[2];
00324
00325     return res;
00326 }
00327
00328 L4_INLINE l4_msgtag_t
00329 l4_scheduler_run_thread_u(l4_cap_idx_t scheduler, l4_cap_idx_t thread,
00330                          l4_sched_param_t const *sp, l4_utcb_t *utcb) L4_NOTHROW
00331 {
00332     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00333     m->mr[0] = L4_SCHEDULER_RUN_THREAD_OP;
00334     m->mr[1] = sp->affinity.gran_offset;
00335     m->mr[2] = sp->affinity.map;
00336     m->mr[3] = sp->prio;
00337     m->mr[4] = sp->quantum;
00338     m->mr[5] = l4_map_obj_control(0, 0);
00339     m->mr[6] = l4_obj_fpage(thread, 0, L4_CAP_FPAGE_RWS).raw;
00340
00341     return l4_ipc_call(scheduler, utcb, l4_msgtag(L4_PROTO_SCHEDULER, 5, 1, 0), L4_IPC_NEVER);
00342 }
00343
00344 L4_INLINE l4_msgtag_t
00345 l4_scheduler_idle_time_u(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00346                        l4_kernel_clock_t *us, l4_utcb_t *utcb) L4_NOTHROW
00347 {
00348     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00349     l4_msgtag_t res;
00350
00351     v->mr[0] = L4_SCHEDULER_IDLE_TIME_OP;
00352     v->mr[1] = cpus->gran_offset;
00353     v->mr[2] = cpus->map;
00354
00355     res = l4_ipc_call(scheduler, utcb,
00356                      l4_msgtag(L4_PROTO_SCHEDULER, 3, 0, 0), L4_IPC_NEVER);
00357
00358     if (l4_msgtag_has_error(res))
00359         return res;
00360
00361     *us = v->mr64[l4_utcb_mr64_idx(0)];
00362
00363     return res;
00364 }
00365
00366
00367 L4_INLINE int
00368 l4_scheduler_is_online_u(l4_cap_idx_t scheduler, l4_umword_t cpu,
00369                        l4_utcb_t *utcb) L4_NOTHROW
00370 {
00371     l4_sched_cpu_set_t s;
00372     l4_msgtag_t r;
00373     s.gran_offset = cpu;
00374     r = l4_scheduler_info_u(scheduler, NULL, &s, NULL, utcb);
00375     if (l4_msgtag_has_error(r) || l4_msgtag_label(r) < 0)
00376         return 0;
00377
00378     return s.map & 1;
00379 }
00380
00381
00382 L4_INLINE l4_msgtag_t
00383 l4_scheduler_info(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00384                  l4_sched_cpu_set_t *cpus) L4_NOTHROW
00385 {

```

```

00386     return l4_scheduler_info_u(scheduler, cpu_max, cpus, NULL, l4_utcb());
00387 }
00388
00389 L4_INLINE l4_msgtag_t
00390 l4_scheduler_info_with_classes(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00391                               l4_sched_cpu_set_t *cpus,
00392                               l4_umword_t *sched_classes) L4_NOTHROW
00393 {
00394     return l4_scheduler_info_u(scheduler, cpu_max, cpus, sched_classes, l4_utcb());
00395 }
00396
00397 L4_INLINE l4_msgtag_t
00398 l4_scheduler_run_thread(l4_cap_idx_t scheduler,
00399                        l4_cap_idx_t thread, l4_sched_param_t const *sp) L4_NOTHROW
00400 {
00401     return l4_scheduler_run_thread_u(scheduler, thread, sp, l4_utcb());
00402 }
00403
00404 L4_INLINE l4_msgtag_t
00405 l4_scheduler_idle_time(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00406                       l4_kernel_clock_t *us) L4_NOTHROW
00407 {
00408     return l4_scheduler_idle_time_u(scheduler, cpus, us, l4_utcb());
00409 }
00410
00411 L4_INLINE int
00412 l4_scheduler_is_online(l4_cap_idx_t scheduler, l4_umword_t cpu) L4_NOTHROW
00413 {
00414     return l4_scheduler_is_online_u(scheduler, cpu, l4_utcb());
00415 }

```

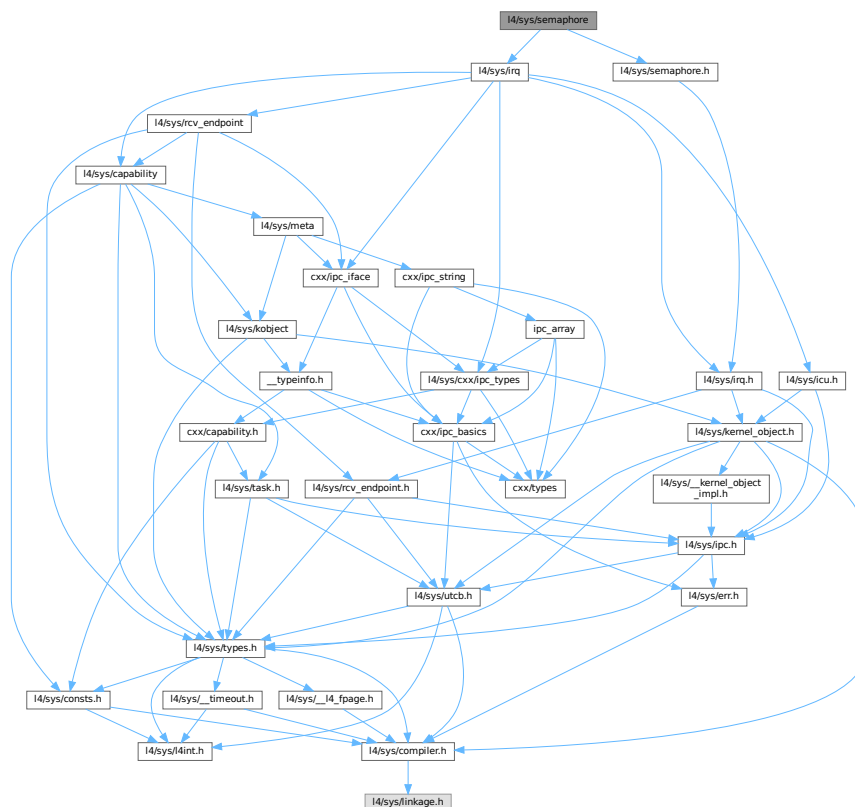
## 16.554 I4/sys/semaphore File Reference

Semaphore class definition.

```
#include <l4/sys/irq>
```

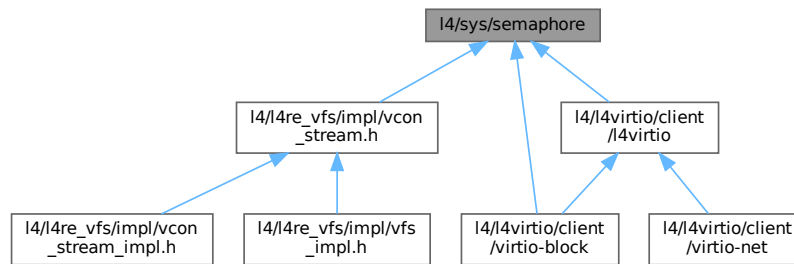
```
#include <l4/sys/semaphore.h>
```

Include dependency graph for semaphore:





This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [L4::Semaphore](#)

*C++ Kernel-provided semaphore interface, see [Kernel-provided semaphore](#) for the C interface.*

## Namespaces

- namespace [L4](#)

[L4](#) low-level kernel interface.

### 16.554.1 Detailed Description

Semaphore class definition.

Definition in file [semaphore](#).

## 16.555 semaphore

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2015 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00025 #include <l4/sys/irq>
00026 #include <l4/sys/semaphore.h>
00027
00028 namespace L4 {

```

```

00029
00054 struct Semaphore : Kobject_t<Semaphore, Triggerable, L4_PROTO_SEMAPHORE>
00055 {
00070     l4_msgtag_t up(l4_utcb_t *utcb = l4_utcb()) noexcept
00071     { return trigger(utcb); }
00072
00090     l4_msgtag_t down(l4_timeout_t timeout = L4_IPC_NEVER,
00091                     l4_utcb_t *utcb = l4_utcb()) noexcept
00092     { return l4_semaphore_down_u(cap(), timeout, utcb); }
00093 };
00094
00095 }

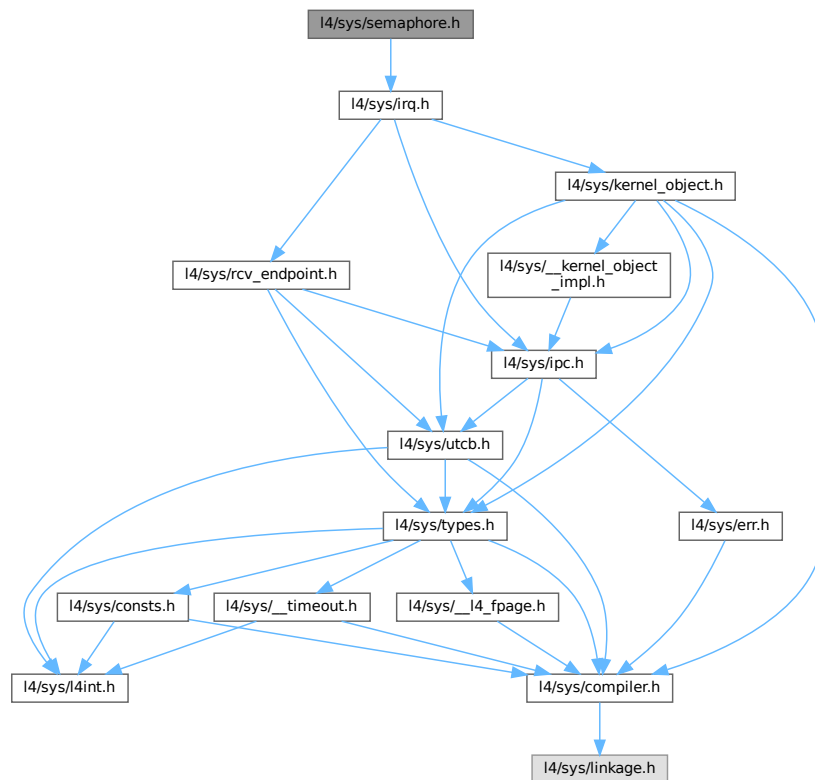
```

## 16.556 l4/sys/semaphore.h File Reference

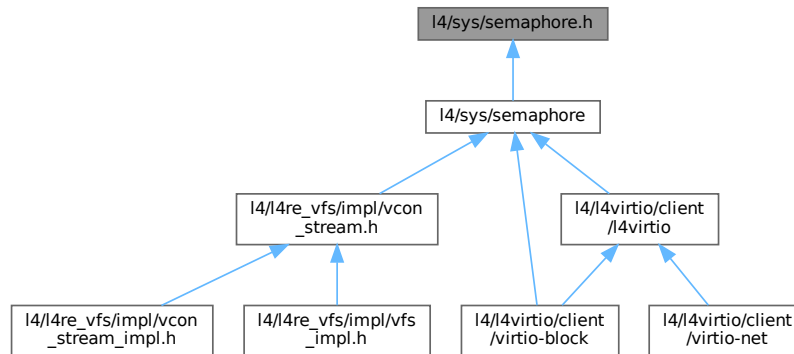
C semaphore interface.

```
#include <l4/sys/irq.h>
```

Include dependency graph for semaphore.h:



This graph shows which files directly or indirectly include this file:



## Functions

- [l4\\_msgtag\\_t l4\\_semaphore\\_up](#) ([l4\\_cap\\_idx\\_t](#) sem) [L4\\_NOTHROW](#)  
Semaphore up operation (wrapper for `trigger()`).
- [l4\\_msgtag\\_t l4\\_semaphore\\_down](#) ([l4\\_cap\\_idx\\_t](#) sem, [l4\\_timeout\\_t](#) timeout) [L4\\_NOTHROW](#)  
Semaphore down operation.

## 16.556.1 Detailed Description

C semaphore interface.

Definition in file [semaphore.h](#).

## 16.557 semaphore.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2015 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00025 #include <l4/sys/irq.h>
00026
00036 enum L4_semaphore_op
00037 {
00038     L4_SEMAPHORE_OP_DOWN    = 0,

```

```

00039 // semaphore up is IRQ_OP_TRIGGER with IRQ/Triggerable protocol
00040 };
00041
00055 L4_INLINE l4_msgtag_t
00056 l4_semaphore_up(l4_cap_idx_t sem) L4_NOTHROW
00057 {
00058     return l4_irq_trigger(sem);
00059 }
00060
00064 L4_INLINE l4_msgtag_t
00065 l4_semaphore_up_u(l4_cap_idx_t sem, l4_utcb_t *utcb) L4_NOTHROW
00066 {
00067     return l4_irq_trigger_u(sem, utcb);
00068 }
00069
00088 L4_INLINE l4_msgtag_t
00089 l4_semaphore_down(l4_cap_idx_t sem, l4_timeout_t timeout) L4_NOTHROW;
00090
00094 L4_INLINE l4_msgtag_t
00095 l4_semaphore_down_u(l4_cap_idx_t sem, l4_timeout_t to,
00096                    l4_utcb_t *utcb) L4_NOTHROW;
00097
00098
00099 L4_INLINE l4_msgtag_t
00100 l4_semaphore_down_u(l4_cap_idx_t sem, l4_timeout_t to,
00101                    l4_utcb_t *utcb) L4_NOTHROW
00102 {
00103     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00104     m->mr[0] = L4_SEMAPHORE_OP_DOWN;
00105     return l4_ipc_call(sem, utcb, l4_msgtag(L4_PROTO_SEMAPHORE, 1, 0, 0), to);
00106 }
00107
00108
00109 L4_INLINE l4_msgtag_t
00110 l4_semaphore_down(l4_cap_idx_t sem, l4_timeout_t to) L4_NOTHROW
00111 {
00112     return l4_semaphore_down_u(sem, to, l4_utcb());
00113 }
00114

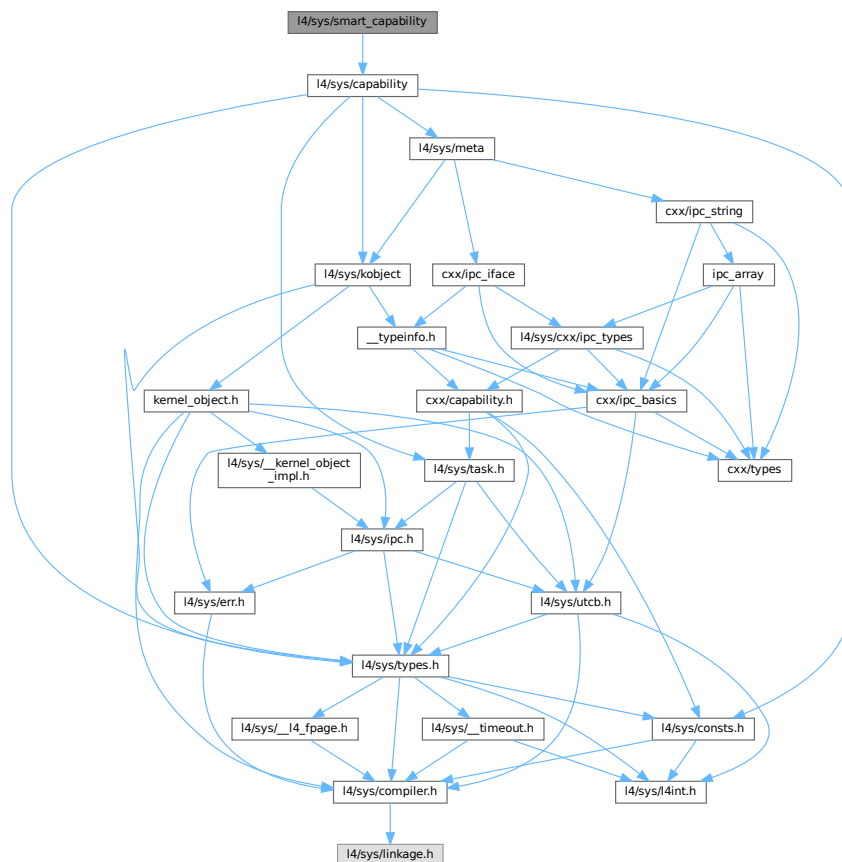
```

## 16.558 l4/sys/smart\_capability File Reference

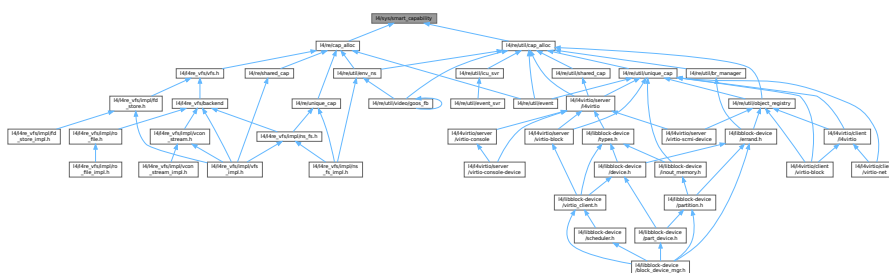
L4::Capability class.

```
#include <linux/sys/capability>
```

Include dependency graph for smart\_capability:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4::Smart_cap< T, SMART >`  
*Smart capability class.*

## Namespaces

- namespace L4  
*L4 low-level kernel interface.*

## Functions

- `template<typename T, typename F, typename SMART >`  
`Smart_cap< T, SMART > L4::cap_cast (Smart_cap< F, SMART > const &c) noexcept`  
*static\_cast for (smart) capabilities.*
- `template<typename T, typename F, typename SMART >`  
`Smart_cap< T, SMART > L4::cap_reinterpret_cast (Smart_cap< F, SMART > const &c) noexcept`  
*reinterpret\_cast for (smart) capabilities.*

### 16.558.1 Detailed Description

L4::Capability class.

#### Author

Alexander Warg [alexander.warg@os.inf.tu-dresden.de](mailto:alexander.warg@os.inf.tu-dresden.de)

Definition in file [smart\\_capability](#).

## 16.559 smart\_capability

[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00009 /*
00010  * (c) 2008-2009 Author(s)
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction. Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License. This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 #pragma once
00027
00028 #include <l4/sys/capability>
00029
00030 namespace L4 {
00031
00032 template< typename T, typename SMART >
00033 class Smart_cap : public Cap_base, private SMART
00034 {
00035 public:
00036     SMART const &smart() const noexcept { return *this; }
00037
00038     void _delete() noexcept
00039     {
00040         SMART::free(const_cast<Smart_cap<T, SMART>&>(*this));
00041     }
00042
00043     Cap<T> release() const noexcept
00044     {
00045         l4_cap_idx_t r = cap();
00046         SMART::invalidate(const_cast<Smart_cap<T, SMART>&>(*this));
00047
00048         return Cap<T>(r);
00049     }
00050
00051     void reset() noexcept
00052     {
```

```

00057     _c = L4_INVALID_CAP;
00058 }
00059
00060 Smart_cap() noexcept : Cap_base(Invalid) {}
00061
00062 Smart_cap(Cap_base::Cap_type t) noexcept : Cap_base(t) {}
00063
00072 template< typename O >
00073 Smart_cap(Cap<O> const &p) noexcept : Cap_base(p.cap())
00074 { Cap<T>::template check_convertible_from<O>(); }
00075
00076 template< typename O >
00077 Smart_cap(Cap<O> const &p, SMART const &smart) noexcept
00078 : Cap_base(p.cap()), SMART(smart)
00079 { Cap<T>::template check_convertible_from<O>(); }
00080
00081 template< typename O >
00082 Smart_cap(Smart_cap<O, SMART> const &o) noexcept
00083 : Cap_base(SMART::copy(o)), SMART(o.smart())
00084 { Cap<T>::template check_convertible_from<O>(); }
00085
00086 Smart_cap(Smart_cap const &o) noexcept
00087 : Cap_base(SMART::copy(o)), SMART(o.smart())
00088 { }
00089
00090 template< typename O >
00091 Smart_cap(typename Cap<O>::Cap_type cap) noexcept : Cap_base(cap)
00092 { Cap<T>::template check_convertible_from<O>(); }
00093
00094 void operator = (typename Cap<T>::Cap_type cap) noexcept
00095 {
00096     _delete();
00097     _c = cap;
00098 }
00099
00100 template< typename O >
00101 void operator = (Smart_cap<O, SMART> const &o) noexcept
00102 {
00103     _delete();
00104     _c = this->SMART::copy(o).cap();
00105     this->SMART::operator = (o.smart());
00106     // return *this;
00107 }
00108
00109 Smart_cap const &operator = (Smart_cap const &o) noexcept
00110 {
00111     if (&o == this)
00112         return *this;
00113
00114     _delete();
00115     _c = this->SMART::copy(o).cap();
00116     this->SMART::operator = (o.smart());
00117     return *this;
00118 }
00119
00120 #if __cplusplus >= 201103L
00121 template< typename O >
00122 Smart_cap(Smart_cap<O, SMART> &&o) noexcept
00123 : Cap_base(o.release()), SMART(o.smart())
00124 { Cap<T>::template check_convertible_from<O>(); }
00125
00126 Smart_cap(Smart_cap &&o) noexcept
00127 : Cap_base(o.release()), SMART(o.smart())
00128 { }
00129
00130 template< typename O >
00131 void operator = (Smart_cap<O, SMART> &&o) noexcept
00132 {
00133     _delete();
00134     _c = o.release().cap();
00135     this->SMART::operator = (o.smart());
00136     // return *this;
00137 }
00138
00139 Smart_cap const &operator = (Smart_cap &&o) noexcept
00140 {
00141     if (&o == this)
00142         return *this;
00143
00144     _delete();
00145     _c = o.release().cap();
00146     this->SMART::operator = (o.smart());
00147     return *this;
00148 }
00149 #endif
00150
00154 Cap<T> operator -> () const noexcept { return Cap<T>(_c); }

```

```

00155
00156     Cap<T> get() const noexcept { return Cap<T>(_c); }
00157
00158     ~Smart_cap() noexcept { _delete(); }
00159 };
00160
00161 template< typename T >
00162 class Weak_cap : public Cap_base
00163 {
00164 public:
00165     Weak_cap() noexcept : Cap_base(Invalid) {}
00166
00167     template< typename O >
00168     Weak_cap(typename Cap<O>::Cap_type t) noexcept : Cap_base(t)
00169     { Cap<T>::template check_convertible_from<O>(); }
00170
00171     template< typename O, typename S >
00172     Weak_cap(Smart_cap<O, S> const &c) noexcept : Cap_base(c.cap())
00173     { Cap<T>::template check_convertible_from<O>(); }
00174
00175     Weak_cap(Weak_cap const &o) noexcept : Cap_base(o) {}
00176
00177     template< typename O >
00178     Weak_cap(Weak_cap<O> const &o) noexcept : Cap_base(o)
00179     { Cap<T>::template check_convertible_from<O>(); }
00180
00181 };
00182
00183 namespace Cap_traits {
00184     template< typename T1, typename T2 >
00185     struct Type { enum { Equal = false }; };
00186
00187     template< typename T1 >
00188     struct Type<T1,T1> { enum { Equal = true }; };
00189 };
00190
00201 template< typename T, typename F, typename SMART >
00202 inline
00203 Smart_cap<T, SMART> cap_cast(Smart_cap<F, SMART> const &c) noexcept
00204 {
00205     Cap<T>::template check_castable_from<F>();
00206     return Smart_cap<T, SMART>(Cap<T>(SMART::copy(c).cap()));
00207 }
00208
00209
00220 template< typename T, typename F, typename SMART >
00221 inline
00222 Smart_cap<T, SMART> cap_reinterpret_cast(Smart_cap<F, SMART> const &c) noexcept
00223 {
00224     return Smart_cap<T, SMART>(Cap<T>(SMART::copy(c).cap()));
00225 }
00226
00227
00228 }

```

## 16.560 l4/sys/task File Reference

Common task related definitions.

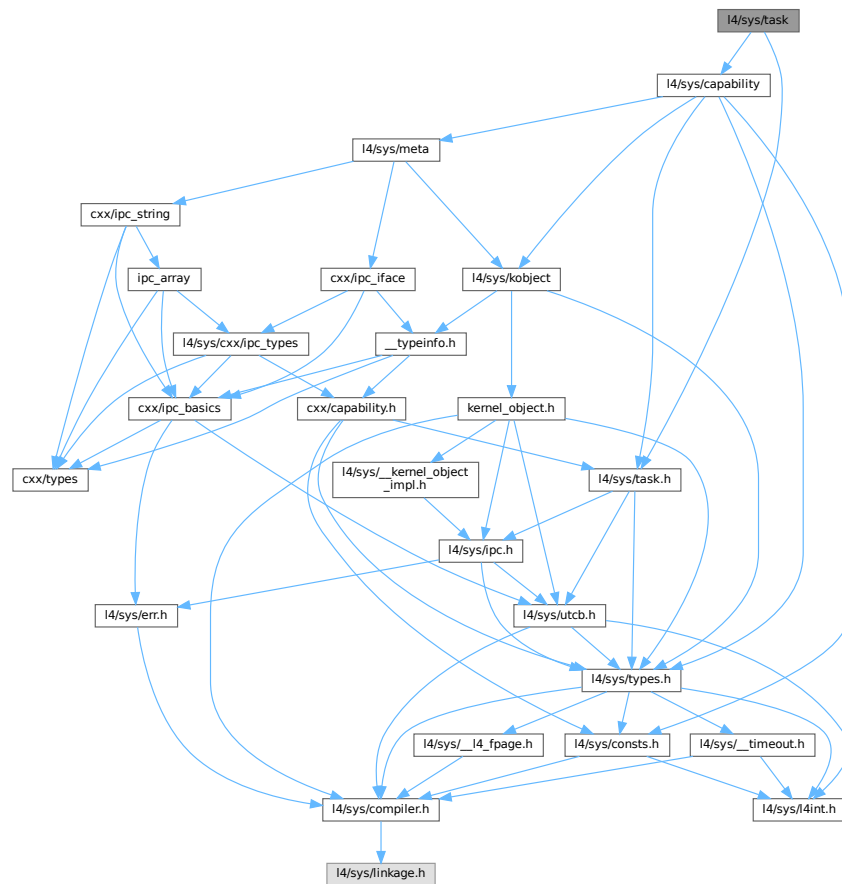
```

#include <l4/sys/task.h>
#include <l4/sys/capability>

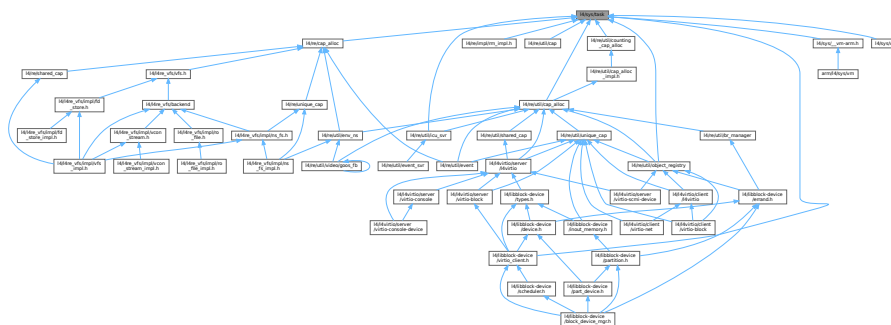
```



Include dependency graph for task:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::Task](#)

*C++ interface of the [Task](#) kernel object, see [Task](#) for the C interface.*

## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

## 16.560.1 Detailed Description

Common task related definitions.

Definition in file [task](#).

## 16.561 task

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include <l4/sys/task.h>
00028 #include <l4/sys/capability>
00029
00030 namespace L4 {
00031
00044 class Task :
00045     public Kobject_t<Task, Kobject, L4_PROTO_TASK,
00046         Type_info::Demand_t<2> >
00047 {
00048 public:
00095     l4_msgtag_t map(Cap<Task> const &src_task,
00096         l4_fpage_t const &snd_fpage, l4_umword_t snd_base,
00097         l4_utcb_t *utcb = l4_utcb()) noexcept
00098     { return l4_task_map_u(cap(), src_task.cap(), snd_fpage, snd_base, utcb); }
00099
00123     l4_msgtag_t unmap(l4_fpage_t const &fpage,
00124         l4_umword_t map_mask,
00125         l4_utcb_t *utcb = l4_utcb()) noexcept
00126     { return l4_task_unmap_u(cap(), fpage, map_mask, utcb); }
00127
00142     l4_msgtag_t unmap_batch(l4_fpage_t const *fpages,
00143         unsigned num_fpages,
00144         l4_umword_t map_mask,
00145         l4_utcb_t *utcb = l4_utcb()) noexcept
00146     { return l4_task_unmap_batch_u(cap(), fpages, num_fpages, map_mask, utcb); }
00147
00168     l4_msgtag_t delete_obj(L4::Cap<void> obj,
00169         l4_utcb_t *utcb = l4_utcb()) noexcept
00170     { return l4_task_delete_obj_u(cap(), obj.cap(), utcb); }
00171
00187     l4_msgtag_t release_cap(L4::Cap<void> cap,
00188         l4_utcb_t *utcb = l4_utcb()) noexcept
00189     { return l4_task_release_cap_u(this->cap(), cap.cap(), utcb); }
00190
00208     l4_msgtag_t cap_valid(Cap<void> const &cap,
00209         l4_utcb_t *utcb = l4_utcb()) noexcept
00210     { return l4_task_cap_valid_u(this->cap(), cap.cap(), utcb); }
00211
```

```

00225  l4_msgtag_t cap_equal(Cap<void> const &cap_a,
00226                        Cap<void> const &cap_b,
00227                        l4_utcb_t *utcb = l4_utcb()) noexcept
00228  { return l4_task_cap_equal_u(cap(), cap_a.cap(), cap_b.cap(), utcb); }
00229
00255  l4_msgtag_t add_ku_mem(l4_fpage_t *fpage,
00256                      l4_utcb_t *utcb = l4_utcb()) noexcept
00257  { return l4_task_add_ku_mem_u(cap(), fpage, utcb); }
00258
00259 };
00260 }
00261
00262

```

## 16.562 task.h

```

00001 /*
00002  * (c) 2018 Adam Lackorzynski <adam@l4re.org>
00003  *
00004  * This file is part of L4Re and distributed under the terms of the
00005  * GNU General Public License 2.
00006  * Please see the COPYING-GPL-2 file for details.
00007  *
00008  * As a special exception, you may use this file as part of a free software
00009  * library without restriction. Specifically, if other files instantiate
00010  * templates or use macros or inline functions from this file, or you compile
00011  * this file and link it with other files to produce an executable, this
00012  * file does not by itself cause the resulting executable to be covered by
00013  * the GNU General Public License. This exception does not however
00014  * invalidate any other reasons why the executable file might be covered by
00015  * the GNU General Public License.
00016  */
00017 #pragma once
00018
00019 #include_next <l4/sys/task.h>
00020 #include <l4/sys/__task-arm.h>

```

## 16.563 l4/sys/task.h File Reference

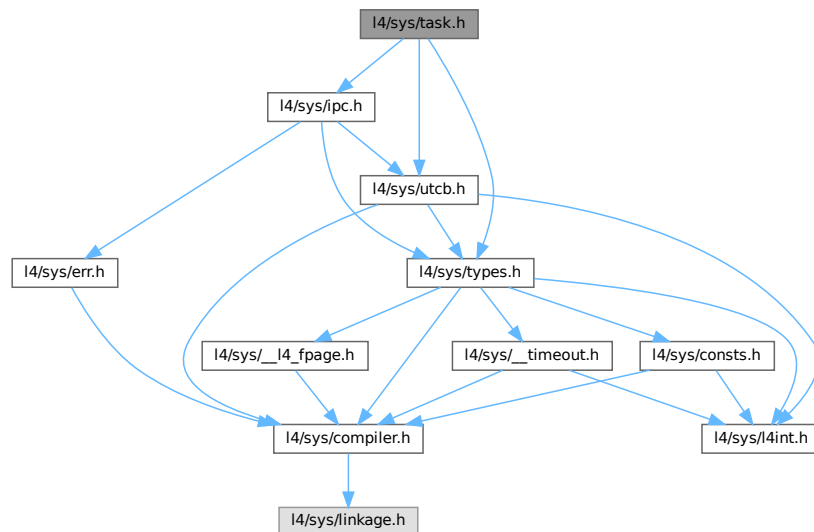
Common task related definitions.

```

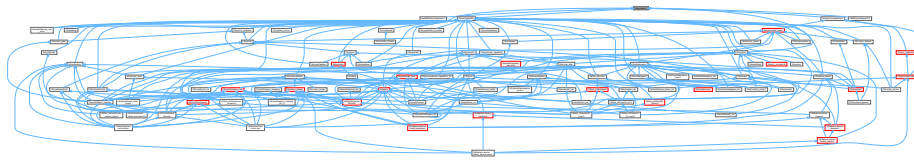
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for task.h:



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum `L4_task_ops` {  
`L4_TASK_MAP_OP` = 0UL , `L4_TASK_UNMAP_OP` = 1UL , `L4_TASK_CAP_INFO_OP` = 2UL ,  
`L4_TASK_ADD_KU_MEM_OP` = 3UL ,  
`L4_TASK_LDT_SET_X86_OP` = 0x11UL , `L4_TASK_MAP_VGICC_ARM_OP` = 0x12UL }

*Operations on task objects.*

## Functions

- `l4_msgtag_t l4_task_map` (`l4_cap_idx_t` dst\_task, `l4_cap_idx_t` src\_task, `l4_fpage_t` snd\_fpage, `l4_umword_t` snd\_base) `L4_NOTHROW`  
*Map resources available in the source task to a destination task.*
- `l4_msgtag_t l4_task_unmap` (`l4_cap_idx_t` task, `l4_fpage_t` fpage, `l4_umword_t` map\_mask) `L4_NOTHROW`  
*Revoke rights from the task.*
- `l4_msgtag_t l4_task_unmap_batch` (`l4_cap_idx_t` task, `l4_fpage_t` const \*fpages, unsigned num\_fpages, `l4_umword_t` map\_mask) `L4_NOTHROW`  
*Revoke rights from a task.*
- `l4_msgtag_t l4_task_delete_obj` (`l4_cap_idx_t` task, `l4_cap_idx_t` obj) `L4_NOTHROW`  
*Release capability and delete object.*
- `l4_msgtag_t l4_task_release_cap` (`l4_cap_idx_t` task, `l4_cap_idx_t` cap) `L4_NOTHROW`  
*Release object capability.*
- `l4_msgtag_t l4_task_cap_valid` (`l4_cap_idx_t` task, `l4_cap_idx_t` cap) `L4_NOTHROW`  
*Check whether a capability is present (refers to an object).*
- `l4_msgtag_t l4_task_cap_equal` (`l4_cap_idx_t` task, `l4_cap_idx_t` cap\_a, `l4_cap_idx_t` cap\_b) `L4_NOTHROW`  
*Test whether two capabilities point to the same object with the same rights.*
- `l4_msgtag_t l4_task_add_ku_mem` (`l4_cap_idx_t` task, `l4_fpage_t` \*ku\_mem) `L4_NOTHROW`  
*Add kernel-user memory.*

## 16.563.1 Detailed Description

Common task related definitions.

Definition in file [task.h](#).

## 16.564 task.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026 #include <l4/sys/types.h>
00027 #include <l4/sys/utcb.h>
00028
00088 L4_INLINE l4_msgtag_t
00089 l4_task_map(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00090            l4_fpage_t snd_fpage, l4_umword_t snd_base) L4_NOTHROW;
00091
00095 L4_INLINE l4_msgtag_t
00096 l4_task_map_u(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00097              l4_fpage_t snd_fpage, l4_umword_t snd_base, l4_utcb_t *utcb) L4_NOTHROW;
00098
00122 L4_INLINE l4_msgtag_t
00123 l4_task_unmap(l4_cap_idx_t task, l4_fpage_t fpage,
00124              l4_umword_t map_mask) L4_NOTHROW;
00125
00129 L4_INLINE l4_msgtag_t
00130 l4_task_unmap_u(l4_cap_idx_t task, l4_fpage_t fpage,
00131                 l4_umword_t map_mask, l4_utcb_t *utcb) L4_NOTHROW;
00132
00152 L4_INLINE l4_msgtag_t
00153 l4_task_unmap_batch(l4_cap_idx_t task, l4_fpage_t const *fpages,
00154                     unsigned num_fpages, l4_umword_t map_mask) L4_NOTHROW;
00155
00159 L4_INLINE l4_msgtag_t
00160 l4_task_unmap_batch_u(l4_cap_idx_t task, l4_fpage_t const *fpages,
00161                       unsigned num_fpages, l4_umword_t map_mask,
00162                       l4_utcb_t *u) L4_NOTHROW;
00163
00185 L4_INLINE l4_msgtag_t
00186 l4_task_delete_obj(l4_cap_idx_t task, l4_cap_idx_t obj) L4_NOTHROW;
00187
00191 L4_INLINE l4_msgtag_t
00192 l4_task_delete_obj_u(l4_cap_idx_t task, l4_cap_idx_t obj,
00193                      l4_utcb_t *u) L4_NOTHROW;
00194
00211 L4_INLINE l4_msgtag_t
00212 l4_task_release_cap(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW;
00213
00217 L4_INLINE l4_msgtag_t
00218 l4_task_release_cap_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00219                       l4_utcb_t *u) L4_NOTHROW;
00220
00221
00239 L4_INLINE l4_msgtag_t
00240 l4_task_cap_valid(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW;
00241
00245 L4_INLINE l4_msgtag_t
00246 l4_task_cap_valid_u(l4_cap_idx_t task, l4_cap_idx_t cap, l4_utcb_t *utcb) L4_NOTHROW;
00247
00260 L4_INLINE l4_msgtag_t
00261 l4_task_cap_equal(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00262                  l4_cap_idx_t cap_b) L4_NOTHROW;
00263
00267 L4_INLINE l4_msgtag_t
00268 l4_task_add_ku_mem_u(l4_cap_idx_t task, l4_fpage_t *ku_mem,
00269                      l4_utcb_t *u) L4_NOTHROW;
00270
00297 L4_INLINE l4_msgtag_t
00298 l4_task_add_ku_mem(l4_cap_idx_t task, l4_fpage_t *ku_mem) L4_NOTHROW;
00299

```

```

00300
00304 L4_INLINE l4_msgtag_t
00305 l4_task_cap_equal_u(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00306                     l4_cap_idx_t cap_b, l4_utcb_t *utcb) L4_NOTHROW;
00307
00312 enum L4_task_ops
00313 {
00314     L4_TASK_MAP_OP          = 0UL,
00315     L4_TASK_UNMAP_OP       = 1UL,
00316     L4_TASK_CAP_INFO_OP    = 2UL,
00317     L4_TASK_ADD_KU_MEM_OP   = 3UL,
00318     L4_TASK_LDT_SET_X86_OP  = 0x11UL,
00319     L4_TASK_MAP_VGICC_ARM_OP = 0x12UL,
00320 };
00321
00322
00323 /* IMPLEMENTATION ----- */
00324
00325 #include <l4/sys/ipc.h>
00326
00327
00328 L4_INLINE l4_msgtag_t
00329 l4_task_map_u(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00330              l4_fpage_t snd_fpage, l4_umword_t snd_base, l4_utcb_t *u) L4_NOTHROW
00331 {
00332     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00333     v->mr[0] = L4_TASK_MAP_OP;
00334     v->mr[3] = l4_map_obj_control(0,0);
00335     v->mr[4] = l4_obj_fpage(src_task, 0, L4_CAP_FPAGE_RWS).raw;
00336     v->mr[1] = snd_base;
00337     v->mr[2] = snd_fpage.raw;
00338     return l4_ipc_call(dst_task, u, l4_msgtag(L4_PROTO_TASK, 3, 1, 0), L4_IPC_NEVER);
00339 }
00340
00341 L4_INLINE l4_msgtag_t
00342 l4_task_unmap_u(l4_cap_idx_t task, l4_fpage_t fpage,
00343                l4_umword_t map_mask, l4_utcb_t *u) L4_NOTHROW
00344 {
00345     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00346     v->mr[0] = L4_TASK_UNMAP_OP;
00347     v->mr[1] = map_mask;
00348     v->mr[2] = fpage.raw;
00349     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 3, 0, 0), L4_IPC_NEVER);
00350 }
00351
00352 L4_INLINE l4_msgtag_t
00353 l4_task_unmap_batch_u(l4_cap_idx_t task, l4_fpage_t const *fpages,
00354                      unsigned num_fpages, l4_umword_t map_mask,
00355                      l4_utcb_t *u) L4_NOTHROW
00356 {
00357     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00358     v->mr[0] = L4_TASK_UNMAP_OP;
00359     v->mr[1] = map_mask;
00360     __builtin_memcpy(&v->mr[2], fpages, num_fpages * sizeof(l4_fpage_t));
00361     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2 + num_fpages, 0, 0), L4_IPC_NEVER);
00362 }
00363
00364 L4_INLINE l4_msgtag_t
00365 l4_task_cap_valid_u(l4_cap_idx_t task, l4_cap_idx_t cap, l4_utcb_t *u) L4_NOTHROW
00366 {
00367     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00368     v->mr[0] = L4_TASK_CAP_INFO_OP;
00369     v->mr[1] = cap & ~1UL;
00370     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0), L4_IPC_NEVER);
00371 }
00372
00373 L4_INLINE l4_msgtag_t
00374 l4_task_cap_equal_u(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00375                    l4_cap_idx_t cap_b, l4_utcb_t *u) L4_NOTHROW
00376 {
00377     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00378     v->mr[0] = L4_TASK_CAP_INFO_OP;
00379     v->mr[1] = cap_a;
00380     v->mr[2] = cap_b;
00381     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 3, 0, 0), L4_IPC_NEVER);
00382 }
00383
00384 L4_INLINE l4_msgtag_t
00385 l4_task_add_ku_mem_u(l4_cap_idx_t task, l4_fpage_t *ku_mem,
00386                    l4_utcb_t *u) L4_NOTHROW
00387 {
00388     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00389     l4_msgtag_t ret;
00390     v->mr[0] = L4_TASK_ADD_KU_MEM_OP;
00391     v->mr[1] = ku_mem->raw;
00392     ret = l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0), L4_IPC_NEVER);
00393     if (!l4_msgtag_has_error(ret))

```

```

00394     {
00395         l4_msgregs_t *v = l4_utcb_mr_u(u);
00396         ku_mem->raw = v->mr[0];
00397     }
00398     return ret;
00399 }
00400
00401
00402
00403 L4_INLINE l4_msgtag_t
00404 l4_task_map(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00405             l4_fpage_t snd_fpage, l4_umword_t snd_base) L4_NOTHROW
00406 {
00407     return l4_task_map_u(dst_task, src_task, snd_fpage, snd_base, l4_utcb());
00408 }
00409
00410 L4_INLINE l4_msgtag_t
00411 l4_task_unmap(l4_cap_idx_t task, l4_fpage_t fpage,
00412              l4_umword_t map_mask) L4_NOTHROW
00413 {
00414     return l4_task_unmap_u(task, fpage, map_mask, l4_utcb());
00415 }
00416
00417 L4_INLINE l4_msgtag_t
00418 l4_task_unmap_batch(l4_cap_idx_t task, l4_fpage_t const *fpages,
00419                    unsigned num_fpages, l4_umword_t map_mask) L4_NOTHROW
00420 {
00421     return l4_task_unmap_batch_u(task, fpages, num_fpages, map_mask,
00422                                  l4_utcb());
00423 }
00424
00425 L4_INLINE l4_msgtag_t
00426 l4_task_delete_obj_u(l4_cap_idx_t task, l4_cap_idx_t obj,
00427                     l4_utcb_t *u) L4_NOTHROW
00428 {
00429     return l4_task_unmap_u(task, l4_obj_fpage(obj, 0, L4_CAP_FPAGE_RWSD),
00430                           L4_FP_DELETE_OBJ, u);
00431 }
00432
00433 L4_INLINE l4_msgtag_t
00434 l4_task_delete_obj(l4_cap_idx_t task, l4_cap_idx_t obj) L4_NOTHROW
00435 {
00436     return l4_task_delete_obj_u(task, obj, l4_utcb());
00437 }
00438
00439
00440 L4_INLINE l4_msgtag_t
00441 l4_task_release_cap_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00442                      l4_utcb_t *u) L4_NOTHROW
00443 {
00444     return l4_task_unmap_u(task, l4_obj_fpage(cap, 0, L4_CAP_FPAGE_RWSD),
00445                           L4_FP_ALL_SPACES, u);
00446 }
00447
00448 L4_INLINE l4_msgtag_t
00449 l4_task_release_cap(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW
00450 {
00451     return l4_task_release_cap_u(task, cap, l4_utcb());
00452 }
00453
00454 L4_INLINE l4_msgtag_t
00455 l4_task_cap_valid(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW
00456 {
00457     return l4_task_cap_valid_u(task, cap, l4_utcb());
00458 }
00459
00460 L4_INLINE l4_msgtag_t
00461 l4_task_cap_equal(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00462                  l4_cap_idx_t cap_b) L4_NOTHROW
00463 {
00464     return l4_task_cap_equal_u(task, cap_a, cap_b, l4_utcb());
00465 }
00466
00467 L4_INLINE l4_msgtag_t
00468 l4_task_add_ku_mem(l4_cap_idx_t task, l4_fpage_t *ku_mem) L4_NOTHROW
00469 {
00470     return l4_task_add_ku_mem_u(task, ku_mem, l4_utcb());
00471 }

```

## 16.565 arm/l4/sys/thread.h File Reference

ARM-specific thread related definitions.

## Enumerations

- enum [L4\\_thread\\_ex\\_regs\\_flags\\_arm](#) { [L4\\_THREAD\\_EX\\_REGS\\_ARM\\_SET\\_EL\\_MASK](#) = 0x3 << 24 , [L4\\_THREAD\\_EX\\_REGS\\_ARM\\_SET\\_EL\\_KEEP](#) = 0x0 << 24 , [L4\\_THREAD\\_EX\\_REGS\\_ARM\\_SET\\_EL\\_EL0](#) = 0x1 << 24 , [L4\\_THREAD\\_EX\\_REGS\\_ARM\\_SET\\_EL\\_EL1](#) = 0x2 << 24 }

*Arm specific [L4::Thread::ex\\_regs\(\)](#) flags.*

## Functions

- [l4\\_msgtag\\_t l4\\_thread\\_arm\\_set\\_tpidruro](#) ([l4\\_cap\\_idx\\_t](#) thread, [l4\\_addr\\_t](#) tpidruro) [L4\\_NOTHROW](#)

*Set the TPIDRURO thread specific register.*

## 16.565.1 Detailed Description

ARM-specific thread related definitions.

Definition in file [thread.h](#).

## 16.566 thread.h

[Go to the documentation of this file.](#)

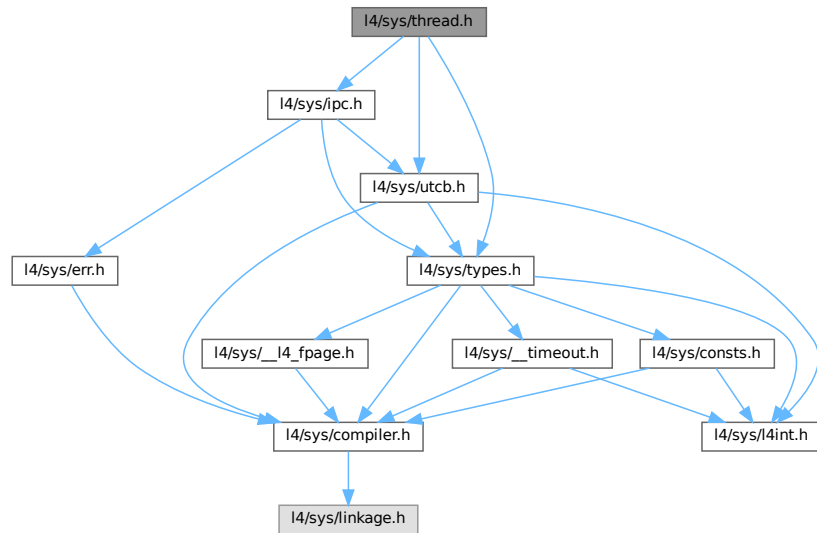
```
00001
00005 /*
00006  * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #pragma once
00023
00024 #include_next <l4/sys/thread.h>
00025
00034 L4_INLINE l4_msgtag_t
00035 l4_thread_arm_set_tpidruro(l4_cap_idx_t thread, l4_addr_t tpidruro) L4_NOTHROW;
00036
00041 L4_INLINE l4_msgtag_t
00042 l4_thread_arm_set_tpidruro_u(l4_cap_idx_t thread, l4_addr_t tpidruro,
00043                             l4_utcb_t *utcb) L4_NOTHROW;
00044
00053 enum L4_thread_ex_regs_flags_arm
00054 {
00056     L4_THREAD_EX_REGS_ARM_SET_EL_MASK      = 0x3 << 24,
00058     L4_THREAD_EX_REGS_ARM_SET_EL_KEEP      = 0x0 << 24,
00060     L4_THREAD_EX_REGS_ARM_SET_EL_EL0       = 0x1 << 24,
00062     L4_THREAD_EX_REGS_ARM_SET_EL_EL1       = 0x2 << 24,
00063 };
00064
00065 /* IMPLEMENTATION ----- */
00066
00067 L4_INLINE l4_msgtag_t
00068 l4_thread_arm_set_tpidruro_u(l4_cap_idx_t thread, l4_addr_t tpidruro,
00069                             l4_utcb_t *utcb) L4_NOTHROW
00070 {
00071     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00072     v->mr[0] = L4_THREAD_ARM_TPIDRURO_OP;
00073     v->mr[1] = tpidruro;
00074     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0),
00075                       L4_IPC_NEVER);
00076 }
00077
00078 L4_INLINE l4_msgtag_t
00079 l4_thread_arm_set_tpidruro(l4_cap_idx_t thread, l4_addr_t tpidruro) L4_NOTHROW
00080 {
00081     return l4_thread_arm_set_tpidruro_u(thread, tpidruro, l4_utcb());
00082 }
```



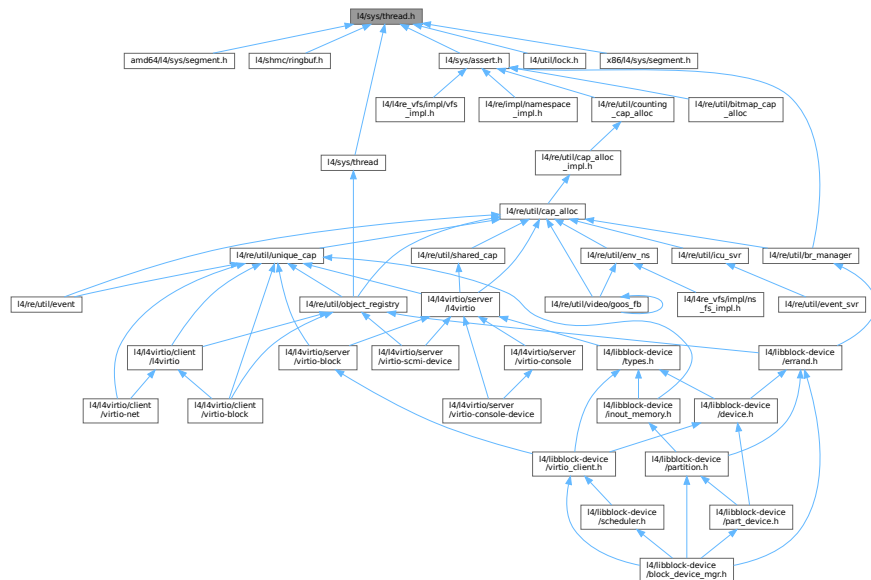
## 16.567 I4/sys/thread.h File Reference

## Common thread related definitions.

```
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
Include dependency graph for thread.h:
```



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum `L4_thread_ops` {  
`L4_THREAD_CONTROL_OP` = 0UL , `L4_THREAD_EX_REGS_OP` = 1UL , `L4_THREAD_SWITCH_OP` = 2UL , `L4_THREAD_STATS_OP` = 3UL ,  
`L4_THREAD_VCPU_RESUME_OP` = 4UL , `L4_THREAD_REGISTER_DELETE_IRQ_OP` = 5UL ,  
`L4_THREAD_MODIFY_SENDER_OP` = 6UL , `L4_THREAD_VCPU_CONTROL_OP` = 7UL ,  
`L4_THREAD_VCPU_CONTROL_EXT_OP` = `L4_THREAD_VCPU_CONTROL_OP` | 0x10000 , `L4_THREAD_X86_GDT_OP` = 0x10UL ,  
`L4_THREAD_ARM_TPIDRURO_OP` = 0x10UL , `L4_THREAD_AMD64_SET_SEGMENT_BASE_OP` = 0x12UL ,  
`L4_THREAD_AMD64_GET_SEGMENT_INFO_OP` = 0x13UL , `L4_THREAD_OPCODE_MASK` = 0xffff }  
*Operations on thread objects.*
- enum `L4_thread_control_flags` {  
`L4_THREAD_CONTROL_SET_PAGER` = 0x0010000 , `L4_THREAD_CONTROL_BIND_TASK` = 0x0200000 ,  
`L4_THREAD_CONTROL_ALIEN` = 0x0400000 , `L4_THREAD_CONTROL_UX_NATIVE` = 0x0800000 ,  
`L4_THREAD_CONTROL_SET_EXC_HANDLER` = 0x1000000 }  
*Flags for the thread control operation.*
- enum `L4_thread_control_mr_indices` {  
`L4_THREAD_CONTROL_MR_IDX_FLAGS` = 0 , `L4_THREAD_CONTROL_MR_IDX_PAGER` = 1 ,  
`L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER` = 2 , `L4_THREAD_CONTROL_MR_IDX_FLAG_VALS` = 4 ,  
`L4_THREAD_CONTROL_MR_IDX_BIND_UTCB` = 5 , `L4_THREAD_CONTROL_MR_IDX_BIND_TASK` = 6  
}
  
*Indices for the values in the message register for thread control.*
- enum `L4_thread_ex_regs_flags` { `L4_THREAD_EX_REGS_CANCEL` = 0x10000UL , `L4_THREAD_EX_REGS_TRIGGER_EXC` = 0x20000UL , `L4_THREAD_EX_REGS_ARCH_MASK` = 0xff000000UL }  
*Flags for the thread ex-regs operation.*

## Functions

- `l4_msgtag_t l4_thread_ex_regs` (`l4_cap_idx_t` thread, `l4_addr_t` ip, `l4_addr_t` sp, `l4_umword_t` flags) `L4_NOTHROW`  
*Exchange basic thread registers.*
- `l4_msgtag_t l4_thread_ex_regs_u` (`l4_cap_idx_t` thread, `l4_addr_t` ip, `l4_addr_t` sp, `l4_umword_t` flags, `l4_utcb_t` \*utcb) `L4_NOTHROW`  
*Exchange basic thread registers.*
- `l4_msgtag_t l4_thread_ex_regs_ret` (`l4_cap_idx_t` thread, `l4_addr_t` \*ip, `l4_addr_t` \*sp, `l4_umword_t` \*flags) `L4_NOTHROW`  
*Exchange basic thread registers and return previous values.*
- `l4_msgtag_t l4_thread_ex_regs_ret_u` (`l4_cap_idx_t` thread, `l4_addr_t` \*ip, `l4_addr_t` \*sp, `l4_umword_t` \*flags, `l4_utcb_t` \*utcb) `L4_NOTHROW`  
*Exchange basic thread registers and return previous values.*
- void `l4_thread_control_start` (void) `L4_NOTHROW`  
*Start a thread control API sequence.*
- void `l4_thread_control_pager` (`l4_cap_idx_t` pager) `L4_NOTHROW`  
*Set the pager.*
- void `l4_thread_control_exc_handler` (`l4_cap_idx_t` exc\_handler) `L4_NOTHROW`  
*Set the exception handler.*
- void `l4_thread_control_bind` (`l4_utcb_t` \*thread\_utcb, `l4_cap_idx_t` task) `L4_NOTHROW`  
*Bind the thread to a task.*
- void `l4_thread_control_alien` (int on) `L4_NOTHROW`  
*Enable alien mode.*
- void `l4_thread_control_ux_host_syscall` (int on) `L4_NOTHROW`

- Enable pass through of native host (Linux) system calls.*
- [l4\\_msgtag\\_t l4\\_thread\\_control\\_commit \(l4\\_cap\\_idx\\_t thread\) L4\\_NOTHROW](#)  
*Commit the thread control parameters.*
- [l4\\_msgtag\\_t l4\\_thread\\_yield \(void\) L4\\_NOTHROW](#)  
*Yield current time slice.*
- [l4\\_msgtag\\_t l4\\_thread\\_switch \(l4\\_cap\\_idx\\_t to\\_thread\) L4\\_NOTHROW](#)  
*Switch to another thread (and donate the remaining time slice).*
- [l4\\_msgtag\\_t l4\\_thread\\_stats\\_time \(l4\\_cap\\_idx\\_t thread, l4\\_kernel\\_clock\\_t \\*us\) L4\\_NOTHROW](#)  
*Get consumed time of thread in  $\mu$ s.*
- [l4\\_msgtag\\_t l4\\_thread\\_vcpu\\_resume\\_start \(void\) L4\\_NOTHROW](#)  
*vCPU return from event handler.*
- [l4\\_msgtag\\_t l4\\_thread\\_vcpu\\_resume\\_commit \(l4\\_cap\\_idx\\_t thread, l4\\_msgtag\\_t tag\) L4\\_NOTHROW](#)  
*Commit vCPU resume.*
- [l4\\_msgtag\\_t l4\\_thread\\_vcpu\\_control \(l4\\_cap\\_idx\\_t thread, l4\\_addr\\_t vcpu\\_state\) L4\\_NOTHROW](#)  
*Enable the vCPU feature for the thread.*
- [l4\\_msgtag\\_t l4\\_thread\\_vcpu\\_control\\_u \(l4\\_cap\\_idx\\_t thread, l4\\_addr\\_t vcpu\\_state, l4\\_utcb\\_t \\*utcb\) L4\\_NOTHROW](#)  
*Enable the vCPU feature for the thread.*
- [l4\\_msgtag\\_t l4\\_thread\\_vcpu\\_control\\_ext \(l4\\_cap\\_idx\\_t thread, l4\\_addr\\_t ext\\_vcpu\\_state\) L4\\_NOTHROW](#)  
*Enable the extended vCPU feature for the thread.*
- [l4\\_msgtag\\_t l4\\_thread\\_vcpu\\_control\\_ext\\_u \(l4\\_cap\\_idx\\_t thread, l4\\_addr\\_t ext\\_vcpu\\_state, l4\\_utcb\\_t \\*utcb\) L4\\_NOTHROW](#)  
*Enable the extended vCPU feature for the thread.*
- [l4\\_msgtag\\_t l4\\_thread\\_register\\_del\\_irq \(l4\\_cap\\_idx\\_t thread, l4\\_cap\\_idx\\_t irq\) L4\\_NOTHROW](#)  
*Register an IRQ that will trigger upon deletion events.*
- [l4\\_msgtag\\_t l4\\_thread\\_modify\\_sender\\_start \(void\) L4\\_NOTHROW](#)  
*Start a thread sender modification sequence.*
- [int l4\\_thread\\_modify\\_sender\\_add \(l4\\_umword\\_t match\\_mask, l4\\_umword\\_t match, l4\\_umword\\_t del\\_bits, l4\\_umword\\_t add\\_bits, l4\\_msgtag\\_t \\*tag\) L4\\_NOTHROW](#)  
*Add a modification pattern to a sender modification sequence.*
- [l4\\_msgtag\\_t l4\\_thread\\_modify\\_sender\\_commit \(l4\\_cap\\_idx\\_t thread, l4\\_msgtag\\_t tag\) L4\\_NOTHROW](#)  
*Apply (commit) a sender modification sequence.*

## 16.567.1 Detailed Description

Common thread related definitions.

Definition in file [thread.h](#).

## 16.568 thread.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.

```

```

00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025  #pragma once
00026
00027  #include <l4/sys/types.h>
00028  #include <l4/sys/utcb.h>
00029  #include <l4/sys/ipc.h>
00030
00090  L4_INLINE l4_msgtag_t
00091  l4_thread_ex_regs(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
00092                  l4_umword_t flags) L4_NOTHROW;
00093
00100  L4_INLINE l4_msgtag_t
00101  l4_thread_ex_regs_u(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
00102                    l4_umword_t flags, l4_utcb_t *utcb) L4_NOTHROW;
00103
00134  L4_INLINE l4_msgtag_t
00135  l4_thread_ex_regs_ret(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
00136                      l4_umword_t *flags) L4_NOTHROW;
00137
00144  L4_INLINE l4_msgtag_t
00145  l4_thread_ex_regs_ret_u(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
00146                        l4_umword_t *flags, l4_utcb_t *utcb) L4_NOTHROW;
00147
00148
00149
00196  L4_INLINE void
00197  l4_thread_control_start(void) L4_NOTHROW;
00198
00203  L4_INLINE void
00204  l4_thread_control_start_u(l4_utcb_t *utcb) L4_NOTHROW;
00205
00215  L4_INLINE void
00216  l4_thread_control_pager(l4_cap_idx_t pager) L4_NOTHROW;
00217
00222  L4_INLINE void
00223  l4_thread_control_pager_u(l4_cap_idx_t pager, l4_utcb_t *utcb) L4_NOTHROW;
00224
00234  L4_INLINE void
00235  l4_thread_control_exc_handler(l4_cap_idx_t exc_handler) L4_NOTHROW;
00236
00241  L4_INLINE void
00242  l4_thread_control_exc_handler_u(l4_cap_idx_t exc_handler,
00243                                l4_utcb_t *utcb) L4_NOTHROW;
00244
00267  L4_INLINE void
00268  l4_thread_control_bind(l4_utcb_t *thread_utcb,
00269                       l4_cap_idx_t task) L4_NOTHROW;
00270
00275  L4_INLINE void
00276  l4_thread_control_bind_u(l4_utcb_t *thread_utcb,
00277                          l4_cap_idx_t task, l4_utcb_t *utcb) L4_NOTHROW;
00278
00302  L4_INLINE void
00303  l4_thread_control_alien(int on) L4_NOTHROW;
00304
00309  L4_INLINE void
00310  l4_thread_control_alien_u(l4_utcb_t *utcb, int on) L4_NOTHROW;
00311
00322  L4_INLINE void
00323  l4_thread_control_ux_host_syscall(int on) L4_NOTHROW;
00324
00329  L4_INLINE void
00330  l4_thread_control_ux_host_syscall_u(l4_utcb_t *utcb, int on) L4_NOTHROW;
00331
00332
00333
00347  L4_INLINE l4_msgtag_t
00348  l4_thread_control_commit(l4_cap_idx_t thread) L4_NOTHROW;
00349
00354  L4_INLINE l4_msgtag_t
00355  l4_thread_control_commit_u(l4_cap_idx_t thread, l4_utcb_t *utcb) L4_NOTHROW;
00356
00363  L4_INLINE l4_msgtag_t
00364  l4_thread_yield(void) L4_NOTHROW;
00365
00374  L4_INLINE l4_msgtag_t
00375  l4_thread_switch(l4_cap_idx_t to_thread) L4_NOTHROW;
00376

```

```

00381 L4_INLINE l4_msgtag_t
00382 l4_thread_switch_u(l4_cap_idx_t to_thread, l4_utcb_t *utcb) L4_NOTHROW;
00383
00384
00385
00395 L4_INLINE l4_msgtag_t
00396 l4_thread_stats_time(l4_cap_idx_t thread, l4_kernel_clock_t *us) L4_NOTHROW;
00397
00402 L4_INLINE l4_msgtag_t
00403 l4_thread_stats_time_u(l4_cap_idx_t thread, l4_kernel_clock_t *us,
00404                        l4_utcb_t *utcb) L4_NOTHROW;
00405
00406
00417 L4_INLINE l4_msgtag_t
00418 l4_thread_vcpu_resume_start(void) L4_NOTHROW;
00419
00424 L4_INLINE l4_msgtag_t
00425 l4_thread_vcpu_resume_start_u(l4_utcb_t *utcb) L4_NOTHROW;
00426
00465 L4_INLINE l4_msgtag_t
00466 l4_thread_vcpu_resume_commit(l4_cap_idx_t thread,
00467                              l4_msgtag_t tag) L4_NOTHROW;
00468
00473 L4_INLINE l4_msgtag_t
00474 l4_thread_vcpu_resume_commit_u(l4_cap_idx_t thread,
00475                                l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;
00476
00477
00497 L4_INLINE l4_msgtag_t
00498 l4_thread_vcpu_control(l4_cap_idx_t thread, l4_addr_t vcpu_state) L4_NOTHROW;
00499
00507 L4_INLINE l4_msgtag_t
00508 l4_thread_vcpu_control_u(l4_cap_idx_t thread, l4_addr_t vcpu_state,
00509                          l4_utcb_t *utcb) L4_NOTHROW;
00510
00542 L4_INLINE l4_msgtag_t
00543 l4_thread_vcpu_control_ext(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state) L4_NOTHROW;
00544
00552 L4_INLINE l4_msgtag_t
00553 l4_thread_vcpu_control_ext_u(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state,
00554                              l4_utcb_t *utcb) L4_NOTHROW;
00555
00556
00578 L4_INLINE l4_msgtag_t
00579 l4_thread_register_del_irq(l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW;
00580
00585 L4_INLINE l4_msgtag_t
00586 l4_thread_register_del_irq_u(l4_cap_idx_t thread, l4_cap_idx_t irq,
00587                              l4_utcb_t *utcb) L4_NOTHROW;
00588
00610 L4_INLINE l4_msgtag_t
00611 l4_thread_modify_sender_start(void) L4_NOTHROW;
00612
00617 L4_INLINE l4_msgtag_t
00618 l4_thread_modify_sender_start_u(l4_utcb_t *u) L4_NOTHROW;
00619
00644 L4_INLINE int
00645 l4_thread_modify_sender_add(l4_umword_t match_mask,
00646                             l4_umword_t match,
00647                             l4_umword_t del_bits,
00648                             l4_umword_t add_bits,
00649                             l4_msgtag_t *tag) L4_NOTHROW;
00650
00655 L4_INLINE int
00656 l4_thread_modify_sender_add_u(l4_umword_t match_mask,
00657                               l4_umword_t match,
00658                               l4_umword_t del_bits,
00659                               l4_umword_t add_bits,
00660                               l4_msgtag_t *tag, l4_utcb_t *u) L4_NOTHROW;
00661
00676 L4_INLINE l4_msgtag_t
00677 l4_thread_modify_sender_commit(l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW;
00678
00683 L4_INLINE l4_msgtag_t
00684 l4_thread_modify_sender_commit_u(l4_cap_idx_t thread, l4_msgtag_t tag,
00685                                  l4_utcb_t *u) L4_NOTHROW;
00686
00693 enum l4_thread_ops
00694 {
00695     L4_THREAD_CONTROL_OP          = 0UL,
00696     L4_THREAD_EX_REGS_OP         = 1UL,
00697     L4_THREAD_SWITCH_OP          = 2UL,
00698     L4_THREAD_STATS_OP           = 3UL,
00699     L4_THREAD_VCPU_RESUME_OP      = 4UL,
00700     L4_THREAD_REGISTER_DELETE_IRQ_OP = 5UL,
00701     L4_THREAD_MODIFY_SENDER_OP    = 6UL,
00702     L4_THREAD_VCPU_CONTROL_OP     = 7UL,

```

```

00703 L4_THREAD_VCPU_CONTROL_EXT_OP      = L4_THREAD_VCPU_CONTROL_OP | 0x10000,
00704 L4_THREAD_X86_GDT_OP                = 0x10UL,
00705 L4_THREAD_ARM_TPIDRURO_OP           = 0x10UL,
00706 L4_THREAD_AMD64_SET_SEGMENT_BASE_OP = 0x12UL,
00707 L4_THREAD_AMD64_GET_SEGMENT_INFO_OP = 0x13UL,
00708 L4_THREAD_OPCODE_MASK               = 0xffff,
00709 };
00710
00721 enum L4_thread_control_flags
00722 {
00724 L4_THREAD_CONTROL_SET_PAGER          = 0x0010000,
00726 L4_THREAD_CONTROL_BIND_TASK         = 0x0200000,
00728 L4_THREAD_CONTROL_ALIEN             = 0x0400000,
00730 L4_THREAD_CONTROL_UX_NATIVE         = 0x0800000,
00732 L4_THREAD_CONTROL_SET_EXC_HANDLER   = 0x1000000,
00733 };
00734
00744 enum L4_thread_control_mr_indices
00745 {
00746 L4_THREAD_CONTROL_MR_IDX_FLAGS      = 0,
00747 L4_THREAD_CONTROL_MR_IDX_PAGER      = 1,
00748 L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER = 2,
00749 L4_THREAD_CONTROL_MR_IDX_FLAG_VALS  = 4,
00750 L4_THREAD_CONTROL_MR_IDX_BIND_UTCB  = 5,
00751 L4_THREAD_CONTROL_MR_IDX_BIND_TASK  = 6,
00752 };
00753
00759 enum L4_thread_ex_regs_flags
00760 {
00761 L4_THREAD_EX_REGS_CANCEL             = 0x10000UL,
00762 L4_THREAD_EX_REGS_TRIGGER_EXCEPTION = 0x20000UL,
00764 L4_THREAD_EX_REGS_ARCH_MASK         = 0xffff00000UL,
00765 };
00766
00767
00768 /* IMPLEMENTATION ----- */
00769
00770 #include <l4/sys/ipc.h>
00771 #include <l4/sys/types.h>
00772
00773 L4_INLINE l4_msgtag_t
00774 l4_thread_ex_regs_u(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
00775                    l4_umword_t flags, l4_utcb_t *utcb) L4_NOTHROW
00776 {
00777     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00778     v->mr[0] = L4_THREAD_EX_REGS_OP | flags;
00779     v->mr[1] = ip;
00780     v->mr[2] = sp;
00781     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 3, 0, 0), L4_IPC_NEVER);
00782 }
00783
00784 L4_INLINE l4_msgtag_t
00785 l4_thread_ex_regs_ret_u(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
00786                        l4_umword_t *flags, l4_utcb_t *utcb) L4_NOTHROW
00787 {
00788     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00789     l4_msgtag_t ret = l4_thread_ex_regs_u(thread, *ip, *sp, *flags, utcb);
00790     if (l4_error_u(ret, utcb))
00791         return ret;
00792
00793     *flags = v->mr[0];
00794     *ip    = v->mr[1];
00795     *sp    = v->mr[2];
00796     return ret;
00797 }
00798
00799 L4_INLINE void
00800 l4_thread_control_start_u(l4_utcb_t *utcb) L4_NOTHROW
00801 {
00802     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00803     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] = L4_THREAD_CONTROL_OP;
00804 }
00805
00806 L4_INLINE void
00807 l4_thread_control_pager_u(l4_cap_idx_t pager, l4_utcb_t *utcb) L4_NOTHROW
00808 {
00809     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00810     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_SET_PAGER;
00811     v->mr[L4_THREAD_CONTROL_MR_IDX_PAGER] = pager;
00812 }
00813
00814 L4_INLINE void
00815 l4_thread_control_exc_handler_u(l4_cap_idx_t exc_handler,
00816                                l4_utcb_t *utcb) L4_NOTHROW
00817 {
00818     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00819     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_SET_EXC_HANDLER;

```

```

00820     v->mr[L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER] = exc_handler;
00821 }
00822
00823 L4_INLINE void
00824 l4_thread_control_bind_u(l4_utcb_t *thread_utcb, l4_cap_idx_t task,
00825                        l4_utcb_t *utcb) L4_NOTHROW
00826 {
00827     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00828     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_BIND_TASK;
00829     v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_UTCB] = (l4_addr_t)thread_utcb;
00830     v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_TASK] = L4_ITEM_MAP;
00831     v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_TASK + 1] = l4_obj_fpage(task, 0, L4_CAP_FPAGE_RWS).raw;
00832 }
00833
00834 L4_INLINE void
00835 l4_thread_control_alien_u(l4_utcb_t *utcb, int on) L4_NOTHROW
00836 {
00837     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00838     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_ALIEN;
00839     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAG_VALS] |= on ? L4_THREAD_CONTROL_ALIEN : 0;
00840 }
00841
00842 L4_INLINE void
00843 l4_thread_control_ux_host_syscall_u(l4_utcb_t *utcb, int on) L4_NOTHROW
00844 {
00845     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00846     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_UX_NATIVE;
00847     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAG_VALS] |= on ? L4_THREAD_CONTROL_UX_NATIVE : 0;
00848 }
00849
00850 L4_INLINE l4_msgtag_t
00851 l4_thread_control_commit_u(l4_cap_idx_t thread, l4_utcb_t *utcb) L4_NOTHROW
00852 {
00853     int items = 0;
00854     if (l4_utcb_mr_u(utcb)->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] & L4_THREAD_CONTROL_BIND_TASK)
00855         items = 1;
00856     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 6, items, 0), L4_IPC_NEVER);
00857 }
00858
00859
00860 L4_INLINE l4_msgtag_t
00861 l4_thread_yield(void) L4_NOTHROW
00862 {
00863     l4_ipc_receive(L4_INVALID_CAP, NULL, L4_IPC_BOTH_TIMEOUT_0);
00864     return l4_msgtag(0, 0, 0, 0);
00865 }
00866
00867 /* Preliminary, to be changed */
00868 L4_INLINE l4_msgtag_t
00869 l4_thread_switch_u(l4_cap_idx_t to_thread, l4_utcb_t *utcb) L4_NOTHROW
00870 {
00871     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00872     v->mr[0] = L4_THREAD_SWITCH_OP;
00873     return l4_ipc_call(to_thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER);
00874 }
00875
00876
00877 L4_INLINE l4_msgtag_t
00878 l4_thread_stats_time_u(l4_cap_idx_t thread, l4_kernel_clock_t *us,
00879                      l4_utcb_t *utcb) L4_NOTHROW
00880 {
00881     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00882     l4_msgtag_t res;
00883
00884     v->mr[0] = L4_THREAD_STATS_OP;
00885
00886     res = l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER);
00887
00888     if (l4_msgtag_has_error(res))
00889         return res;
00890
00891     *us = v->mr64[l4_utcb_mr64_idx(0)];
00892
00893     return res;
00894 }
00895
00896 L4_INLINE l4_msgtag_t
00897 l4_thread_vcpu_resume_start_u(l4_utcb_t *utcb) L4_NOTHROW
00898 {
00899     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00900     v->mr[0] = L4_THREAD_VCPU_RESUME_OP;
00901     return l4_msgtag(L4_PROTO_THREAD, 1, 0, 0);
00902 }
00903
00904 L4_INLINE l4_msgtag_t
00905 l4_thread_vcpu_resume_commit_u(l4_cap_idx_t thread,
00906                               l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW

```

```

00907 {
00908     return l4_ipc_call(thread, utcb, tag, L4_IPC_NEVER);
00909 }
00910
00911 L4_INLINE l4_msgtag_t
00912 l4_thread_ex_regs(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
00913                  l4_umword_t flags) L4_NOTHROW
00914 {
00915     return l4_thread_ex_regs_u(thread, ip, sp, flags, l4_utcb());
00916 }
00917
00918 L4_INLINE l4_msgtag_t
00919 l4_thread_ex_regs_ret(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
00920                      l4_umword_t *flags) L4_NOTHROW
00921 {
00922     return l4_thread_ex_regs_ret_u(thread, ip, sp, flags, l4_utcb());
00923 }
00924
00925 L4_INLINE void
00926 l4_thread_control_start(void) L4_NOTHROW
00927 {
00928     l4_thread_control_start_u(l4_utcb());
00929 }
00930
00931 L4_INLINE void
00932 l4_thread_control_pager(l4_cap_idx_t pager) L4_NOTHROW
00933 {
00934     l4_thread_control_pager_u(pager, l4_utcb());
00935 }
00936
00937 L4_INLINE void
00938 l4_thread_control_exc_handler(l4_cap_idx_t exc_handler) L4_NOTHROW
00939 {
00940     l4_thread_control_exc_handler_u(exc_handler, l4_utcb());
00941 }
00942
00943
00944 L4_INLINE void
00945 l4_thread_control_bind(l4_utcb_t *thread_utcb, l4_cap_idx_t task) L4_NOTHROW
00946 {
00947     l4_thread_control_bind_u(thread_utcb, task, l4_utcb());
00948 }
00949
00950 L4_INLINE void
00951 l4_thread_control_alien(int on) L4_NOTHROW
00952 {
00953     l4_thread_control_alien_u(l4_utcb(), on);
00954 }
00955
00956 L4_INLINE void
00957 l4_thread_control_ux_host_syscall(int on) L4_NOTHROW
00958 {
00959     l4_thread_control_ux_host_syscall_u(l4_utcb(), on);
00960 }
00961
00962 L4_INLINE l4_msgtag_t
00963 l4_thread_control_commit(l4_cap_idx_t thread) L4_NOTHROW
00964 {
00965     return l4_thread_control_commit_u(thread, l4_utcb());
00966 }
00967
00968
00969
00970
00971 L4_INLINE l4_msgtag_t
00972 l4_thread_switch(l4_cap_idx_t to_thread) L4_NOTHROW
00973 {
00974     return l4_thread_switch_u(to_thread, l4_utcb());
00975 }
00976
00977
00978
00979
00980 L4_INLINE l4_msgtag_t
00981 l4_thread_stats_time(l4_cap_idx_t thread, l4_kernel_clock_t *us) L4_NOTHROW
00982 {
00983     return l4_thread_stats_time_u(thread, us, l4_utcb());
00984 }
00985
00986 L4_INLINE l4_msgtag_t
00987 l4_thread_vcpu_resume_start(void) L4_NOTHROW
00988 {
00989     return l4_thread_vcpu_resume_start_u(l4_utcb());
00990 }
00991
00992 L4_INLINE l4_msgtag_t
00993 l4_thread_vcpu_resume_commit(l4_cap_idx_t thread,

```



```

00994         l4_msgtag_t tag) L4_NOTHROW
00995 {
00996     return l4_thread_vcpu_resume_commit_u(thread, tag, l4_utcb());
00997 }
00998
00999
01000 L4_INLINE l4_msgtag_t
01001 l4_thread_register_del_irq_u(l4_cap_idx_t thread, l4_cap_idx_t irq,
01002                             l4_utcb_t *u) L4_NOTHROW
01003 {
01004     l4_msg_regs_t *m = l4_utcb_mr_u(u);
01005     m->mr[0] = L4_THREAD_REGISTER_DELETE_IRQ_OP;
01006     m->mr[1] = l4_map_obj_control(0, 0);
01007     m->mr[2] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).raw;
01008     return l4_ipc_call(thread, u, l4_msgtag(L4_PROTO_THREAD, 1, 1, 0), L4_IPC_NEVER);
01009 }
01010
01011
01012 L4_INLINE l4_msgtag_t
01013 l4_thread_register_del_irq(l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW
01014 {
01015     return l4_thread_register_del_irq_u(thread, irq, l4_utcb());
01016 }
01017
01018
01019 L4_INLINE l4_msgtag_t
01020 l4_thread_vcpu_control_u(l4_cap_idx_t thread, l4_addr_t vcpu_state,
01021                         l4_utcb_t *utcb) L4_NOTHROW
01022 {
01023     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
01024     v->mr[0] = L4_THREAD_VCPU_CONTROL_OP;
01025     v->mr[1] = vcpu_state;
01026     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER);
01027 }
01028
01029 L4_INLINE l4_msgtag_t
01030 l4_thread_vcpu_control(l4_cap_idx_t thread, l4_addr_t vcpu_state) L4_NOTHROW
01031 { return l4_thread_vcpu_control_u(thread, vcpu_state, l4_utcb()); }
01032
01033
01034 L4_INLINE l4_msgtag_t
01035 l4_thread_vcpu_control_ext_u(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state,
01036                             l4_utcb_t *utcb) L4_NOTHROW
01037 {
01038     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
01039     v->mr[0] = L4_THREAD_VCPU_CONTROL_EXT_OP;
01040     v->mr[1] = ext_vcpu_state;
01041     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER);
01042 }
01043
01044 L4_INLINE l4_msgtag_t
01045 l4_thread_vcpu_control_ext(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state) L4_NOTHROW
01046 { return l4_thread_vcpu_control_ext_u(thread, ext_vcpu_state, l4_utcb()); }
01047
01048 L4_INLINE l4_msgtag_t
01049 l4_thread_modify_sender_start_u(l4_utcb_t *u) L4_NOTHROW
01050 {
01051     l4_msg_regs_t *m = l4_utcb_mr_u(u);
01052     m->mr[0] = L4_THREAD_MODIFY_SENDER_OP;
01053     return l4_msgtag(L4_PROTO_THREAD, 1, 0, 0);
01054 }
01055
01056 L4_INLINE int
01057 l4_thread_modify_sender_add_u(l4_umword_t match_mask,
01058                              l4_umword_t match,
01059                              l4_umword_t del_bits,
01060                              l4_umword_t add_bits,
01061                              l4_msgtag_t *tag, l4_utcb_t *u) L4_NOTHROW
01062 {
01063     l4_msg_regs_t *m = l4_utcb_mr_u(u);
01064     unsigned w = l4_msgtag_words(*tag);
01065     if (w >= L4_UTCB_GENERIC_DATA_SIZE - 4)
01066         return -L4_ENOMEM;
01067
01068     m->mr[w] = match_mask;
01069     m->mr[w+1] = match;
01070     m->mr[w+2] = del_bits;
01071     m->mr[w+3] = add_bits;
01072
01073     *tag = l4_msgtag(l4_msgtag_label(*tag), w + 4, 0, 0);
01074
01075     return 0;
01076 }
01077
01078 L4_INLINE l4_msgtag_t
01079 l4_thread_modify_sender_commit_u(l4_cap_idx_t thread, l4_msgtag_t tag,
01080                                 l4_utcb_t *u) L4_NOTHROW

```

```

01081 {
01082     return l4_ipc_call(thread, u, tag, L4_IPC_NEVER);
01083 }
01084
01085 L4_INLINE l4_msgtag_t
01086 l4_thread_modify_sender_start(void) L4_NOTHROW
01087 {
01088     return l4_thread_modify_sender_start_u(l4_utcb());
01089 }
01090
01091 L4_INLINE int
01092 l4_thread_modify_sender_add(l4_umword_t match_mask,
01093                             l4_umword_t match,
01094                             l4_umword_t del_bits,
01095                             l4_umword_t add_bits,
01096                             l4_msgtag_t *tag) L4_NOTHROW
01097 {
01098     return l4_thread_modify_sender_add_u(match_mask, match,
01099                                           del_bits, add_bits, tag, l4_utcb());
01100 }
01101
01102 L4_INLINE l4_msgtag_t
01103 l4_thread_modify_sender_commit(l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW
01104 {
01105     return l4_thread_modify_sender_commit_u(thread, tag, l4_utcb());
01106 }

```

## 16.569 I4/util/thread.h File Reference

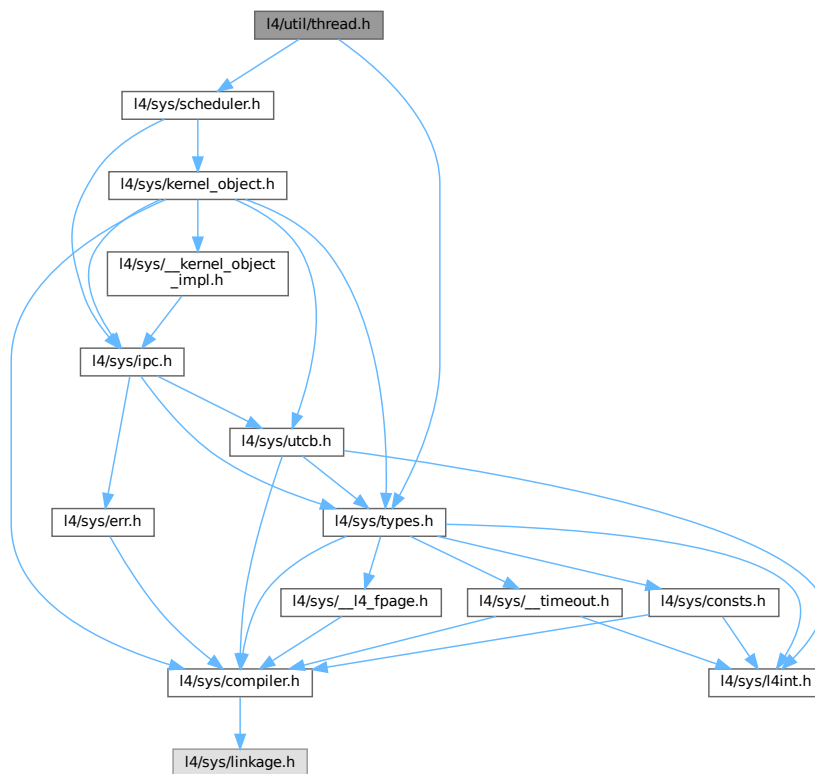
Low-level Thread Functions.

```

#include <l4/sys/types.h>
#include <l4/sys/scheduler.h>

```

Include dependency graph for thread.h:



## Macros

- `#define __L4UTIL_THREAD_FUNC(name) void L4_NORETURN name(void)`  
*Defines a wrapper function that sets up the registers according to the calling conventions for the architecture.*

## 16.569.1 Detailed Description

Low-level Thread Functions.

### Date

1997

### Author

Sebastian Schönberg

Definition in file [thread.h](#).

## 16.569.2 Macro Definition Documentation

### 16.569.2.1 \_\_L4UTIL\_THREAD\_FUNC

```
#define __L4UTIL_THREAD_FUNC(  
    name ) void L4_NORETURN name(void)
```

Defines a wrapper function that sets up the registers according to the calling conventions for the architecture.

Use this as a function header when starting a low-level thread where only stack and instruction pointer are in a well-defined state.

Example:

```
L4UTIL_THREAD_FUNC(helper_thread) { l4_infinite_loop(); }
```

```
thread_cap->ex_regs((l4_umword_t)helper_thread, stack_addr);
```

Definition at line 72 of file [thread.h](#).

## 16.570 thread.h

[Go to the documentation of this file.](#)

```

00001
00008 /*
00009  * (c) 2003-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
00014  */
00015
00016 #ifndef __L4_THREAD_H
00017 #define __L4_THREAD_H
00018
00019 #include <l4/sys/types.h>
00020 #include <l4/sys/scheduler.h>
00021
00022 EXTERN_C_BEGIN
00023
00046 L4_CV long
00047 l4util_create_thread(l4_cap_idx_t id, l4_utcb_t *thread_utcb,
00048                     l4_cap_idx_t factory,
00049                     l4_umword_t pc, l4_umword_t sp, l4_cap_idx_t pager,
00050                     l4_cap_idx_t task,
00051                     l4_cap_idx_t scheduler, l4_sched_param_t scp) L4_NOTHROW;
00052
00053 EXTERN_C_END
00054
00055 #ifndef L4UTIL_THREAD_FUNC
00072 #define __L4UTIL_THREAD_FUNC(name) void L4_NORETURN name(void)
00073 #define L4UTIL_THREAD_FUNC(name) __L4UTIL_THREAD_FUNC(name)
00074 #define __L4UTIL_THREAD_STATIC_FUNC(name) static L4_NORETURN void name(void)
00075 #define L4UTIL_THREAD_STATIC_FUNC(name) __L4UTIL_THREAD_STATIC_FUNC(name)
00076 #endif
00077
00078 #endif /* __L4_THREAD_H */

```

## 16.571 l4/sys/typeinfo\_svr File Reference

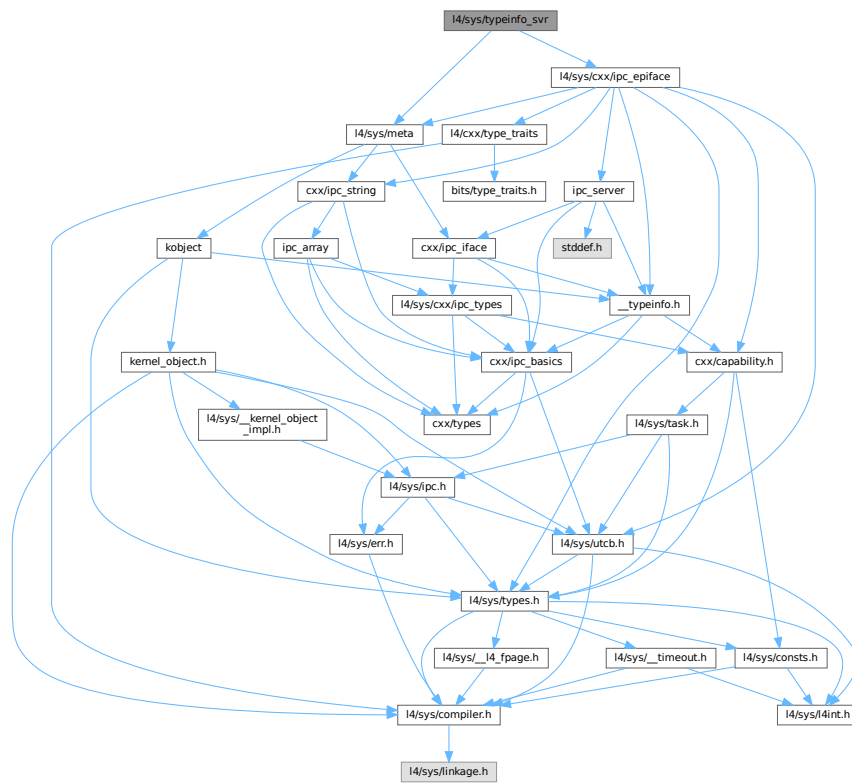
Type information server template.

```

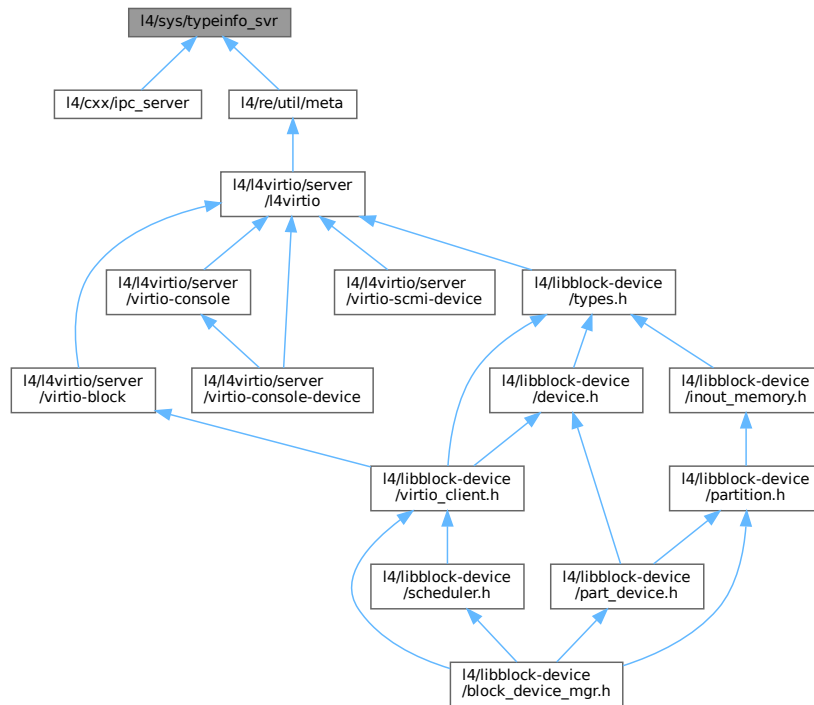
#include <l4/sys/meta>
#include <l4/sys/cxx/ipc_epiface>

```

Include dependency graph for typeidinfo\_svr:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

## 16.571.1 Detailed Description

Type information server template.

Definition in file [typeinfo\\_svr](#).

## 16.572 typeinfo\_svr

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=c++: -- Mode: C++ --
00002 /*
00003  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
  
```

```

00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023
00024 #pragma once
00025
00026 #include <l4/sys/meta>
00027 #include <l4/sys/cxx/ipc_epiface>
00028
00029 namespace L4 { namespace Util {
00030
00031 template<typename KO, typename IOS>
00032 long handle_meta_request(IOS &ios)
00033 {
00034     using L4::Ipc::Msg::dispatch_call;
00035     typedef L4::Ipc::Detail::Meta_svr<KO> Msvr;
00036     typedef L4::Meta::Rpc<KO> Rpc;
00037     Msvr *svr = nullptr;
00038     l4_msgtag_t tag = dispatch_call<Rpc>(svr, ios.utcb(), ios.tag(), 0);
00039     ios.set_ipc_params(tag);
00040     return tag.label();
00041 }
00042
00043 }}

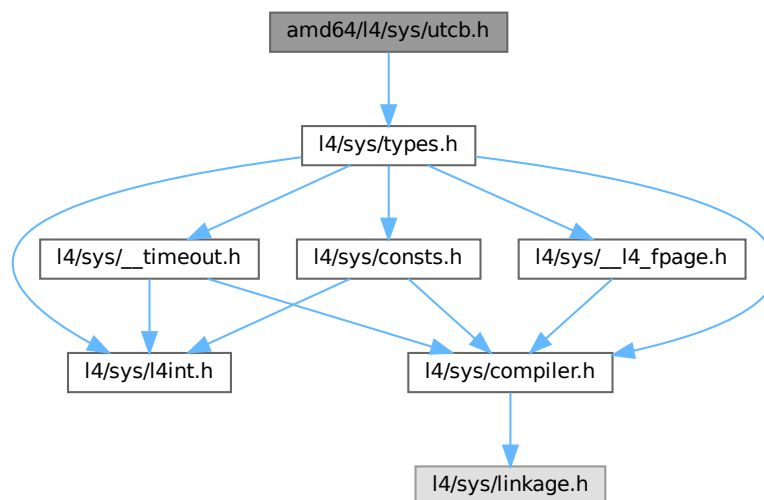
```

## 16.573 amd64/l4/sys/utcb.h File Reference

UTCB definitions for amd64.

```
#include <l4/sys/types.h>
```

Include dependency graph for utcb.h:



### Data Structures

- struct `l4_exc_regs_t`  
UTCB structure for exceptions.

## Typedefs

- typedef struct [l4\\_exc\\_regs\\_t](#) [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

## Enumerations

- enum [L4\\_utcb\\_consts\\_amd64](#)  
*UTCB constants for AMD64.*

## Functions

- [l4\\_umword\\_t](#) [l4\\_utcb\\_exc\\_pc](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)  
*Access function to get the program counter of the exception state.*
- void [l4\\_utcb\\_exc\\_pc\\_set](#) ([l4\\_exc\\_regs\\_t](#) \*u, [l4\\_addr\\_t](#) pc) [L4\\_NOTHROW](#)  
*Set the program counter register in the exception state.*
- [l4\\_umword\\_t](#) [l4\\_utcb\\_exc\\_typeval](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)  
*Get the value out of an exception UTCB that describes the type of exception.*
- int [l4\\_utcb\\_exc\\_is\\_pf](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)  
*Check whether an exception IPC is a page fault.*
- [l4\\_addr\\_t](#) [l4\\_utcb\\_exc\\_pfa](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)  
*Function to get the L4 style page fault address out of an exception.*
- int [l4\\_utcb\\_exc\\_is\\_ex\\_regs\\_exception](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)  
*Check whether an exception IPC was triggered via [l4\\_thread\\_ex\\_regs\(\)](#).*

## 16.573.1 Detailed Description

UTCB definitions for amd64.

Definition in file [utcb.h](#).

## 16.574 utcb.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 /*****
00025 #ifndef __L4_SYS__INCLUDE__ARCH_AMD64__UTCB_H__
00026 #define __L4_SYS__INCLUDE__ARCH_AMD64__UTCB_H__
00027
```



```

00028 #include <l4/sys/types.h>
00029
00039 enum L4_utcb_consts_amd64
00040 {
00041     L4_UTCB_EXCEPTION_REGS_SIZE    = 26,
00042     L4_UTCB_GENERIC_DATA_SIZE      = 63,
00043     L4_UTCB_GENERIC_BUFFERS_SIZE   = 58,
00044
00045     L4_UTCB_MSG_REGS_OFFSET        = 0,
00046     L4_UTCB_BUF_REGS_OFFSET        = 64 * sizeof(l4_umword_t),
00047     L4_UTCB_THREAD_REGS_OFFSET     = 123 * sizeof(l4_umword_t),
00048
00049     L4_UTCB_INHERIT_FPU             = 1UL « 24,
00050     L4_UTCB_OFFSET                  = 1024,
00051 };
00052
00057 typedef struct l4_exc_regs_t
00058 {
00059     l4_umword_t r15;
00060     l4_umword_t r14;
00061     l4_umword_t r13;
00062     l4_umword_t r12;
00063     l4_umword_t r11;
00064     l4_umword_t r10;
00065     l4_umword_t r9;
00066     l4_umword_t r8;
00067     l4_umword_t rdi;
00068     l4_umword_t rsi;
00069     l4_umword_t rbp;
00070     l4_umword_t pfa;
00071     l4_umword_t rbx;
00072     l4_umword_t rdx;
00073     l4_umword_t rcx;
00074     l4_umword_t rax;
00076     l4_umword_t trapno;
00077     l4_umword_t err;
00078     l4_umword_t ip;
00079     l4_umword_t dummy1;
00080     l4_umword_t flags;
00081     l4_umword_t sp;
00082     l4_umword_t ss;
00083     l4_umword_t fs_base;
00084     l4_umword_t gs_base;
00085     l4_uint16_t ds, es, fs, gs;
00086 } l4_exc_regs_t;
00087
00088
00089 #include_next <l4/sys/utcb.h>
00090
00091 /*
00092  * =====
00093  * Implementations.
00094  */
00095
00096 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
00097 {
00098     l4_utcb_t *res;
00099     __asm__ ( "mov %%gs:0, %0 \n" : "=r"(res));
00100     return res;
00101 }
00102
00103 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW
00104 {
00105     return u->ip;
00106 }
00107
00108 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW
00109 {
00110     u->ip = pc;
00111 }
00112
00113 L4_INLINE l4_umword_t l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW
00114 {
00115     return u->trapno;
00116 }
00117
00118 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW
00119 {
00120     return u->trapno == 14;
00121 }
00122
00123 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW
00124 {
00125     return (u->pfa & ~7UL) | (u->err & 2);
00126 }
00127
00128 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW

```

```

00129 {
00130     return l4_utcb_exc_typeval(u) == 0xff;
00131 }
00132
00133 #endif /* ! __L4_SYS__INCLUDE__ARCH_AMD64__UTCB_H__ */

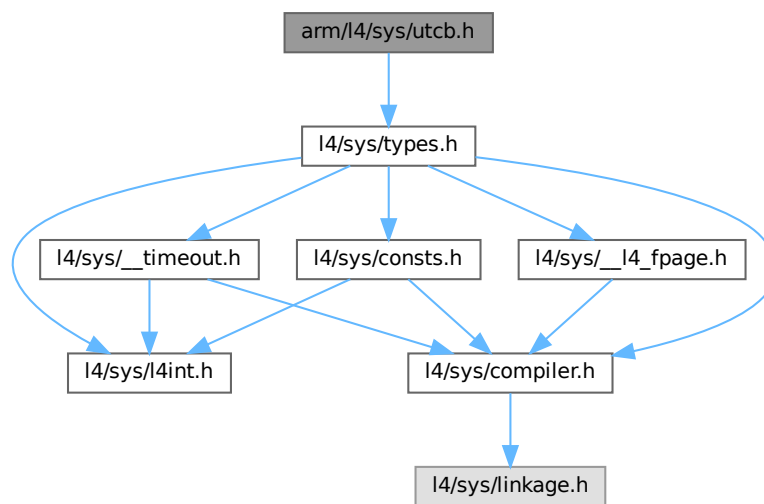
```

## 16.575 arm/l4/sys/utcb.h File Reference

UTCB definitions for ARM.

```
#include <l4/sys/types.h>
```

Include dependency graph for utcb.h:



### Data Structures

- struct [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

### Typedefs

- typedef struct [l4\\_exc\\_regs\\_t](#) [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

### Enumerations

- enum [L4\\_utcb\\_consts\\_arm](#)  
*UTCB constants for ARM.*

## Functions

- [l4\\_umword\\_t l4\\_utcb\\_exc\\_pc \(l4\\_exc\\_regs\\_t const \\*u\) L4\\_NOTHROW](#)  
Access function to get the program counter of the exception state.
- [void l4\\_utcb\\_exc\\_pc\\_set \(l4\\_exc\\_regs\\_t \\*u, l4\\_addr\\_t pc\) L4\\_NOTHROW](#)  
Set the program counter register in the exception state.
- [l4\\_umword\\_t l4\\_utcb\\_exc\\_typeval \(l4\\_exc\\_regs\\_t const \\*u\) L4\\_NOTHROW](#)  
Get the value out of an exception UTCB that describes the type of exception.
- [int l4\\_utcb\\_exc\\_is\\_pf \(l4\\_exc\\_regs\\_t const \\*u\) L4\\_NOTHROW](#)  
Check whether an exception IPC is a page fault.
- [l4\\_addr\\_t l4\\_utcb\\_exc\\_pfa \(l4\\_exc\\_regs\\_t const \\*u\) L4\\_NOTHROW](#)  
Function to get the L4 style page fault address out of an exception.
- [int l4\\_utcb\\_exc\\_is\\_ex\\_regs\\_exception \(l4\\_exc\\_regs\\_t const \\*u\) L4\\_NOTHROW](#)  
Check whether an exception IPC was triggered via [l4\\_thread\\_ex\\_regs\(\)](#).

## 16.575.1 Detailed Description

UTCB definitions for ARM.

Definition in file [utcb.h](#).

## 16.576 utcb.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #ifndef __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__
00025 #define __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__
00026
00027 #include <l4/sys/types.h>
00028
00038 typedef struct l4_exc_regs_t
00039 {
00040     l4_umword_t pfa;
00041     l4_umword_t err;
00043     l4_umword_t r[13];
00044     l4_umword_t sp;
00045     l4_umword_t ulr;
00046     l4_umword_t dummy1;
00047     l4_umword_t pc;
00048     l4_umword_t cpsr;
00049     l4_umword_t tpidruro;
00050     l4_umword_t tpidrurw;
00051 } l4_exc_regs_t;
00052
00058 enum l4_utcb_consts_arm
00059 {
00060     L4_UTCB_EXCEPTION_REGS_SIZE = sizeof(l4_exc_regs_t) / sizeof(l4_umword_t),
00061     L4_UTCB_GENERIC_DATA_SIZE = 63,

```

```

00062 L4_UTCB_GENERIC_BUFFERS_SIZE = 58,
00063
00064 L4_UTCB_MSG_REGS_OFFSET = 0,
00065 L4_UTCB_BUF_REGS_OFFSET = 64 * sizeof(l4_umword_t),
00066 L4_UTCB_THREAD_REGS_OFFSET = 123 * sizeof(l4_umword_t),
00067
00068 L4_UTCB_INHERIT_FPU = 1UL < 24,
00069
00070 L4_UTCB_OFFSET = 512,
00071 };
00072
00073 #include_next <l4/sys/utcb.h>
00074
00075 /*
00076  * =====
00077  * Implementations.
00078  */
00079
00080 #ifdef __GNUC__
00081 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
00082 {
00083     # if defined(__ARM_ARCH) && __ARM_ARCH >= 7
00084         l4_utcb_t *utcb;
00085         __asm__ ("mrc p15, 0, %0, c13, c0, 2" : "=r" (utcb)); // TPIDRURW
00086     #else
00087         register l4_utcb_t *utcb __asm__ ("r0");
00088         __asm__ ("mov lr, pc\n"
00089                 "mvn pc, #0xff\n" // write 0xffffffff00 to pc
00090                 : "=r" (utcb) : : "lr");
00091     #endif
00092     return utcb;
00093 }
00094 #endif
00095
00096 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW
00097 {
00098     return u->pc;
00099 }
00100
00101 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW
00102 {
00103     u->pc = pc;
00104 }
00105
00106 L4_INLINE l4_umword_t l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW
00107 {
00108     return u->err >> 26;
00109 }
00110
00111 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW
00112 {
00113     return ((u->err >> 26) & 0x30) == 0x20;
00114 }
00115
00116 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW
00117 {
00118     return (u->pfa & ~7UL) | ((u->err >> 5) & 2);
00119 }
00120
00121 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW
00122 {
00123     return l4_utcb_exc_typeval(u) == 0x3e;
00124 }
00125
00126 #endif /* ! __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__ */

```

## 16.577 l4/sys/utcb.h File Reference

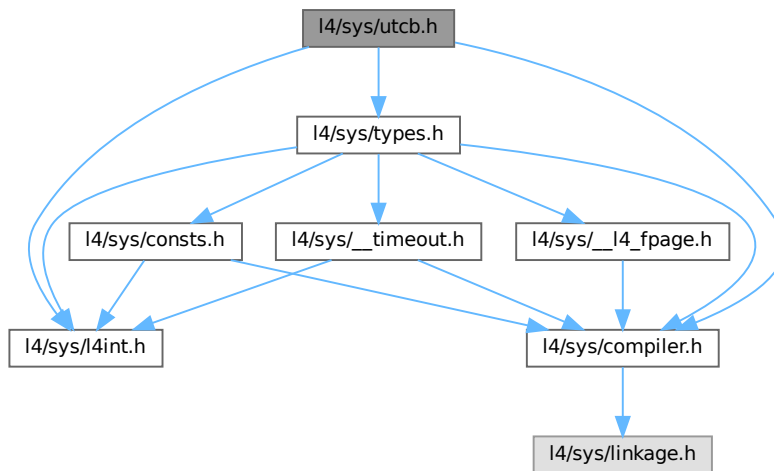
UTCB definitions.

```

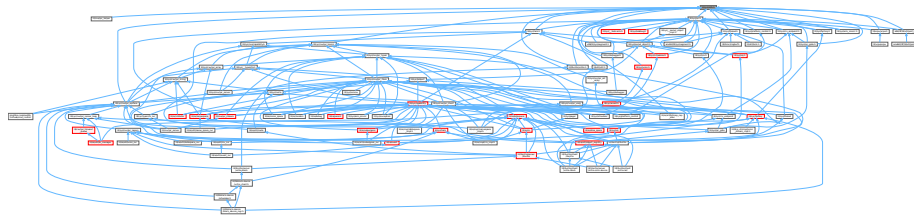
#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>

```

Include dependency graph for utcb.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- union [l4\\_msg\\_regs\\_t](#)  
*Encapsulation of the message-register block in the UTCB.*
- struct [l4\\_buf\\_regs\\_t](#)  
*Encapsulation of the buffer-registers block in the UTCB.*
- struct [l4\\_thread\\_regs\\_t](#)  
*Encapsulation of the thread-control-register block of the UTCB.*

## Typedefs

- typedef struct [l4\\_utcb\\_t](#) [l4\\_utcb\\_t](#)  
*Opaque type for the UTCB.*
- typedef union [l4\\_msg\\_regs\\_t](#) [l4\\_msg\\_regs\\_t](#)  
*Encapsulation of the message-register block in the UTCB.*
- typedef struct [l4\\_buf\\_regs\\_t](#) [l4\\_buf\\_regs\\_t](#)  
*Encapsulation of the buffer-registers block in the UTCB.*
- typedef struct [l4\\_thread\\_regs\\_t](#) [l4\\_thread\\_regs\\_t](#)  
*Encapsulation of the thread-control-register block of the UTCB.*

## Functions

- [l4\\_utcb\\_t](#) \* [l4\\_utcb](#) (void) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
Get the UTCB address.
- [l4\\_msg\\_regs\\_t](#) \* [l4\\_utcb\\_mr](#) (void) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
Get the message-register block of a UTCB.
- [l4\\_buf\\_regs\\_t](#) \* [l4\\_utcb\\_br](#) (void) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
Get the buffer-register block of a UTCB.
- [l4\\_thread\\_regs\\_t](#) \* [l4\\_utcb\\_tcr](#) (void) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
Get the thread-control-register block of a UTCB.
- [l4\\_exc\\_regs\\_t](#) \* [l4\\_utcb\\_exc](#) (void) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
Get the message-register block of a UTCB (for an exception IPC).
- [l4\\_umword\\_t](#) [l4\\_utcb\\_exc\\_pc](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
Access function to get the program counter of the exception state.
- void [l4\\_utcb\\_exc\\_pc\\_set](#) ([l4\\_exc\\_regs\\_t](#) \*u, [l4\\_addr\\_t](#) pc) [L4\\_NOTHROW](#)  
Set the program counter register in the exception state.
- unsigned long [l4\\_utcb\\_exc\\_typeval](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
Get the value out of an exception UTCB that describes the type of exception.
- int [l4\\_utcb\\_exc\\_is\\_pf](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
Check whether an exception IPC is a page fault.
- [l4\\_addr\\_t](#) [l4\\_utcb\\_exc\\_pfa](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
Function to get the L4 style page fault address out of an exception.
- int [l4\\_utcb\\_exc\\_is\\_ex\\_regs\\_exception](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
Check whether an exception IPC was triggered via [l4\\_thread\\_ex\\_regs\(\)](#).
- void [l4\\_utcb\\_inherit\\_fpu](#) (int switch\_on) [L4\\_NOTHROW](#)  
Enable or disable inheritance of FPU state to receiver.
- [l4\\_timeout\\_s](#) [l4\\_timeout\\_abs](#) ([l4\\_kernel\\_clock\\_t](#) pint, int br) [L4\\_NOTHROW](#)  
Set an absolute timeout.
- unsigned [l4\\_utcb\\_mr64\\_idx](#) (unsigned idx) [L4\\_NOTHROW](#)  
Get index into 64bit message registers alias from native-sized index.

## 16.577.1 Detailed Description

UTCB definitions.

Definition in file [utcb.h](#).

## 16.578 utcb.h

[Go to the documentation of this file.](#)

```

00001 /*****
00007 */
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011 *      economic rights: Technische Universität Dresden (Germany)
00012 *
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this

```

```

00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License. This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026  /*****
00027  #ifndef _L4_SYS_UTCB_H
00028  #define _L4_SYS_UTCB_H
00029
00030 #include <l4/sys/types.h>
00031 #include <l4/sys/compiler.h>
00032 #include <l4/sys/l4int.h>
00033
00067 typedef struct l4_utcb_t l4_utcb_t;
00068
00078 typedef union l4_msg_regs_t
00079 {
00080     l4_umword_t mr[L4_UTCB_GENERIC_DATA_SIZE];
00081     l4_uint64_t mr64[L4_UTCB_GENERIC_DATA_SIZE / (sizeof(l4_uint64_t)/sizeof(l4_umword_t))];
00082 } l4_msg_regs_t;
00083
00093 typedef struct l4_buf_regs_t
00094 {
00096     l4_umword_t bdr;
00097
00099     l4_umword_t br[L4_UTCB_GENERIC_BUFFERS_SIZE];
00100 } l4_buf_regs_t;
00101
00110 typedef struct l4_thread_regs_t
00111 {
00113     l4_umword_t error;
00115     l4_umword_t free_marker;
00117     l4_umword_t user[3];
00118 } l4_thread_regs_t;
00119
00120 __BEGIN_DECLS
00121
00132 L4_INLINE l4_utcb_t *l4_utcb_wrap(void) L4_NOTHROW L4_PURE;
00133
00139 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW L4_PURE;
00140
00145 L4_INLINE l4_utcb_t *l4_utcb(void) L4_NOTHROW L4_PURE;
00146
00152 L4_INLINE l4_msg_regs_t *l4_utcb_mr(void) L4_NOTHROW L4_PURE;
00153
00158 L4_INLINE l4_msg_regs_t *l4_utcb_mr_u(l4_utcb_t *u) L4_NOTHROW L4_PURE;
00159
00166 L4_INLINE l4_buf_regs_t *l4_utcb_br(void) L4_NOTHROW L4_PURE;
00167
00172 L4_INLINE l4_buf_regs_t *l4_utcb_br_u(l4_utcb_t *u) L4_NOTHROW L4_PURE;
00173
00179 L4_INLINE l4_thread_regs_t *l4_utcb_tcr(void) L4_NOTHROW L4_PURE;
00180
00185 L4_INLINE l4_thread_regs_t *l4_utcb_tcr_u(l4_utcb_t *u) L4_NOTHROW L4_PURE;
00186
00199 L4_INLINE l4_exc_regs_t *l4_utcb_exc(void) L4_NOTHROW L4_PURE;
00200
00205 L4_INLINE l4_exc_regs_t *l4_utcb_exc_u(l4_utcb_t *u) L4_NOTHROW L4_PURE;
00206
00214 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;
00215
00224 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW;
00225
00230 L4_INLINE unsigned long l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;
00231
00241 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;
00242
00247 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;
00248
00249
00260 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;
00261
00266 L4_INLINE void l4_utcb_inherit_fpu(int switch_on) L4_NOTHROW;
00267
00271 L4_INLINE void l4_utcb_inherit_fpu_u(l4_utcb_t *u, int switch_on) L4_NOTHROW;
00272
00287 L4_INLINE
00288 l4_timeout_s l4_timeout_abs_u(l4_kernel_clock_t pint, int br,
00289                               l4_utcb_t *utcb) L4_NOTHROW;
00303 L4_INLINE
00304 l4_timeout_s l4_timeout_abs(l4_kernel_clock_t pint, int br) L4_NOTHROW;
00305
00313 L4_INLINE
00314 unsigned l4_utcb_mr64_idx(unsigned idx) L4_NOTHROW;
00315
00316 /*****
00317  * Implementations

```

```

00318  *****/
00319
00320 L4_INLINE l4_msg_regs_t *l4_utcb_mr_u(l4_utcb_t *u) L4_NOTHROW
00321 { return (l4_msg_regs_t*)((char*)u + L4_UTCB_MSG_REGS_OFFSET); }
00322
00323 L4_INLINE l4_buf_regs_t *l4_utcb_br_u(l4_utcb_t *u) L4_NOTHROW
00324 { return (l4_buf_regs_t*)((char*)u + L4_UTCB_BUF_REGS_OFFSET); }
00325
00326 L4_INLINE l4_thread_regs_t *l4_utcb_tcr_u(l4_utcb_t *u) L4_NOTHROW
00327 { return (l4_thread_regs_t*)((char*)u + L4_UTCB_THREAD_REGS_OFFSET); }
00328
00329 L4_INLINE l4_exc_regs_t *l4_utcb_exc_u(l4_utcb_t *u) L4_NOTHROW
00330 { return (l4_exc_regs_t*)((char*)u + L4_UTCB_MSG_REGS_OFFSET); }
00331
00332 L4_INLINE void l4_utcb_inherit_fpu_u(l4_utcb_t *u, int switch_on) L4_NOTHROW
00333 {
00334     if (switch_on)
00335         l4_utcb_br_u(u)->bdr |= L4_UTCB_INHERIT_FPU;
00336     else
00337         l4_utcb_br_u(u)->bdr &= ~L4_UTCB_INHERIT_FPU;
00338 }
00339
00340 L4_INLINE l4_utcb_t *l4_utcb(void) L4_NOTHROW
00341 {
00342     #ifdef L4SYS_USE_UTCB_WRAP
00343         return l4_utcb_wrap();
00344     #else
00345         return l4_utcb_direct();
00346     #endif
00347 }
00348
00349
00350
00351
00352 L4_INLINE l4_msg_regs_t *l4_utcb_mr(void) L4_NOTHROW
00353 { return l4_utcb_mr_u(l4_utcb()); }
00354
00355 L4_INLINE l4_buf_regs_t *l4_utcb_br(void) L4_NOTHROW
00356 { return l4_utcb_br_u(l4_utcb()); }
00357
00358 L4_INLINE l4_thread_regs_t *l4_utcb_tcr(void) L4_NOTHROW
00359 { return l4_utcb_tcr_u(l4_utcb()); }
00360
00361 L4_INLINE l4_exc_regs_t *l4_utcb_exc(void) L4_NOTHROW
00362 { return l4_utcb_exc_u(l4_utcb()); }
00363
00364 L4_INLINE void l4_utcb_inherit_fpu(int switch_on) L4_NOTHROW
00365 { l4_utcb_inherit_fpu_u(l4_utcb(), switch_on); }
00366
00367 L4_INLINE
00368 l4_timeout_s l4_timeout_abs_u(l4_kernel_clock_t val, int pos,
00369                               l4_utcb_t *utcb) L4_NOTHROW
00370 {
00371     union T
00372     {
00373         l4_kernel_clock_t t;
00374         l4_umword_t m[sizeof(l4_kernel_clock_t)/sizeof(l4_umword_t)];
00375     };
00376     l4_timeout_s to;
00377     to.t = 0x8000 | pos;
00378     ((union T*)(l4_utcb_br_u(utcb)->br + pos))->t = val;
00379     return to;
00380 }
00381
00382 L4_INLINE
00383 l4_timeout_s l4_timeout_abs(l4_kernel_clock_t val, int pos) L4_NOTHROW
00384 { return l4_timeout_abs_u(val, pos, l4_utcb()); }
00385
00386 L4_INLINE unsigned l4_utcb_mr64_idx(unsigned idx) L4_NOTHROW
00387 { return idx / (sizeof(l4_uint64_t) / sizeof(l4_umword_t)); }
00388
00389 __END_DECLS
00390
00391 #endif /* !_L4_SYS_UTCB_H */

```

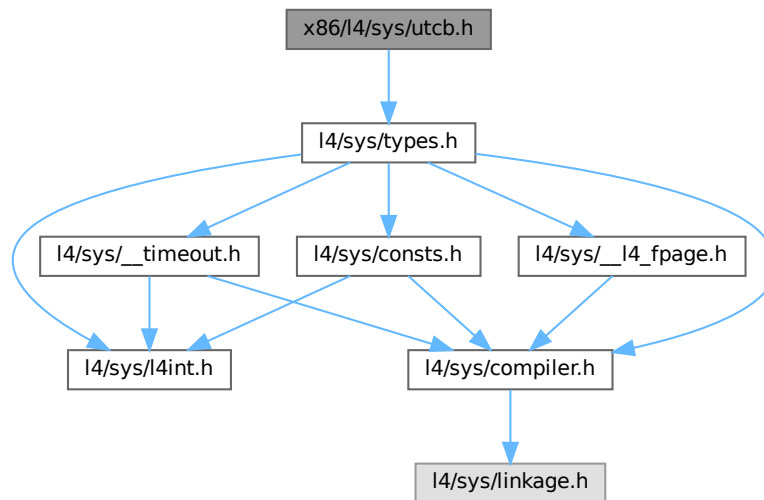
## 16.579 x86/I4/sys/utcb.h File Reference

UTCB definitions for X86.



```
#include <l4/sys/types.h>
```

Include dependency graph for utcb.h:



## Data Structures

- struct `l4_exc_regs_t`  
*UTCB structure for exceptions.*

## Typedefs

- typedef struct `l4_exc_regs_t` `l4_exc_regs_t`  
*UTCB structure for exceptions.*

## Enumerations

- enum `L4_utcb_consts_x86` {  
`L4_UTCB_EXCEPTION_REGS_SIZE = 19` , `L4_UTCB_GENERIC_DATA_SIZE = 63` , `L4_UTCB_GENERIC_BUFFERS_SIZE = 58` , `L4_UTCB_MSG_REGS_OFFSET = 0` ,  
`L4_UTCB_BUF_REGS_OFFSET = 64 * sizeof(l4_umword_t)` , `L4_UTCB_THREAD_REGS_OFFSET = 123 * sizeof(l4_umword_t)` , `L4_UTCB_INHERIT_FPU = 1UL << 24` , `L4_UTCB_OFFSET = 512` }  
*UTCB constants for x86.*

## Functions

- `l4_umword_t l4_utcb_exc_pc (l4_exc_regs_t const *u)` `L4_NOTHROW`  
*Access function to get the program counter of the exception state.*
- `void l4_utcb_exc_pc_set (l4_exc_regs_t *u, l4_addr_t pc)` `L4_NOTHROW`  
*Set the program counter register in the exception state.*
- `l4_umword_t l4_utcb_exc_typeval (l4_exc_regs_t const *u)` `L4_NOTHROW`

Get the value out of an exception UTCB that describes the type of exception.

- `int l4_utcb_exc_is_pf (l4_exc_regs_t const *u) L4_NOTHROW`

Check whether an exception IPC is a page fault.

- `l4_addr_t l4_utcb_exc_pfa (l4_exc_regs_t const *u) L4_NOTHROW`

Function to get the L4 style page fault address out of an exception.

- `int l4_utcb_exc_is_ex_regs_exception (l4_exc_regs_t const *u) L4_NOTHROW`

Check whether an exception IPC was triggered via `l4_thread_ex_regs()`.

## 16.579.1 Detailed Description

UTCB definitions for X86.

Definition in file [utcb.h](#).

## 16.580 utcb.h

[Go to the documentation of this file.](#)

```
00001 /*****
00002  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019 /*****
00020  * #ifndef __L4_SYS__INCLUDE__ARCH_X86__UTCB_H__
00021  * #define __L4_SYS__INCLUDE__ARCH_X86__UTCB_H__
00022  * #include <l4/sys/types.h>
00023  *
00024  * enum l4_utcb_consts_x86
00025  * {
00026  *     L4_UTCB_EXCEPTION_REGS_SIZE      = 19,
00027  *     L4_UTCB_GENERIC_DATA_SIZE        = 63,
00028  *     L4_UTCB_GENERIC_BUFFERS_SIZE     = 58,
00029  *     L4_UTCB_MSG_REGS_OFFSET          = 0,
00030  *     L4_UTCB_BUF_REGS_OFFSET          = 64 * sizeof(l4_umword_t),
00031  *     L4_UTCB_THREAD_REGS_OFFSET       = 123 * sizeof(l4_umword_t),
00032  *     L4_UTCB_INHERIT_FPU               = 1UL < 24,
00033  *     L4_UTCB_OFFSET                   = 512,
00034  * };
00035  * typedef struct l4_exc_regs_t
00036  * {
00037  *     l4_umword_t es;
00038  *     l4_umword_t ds;
00039  *     l4_umword_t gs;
00040  *     l4_umword_t fs;
00041  *     l4_umword_t edi;
00042  *     l4_umword_t esi;
00043  *     l4_umword_t ebp;
00044  *     l4_umword_t pfa;
00045  * }
```

```

00083     l4_umword_t ebx;
00084     l4_umword_t edx;
00085     l4_umword_t ecx;
00086     l4_umword_t eax;
00088     l4_umword_t trapno;
00089     l4_umword_t err;
00091     l4_umword_t ip;
00092     l4_umword_t dummy1;
00093     l4_umword_t flags;
00094     l4_umword_t sp;
00095     l4_umword_t ss;
00096 } l4_exc_regs_t;
00097
00098 #include_next <l4/sys/utcb.h>
00099
00100 /*
00101  * =====
00102  * Implementations.
00103  */
00104
00105 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
00106 {
00107     l4_utcb_t *utcb;
00108     __asm__ ("mov %%fs:0, %0" : "=r" (utcb));
00109     return utcb;
00110 }
00111
00112 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW
00113 {
00114     return u->ip;
00115 }
00116
00117 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW
00118 {
00119     u->ip = pc;
00120 }
00121
00122 L4_INLINE void l4_utcb_exc_sp_set(l4_exc_regs_t *u, l4_addr_t sp) L4_NOTHROW
00123 {
00124     u->sp = sp;
00125 }
00126
00127 L4_INLINE l4_umword_t l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW
00128 {
00129     return u->trapno;
00130 }
00131
00132 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW
00133 {
00134     return u->trapno == 14;
00135 }
00136
00137 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW
00138 {
00139     return (u->pfa & ~7UL) | (u->err & 2);
00140 }
00141
00142 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW
00143 {
00144     return l4_utcb_exc_typeval(u) == 0xff;
00145 }
00146
00147 #endif /* ! __L4_SYS__INCLUDE__ARCH_X86__UTCB_H__ */

```

## 16.581 l4/sys/vcon File Reference

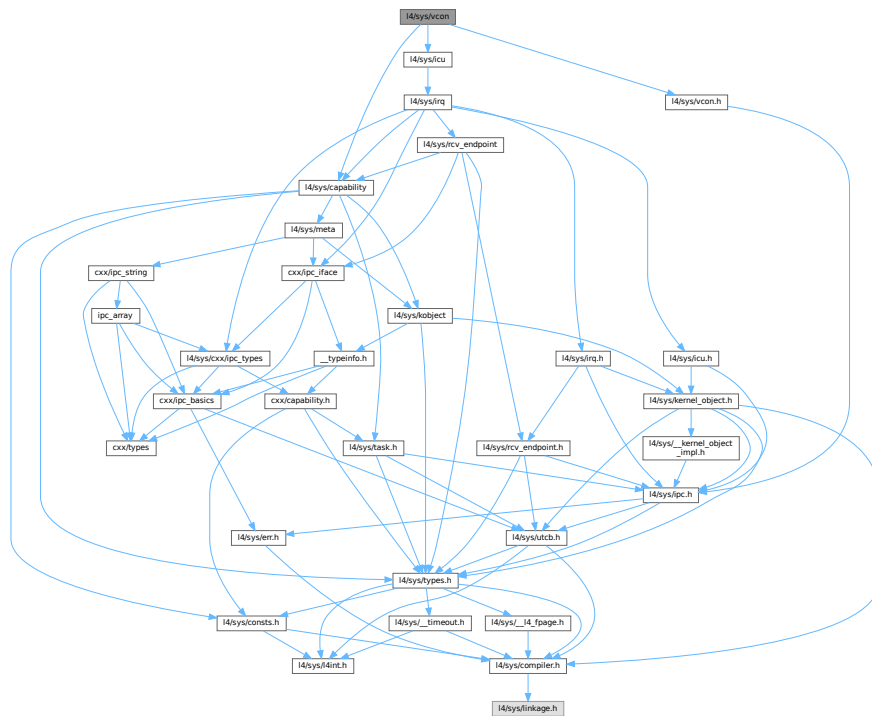
C++ Virtual console interface.

```

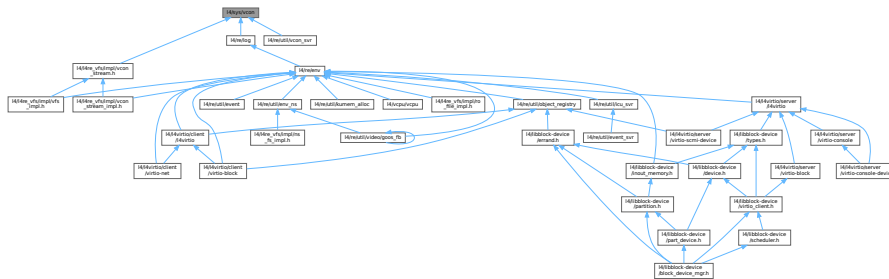
#include <l4/sys/icu>
#include <l4/sys/vcon.h>
#include <l4/sys/capability>

```

Include dependency graph for vcon:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4::Vcon`  
*C++ L4 Vcon interface, see [Virtual Console](#) for the C interface.*

## Namespaces

- namespace **L4**  
*L4 low-level kernel interface.*

## 16.581.1 Detailed Description

C++ Virtual console interface.

Definition in file [vcon](#).

## 16.582 vcon

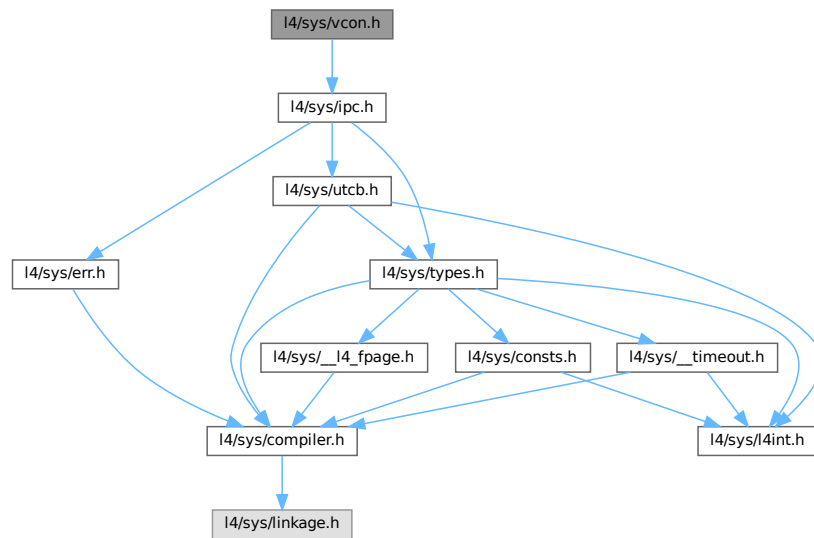
[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/sys/icu>
00028 #include <l4/sys/vcon.h>
00029 #include <l4/sys/capability>
00030
00031 namespace L4 {
00032
00056 class Vcon :
00057     public Kobject_t<Vcon, Icu, L4_PROTO_LOG>
00058 {
00059 public:
00075     l4_msgtag_t
00076     send(char const *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00077     { return l4_vcon_send_u(cap(), buf, size, utcb); }
00078
00089     long
00090     write(char const *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00091     { return l4_vcon_write_u(cap(), buf, size, utcb); }
00092
00108     int
00109     read(char *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00110     { return l4_vcon_read_u(cap(), buf, size, utcb); }
00111
00135     int
00136     read_with_flags(char *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00137     { return l4_vcon_read_with_flags_u(cap(), buf, size, utcb); }
00138
00148     l4_msgtag_t
00149     set_attr(l4_vcon_attr_t const *attr, l4_utcb_t *utcb = l4_utcb()) const noexcept
00150     { return l4_vcon_set_attr_u(cap(), attr, utcb); }
00151
00161     l4_msgtag_t
00162     get_attr(l4_vcon_attr_t *attr, l4_utcb_t *utcb = l4_utcb()) const noexcept
00163     { return l4_vcon_get_attr_u(cap(), attr, utcb); }
00164
00165     typedef L4::Typeid::Raw_ipc<Vcon> Rpcs;
00166 };
00167
00168 }
```

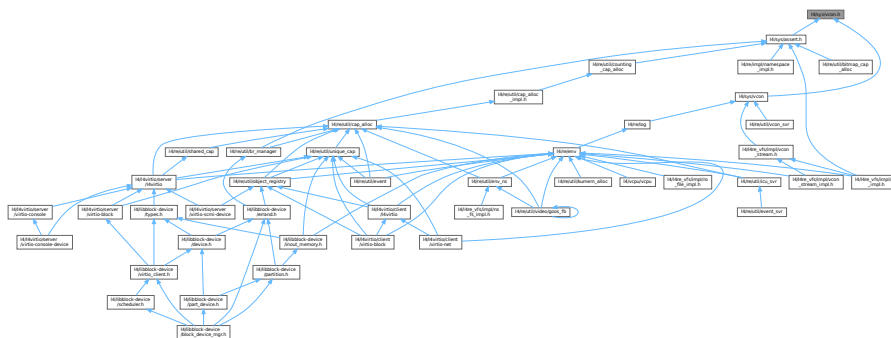
## 16.583 l4/sys/vcon.h File Reference

Virtual console interface.

```
#include <l4/sys/ipc.h>
Include dependency graph for vcon.h:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4\\_vcon\\_attr\\_t](#)  
*Vcon attribute structure.*

## Typedefs

- typedef struct [l4\\_vcon\\_attr\\_t](#) [l4\\_vcon\\_attr\\_t](#)  
*Vcon attribute structure.*

## Enumerations

- enum `L4_vcon_size_consts` { `L4_VCON_WRITE_SIZE` = (L4\_UTCB\_GENERIC\_DATA\_SIZE - 2) \* sizeof(l4\_umword\_t) , `L4_VCON_READ_SIZE` = (L4\_UTCB\_GENERIC\_DATA\_SIZE - 1) \* sizeof(l4\_umword\_t) }  
*Size constants.*
- enum `L4_vcon_read_flags` { `L4_VCON_READ_SIZE_MASK` = 0x3ffffff , `L4_VCON_READ_STAT_BREAK` = 1 << 30 , `L4_VCON_READ_STAT_DONE` = 1 << 31 }  
*Vcon read flags.*
- enum `L4_vcon_i_flags` { `L4_VCON_INLCR` = 000100 , `L4_VCON_IGNCR` = 000200 , `L4_VCON_ICRNL` = 000400 }  
*Input flags.*
- enum `L4_vcon_o_flags` { `L4_VCON_ONLCR` = 000004 , `L4_VCON_OCRNL` = 000010 , `L4_VCON_ONLRET` = 000040 }  
*Output flags.*
- enum `L4_vcon_l_flags` { `L4_VCON_ICANON` = 000002 , `L4_VCON_ECHO` = 000010 }  
*Local flags.*
- enum `L4_vcon_ops` { `L4_VCON_WRITE_OP` = 0UL , `L4_VCON_READ_OP` = 1UL , `L4_VCON_SET_ATTR_OP` = 2UL , `L4_VCON_GET_ATTR_OP` = 3UL }  
*Operations on vcon objects.*

## Functions

- `l4_msgtag_t l4_vcon_send` (l4\_cap\_idx\_t vcon, char const \*buf, unsigned size) `L4_NOTHROW`  
*Send data to virtual console.*
- `l4_msgtag_t l4_vcon_send_u` (l4\_cap\_idx\_t vcon, char const \*buf, unsigned size, l4\_utcb\_t \*utcb) `L4_NOTHROW`  
*Send data to *this* virtual console.*
- `long l4_vcon_write` (l4\_cap\_idx\_t vcon, char const \*buf, unsigned size) `L4_NOTHROW`  
*Write data to virtual console.*
- `long l4_vcon_write_u` (l4\_cap\_idx\_t vcon, char const \*buf, unsigned size, l4\_utcb\_t \*utcb) `L4_NOTHROW`  
*Write data to *this* virtual console.*
- `int l4_vcon_read` (l4\_cap\_idx\_t vcon, char \*buf, unsigned size) `L4_NOTHROW`  
*Read data from virtual console.*
- `int l4_vcon_read_u` (l4\_cap\_idx\_t vcon, char \*buf, unsigned size, l4\_utcb\_t \*utcb) `L4_NOTHROW`  
*Read data from *this* virtual console.*
- `int l4_vcon_read_with_flags` (l4\_cap\_idx\_t vcon, char \*buf, unsigned size) `L4_NOTHROW`  
*Read data from virtual console, extended version including flags.*
- `l4_msgtag_t l4_vcon_set_attr` (l4\_cap\_idx\_t vcon, l4\_vcon\_attr\_t const \*attr) `L4_NOTHROW`  
*Set attributes of a Vcon.*
- `l4_msgtag_t l4_vcon_set_attr_u` (l4\_cap\_idx\_t vcon, l4\_vcon\_attr\_t const \*attr, l4\_utcb\_t \*utcb) `L4_NOTHROW`  
*Set the attributes of *this* virtual console.*
- `l4_msgtag_t l4_vcon_get_attr` (l4\_cap\_idx\_t vcon, l4\_vcon\_attr\_t \*attr) `L4_NOTHROW`  
*Get attributes of a Vcon.*
- `l4_msgtag_t l4_vcon_get_attr_u` (l4\_cap\_idx\_t vcon, l4\_vcon\_attr\_t \*attr, l4\_utcb\_t \*utcb) `L4_NOTHROW`  
*Get attributes of *this* virtual console.*
- `void l4_vcon_set_attr_raw` (l4\_vcon\_attr\_t \*attr) `L4_NOTHROW`  
*Set terminal attributes to disable all special processing.*

## 16.583.1 Detailed Description

Virtual console interface.

Definition in file [vcon.h](#).

## 16.583.2 Enumeration Type Documentation

### 16.583.2.1 L4\_vcon\_read\_flags

```
enum L4_vcon_read_flags
```

Vcon read flags.

Enumerator

L4_VCON_READ_SIZE_MASK	Size mask.
L4_VCON_READ_STAT_BREAK	Break condition flag.
L4_VCON_READ_STAT_DONE	Done condition flag.

Definition at line 179 of file [vcon.h](#).

## 16.584 vcon.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *                Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *                Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00009  *                economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/sys/ipc.h>
00027
00067 L4_INLINE l4_msgtag_t
00068 l4_vcon_send(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW;
00069
00076 L4_INLINE l4_msgtag_t
00077 l4_vcon_send_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW;
00078
00090 L4_INLINE long
00091 l4_vcon_write(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW;
00092
00099 L4_INLINE long
00100 l4_vcon_write_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW;
00101
00106 enum L4_vcon_size_consts
00107 {
```



```

00109 L4_VCON_WRITE_SIZE = (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(l4_umword_t),
00111 L4_VCON_READ_SIZE  = (L4_UTCB_GENERIC_DATA_SIZE - 1) * sizeof(l4_umword_t),
00112 };
00113
00129 L4_INLINE int
00130 l4_vcon_read(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW;
00131
00138 L4_INLINE int
00139 l4_vcon_read_u(l4_cap_idx_t vcon, char *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW;
00140
00166 L4_INLINE int
00167 l4_vcon_read_with_flags(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW;
00168
00172 L4_INLINE int
00173 l4_vcon_read_with_flags_u(l4_cap_idx_t vcon, char *buf, unsigned size,
00174                           l4_utcb_t *utcb) L4_NOTHROW;
00175
00179 enum l4_vcon_read_flags
00180 {
00181     L4_VCON_READ_SIZE_MASK = 0x3fffffff,
00182     L4_VCON_READ_STAT_BREAK = 1 << 30,
00183     L4_VCON_READ_STAT_DONE = 1 << 31,
00184 };
00185
00196 typedef struct l4_vcon_attr_t
00197 {
00198     l4_umword_t i_flags;
00199     l4_umword_t o_flags;
00200     l4_umword_t l_flags;
00201
00202 #ifdef __cplusplus
00209     inline void set_raw();
00210 #endif
00211 } l4_vcon_attr_t;
00212
00217 enum l4_vcon_i_flags
00218 {
00219     L4_VCON_INLCR = 000100,
00220     L4_VCON_IGNCR = 000200,
00221     L4_VCON_ICRNL = 000400,
00222 };
00223
00228 enum l4_vcon_o_flags
00229 {
00230     L4_VCON_ONLCR = 000004,
00231     L4_VCON_OCRNL = 000010,
00232     L4_VCON_ONLRET = 000040,
00233 };
00234
00239 enum l4_vcon_l_flags
00240 {
00241     L4_VCON_ICANON = 000002,
00242     L4_VCON_ECHO = 000010,
00243 };
00244
00253 L4_INLINE l4_msgtag_t
00254 l4_vcon_set_attr(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr) L4_NOTHROW;
00255
00262 L4_INLINE l4_msgtag_t
00263 l4_vcon_set_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr,
00264                   l4_utcb_t *utcb) L4_NOTHROW;
00265
00274 L4_INLINE l4_msgtag_t
00275 l4_vcon_get_attr(l4_cap_idx_t vcon, l4_vcon_attr_t *attr) L4_NOTHROW;
00276
00283 L4_INLINE l4_msgtag_t
00284 l4_vcon_get_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t *attr,
00285                   l4_utcb_t *utcb) L4_NOTHROW;
00286
00292 L4_INLINE void
00293 l4_vcon_set_attr_raw(l4_vcon_attr_t *attr) L4_NOTHROW;
00294
00295
00300 enum l4_vcon_ops
00301 {
00302     L4_VCON_WRITE_OP = 0UL,
00303     L4_VCON_READ_OP = 1UL,
00304     L4_VCON_SET_ATTR_OP = 2UL,
00305     L4_VCON_GET_ATTR_OP = 3UL,
00306 };
00307
00308 /***** Implementations *****/
00309
00310 L4_INLINE l4_msgtag_t
00311 l4_vcon_send_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW
00312 {
00313     l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);

```

```

00314     mr->mr[0] = L4_VCON_WRITE_OP;
00315     mr->mr[1] = size;
00316     __builtin_memcpy(&mr->mr[2], buf, size);
00317     return l4_ipc_send(vcon, utcb,
00318                       l4_msgtag(L4_PROTO_LOG, 2 + l4_bytes_to_mwords(size),
00319                                0, L4_MSGTAG_SCHEDULE),
00320                       L4_IPC_NEVER);
00321 }
00322
00323 L4_INLINE l4_msgtag_t
00324 l4_vcon_send(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW
00325 {
00326     return l4_vcon_send_u(vcon, buf, size, l4_utcb());
00327 }
00328
00329 L4_INLINE long
00330 l4_vcon_write_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW
00331 {
00332     l4_msgtag_t t;
00333
00334     if (size > L4_VCON_WRITE_SIZE)
00335         size = L4_VCON_WRITE_SIZE;
00336
00337     t = l4_vcon_send_u(vcon, buf, size, utcb);
00338     if (l4_msgtag_has_error(t))
00339         return l4_error(t);
00340
00341     return (long) size;
00342 }
00343
00344 L4_INLINE long
00345 l4_vcon_write(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW
00346 {
00347     return l4_vcon_write_u(vcon, buf, size, l4_utcb());
00348 }
00349
00350 L4_INLINE int
00351 l4_vcon_read_with_flags_u(l4_cap_idx_t vcon, char *buf, unsigned size,
00352                          l4_utcb_t *utcb) L4_NOTHROW
00353 {
00354     int ret;
00355     unsigned r;
00356     l4_msg_regs_t *mr;
00357
00358     mr = l4_utcb_mr_u(utcb);
00359     mr->mr[0] = (size << 16) | L4_VCON_READ_OP;
00360
00361     ret = l4_error_u(l4_ipc_call(vcon, utcb,
00362                                l4_msgtag(L4_PROTO_LOG, 1, 0, 0),
00363                                L4_IPC_NEVER),
00364                    utcb);
00365     if (ret < 0)
00366         return ret;
00367
00368     r = mr->mr[0] & L4_VCON_READ_SIZE_MASK;
00369
00370     if (!(mr->mr[0] & L4_VCON_READ_STAT_DONE)) // !eof
00371         ret = size + 1;
00372     else if (r < size)
00373         ret = r;
00374     else
00375         ret = size;
00376
00377     if (L4_LIKELY(buf != NULL))
00378         __builtin_memcpy(buf, &mr->mr[1], r < size ? r : size);
00379
00380     return ret | (mr->mr[0] & ~(L4_VCON_READ_STAT_DONE | L4_VCON_READ_SIZE_MASK));
00381 }
00382
00383 L4_INLINE int
00384 l4_vcon_read_with_flags(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW
00385 {
00386     return l4_vcon_read_with_flags_u(vcon, buf, size, l4_utcb());
00387 }
00388
00389 L4_INLINE int
00390 l4_vcon_read_u(l4_cap_idx_t vcon, char *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW
00391 {
00392     int r = l4_vcon_read_with_flags_u(vcon, buf, size, utcb);
00393     if (r < 0)
00394         return r;
00395
00396     return r & L4_VCON_READ_SIZE_MASK;
00397 }
00398
00399 L4_INLINE int
00400 l4_vcon_read(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW

```

```

00401 {
00402     return l4_vcon_read_u(vcon, buf, size, l4_utcb());
00403 }
00404
00405 L4_INLINE l4_msgtag_t
00406 l4_vcon_set_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr,
00407                   l4_utcb_t *utcb) L4_NOTHROW
00408 {
00409     l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00410
00411     mr->mr[0] = L4_VCON_SET_ATTR_OP;
00412     __builtin_memcpy(&mr->mr[1], attr, sizeof(*attr));
00413
00414     return l4_ipc_call(vcon, utcb,
00415                       l4_msgtag(L4_PROTO_LOG, 4, 0, 0),
00416                       L4_IPC_NEVER);
00417 }
00418
00419 L4_INLINE l4_msgtag_t
00420 l4_vcon_set_attr(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr) L4_NOTHROW
00421 {
00422     return l4_vcon_set_attr_u(vcon, attr, l4_utcb());
00423 }
00424
00425 L4_INLINE l4_msgtag_t
00426 l4_vcon_get_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t *attr,
00427                   l4_utcb_t *utcb) L4_NOTHROW
00428 {
00429     l4_msgtag_t res;
00430     l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00431
00432     mr->mr[0] = L4_VCON_GET_ATTR_OP;
00433
00434     res = l4_ipc_call(vcon, utcb,
00435                      l4_msgtag(L4_PROTO_LOG, 1, 0, 0),
00436                      L4_IPC_NEVER);
00437     if (l4_error_u(res, utcb) >= 0)
00438         __builtin_memcpy(attr, &mr->mr[1], sizeof(*attr));
00439
00440     return res;
00441 }
00442
00443 L4_INLINE l4_msgtag_t
00444 l4_vcon_get_attr(l4_cap_idx_t vcon, l4_vcon_attr_t *attr) L4_NOTHROW
00445 {
00446     return l4_vcon_get_attr_u(vcon, attr, l4_utcb());
00447 }
00448
00449 L4_INLINE void
00450 l4_vcon_set_attr_raw(l4_vcon_attr_t *attr) L4_NOTHROW
00451 {
00452     attr->i_flags = 0;
00453     attr->o_flags = 0;
00454     attr->l_flags = 0;
00455 }
00456
00457 #ifdef __cplusplus
00458 inline void
00459 l4_vcon_attr_t::set_raw()
00460 { l4_vcon_set_attr_raw(this); }
00461 #endif

```

## 16.585 l4/sys/vcpu.h File Reference

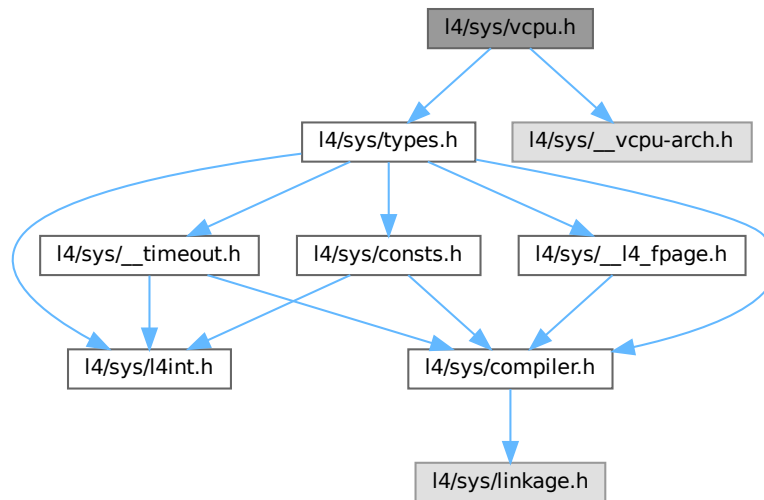
### vCPU API

```

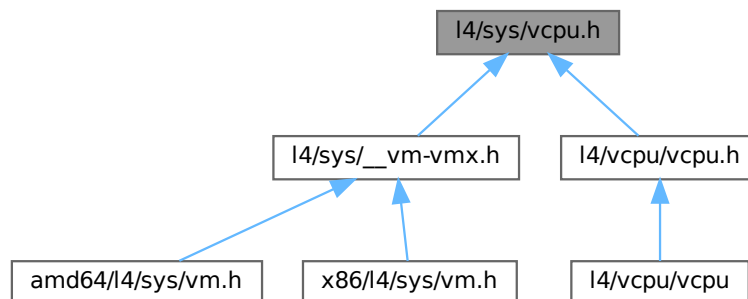
#include <l4/sys/types.h>
#include <l4/sys/__vcpu-arch.h>

```

Include dependency graph for `vcpu.h`:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `l4_vcpu_state_t`  
*State of a vCPU.*

## Typedefs

- typedef struct `l4_vcpu_state_t` `l4_vcpu_state_t`  
*State of a vCPU.*

## Enumerations

- enum `L4_vcpu_state_flags` {  
`L4_VCPU_F_IRQ` = 0x01 , `L4_VCPU_F_PAGE_FAULTS` = 0x02 , `L4_VCPU_F_EXCEPTIONS` = 0x04 ,  
`L4_VCPU_F_USER_MODE` = 0x20 ,  
`L4_VCPU_F_FPU_ENABLED` = 0x80 }  
*State flags of a vCPU.*
- enum `L4_vcpu_sticky_flags` { `L4_VCPU_SF_IRQ_PENDING` = 0x01 }  
*Sticky flags of a vCPU.*

## Functions

- int `l4_vcpu_check_version` (`l4_vcpu_state_t` const \*vcpu) `L4_NOTHROW`  
*Check if a vCPU state has the right version.*

### 16.585.1 Detailed Description

vCPU API

Definition in file [vcpu.h](#).

### 16.585.2 Function Documentation

#### 16.585.2.1 l4\_vcpu\_check\_version()

```
int l4_vcpu_check_version (
    l4_vcpu_state_t const * vcpu ) [inline]
```

Check if a vCPU state has the right version.

#### Parameters

<code>vcpu</code>	A pointer to an initialized vCPU state.
-------------------	---

#### Return values

1	If the vCPU state has a matching version ID for the current vCPU user-level structures.
0	If the vCPU state has a different (incompatible) version ID than the current vCPU user-level structures.

Definition at line 202 of file [vcpu.h](#).

References [L4\\_VCPU\\_STATE\\_VERSION](#), and [l4\\_vcpu\\_state\\_t::version](#).

## 16.586 vcpu.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00023 #pragma once
00024
00025 #include <l4/sys/types.h>
00026 #include <l4/sys/__vcpu-arch.h>
00027
00086 typedef struct l4_vcpu_state_t
00087 {
00088     l4_umword_t      version;
00091     l4_umword_t      user_data[7];
00092     l4_vcpu_regs_t   r;
00093     l4_vcpu_ipc_regs_t i;
00094
00095     l4_uint16_t      state;
00096     l4_uint16_t      saved_state;
00097     l4_uint16_t      sticky_flags;
00098     l4_uint16_t      _reserved;
00099
00100     l4_cap_idx_t      user_task;
00101
00102     l4_umword_t      entry_sp;
00103     l4_umword_t      entry_ip;
00104     l4_umword_t      reserved_sp;
00105     l4_vcpu_arch_state_t arch_state;
00106 } l4_vcpu_state_t;
00107
00112 enum L4_vcpu_state_flags
00113 {
00125     L4_VCPU_F_IRQ          = 0x01,
00126
00140     L4_VCPU_F_PAGE_FAULTS = 0x02,
00141
00153     L4_VCPU_F_EXCEPTIONS  = 0x04,
00154
00163     L4_VCPU_F_USER_MODE   = 0x20,
00164
00171     L4_VCPU_F_FPU_ENABLED = 0x80,
00172 };
00173
00178 enum L4_vcpu_sticky_flags
00179 {
00182     L4_VCPU_SF_IRQ_PENDING = 0x01,
00183 };
00184
00196 L4_INLINE int
00197 l4_vcpu_check_version(l4_vcpu_state_t const *vcpu) L4_NOTHROW;
00198
00199 /* IMPLEMENTATION: -----*/
00200
00201 L4_INLINE int
00202 l4_vcpu_check_version(l4_vcpu_state_t const *vcpu) L4_NOTHROW
00203 {
00204     return vcpu->version == L4_VCPU_STATE_VERSION;
00205 }

```

## 16.587 l4/vcpu/vcpu.h File Reference

vCPU support library (C interface).

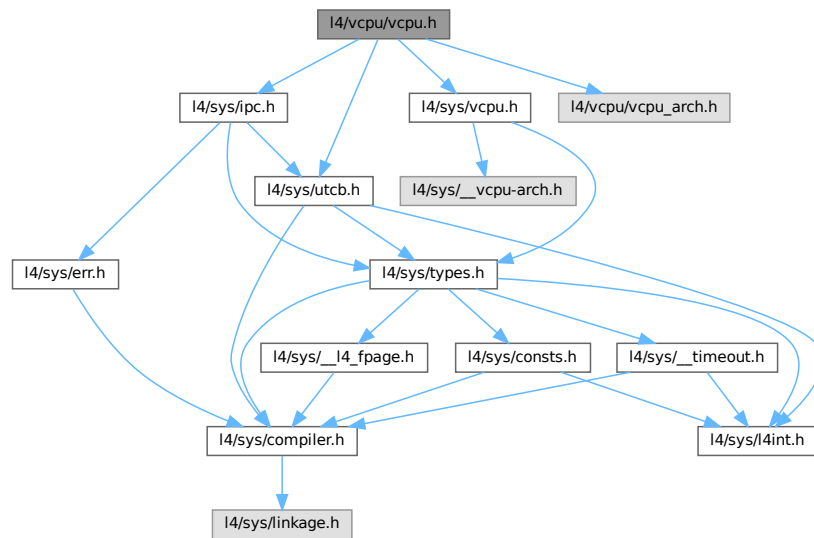
```

#include <l4/sys/vcpu.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>

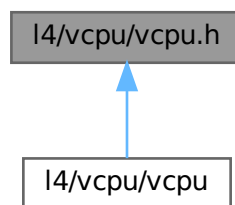
```

```
#include <l4/vcpu/vcpu_arch.h>
```

Include dependency graph for vcpu.h:



This graph shows which files directly or indirectly include this file:



## Functions

- void [l4vcpu\\_irq\\_disable](#) ([l4\\_vcpu\\_state\\_t](#) \*vcpu) [L4\\_NOTHROW](#)  
*Disable a vCPU for event delivery.*
- unsigned [l4vcpu\\_irq\\_disable\\_save](#) ([l4\\_vcpu\\_state\\_t](#) \*vcpu) [L4\\_NOTHROW](#)  
*Disable a vCPU for event delivery and return previous state.*
- void [l4vcpu\\_irq\\_enable](#) ([l4\\_vcpu\\_state\\_t](#) \*vcpu, [l4\\_utcb\\_t](#) \*utcb, [l4vcpu\\_event\\_hndl\\_t](#) do\_event\_work\_cb, [l4vcpu\\_setup\\_ipc\\_t](#) setup\_ipc) [L4\\_NOTHROW](#)  
*Enable a vCPU for event delivery.*
- void [l4vcpu\\_irq\\_restore](#) ([l4\\_vcpu\\_state\\_t](#) \*vcpu, unsigned s, [l4\\_utcb\\_t](#) \*utcb, [l4vcpu\\_event\\_hndl\\_t](#) do\_↔ event\_work\_cb, [l4vcpu\\_setup\\_ipc\\_t](#) setup\_ipc) [L4\\_NOTHROW](#)  
*Restore a previously saved IRQ/event state.*

- void `l4vcpu_wait_for_event` (`l4_vcpu_state_t` \*vcpu, `l4_utcb_t` \*utcb, `l4vcpu_event_hndl_t` do\_event\_work\_cb, `l4vcpu_setup_ipc_t` setup\_ipc) `L4_NOTHROW`  
Wait for event.
- void `l4vcpu_print_state` (const `l4_vcpu_state_t` \*vcpu, const char \*prefix) `L4_NOTHROW`  
Print the state of a vCPU.
- int `l4vcpu_is_irq_entry` (`l4_vcpu_state_t` const \*vcpu) `L4_NOTHROW`  
Return whether the entry reason was an IRQ/IPC message.
- int `l4vcpu_is_page_fault_entry` (`l4_vcpu_state_t` const \*vcpu) `L4_NOTHROW`  
Return whether the entry reason was a page fault.
- int `l4vcpu_ext_alloc` (`l4_vcpu_state_t` \*\*vcpu, `l4_addr_t` \*ext\_state, `l4_cap_idx_t` task, `l4_cap_idx_t` regmgr) `L4_NOTHROW`  
Allocate state area for an extended vCPU.

## 16.587.1 Detailed Description

vCPU support library (C interface).

Definition in file `vcpu.h`.

## 16.588 vcpu.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00003  *      economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00022 #pragma once
00023
00024 #include <l4/sys/vcpu.h>
00025 #include <l4/sys/utcb.h>
00026
00027 __BEGIN_DECLS
00028
00044 typedef void (*l4vcpu_event_hndl_t) (l4_vcpu_state_t *vcpu);
00045 typedef void (*l4vcpu_setup_ipc_t) (l4_utcb_t *utcb);
00046
00053 L4_CV L4_INLINE
00054 void
00055 l4vcpu_irq_disable(l4_vcpu_state_t *vcpu) L4_NOTHROW;
00056
00065 L4_CV L4_INLINE
00066 unsigned
00067 l4vcpu_irq_disable_save(l4_vcpu_state_t *vcpu) L4_NOTHROW;
00068
00081 L4_CV L4_INLINE
00082 void
00083 l4vcpu_irq_enable(l4_vcpu_state_t *vcpu, l4_utcb_t *utcb,
00084                  l4vcpu_event_hndl_t do_event_work_cb,
00085                  l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW;
00086
00101 L4_CV L4_INLINE
00102 void
00103 l4vcpu_irq_restore(l4_vcpu_state_t *vcpu, unsigned s,
00104                  l4_utcb_t *utcb,
00105                  l4vcpu_event_hndl_t do_event_work_cb,
```



```

00106         l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW;
00107
00121 L4_CV L4_INLINE
00122 void
00123 l4vcpu_wait(l4_vcpu_state_t *vcpu, l4_utcb_t *utcb,
00124             l4_timeout_t to,
00125             l4vcpu_event_hndl_t do_event_work_cb,
00126             l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW;
00127
00141 L4_CV L4_INLINE
00142 void
00143 l4vcpu_wait_for_event(l4_vcpu_state_t *vcpu, l4_utcb_t *utcb,
00144                      l4vcpu_event_hndl_t do_event_work_cb,
00145                      l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW;
00146
00147
00155 L4_CV void
00156 l4vcpu_print_state(const l4_vcpu_state_t *vcpu, const char *prefix) L4_NOTHROW;
00157
00161 L4_CV void
00162 l4vcpu_print_state_arch(const l4_vcpu_state_t *vcpu, const char *prefix) L4_NOTHROW;
00163
00164
00173 L4_CV L4_INLINE
00174 int
00175 l4vcpu_is_irq_entry(l4_vcpu_state_t const *vcpu) L4_NOTHROW;
00176
00185 L4_CV L4_INLINE
00186 int
00187 l4vcpu_is_page_fault_entry(l4_vcpu_state_t const *vcpu) L4_NOTHROW;
00188
00200 L4_CV int
00201 l4vcpu_ext_alloc(l4_vcpu_state_t **vcpu, l4_addr_t *ext_state,
00202                 l4_cap_idx_t task, l4_cap_idx_t regmgr) L4_NOTHROW;
00203
00204 /* ===== */
00205 /* Implementations */
00206
00207 #include <l4/sys/ipc.h>
00208 #include <l4/vcpu/vcpu_arch.h>
00209
00210 L4_CV L4_INLINE
00211 void
00212 l4vcpu_irq_disable(l4_vcpu_state_t *vcpu) L4_NOTHROW
00213 {
00214     vcpu->state &= ~L4_VCPU_F_IRQ;
00215     l4_barrier();
00216 }
00217
00218 L4_CV L4_INLINE
00219 unsigned
00220 l4vcpu_irq_disable_save(l4_vcpu_state_t *vcpu) L4_NOTHROW
00221 {
00222     unsigned s = vcpu->state;
00223     l4vcpu_irq_disable(vcpu);
00224     return s;
00225 }
00226
00227 L4_CV L4_INLINE
00228 void
00229 l4vcpu_wait(l4_vcpu_state_t *vcpu, l4_utcb_t *utcb,
00230             l4_timeout_t to,
00231             l4vcpu_event_hndl_t do_event_work_cb,
00232             l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW
00233 {
00234     l4vcpu_irq_disable(vcpu);
00235     setup_ipc(utcb);
00236     vcpu->i.tag = l4_ipc_wait(utcb, &vcpu->i.label, to);
00237     if (L4_LIKELY(!l4_msgtag_has_error(vcpu->i.tag)))
00238         do_event_work_cb(vcpu);
00239 }
00240
00241 L4_CV L4_INLINE
00242 void
00243 l4vcpu_irq_enable(l4_vcpu_state_t *vcpu, l4_utcb_t *utcb,
00244                  l4vcpu_event_hndl_t do_event_work_cb,
00245                  l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW
00246 {
00247     if (!(vcpu->state & L4_VCPU_F_IRQ))
00248     {
00249         setup_ipc(utcb);
00250         l4_barrier();
00251     }
00252     while (1)
00253     {
00254         vcpu->state |= L4_VCPU_F_IRQ;

```

```

00256     l4_barrier();
00257
00258     if (L4_LIKELY(!(vcpu->sticky_flags & L4_VCPU_SF_IRQ_PENDING)))
00259         break;
00260
00261     l4vcpu_wait(vcpu, utcb, L4_IPC_BOTH_TIMEOUT_0,
00262                do_event_work_cb, setup_ipc);
00263 }
00264 }
00265
00266 L4_CV L4_INLINE
00267 void
00268 l4vcpu_irq_restore(l4_vcpu_state_t *vcpu, unsigned s,
00269                  l4_utcb_t *utcb,
00270                  l4vcpu_event_hndl_t do_event_work_cb,
00271                  l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW
00272 {
00273     if (s & L4_VCPU_F_IRQ)
00274         l4vcpu_irq_enable(vcpu, utcb, do_event_work_cb, setup_ipc);
00275     else if (vcpu->state & L4_VCPU_F_IRQ)
00276         l4vcpu_irq_disable(vcpu);
00277 }
00278
00279 L4_CV L4_INLINE
00280 void
00281 l4vcpu_wait_for_event(l4_vcpu_state_t *vcpu, l4_utcb_t *utcb,
00282                     l4vcpu_event_hndl_t do_event_work_cb,
00283                     l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW
00284 {
00285     l4vcpu_wait(vcpu, utcb, L4_IPC_NEVER, do_event_work_cb, setup_ipc);
00286 }
00287
00288 __END_DECLS

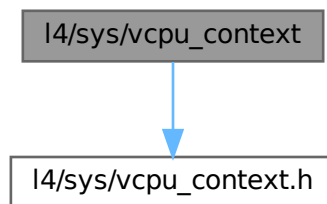
```

## 16.589 l4/sys/vcpu\_context File Reference

Hardware vCPU context interface.

#include <l4/sys/vcpu\_context.h>

Include dependency graph for vcpu\_context:



### Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

### 16.589.1 Detailed Description

Hardware vCPU context interface.

Definition in file [vcpu\\_context](#).

## 16.590 vcpu\_context

[Go to the documentation of this file.](#)

```

00001
00006 #pragma once
00007
00008 #include <l4/sys/vcpu_context.h>
00009
00010 namespace L4 {
00011
00012 class Vcpu_context :
00013     public Kobject_t<Vcpu_context, Kobject, L4_PROTO_VCPU_CONTEXT>
00014 {
00015 public:
00016     Vcpu_context(Vcpu_context const &) = delete;
00017     void operator = (Vcpu_context const &) = delete;
00018
00019 protected:
00020     Vcpu_context();
00021 };
00022
00023 };

```

## 16.591 vcpu\_context.h

```

00001
00007 #pragma once

```

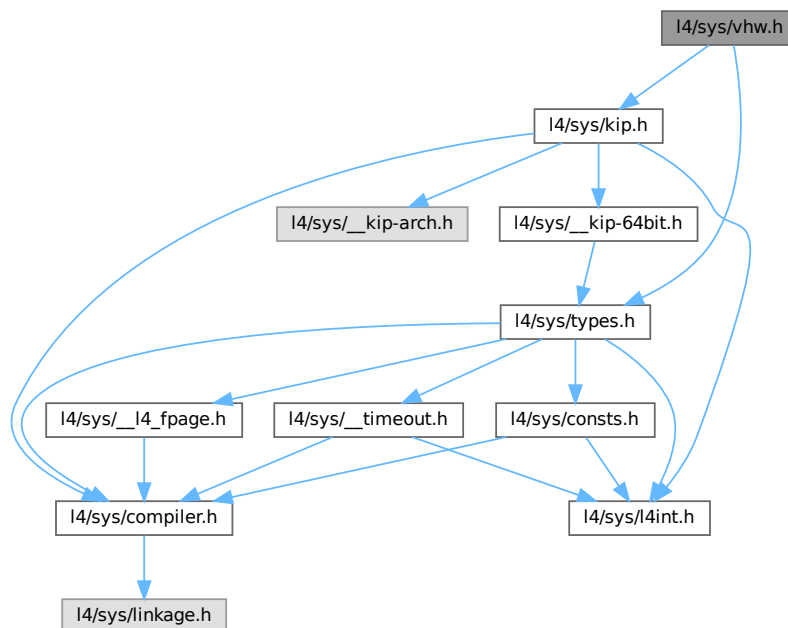
## 16.592 l4/sys/vhw.h File Reference

Descriptors for virtual hardware (under UX).

```
#include <l4/sys/types.h>
```

```
#include <l4/sys/kip.h>
```

Include dependency graph for vhw.h:



## Data Structures

- struct [l4\\_vhw\\_entry](#)  
*Description of a device.*
- struct [l4\\_vhw\\_descriptor](#)  
*Virtual hardware devices description.*

## Enumerations

- enum [l4\\_vhw\\_entry\\_type](#) { [L4\\_TYPE\\_VHW\\_NONE](#), [L4\\_TYPE\\_VHW\\_FRAMEBUFFER](#), [L4\\_TYPE\\_VHW\\_INPUT](#), [L4\\_TYPE\\_VHW\\_NET](#) }  
*Type of device.*

## 16.592.1 Detailed Description

Descriptors for virtual hardware (under UX).

Definition in file [vhw.h](#).

## 16.593 vhw.h

[Go to the documentation of this file.](#)

```
00001 /*****
00007 */
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010 *      economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 /*****
00026 #ifndef _L4_SYS_VHW_H
00027 #define _L4_SYS_VHW_H
00028
00029 #include <l4/sys/types.h>
00030 #include <l4/sys/kip.h>
00031
00044 enum l4_vhw_entry_type {
00045     L4_TYPE_VHW_NONE,
00046     L4_TYPE_VHW_FRAMEBUFFER,
00047     L4_TYPE_VHW_INPUT,
00048     L4_TYPE_VHW_NET,
00049 };
00050
00055 struct l4_vhw_entry {
00056     enum l4_vhw_entry_type type;
00057     l4_uint32_t provider_pid;
00059     l4_addr_t mem_start;
00060     l4_addr_t mem_size;
00062     l4_uint32_t irq_no;
00063     l4_uint32_t fd;
00064 };
00065
00070 struct l4_vhw_descriptor {
00071     l4_uint32_t magic;
00072     l4_uint8_t version;
```

```

00073  l4_uint8_t  count;
00074  l4_uint8_t  pad1;
00075  l4_uint8_t  pad2;
00077  struct l4_vhw_entry descs[];
00078  };
00079
00080  enum {
00081      L4_VHW_MAGIC = 0x56687765,
00082  };
00083
00084  static inline struct l4_vhw_descriptor *
00085  l4_vhw_get(l4_kernel_info_t const *kip) L4_NOTHROW
00086  {
00087      struct l4_vhw_descriptor *v
00088          = (struct l4_vhw_descriptor *)(((unsigned long)kip) + kip->vhw_offset);
00089
00090      if (v->magic == L4_VHW_MAGIC)
00091          return v;
00092
00093      return NULL;
00094  }
00095
00096  static inline struct l4_vhw_entry *
00097  l4_vhw_get_entry(struct l4_vhw_descriptor *v, int entry) L4_NOTHROW
00098  {
00099      return v->descs + entry;
00100  }
00101
00102  static inline struct l4_vhw_entry *
00103  l4_vhw_get_entry_type(struct l4_vhw_descriptor *v, enum l4_vhw_entry_type t) L4_NOTHROW
00104  {
00105      int i;
00106      struct l4_vhw_entry *e = v->descs;
00107
00108      for (i = 0; i < v->count; i++, e++)
00109          if (e->type == t)
00110              return e;
00111
00112      return NULL;
00113  }
00114
00115  #endif /* ! _L4_SYS_VHW_H */

```

## 16.594 vm

```

00001  // vi:set ft=cpp: -*- Mode: C++ -*-
00002  /*
00003   * (c) 2018 Adam Lackorzynski <adam@l4re.org>
00004   *
00005   * This file is part of L4Re and distributed under the terms of the
00006   * GNU General Public License 2.
00007   * Please see the COPYING-GPL-2 file for details.
00008   *
00009   * As a special exception, you may use this file as part of a free software
00010   * library without restriction. Specifically, if other files instantiate
00011   * templates or use macros or inline functions from this file, or you compile
00012   * this file and link it with other files to produce an executable, this
00013   * file does not by itself cause the resulting executable to be covered by
00014   * the GNU General Public License. This exception does not however
00015   * invalidate any other reasons why the executable file might be covered by
00016   * the GNU General Public License.
00017   */
00018  #pragma once
00019
00020  #include <l4/sys/__vm-arm.h>

```

## 16.595 l4/sys/vm File Reference

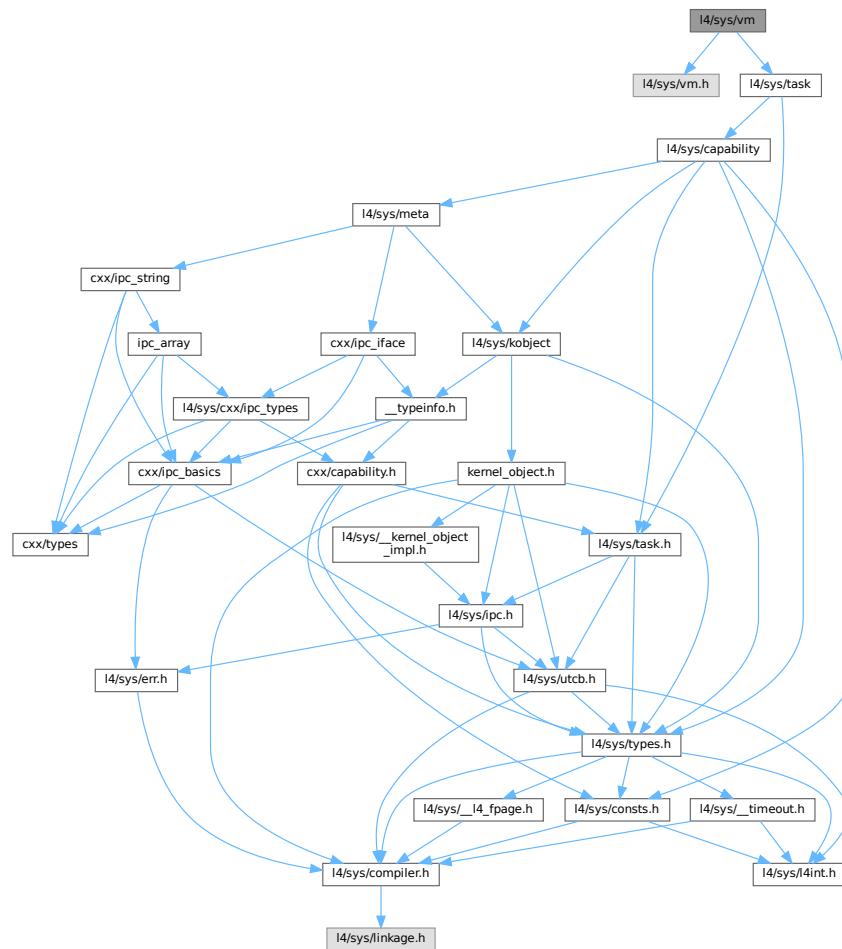
Virtualization interface.

```

#include <l4/sys/vm.h>
#include <l4/sys/task>

```

Include dependency graph for `vm`:



## Data Structures

- class [L4::Vm](#)  
*Virtual machine host address space.*

## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

## 16.595.1 Detailed Description

Virtualization interface.

Definition in file [vm](#).

## 16.596 vm

[Go to the documentation of this file.](#)

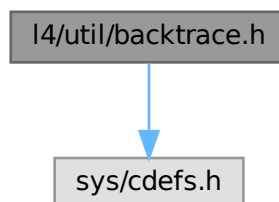
```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include <l4/sys/vm.h>
00028 #include <l4/sys/task>
00029
00030 namespace L4 {
00031
00041 class Vm : public Kobject_t<Vm, Task, L4_PROTO_VM>
00042 {
00043 protected:
00044     Vm();
00045
00046 private:
00047     Vm(Vm const &);
00048     void operator = (Vm const &);
00049 };
00050
00051 };
```

## 16.597 l4/util/backtrace.h File Reference

Backtrace.

```
#include <sys/cdefs.h>
```

Include dependency graph for backtrace.h:







```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU Lesser General Public License 2.1.
00011  * Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013 #pragma once
00014
00015 #include <sys/cdefs.h>
00016
00017 __BEGIN_DECLS
00018
00026 int l4util_backtrace(void **pc_array, int max_len);
00027
00028 __END_DECLS

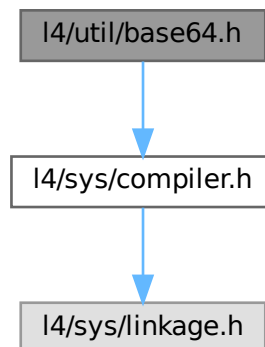
```

## 16.599 l4/util/base64.h File Reference

base 64 encoding and decoding functions adapted from Bob Trower 08/04/01

```
#include <l4/sys/compiler.h>
```

Include dependency graph for base64.h:



### Functions

- void **base64\_encode** (const char \*infile, unsigned int in\_size, char \*\*outfile)  
*base-64-encode string infile*
- void **base64\_decode** (const char \*infile, unsigned int in\_size, char \*\*outfile)  
*decode base-64-encoded string infile*

### 16.599.1 Detailed Description

base 64 encoding and decoding functions adapted from Bob Trower 08/04/01

**Date**

04/26/2002

**Author**Joerg Nothnagel [jn6@os.inf.tu-dresden.de](mailto:jn6@os.inf.tu-dresden.de)Definition in file [base64.h](#).

## 16.600 base64.h

[Go to the documentation of this file.](#)

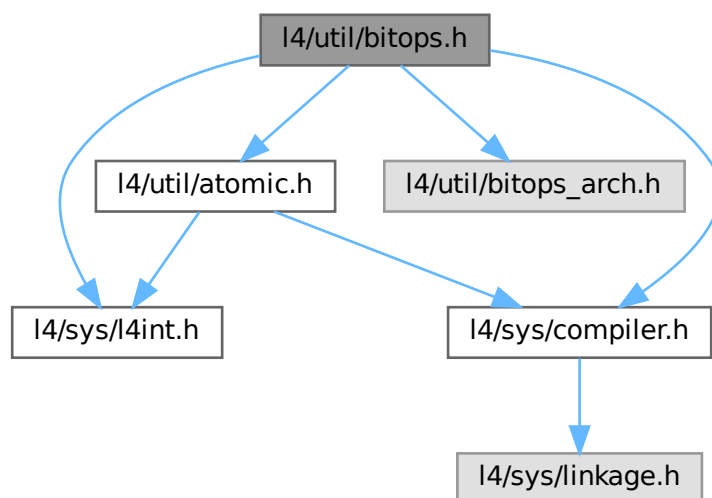
```
00001
00010 /*
00011  * (c) 2008-2009 Author(s)
00012  *     economic rights: Technische Universität Dresden (Germany)
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU Lesser General Public License 2.1.
00015  * Please see the COPYING-LGPL-2.1 file for details.
00016  */
00017
00018 #ifndef B64_EN_DECODE
00019 #define B64_EN_DECODE
00020
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00041 L4_CV void base64_encode( const char *infile, unsigned int in_size, char **outfile);
00042
00053 L4_CV void base64_decode(const char *infile, unsigned int in_size, char **outfile);
00054
00055 EXTERN_C_END
00056
00058 #endif //B64_EN_DECODE
```

## 16.601 l4/util/bitops.h File Reference

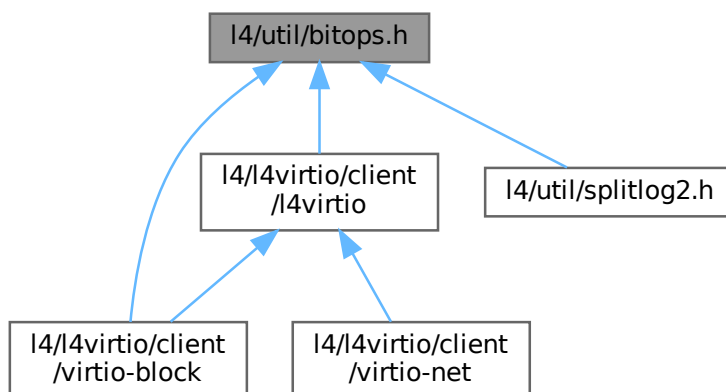
bit manipulation functions

```
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
#include <l4/util/bitops_arch.h>
#include <l4/util/atomic.h>
```

Include dependency graph for bitops.h:



This graph shows which files directly or indirectly include this file:



## Macros

- `#define l4util_test_and_clear_bit(b, dest) l4util_btr(b, dest)`  
*define some more usual names*

## Functions

- void [l4util\\_set\\_bit](#) (int b, volatile [l4\\_umword\\_t](#) \*dest)  
*Set bit in memory.*
- void [l4util\\_clear\\_bit](#) (int b, volatile [l4\\_umword\\_t](#) \*dest)  
*Clear bit in memory.*
- void [l4util\\_complement\\_bit](#) (int b, volatile [l4\\_umword\\_t](#) \*dest)  
*Complement bit in memory.*
- int [l4util\\_test\\_bit](#) (int b, const volatile [l4\\_umword\\_t](#) \*dest)  
*Test bit (return value of bit)*
- int [l4util\\_bts](#) (int b, volatile [l4\\_umword\\_t](#) \*dest)  
*Bit test and set.*
- int [l4util\\_btr](#) (int b, volatile [l4\\_umword\\_t](#) \*dest)  
*Bit test and reset.*
- int [l4util\\_btc](#) (int b, volatile [l4\\_umword\\_t](#) \*dest)  
*Bit test and complement.*
- int [l4util\\_bsr](#) ([l4\\_umword\\_t](#) word)  
*Bit scan reverse.*
- int [l4util\\_bsf](#) ([l4\\_umword\\_t](#) word)  
*Bit scan forward.*
- int [l4util\\_find\\_first\\_set\\_bit](#) (const void \*dest, [l4\\_size\\_t](#) size)  
*Find the first set bit in a memory region.*
- int [l4util\\_find\\_first\\_zero\\_bit](#) (const void \*dest, [l4\\_size\\_t](#) size)  
*Find the first zero bit in a memory region.*
- int [l4util\\_next\\_power2](#) (unsigned long val)  
*Find the next power of 2 for a given number.*

### 16.601.1 Detailed Description

bit manipulation functions

#### Date

07/03/2001

#### Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de)

Definition in file [bitops.h](#).

## 16.602 bitops.h

[Go to the documentation of this file.](#)

```

00001 /*****
00009 */
00010 * (c) 2000-2009 Author(s)
00011 *     economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 /*****
00018 #ifndef __L4UTIL__INCLUDE__BITOPS_H__
00019 #define __L4UTIL__INCLUDE__BITOPS_H__
00020
00021 /* L4 includes */
00022 #include <l4/sys/l4int.h>
00023 #include <l4/sys/compiler.h>
00024
00026 #define l4util_test_and_clear_bit(b, dest)  l4util_btr(b, dest)
00027 #define l4util_test_and_set_bit(b, dest)    l4util_bts(b, dest)
00028 #define l4util_test_and_change_bit(b, dest) l4util_btc(b, dest)
00029 #define l4util_log2(word)                  l4util_bsr(word)
00030
00031 /*****
00032 *** Prototypes
00033 *****/
00034
00035 EXTERN_C_BEGIN
00036
00049 L4_INLINE void
00050 l4util_set_bit(int b, volatile l4_umword_t * dest);
00051
00059 L4_INLINE void
00060 l4util_clear_bit(int b, volatile l4_umword_t * dest);
00061
00069 L4_INLINE void
00070 l4util_complement_bit(int b, volatile l4_umword_t * dest);
00071
00081 L4_INLINE int
00082 l4util_test_bit(int b, const volatile l4_umword_t * dest);
00083
00095 L4_INLINE int
00096 l4util_bts(int b, volatile l4_umword_t * dest);
00097
00109 L4_INLINE int
00110 l4util_btr(int b, volatile l4_umword_t * dest);
00111
00123 L4_INLINE int
00124 l4util_btc(int b, volatile l4_umword_t * dest);
00125
00137 L4_INLINE int
00138 l4util_bsr(l4_umword_t word);
00139
00151 L4_INLINE int
00152 l4util_bsf(l4_umword_t word);
00153
00165 L4_INLINE int
00166 l4util_find_first_set_bit(const void * dest, l4_size_t size);
00167
00179 L4_INLINE int
00180 l4util_find_first_zero_bit(const void * dest, l4_size_t size);
00181
00182
00191 L4_INLINE int
00192 l4util_next_power2(unsigned long val);
00193
00194 EXTERN_C_END
00195
00196 /*****
00197 *** Implementation of specific version
00198 *****/
00199
00200 #include <l4/util/bitops_arch.h>
00201
00202 /*****
00203 *** Generic implementations
00204 *****/
00205
00206 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_SET_BIT
00207 #include <l4/util/atomic.h>
00208 L4_INLINE void
00209 l4util_set_bit(int b, volatile l4_umword_t * dest)
00210 {

```

```

00211     l4_umword_t oldval, newval;
00212
00213     dest += b / (sizeof(*dest) * 8); /* advance dest to the proper element */
00214     b     &= sizeof(*dest) * 8 - 1; /* modulo; cut off all upper bits */
00215
00216     do
00217     {
00218         oldval = *dest;
00219         newval = oldval | (1UL < b);
00220     }
00221     while (!l4util_cmpxchg(dest, oldval, newval));
00222 }
00223 #endif
00224
00225 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
00226 #include <l4/util/atomic.h>
00227 L4_INLINE void
00228 l4util_clear_bit(int b, volatile l4_umword_t * dest)
00229 {
00230     l4_umword_t oldval, newval;
00231
00232     dest += b / (sizeof(*dest) * 8);
00233     b     &= sizeof(*dest) * 8 - 1;
00234
00235     do
00236     {
00237         oldval = *dest;
00238         newval = oldval & ~(1UL < b);
00239     }
00240     while (!l4util_cmpxchg(dest, oldval, newval));
00241 }
00242 #endif
00243
00244 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
00245 L4_INLINE int
00246 l4util_test_bit(int b, const volatile l4_umword_t * dest)
00247 {
00248     dest += b / (sizeof(*dest) * 8);
00249     b     &= sizeof(*dest) * 8 - 1;
00250
00251     return (*dest >> b) & 1;
00252 }
00253 #endif
00254
00255 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
00256 #include <l4/util/atomic.h>
00257 L4_INLINE int
00258 l4util_bts(int b, volatile l4_umword_t * dest)
00259 {
00260     l4_umword_t oldval, newval;
00261
00262     dest += b / (sizeof(*dest) * 8);
00263     b     &= sizeof(*dest) * 8 - 1;
00264
00265     do
00266     {
00267         oldval = *dest;
00268         newval = oldval | (1UL < b);
00269     }
00270     while (!l4util_cmpxchg(dest, oldval, newval));
00271
00272     /* Return old bit */
00273     return (oldval >> b) & 1;
00274 }
00275 #endif
00276
00277 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
00278 #include <l4/util/atomic.h>
00279 L4_INLINE int
00280 l4util_btr(int b, volatile l4_umword_t * dest)
00281 {
00282     l4_umword_t oldval, newval;
00283
00284     dest += b / (sizeof(*dest) * 8);
00285     b     &= sizeof(*dest) * 8 - 1;
00286
00287     do
00288     {
00289         oldval = *dest;
00290         newval = oldval & ~(1UL < b);
00291     }
00292     while (!l4util_cmpxchg(dest, oldval, newval));
00293
00294     /* Return old bit */
00295     return (oldval >> b) & 1;
00296 }
00297 #endif

```

```

00298
00299 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
00300 L4_INLINE int
00301 l4util_bsr(l4_umword_t word)
00302 {
00303     int i;
00304     if (!word)
00305         return -1;
00306     for (i = 8 * sizeof(word) - 1; i >= 0; i--)
00307         if ((1UL << i) & word)
00308             return i;
00309     __builtin_unreachable();
00310 }
00311 #endif
00312
00313 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
00314 L4_INLINE int
00315 l4util_bsf(l4_umword_t word)
00316 {
00317     unsigned int i;
00318     if (!word)
00319         return -1;
00320     for (i = 0; i < sizeof(word) * 8; i++)
00321         if ((1UL << i) & word)
00322             return i;
00323     __builtin_unreachable();
00324 }
00325 #endif
00326
00327 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
00328 L4_INLINE int
00329 l4util_find_first_zero_bit(const void * dest, l4_size_t size)
00330 {
00331     l4_size_t i, j;
00332     unsigned long *v = (unsigned long*)dest;
00333     if (!size)
00334         return 0;
00335     size = (size + 31) & ~0x1f; /* Grmbl: adapt to x86 implementation... */
00336     for (i = j = 0; i < size; i++, j++)
00337     {
00338         if (j >= sizeof(*v) * 8)
00339         {
00340             j = 0;
00341             v++;
00342         }
00343         if (!(1UL << j) & *v)
00344             return i;
00345     }
00346     return size + 1;
00347 }
00348 #endif
00349
00350 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
00351 L4_INLINE void
00352 l4util_complement_bit(int b, volatile l4_umword_t * dest)
00353 {
00354     dest += b / (sizeof(*dest) * 8);
00355     b &= sizeof(*dest) * 8 - 1;
00356     *dest ^= 1UL << b;
00357 }
00358 #endif
00359
00360 /*
00361  * Adapted from:
00362  * http://en.wikipedia.org/wiki/Power_of_two#Algorithm_to_find_the_next-highest_power_of_two
00363  */
00364 L4_INLINE int
00365 l4util_next_power2(unsigned long val)
00366 {
00367     unsigned i;
00368     if (val == 0)
00369         return 1;
00370     val--;
00371     for (i=1; i < sizeof(unsigned long)*8; i<=1)
00372         val = val | val >> i;

```

```

00385
00386     return val+1;
00387 }
00388
00389
00390 /* Non-implemented version, catch with a linker warning */
00391
00392 extern int __this_l4util_bitops_function_is_not_implemented_for_this_arch__sorry(void);
00393
00394 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
00395 L4_INLINE int
00396 l4util_btc(int b, volatile l4_umword_t * dest)
00397 { (void)b; (void)dest; __this_l4util_bitops_function_is_not_implemented_for_this_arch__sorry(); return
0; }
00398 #endif
00399
00400 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
00401 L4_INLINE int
00402 l4util_find_first_set_bit(const void * dest, l4_size_t size)
00403 { (void)dest; (void)size; __this_l4util_bitops_function_is_not_implemented_for_this_arch__sorry();
return 0; }
00404 #endif
00405
00406 #endif /* ! __L4UTIL__INCLUDE__BITOPS_H__ */

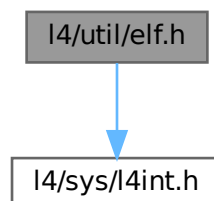
```

## 16.603 l4/util/elf.h File Reference

ELF definition.

```
#include <l4/sys/l4int.h>
```

Include dependency graph for elf.h:



### Data Structures

- struct [Elf32\\_Ehdr](#)  
*ELF32 header.*
- struct [Elf64\\_Ehdr](#)  
*ELF64 header.*
- struct [Elf32\\_Shdr](#)  
*ELF32 section header.*
- struct [Elf64\\_Shdr](#)  
*ELF64 section header.*
- struct [Elf32\\_Phdr](#)  
*ELF32 program header.*
- struct [Elf64\\_Phdr](#)



- *ELF64 program header.*
- struct [Elf32\\_Dyn](#)
  - *ELF32 dynamic entry.*
- struct [Elf64\\_Dyn](#)
  - *ELF64 dynamic entry.*
- struct [Elf32\\_Rel](#)
  - *ELF32 relocation entry w/o addend.*
- struct [Elf32\\_Rela](#)
  - *ELF32 relocation entry w/ addend.*
- struct [Elf64\\_Rel](#)
  - *ELF64 relocation entry w/o addend.*
- struct [Elf64\\_Rela](#)
  - *ELF64 relocation entry w/ addend.*
- struct [Elf32\\_Sym](#)
  - *ELF32 symbol table entry.*
- struct [Elf64\\_Sym](#)
  - *ELF64 symbol table entry.*
- struct [Elf32\\_Auxv](#)
  - *Auxiliary vector (32-bit).*
- struct [Elf64\\_Auxv](#)
  - *Auxiliary vector (64-bit).*

## Macros

- #define **ElfW**(type) \_ElfW(Elf, 32, type)
  - *Use 64 or 32 bits types depending on the target architecture.*
- #define **ELF32\_R\_SYM**(i) ((i)>>8)
  - *Symbol table index.*
- #define **ELF32\_R\_TYPE**(i) ((unsigned char)(i))
- #define **ELF32\_R\_INFO**(s, t) (((s)<<8)+(unsigned char)(t))
  - *Create info from symbol table index + type.*
- #define **ELF64\_R\_SYM**(i) ((i)>>32)
  - *Symbol table index.*
- #define **ELF64\_R\_TYPE**(i) ((i)&0xffffffffL)
- #define **ELF64\_R\_INFO**(s, t) (((s)<<32)+(t)&0xffffffffL)
  - *Create info from symbol table index + type.*
- #define **ELF32\_ST\_BIND**(i) ((i)>>4)
- #define **ELF32\_ST\_TYPE**(i) ((i)&0xf)
- #define **ELF32\_ST\_INFO**(b, t) (((b)<<4)+((t)&0xf))
  - *Make info from bind + type.*
- #define **ELF64\_ST\_BIND**(i) ((i)>>4)
- #define **ELF64\_ST\_TYPE**(i) ((i)&0xf)
- #define **ELF64\_ST\_INFO**(b, t) (((b)<<4)+((t)&0xf))
  - *Make info from bind + type.*

## Typedefs

- typedef struct [Elf32\\_Auxv](#) **Elf32\_Auxv**  
*Auxiliary vector (32-bit).*
- typedef struct [Elf64\\_Auxv](#) **Elf64\_Auxv**  
*Auxiliary vector (64-bit).*

## ELF types

- typedef [l4\\_uint32\\_t](#) **Elf32\_Addr**  
*size 4 align 4*
- typedef [l4\\_uint32\\_t](#) **Elf32\_Off**  
*size 4 align 4*
- typedef [l4\\_uint16\\_t](#) **Elf32\_Half**  
*size 2 align 2*
- typedef [l4\\_uint32\\_t](#) **Elf32\_Word**  
*size 4 align 4*
- typedef [l4\\_int32\\_t](#) **Elf32\_Sword**  
*size 4 align 4*
- typedef [l4\\_uint64\\_t](#) **Elf64\_Addr**  
*size 8 align 8*
- typedef [l4\\_uint64\\_t](#) **Elf64\_Off**  
*size 8 align 8*
- typedef [l4\\_uint16\\_t](#) **Elf64\_Half**  
*size 2 align 2*
- typedef [l4\\_uint32\\_t](#) **Elf64\_Word**  
*size 4 align 4*
- typedef [l4\\_int32\\_t](#) **Elf64\_Sword**  
*size 4 align 4*
- typedef [l4\\_uint64\\_t](#) **Elf64\_Xword**  
*size 8 align 8*
- typedef [l4\\_int64\\_t](#) **Elf64\_Sxword**  
*size 8 align 8*

## Enumerations

- enum { [EI\\_NIDENT](#) = 16 }
- enum [Elf\\_ETs](#) {  
[ET\\_NONE](#) = 0 , [ET\\_REL](#) = 1 , [ET\\_EXEC](#) = 2 , [ET\\_DYN](#) = 3 ,  
[ET\\_CORE](#) = 4 , [ET\\_LOPROC](#) = 0xff00 , [ET\\_HIPROC](#) = 0xffff }
- enum [Elf\\_EMs](#) {  
[EM\\_NONE](#) = 0 , [EM\\_M32](#) = 1 , [EM\\_SPARC](#) = 2 , [EM\\_386](#) = 3 ,  
[EM\\_68K](#) = 4 , [EM\\_88K](#) = 5 , [EM\\_860](#) = 7 , [EM\\_MIPS](#) = 8 ,  
[EM\\_MIPS\\_RS4\\_BE](#) = 10 , [EM\\_SPARC64](#) = 11 , [EM\\_PARISC](#) = 15 , [EM\\_VPP500](#) = 17 ,  
[EM\\_SPARC32PLUS](#) = 18 , [EM\\_960](#) = 19 , [EM\\_PPC](#) = 20 , [EM\\_V800](#) = 36 ,  
[EM\\_FR20](#) = 37 , [EM\\_RH32](#) = 38 , [EM\\_RCE](#) = 39 , [EM\\_ARM](#) = 40 ,  
[EM\\_ALPHA](#) = 41 , [EM\\_SH](#) = 42 , [EM\\_SPARCV9](#) = 43 , [EM\\_TRICORE](#) = 44 ,  
[EM\\_ARC](#) = 45 , [EM\\_H8\\_300](#) = 46 , [EM\\_H8\\_300H](#) = 47 , [EM\\_H8S](#) = 48 ,  
[EM\\_H8\\_500](#) = 49 , [EM\\_IA\\_64](#) = 50 , [EM\\_MIPS\\_X](#) = 51 , [EM\\_COLDFIRE](#) = 52 ,  
[EM\\_68HC12](#) = 53 , [EM\\_X86\\_64](#) = 62 , [EM\\_PDSP](#) = 63 , [EM\\_FX66](#) = 66 ,  
[EM\\_ST9PLUS](#) = 67 , [EM\\_ST7](#) = 68 , [EM\\_68HC16](#) = 69 , [EM\\_68HC11](#) = 70 ,  
[EM\\_68HC08](#) = 71 , [EM\\_68HC05](#) = 72 , [EM\\_SVX](#) = 73 , [EM\\_ST19](#) = 74 ,  
[EM\\_VAX](#) = 75 , [EM\\_CRIS](#) = 76 , [EM\\_JAVELIN](#) = 77 , [EM\\_FIREPATH](#) = 78 ,  
[EM\\_ZSP](#) = 79 , [EM\\_MMIX](#) = 80 , [EM\\_HUANY](#) = 81 , [EM\\_PRISM](#) = 82 ,  
[EM\\_AVR](#) = 83 , [EM\\_FR30](#) = 84 , [EM\\_D10V](#) = 85 , [EM\\_D30V](#) = 86 ,  
[EM\\_V850](#) = 87 , [EM\\_M32R](#) = 88 , [EM\\_MN10300](#) = 89 , [EM\\_MN10200](#) = 90 ,  
[EM\\_PJ](#) = 91 , [EM\\_OPENRISC](#) = 92 , [EM\\_ARC\\_A5](#) = 93 , [EM\\_XTENSA](#) = 94 ,  
[EM\\_ALTERA\\_NIOS2](#) = 113 , [EM\\_AARCH64](#) = 183 , [EM\\_TILEPRO](#) = 188 , [EM\\_MICROBLAZE](#) = 189 ,  
[EM\\_TILEGX](#) = 191 , [EM\\_RISCV](#) = 243 , [EM\\_NUM](#) = 244 }

*Required architecture.*

- enum `Elf_EVs` { `EV_NONE` = 0 , `EV_CURRENT` = 1 }

*Object file version.*

- enum `Elf_EIs` {  
`EI_MAG0` = 0 , `EI_MAG1` = 1 , `EI_MAG2` = 2 , `EI_MAG3` = 3 ,  
`EI_CLASS` = 4 , `EI_DATA` = 5 , `EI_VERSION` = 6 , `EI_OSABI` = 7 ,  
`EI_ABIVERSION` = 8 , `EI_PAD` = 9 }

*Identification Indices.*

- enum `Elf_MAGs` { `ELFMAG0` = 0x7f , `ELFMAG1` = 'E' , `ELFMAG2` = 'L' , `ELFMAG3` = 'F' }

*Magic number.*

- enum `Elf_Classes` { `ELFCLASSNONE` = 0 , `ELFCLASS32` = 1 , `ELFCLASS64` = 2 , `ELFCLASSNUM` = 3 }

*File class or capacity.*

- enum `Elf_DATAs` { `ELFDATANONE` = 0 , `ELFDATA2LSB` = 1 , `ELFDATA2MSB` = 2 , `ELFDATANUM` = 3 }

*Data encoding.*

- enum `Elf_OSABIs` {  
`ELFOSABI_NONE` = 0 , `ELFOSABI_SYSV` = 0 , `ELFOSABI_HPUX` = 1 , `ELFOSABI_NETBSD` = 2 ,  
`ELFOSABI_LINUX` = 3 , `ELFOSABI_SOLARIS` = 6 , `ELFOSABI_AIX` = 7 , `ELFOSABI_IRIX` = 8 ,  
`ELFOSABI_FREEBSD` = 9 , `ELFOSABI_TRU64` = 10 , `ELFOSABI_MODESTO` = 11 , `ELFOSABI_OPENBSD`  
= 12 ,  
`ELFOSABI_ARM` = 97 , `ELFOSABI_STANDALONE` = 255 }

*Identify operating system and ABI to which the object is targeted.*

- enum `Elf_SHNs` {  
`SHN_UNDEF` = 0 , `SHN_LORESERVE` = 0xff00 , `SHN_LOPROC` = 0xff00 , `SHN_HIPROC` = 0xff1f ,  
`SHN_ABS` = 0xffff , `SHN_COMMON` = 0xffff2 , `SHN_HIRESERVE` = 0xffff }

*Special section indexes.*

- enum `Elf_SHTs` {  
`SHT_NULL` = 0 , `SHT_PROGBITS` = 1 , `SHT_SYMTAB` = 2 , `SHT_STRTAB` = 3 ,  
`SHT_RELA` = 4 , `SHT_HASH` = 5 , `SHT_DYNAMIC` = 6 , `SHT_NOTE` = 7 ,  
`SHT_NOBITS` = 8 , `SHT_REL` = 9 , `SHT_SHLIB` = 10 , `SHT_DYNSYM` = 11 ,  
`SHT_INIT_ARRAY` = 14 , `SHT_FINI_ARRAY` = 15 , `SHT_PREINIT_ARRAY` = 16 , `SHT_GROUP` = 17 ,  
`SHT_SYMTAB_SHNDX` = 18 , `SHT_NUM` = 19 , `SHT_LOOS` = 0x60000000 , `SHT_HIOS` = 0x6fffffff ,  
`SHT_LOPROC` = 0x70000000 , `SHT_HIPROC` = 0x7fffffff , `SHT_LOUSER` = 0x80000000 , `SHT_HIUSER` =  
0xffffffff }

*Section type.*

- enum `Elf_SHFs` {  
`SHF_WRITE` = 0x1 , `SHF_ALLOC` = 0x2 , `SHF_EXECINSTR` = 0x4 , `SHF_MERGE` = 0x10 ,  
`SHF_STRINGS` = 0x20 , `SHF_INFO_LINK` = 0x40 , `SHF_LINK_ORDER` = 0x80 , `SHF_OS_NONCONFORMING`  
= 0x100 ,  
`SHF_GROUP` = 0x200 , `SHF_TLS` = 0x400 , `SHF_MASKOS` = 0x0ff00000 , `SHF_MASKPROC` = 0xf0000000  
}

*Section attribute flags.*

- enum `Elf_PTs` {  
`PT_NULL` = 0 , `PT_LOAD` = 1 , `PT_DYNAMIC` = 2 , `PT_INTERP` = 3 ,  
`PT_NOTE` = 4 , `PT_SHLIB` = 5 , `PT_PHDR` = 6 , `PT_TLS` = 7 ,  
`PT_NUM` = 8 , `PT_LOOS` = 0x60000000 , `PT_HIOS` = 0x6fffffff , `PT_LOPROC` = 0x70000000 ,  
`PT_HIPROC` = 0x7fffffff , `PT_GNU_EH_FRAME` = `PT_LOOS` + 0x474e550 , `PT_GNU_STACK` = `PT_LOOS`  
+ 0x474e551 , `PT_GNU_RELRO` = `PT_LOOS` + 0x474e552 ,  
`PT_L4_STACK` = `PT_LOOS` + 0x12 , `PT_L4_KIP` = `PT_LOOS` + 0x13 , `PT_L4_AUX` = `PT_LOOS` + 0x14 }

*Segment types.*

- enum `Elf_PFs` {  
`PF_X` = 0x1 , `PF_W` = 0x2 , `PF_R` = 0x4 , `PF_MASKOS` = 0x0ff00000 ,  
`PF_MASKPROC` = 0x7fffffff }

*Segment permissions.*

- enum `Elf_NTs_core` {  
`NT_PRSTATUS` = 1 , `NT_FPREGSET` = 2 , `NT_PRPSINFO` = 3 , `NT_PRXREG` = 4 ,  
`NT_TASKSTRUCT` = 4 , `NT_PLATFORM` = 5 , `NT_AUXV` = 6 , `NT_GWINDOWS` = 7 ,  
`NT_ASRS` = 8 , `NT_PSTATUS` = 10 , `NT_PSINFO` = 13 , `NT_PRCRED` = 14 ,  
`NT_UTSNAME` = 15 , `NT_LWPSTATUS` = 16 , `NT_LWPSINFO` = 17 , `NT_PRFPXREG` = 20 }  
*Legal values for note segment descriptor types for core files.*
- enum `Elf_NTs_obj` { `NT_VERSION` = 1 }  
*Legal values for the note segment descriptor types for object files.*
- enum `Elf_DTs` {  
`DT_NULL` = 0 , `DT_NEEDED` = 1 , `DT_PLTRELSZ` = 2 , `DT_PLTGOT` = 3 ,  
`DT_HASH` = 4 , `DT_STRTAB` = 5 , `DT_SYMTAB` = 6 , `DT_RELA` = 7 ,  
`DT_RELASZ` = 8 , `DT_RELAENT` = 9 , `DT_STRSZ` = 10 , `DT_SYMENT` = 11 ,  
`DT_INIT` = 12 , `DT_FINI` = 13 , `DT_SONAME` = 14 , `DT_RPATH` = 15 ,  
`DT_SYMBOLIC` = 16 , `DT_REL` = 17 , `DT_RELSZ` = 18 , `DT_RELENT` = 19 ,  
`DT_PTRREL` = 20 , `DT_DEBUG` = 21 , `DT_TEXTREL` = 22 , `DT_JMPREL` = 23 ,  
`DT_BIND_NOW` = 24 , `DT_INIT_ARRAY` = 25 , `DT_FINI_ARRAY` = 26 , `DT_INIT_ARRAYSZ` = 27 ,  
`DT_FINI_ARRAYSZ` = 28 , `DT_RUNPATH` = 29 , `DT_FLAGS` = 30 , `DT_ENCODING` = 32 ,  
`DT_PREINIT_ARRAY` = 32 , `DT_PREINIT_ARRAYSZ` = 33 , `DT_NUM` = 34 , `DT_LOOS` = 0x6000000d ,  
`DT_HIOS` = 0x6ffff000 , `DT_LOPROC` = 0x70000000 , `DT_HIPROC` = 0x7fffffff }  
*Dynamic Array Tags.*
- enum `Elf_DFs` {  
`DF_ORIGIN` = 0x00000001 , `DF_SYMBOLIC` = 0x00000002 , `DF_TEXTREL` = 0x00000004 ,  
`DF_BIND_NOW` = 0x00000008 ,  
`DF_STATIC_TLS` = 0x00000010 }  
*Values of Elf32\_Dyn.d\_un.d\_val, Elf64\_Dyn.d\_un.d\_val in the DT\_FLAGS entry.*
- enum `Elf_DF_1s` {  
`DF_1_NOW` = 0x00000001 , `DF_1_GLOBAL` = 0x00000002 , `DF_1_GROUP` = 0x00000004 ,  
`DF_1_NODELETE` = 0x00000008 ,  
`DF_1_LOADFLTR` = 0x00000010 , `DF_1_INITFIRST` = 0x00000020 , `DF_1_NOOPEN` = 0x00000040 ,  
`DF_1_ORIGIN` = 0x00000080 ,  
`DF_1_DIRECT` = 0x00000100 , `DF_1_TRANS` = 0x00000200 , `DF_1_INTERPOSE` = 0x00000400 ,  
`DF_1_NODEFLIB` = 0x00000800 ,  
`DF_1_NODUMP` = 0x00001000 , `DF_1_CONFALT` = 0x00002000 , `DF_1_ENDFILTEE` = 0x00004000 ,  
`DF_1_DISPRELDNE` = 0x00008000 ,  
`DF_1_DISPRELPND` = 0x00010000 }  
*State flags selectable in the Elf32\_Dyn.d\_un.d\_val / Elf64\_Dyn.d\_un.d\_val element of the DT\_FLAGS\_1 entry in the dynamic section.*
- enum `Elf_DTF_1s`  
*Flags for the feature selection in DT\_FEATURE\_1.*
- enum `Elf_DF_P1s` { `DF_P1_LAZYLOAD` = 0x00000001 , `DF_P1_GROUPPERM` = 0x00000002 }  
*Flags in the DT\_POSFLAG\_1 entry effecting only the next DT\_\* entry.*
- enum `Elf_R_386_s` {  
`R_386_NONE` = 0 , `R_386_32` = 1 , `R_386_PC32` = 2 , `R_386_GOT32` = 3 ,  
`R_386_PLT32` = 4 , `R_386_COPY` = 5 , `R_386_GLOB_DAT` = 6 , `R_386_JMP_SLOT` = 7 ,  
`R_386_RELATIVE` = 8 , `R_386_GOTOFF` = 9 , `R_386_GOTPC` = 10 , `R_386_32PLT` = 11 ,  
`R_386_TLS_TPOFF` = 14 , `R_386_TLS_IE` = 15 , `R_386_TLS_GOTIE` = 16 , `R_386_TLS_LE` = 17 ,  
`R_386_TLS_GD` = 18 , `R_386_TLS_LDM` = 19 , `R_386_16` = 20 , `R_386_PC16` = 21 ,  
`R_386_8` = 22 , `R_386_PC8` = 23 , `R_386_TLS_GD_32` = 24 , `R_386_TLS_GD_PUSH` = 25 ,  
`R_386_TLS_GD_CALL` = 26 , `R_386_TLS_GD_POP` = 27 , `R_386_TLS_LDM_32` = 28 , `R_386_TLS_LDM_PUSH`  
= 29 ,  
`R_386_TLS_LDM_CALL` = 30 , `R_386_TLS_LDM_POP` = 31 , `R_386_TLS_LDO_32` = 32 , `R_386_TLS_IE_32`  
= 33 ,  
`R_386_TLS_LE_32` = 34 , `R_386_TLS_DTPMOD32` = 35 , `R_386_TLS_DTPOFF32` = 36 , `R_386_TLS_TPOFF32`  
= 37 ,  
`R_386_NUM` = 38 }  
*Relocation types (processor specific).*

- enum [Elf\\_EF\\_ARM\\_s](#) { }  
*ARM specific declarations.*
- enum [Elf\\_STT\\_ARM\\_s](#)  
*Additional symbol types for Thumb.*
- enum [Elf\\_SHF\\_s\\_ARM](#) { [SHF\\_ARM\\_ENTRYSECT](#) = 0x10000000 , [SHF\\_ARM\\_COMDEF](#) = 0x80000000 }
- enum [Elf\\_ARM\\_SBs](#) { [PF\\_ARM\\_SB](#) = 0x10000000 }
- enum [Elf\\_R\\_ARM\\_s](#) {  
[R\\_ARM\\_NONE](#) = 0 , [R\\_ARM\\_PC24](#) = 1 , [R\\_ARM\\_ABS32](#) = 2 , [R\\_ARM\\_REL32](#) = 3 ,  
[R\\_ARM\\_PC13](#) = 4 , [R\\_ARM\\_ABS16](#) = 5 , [R\\_ARM\\_ABS12](#) = 6 , [R\\_ARM\\_THM\\_ABS5](#) = 7 ,  
[R\\_ARM\\_ABS8](#) = 8 , [R\\_ARM\\_SBREL32](#) = 9 , [R\\_ARM\\_THM\\_PC22](#) = 10 , [R\\_ARM\\_THM\\_PC8](#) = 11 ,  
[R\\_ARM\\_AMP\\_VCALL9](#) = 12 , [R\\_ARM\\_SWI24](#) = 13 , [R\\_ARM\\_THM\\_SWI8](#) = 14 , [R\\_ARM\\_XPC25](#) = 15 ,  
[R\\_ARM\\_THM\\_XPC22](#) = 16 , [R\\_ARM\\_COPY](#) = 20 , [R\\_ARM\\_GLOB\\_DAT](#) = 21 , [R\\_ARM\\_JUMP\\_SLOT](#) = 22 ,  
[R\\_ARM\\_RELATIVE](#) = 23 , [R\\_ARM\\_GOTOFF](#) = 24 , [R\\_ARM\\_GOTPC](#) = 25 , [R\\_ARM\\_GOT32](#) = 26 ,  
[R\\_ARM\\_PLT32](#) = 27 , [R\\_ARM\\_ALU\\_PCREL\\_7\\_0](#) = 32 , [R\\_ARM\\_ALU\\_PCREL\\_15\\_8](#) = 33 , [R\\_ARM\\_↵](#)  
[ALU\\_PCREL\\_23\\_15](#) = 34 ,  
[R\\_ARM\\_LDR\\_SBREL\\_11\\_0](#) = 35 , [R\\_ARM\\_ALU\\_SBREL\\_19\\_12](#) = 36 , [R\\_ARM\\_ALU\\_SBREL\\_27\\_20](#) =  
37 , [R\\_ARM\\_GNU\\_VTENTRY](#) = 100 ,  
[R\\_ARM\\_GNU\\_VTINHERIT](#) = 101 , [R\\_ARM\\_THM\\_PC11](#) = 102 , [R\\_ARM\\_THM\\_PC9](#) = 103 , [R\\_ARM\\_↵](#)  
[RXPC25](#) = 249 ,  
[R\\_ARM\\_RSBREL32](#) = 250 , [R\\_ARM\\_THM\\_RPC22](#) = 251 , [R\\_ARM\\_RREL32](#) = 252 , [R\\_ARM\\_RABS22](#) =  
253 ,  
[R\\_ARM\\_RPC24](#) = 254 , [R\\_ARM\\_RBASE](#) = 255 , [R\\_ARM\\_NUM](#) = 256 }  
*ARM relocations.*
- enum [Elf\\_R\\_AARCH64\\_s](#) { [R\\_AARCH64\\_NONE](#) = 0 , [R\\_AARCH64\\_RELATIVE](#) = 1027 }
- enum [Elf\\_R\\_X86\\_64\\_s](#) {  
[R\\_X86\\_64\\_NONE](#) = 0 , [R\\_X86\\_64\\_64](#) = 1 , [R\\_X86\\_64\\_PC32](#) = 2 , [R\\_X86\\_64\\_GOT32](#) = 3 ,  
[R\\_X86\\_64\\_PLT32](#) = 4 , [R\\_X86\\_64\\_COPY](#) = 5 , [R\\_X86\\_64\\_GLOB\\_DAT](#) = 6 , [R\\_X86\\_64\\_JUMP\\_SLOT](#) = 7 ,  
[R\\_X86\\_64\\_RELATIVE](#) = 8 , [R\\_X86\\_64\\_GOTPCREL](#) = 9 , [R\\_X86\\_64\\_32](#) = 10 , [R\\_X86\\_64\\_32S](#) = 11 ,  
[R\\_X86\\_64\\_16](#) = 12 , [R\\_X86\\_64\\_PC16](#) = 13 , [R\\_X86\\_64\\_8](#) = 14 , [R\\_X86\\_64\\_PC8](#) = 15 ,  
[R\\_X86\\_64\\_DTPMOD64](#) = 16 , [R\\_X86\\_64\\_DTPOFF64](#) = 17 , [R\\_X86\\_64\\_TPOFF64](#) = 18 , [R\\_X86\\_64\\_TLSGD](#)  
= 19 ,  
[R\\_X86\\_64\\_TLSLD](#) = 20 , [R\\_X86\\_64\\_DTPOFF32](#) = 21 , [R\\_X86\\_64\\_GOTTPOFF](#) = 22 , [R\\_X86\\_64\\_TPOFF32](#)  
= 23 ,  
[R\\_X86\\_64\\_NUM](#) = 24 }  
*AMD x86-64 relocations.*
- enum [Elf\\_STNs](#)  
*Symbol Table Entry.*
- enum [Elf\\_STBs](#) {  
[STB\\_LOCAL](#) = 0 , [STB\\_GLOBAL](#) = 1 , [STB\\_WEAK](#) = 2 , [STB\\_LOOS](#) = 10 ,  
[STB\\_HIOS](#) = 12 , [STB\\_LOPROC](#) = 13 , [STB\\_HIPROC](#) = 15 }  
*Symbol Binding.*
- enum [Elf\\_STTs](#) {  
[STT\\_NOTYPE](#) = 0 , [STT\\_OBJECT](#) = 1 , [STT\\_FUNC](#) = 2 , [STT\\_SECTION](#) = 3 ,  
[STT\\_FILE](#) = 4 , [STT\\_LOOS](#) = 10 , [STT\\_HIOS](#) = 12 , [STT\\_LOPROC](#) = 13 ,  
[STT\\_HIPROC](#) = 15 }  
*Symbol Types.*
- enum [Elf\\_ATs](#) {  
[AT\\_NULL](#) = 0 , [AT\\_IGNORE](#) = 1 , [AT\\_EXECFD](#) = 2 , [AT\\_PHDR](#) = 3 ,  
[AT\\_PHENT](#) = 4 , [AT\\_PHNUM](#) = 5 , [AT\\_PAGESZ](#) = 6 , [AT\\_BASE](#) = 7 ,  
[AT\\_FLAGS](#) = 8 , [AT\\_ENTRY](#) = 9 , [AT\\_NOTELF](#) = 10 , [AT\\_UID](#) = 11 ,  
[AT\\_EUID](#) = 12 , [AT\\_GID](#) = 13 , [AT\\_EGID](#) = 14 , [AT\\_L4\\_AUX](#) = 0xf0 ,  
[AT\\_L4\\_ENV](#) = 0xf1 }  
*Legal values for [Elf32\\_Auxv.atype](#) / [Elf64\\_Auxv.atype](#).*

## 16.603.1 Detailed Description

ELF definition.

Date

08/18/2000

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de) Alexander Warg [awl1@os.inf.tu-dresden.de](mailto:awl1@os.inf.tu-dresden.de)

Many structs from "Executable and Linkable Format (ELF)", Portable Formats Specification, Version 1.1 and "System V Application Binary Interface - DRAFT - April 29, 1998" The Santa Cruz Operation, Inc. (see <http://www.sco.com/developer/gabi/contents.html>)

Definition in file [elf.h](#).

## 16.604 elf.h

[Go to the documentation of this file.](#)

```

00001
00019 /*
00020  * (c) 2008-2009 Author(s)
00021  *      economic rights: Technische Universität Dresden (Germany)
00022  * This file is part of TUD:OS and distributed under the terms of the
00023  * GNU Lesser General Public License 2.1.
00024  * Please see the COPYING-LGPL-2.1 file for details.
00025  */
00026
00027 /* (c) 2003-2006 Technische Universitaet Dresden
00028  * This file is part of the exec package, which is distributed under
00029  * the terms of the GNU General Public License 2. Please see the
00030  * COPYING file for details. */
00031
00032 #pragma once
00033
00034 #include <14/sys/l4int.h>
00035
00046 typedef l4_uint32_t      Elf32_Addr;
00047 typedef l4_uint32_t      Elf32_Off;
00048 typedef l4_uint16_t      Elf32_Half;
00049 typedef l4_uint32_t      Elf32_Word;
00050 typedef l4_int32_t       Elf32_Sword;
00051 typedef l4_uint64_t      Elf64_Addr;
00052 typedef l4_uint64_t      Elf64_Off;
00053 typedef l4_uint16_t      Elf64_Half;
00054 typedef l4_uint32_t      Elf64_Word;
00055 typedef l4_int32_t       Elf64_Sword;
00056 typedef l4_uint64_t      Elf64_Xword;
00057 typedef l4_int64_t       Elf64_Sxword;
00064 #if L4_MWORD_BITS == 64
00065 # define ElfW(type)      _ElfW(Elf, 64, type)
00066 #else
00067 # define ElfW(type)      _ElfW(Elf, 32, type)
00068 #endif
00069 #define _ElfW(e,w,t)      __ElfW(e, w, _##t)
00070 #define __ElfW(e,w,t)    e##w##t
00071
00072 #if defined(ARCH_x86)
00073 # define L4_ARCH_EI_DATA      ELFDATA2LSB
00074 # define L4_ARCH_E_MACHINE    EM_386
00075 # define L4_ARCH_EI_CLASS     ELFCLASS32
00076 #elif defined(ARCH_amd64)
00077 # define L4_ARCH_EI_DATA      ELFDATA2LSB
00078 # define L4_ARCH_E_MACHINE    EM_X86_64
00079 # define L4_ARCH_EI_CLASS     ELFCLASS64
00080 #elif defined(ARCH_arm)
00081 # define L4_ARCH_EI_DATA      ELFDATA2LSB

```

```

00082 # define L4_ARCH_E_MACHINE      EM_ARM
00083 # define L4_ARCH_EI_CLASS          ELFCLASS32
00084 #elif defined(ARCH_arm64)
00085 # define L4_ARCH_EI_DATA            ELFDATA2LSB
00086 # define L4_ARCH_E_MACHINE          EM_AARCH64
00087 # define L4_ARCH_EI_CLASS            ELFCLASS64
00088 #elif defined(ARCH_ppc32)
00089 # define L4_ARCH_EI_DATA            ELFDATA2MSB
00090 # define L4_ARCH_E_MACHINE          EM_PPC
00091 # define L4_ARCH_EI_CLASS            ELFCLASS32
00092 #elif defined(ARCH_sparc)
00093 # define L4_ARCH_EI_DATA            ELFDATA2MSB
00094 # define L4_ARCH_E_MACHINE          EM_SPARC
00095 # define L4_ARCH_EI_CLASS            ELFCLASS32
00096 #elif defined(ARCH_mips)
00097 # define L4_ARCH_EI_DATA            ELFDATA2LSB
00098 # define L4_ARCH_E_MACHINE          EM_MIPS
00099 # ifdef __mips64
00100 #   define L4_ARCH_EI_CLASS          ELFCLASS64
00101 # else
00102 #   define L4_ARCH_EI_CLASS          ELFCLASS32
00103 # endif
00104 #elif defined(ARCH_riscv)
00105 # define L4_ARCH_EI_DATA            ELFDATA2LSB
00106 # define L4_ARCH_E_MACHINE          EM_RISCV
00107 # if __riscv_xlen == 64
00108 #   define L4_ARCH_EI_CLASS          ELFCLASS64
00109 # else
00110 #   define L4_ARCH_EI_CLASS          ELFCLASS32
00111 # endif
00112 #else
00113 # warning elf.h: Unsupported build architecture!
00114 #endif
00115
00116
00121 enum
00122 {
00123     EI_NIDENT                = 16,
00124 };
00125
00129 typedef struct
00130 {
00131     unsigned char e_ident[EI_NIDENT];
00132     Elf32_Half    e_type;
00133     Elf32_Half    e_machine;
00134     Elf32_Word    e_version;
00135     Elf32_Addr    e_entry;
00136     Elf32_Off     e_phoff;
00137     Elf32_Off     e_shoff;
00138     Elf32_Word    e_flags;
00139     Elf32_Half    e_ehsize;
00140     Elf32_Half    e_phentsize;
00141     Elf32_Half    e_phnum;
00142     Elf32_Half    e_shentsize;
00143     Elf32_Half    e_shnum;
00144     Elf32_Half    e_shstrndx;
00145 } Elf32_Ehdr;
00146
00150 typedef struct
00151 {
00152     unsigned char e_ident[EI_NIDENT];
00153     Elf64_Half    e_type;
00154     Elf64_Half    e_machine;
00155     Elf64_Word    e_version;
00156     Elf64_Addr    e_entry;
00157     Elf64_Off     e_phoff;
00158     Elf64_Off     e_shoff;
00159     Elf64_Word    e_flags;
00160     Elf64_Half    e_ehsize;
00161     Elf64_Half    e_phentsize;
00162     Elf64_Half    e_phnum;
00163     Elf64_Half    e_shentsize;
00164     Elf64_Half    e_shnum;
00165     Elf64_Half    e_shstrndx;
00166 } Elf64_Ehdr;
00167
00172 enum Elf_ETs
00173 {
00174     ET_NONE                = 0,
00175     ET_REL                  = 1,
00176     ET_EXEC                 = 2,
00177     ET_DYN                  = 3,
00178     ET_CORE                 = 4,
00179     ET_LOPROC               = 0xff00,
00180     ET_HIPROC               = 0xffff,
00181 };
00182

```

```

00187 enum Elf_EMs
00188 {
00189     EM_NONE           = 0,
00190     EM_M32            = 1,
00191     EM_SPARC          = 2,
00192     EM_386           = 3,
00193     EM_68K           = 4,
00194     EM_88K           = 5,
00195     EM_860           = 7,
00196     EM_MIPS          = 8,
00197     EM_MIPS_RS4_BE   = 10,
00198     EM_SPARC64       = 11,
00199     EM_PARISC        = 15,
00200     EM_VPP500        = 17,
00201     EM_SPARC32PLUS   = 18,
00202     EM_960           = 19,
00203     EM_PPC           = 20,
00204     EM_V800          = 36,
00205     EM_FR20          = 37,
00206     EM_RH32          = 38,
00207     EM_RCE           = 39,
00208     EM_ARM           = 40,
00209     EM_ALPHA         = 41,
00210     EM_SH            = 42,
00211     EM_SPARCV9       = 43,
00212     EM_TRICORE       = 44,
00213     EM_ARC           = 45,
00214     EM_H8_300        = 46,
00215     EM_H8_300H       = 47,
00216     EM_H8S           = 48,
00217     EM_H8_500        = 49,
00218     EM_IA_64         = 50,
00219     EM_MIPS_X        = 51,
00220     EM_COLDFIRE      = 52,
00221     EM_68HC12        = 53,
00222     EM_X86_64        = 62,
00223     EM_PDSP          = 63,
00224     EM_FX66          = 66,
00225     EM_ST9PLUS       = 67,
00226     EM_ST7           = 68,
00227     EM_68HC16        = 69,
00228     EM_68HC11        = 70,
00229     EM_68HC08        = 71,
00230     EM_68HC05        = 72,
00231     EM_SVX           = 73,
00232     EM_ST19          = 74,
00233     EM_VAX           = 75,
00234     EM_CRIS          = 76,
00235     EM_JAVELIN       = 77,
00236     EM_FIREPATH      = 78,
00237     EM_ZSP           = 79,
00238     EM_MMIX          = 80,
00239     EM_HUANY         = 81,
00240     EM_PRISM         = 82,
00241     EM_AVR           = 83,
00242     EM_FR30          = 84,
00243     EM_D10V          = 85,
00244     EM_D30V          = 86,
00245     EM_V850          = 87,
00246     EM_M32R          = 88,
00247     EM_MN10300       = 89,
00248     EM_MN10200       = 90,
00249     EM_PJ            = 91,
00250     EM_OPENRISC      = 92,
00251     EM_ARC_A5        = 93,
00252     EM_XTENSA        = 94,
00253     EM_ALTERA_NIOS2   = 113,
00254     EM_AARCH64       = 183,
00255     EM_TILEPRO       = 188,
00256     EM_MICROBLAZE    = 189,
00257     EM_TILEGX        = 191,
00258     EM_RISCV         = 243,
00259     EM_NUM           = 244,
00260 };
00261
00262 #if 0
00263 #define EM_ALPHA      0x9026 /* interim value used by Linux until the
00264                               committee comes up with a final number */
00265 #define EM_S390       0xA390 /* interim value used for IBM S390 */
00266 #endif
00267
00270 enum Elf_EVs
00271 {
00272     EV_NONE           = 0,
00273     EV_CURRENT        = 1,
00274 };
00275

```



```

00278 enum Elf_EIs
00279 {
00280     EI_MAG0           = 0,
00281     EI_MAG1           = 1,
00282     EI_MAG2           = 2,
00283     EI_MAG3           = 3,
00284     EI_CLASS          = 4,
00285     EI_DATA           = 5,
00286     EI_VERSION        = 6,
00287     EI_OSABI          = 7,
00288     EI_ABIVERSION     = 8,
00289     EI_PAD            = 9,
00290 };
00291
00293 enum Elf_MAGs
00294 {
00295     ELFMAG0           = 0x7f,
00296     ELFMAG1           = 'E',
00297     ELFMAG2           = 'L',
00298     ELFMAG3           = 'F',
00299 };
00300
00302 enum Elf_CLASSES
00303 {
00304     ELFCLASSNONE      = 0,
00305     ELFCLASS32        = 1,
00306     ELFCLASS64        = 2,
00307     ELFCLASSNUM       = 3,
00308 };
00309
00311 enum Elf_DATAs
00312 {
00313     ELFDATANONE       = 0,
00314     ELFDATA2LSB       = 1,
00315     ELFDATA2MSB       = 2,
00316     ELFDATANUM        = 3,
00317 };
00318
00320 enum Elf_OSABIs
00321 {
00322     ELFOSABI_NONE     = 0,
00323     ELFOSABI_SYSV     = 0,
00324     ELFOSABI_HPUX     = 1,
00325     ELFOSABI_NETBSD   = 2,
00326     ELFOSABI_LINUX    = 3,
00327     ELFOSABI_SOLARIS  = 6,
00328     ELFOSABI_AIX      = 7,
00329     ELFOSABI_IRIX     = 8,
00330     ELFOSABI_FREEBSD  = 9,
00331     ELFOSABI_TRU64    = 10,
00332     ELFOSABI_MODESTO  = 11,
00333     ELFOSABI_OPENBSD  = 12,
00334     ELFOSABI_ARM      = 97,
00335     ELFOSABI_STANDALONE = 255,
00336 };
00337
00339 enum Elf_SHNs
00340 {
00341     SHN_UNDEF         = 0,
00342     SHN_LORESERVE     = 0xff00,
00343     SHN_LOPROC        = 0xff00,
00344     SHN_HIPROC        = 0xff1f,
00345     SHN_ABS           = 0xffff1,
00346     SHN_COMMON        = 0xffff2,
00347     SHN_HIRESERVE     = 0xfffff,
00348 };
00349
00351 typedef struct
00352 {
00353     Elf32_Word      sh_name;
00354     Elf32_Word      sh_type;
00355     Elf32_Word      sh_flags;
00356     Elf32_Addr      sh_addr;
00357     Elf32_Off       sh_offset;
00358     Elf32_Word      sh_size;
00359     Elf32_Word      sh_link;
00360     Elf32_Word      sh_info;
00361     Elf32_Word      sh_addralign;
00362     Elf32_Word      sh_entsize;
00363 } Elf32_Shdr;
00364
00366 typedef struct
00367 {
00368     Elf64_Word      sh_name;
00369     Elf64_Word      sh_type;
00370     Elf64_Xword     sh_flags;
00371     Elf64_Addr      sh_addr;

```

```

00372 Elf64_Off          sh_offset;
00373 Elf64_Xword        sh_size;
00374 Elf64_Word         sh_link;
00375 Elf64_Word         sh_info;
00376 Elf64_Xword        sh_addralign;
00377 Elf64_Xword        sh_entsize;
00378 } Elf64_Shdr;
00379
00381 enum Elf_SHTs
00382 {
00383     SHT_NULL          = 0,
00384     SHT_PROGBITS      = 1,
00385     SHT_SYMTAB        = 2,
00386     SHT_STRTAB        = 3,
00387     SHT_RELA          = 4,
00388     SHT_HASH          = 5,
00389     SHT_DYNAMIC        = 6,
00390     SHT_NOTE          = 7,
00391     SHT_NOBITS        = 8,
00392     SHT_REL           = 9,
00393     SHT_SHLIB         = 10,
00394     SHT_DYNSYM        = 11,
00395     SHT_INIT_ARRAY    = 14,
00396     SHT_FINI_ARRAY    = 15,
00397     SHT_PREINIT_ARRAY = 16,
00398     SHT_GROUP         = 17,
00399     SHT_SYMTAB_SHNDX  = 18,
00400     SHT_NUM           = 19,
00401     SHT_LOOS          = 0x60000000,
00402     SHT_HIOS          = 0x6fffffff,
00403     SHT_LOPROC        = 0x70000000,
00404     SHT_HIPOC        = 0x7fffffff,
00405     SHT_LOUSER        = 0x80000000,
00406     SHT_HIUSER        = 0xffffffff,
00407 };
00408
00410 enum Elf_SHFs
00411 {
00412     SHF_WRITE          = 0x1,
00413     SHF_ALLOC          = 0x2,
00414     SHF_EXECINSTR      = 0x4,
00415     SHF_MERGE          = 0x10,
00416     SHF_STRINGS        = 0x20,
00417     SHF_INFO_LINK      = 0x40,
00418     SHF_LINK_ORDER     = 0x80,
00419     SHF_OS_NONCONFORMING = 0x100,
00421     SHF_GROUP          = 0x200,
00422     SHF_TLS            = 0x400,
00423     SHF_MASKOS         = 0x0ff00000,
00424     SHF_MASKPROC       = 0xf0000000,
00425 };
00426
00427
00429 typedef struct
00430 {
00431     Elf32_Word    p_type;
00432     Elf32_Off     p_offset;
00433     Elf32_Addr    p_vaddr;
00434     Elf32_Addr    p_paddr;
00435     Elf32_Word    p_filesz;
00436     Elf32_Word    p_memsz;
00437     Elf32_Word    p_flags;
00438     Elf32_Word    p_align;
00439 } Elf32_Phdr;
00440
00442 typedef struct
00443 {
00444     Elf64_Word    p_type;
00445     Elf64_Word    p_flags;
00446     Elf64_Off     p_offset;
00447     Elf64_Addr    p_vaddr;
00448     Elf64_Addr    p_paddr;
00449     Elf64_Xword   p_filesz;
00450     Elf64_Xword   p_memsz;
00451     Elf64_Xword   p_align;
00452 } Elf64_Phdr;
00453
00455 enum Elf_PTs
00456 {
00457     PT_NULL          = 0,
00458     PT_LOAD          = 1,
00459     PT_DYNAMIC        = 2,
00460     PT_INTERP        = 3,
00461     PT_NOTE          = 4,
00462     PT_SHLIB         = 5,
00463     PT_PHDR          = 6,
00464     PT_TLS           = 7,

```

```

00465 PT_NUM                = 8,
00466 PT_LOOS                = 0x60000000,
00467 PT_HIOS                = 0x6fffffff,
00468 PT_LOPROC              = 0x70000000,
00469 PT_HIPROC              = 0x7fffffff,
00471 PT_GNU_EH_FRAME       = PT_LOOS + 0x474e550,
00472 PT_GNU_STACK          = PT_LOOS + 0x474e551,
00473 PT_GNU_RELRO           = PT_LOOS + 0x474e552,
00475 PT_L4_STACK          = PT_LOOS + 0x12,
00476 PT_L4_KIP             = PT_LOOS + 0x13,
00477 PT_L4_AUX             = PT_LOOS + 0x14,
00478 };
00479
00481 enum ELF_PFs
00482 {
00483     PF_X                = 0x1,
00484     PF_W                = 0x2,
00485     PF_R                = 0x4,
00486     PF_MASKOS           = 0x0ff00000,
00487     PF_MASKPROC         = 0x7fffffff,
00488 };
00489
00491 enum Elf_NTs_core
00492 {
00493     NT_PRSTATUS         = 1,
00494     NT_FPREGSET         = 2,
00495     NT_PRPSINFO         = 3,
00496     NT_PRXREG           = 4,
00497     NT_TASKSTRUCT       = 4,
00498     NT_PLATFORM         = 5,
00499     NT_AUXV              = 6,
00500     NT_GWINDOWS         = 7,
00501     NT_ASRS             = 8,
00502     NT_PSTATUS          = 10,
00503     NT_PSINFO           = 13,
00504     NT_PRCRED           = 14,
00505     NT_UTSNAME          = 15,
00506     NT_LWPSTATUS        = 16,
00507     NT_LWPSINFO         = 17,
00508     NT_PRFPXREG         = 20,
00509 };
00510
00512 enum Elf_NTs_obj
00513 {
00514     NT_VERSION           = 1,
00515 };
00516
00518 typedef struct
00519 {
00520     Elf32_Sword    d_tag;
00521     union
00522     {
00523         Elf32_Word    d_val;
00524         Elf32_Addr    d_ptr;
00525     } d_un;
00526 } Elf32_Dyn;
00527
00529 typedef struct
00530 {
00531     Elf64_Sxword    d_tag;
00532     union
00533     {
00534         Elf64_Xword    d_val;
00535         Elf64_Addr    d_ptr;
00536     } d_un;
00537 } Elf64_Dyn;
00538
00540 enum Elf_DTs
00541 {
00542     DT_NULL            = 0,
00543     DT_NEEDED          = 1,
00544     DT_PLTRELSZ        = 2,
00545     DT_PLTGOT          = 3,
00546     DT_HASH            = 4,
00547     DT_STRTAB          = 5,
00548     DT_SYMTAB          = 6,
00549     DT_RELA            = 7,
00550     DT_RELASZ          = 8,
00551     DT_RELAENT          = 9,
00552     DT_STRSZ           = 10,
00553     DT_SYMENT           = 11,
00554     DT_INIT            = 12,
00555     DT_FINI            = 13,
00556     DT_SONAME          = 14,
00557     DT_RPATH           = 15,
00558     DT_SYMBOLIC         = 16,
00559     DT_REL              = 17,

```

```

00560 DT_RELSZ                = 18,
00561 DT_RELENT              = 19,
00562 DT_PTRREL              = 20,
00563 DT_DEBUG                = 21,
00564 DT_TEXTREL              = 22,
00565 DT_JMPREL               = 23,
00566 DT_BIND_NOW             = 24,
00567 DT_INIT_ARRAY           = 25,
00568 DT_FINI_ARRAY           = 26,
00569 DT_INIT_ARRAYSZ         = 27,
00570 DT_FINI_ARRAYSZ         = 28,
00571 DT_RUNPATH              = 29,
00572 DT_FLAGS                 = 30,
00573 DT_ENCODING              = 32,
00574 DT_PREINIT_ARRAY        = 32,
00575 DT_PREINIT_ARRAYSZ      = 33,
00576 DT_NUM                   = 34,
00577 DT_LOOS                  = 0x6000000d,
00578 DT_HIOS                  = 0x6ffff000,
00579 DT_LOPROC                = 0x70000000,
00580 DT_HIPROC                = 0x7fffffff,
00581 };
00582
00586 enum Elf_DFs
00587 {
00588     DF_ORIGIN              = 0x00000001,
00589     DF_SYMBOLIC            = 0x00000002,
00590     DF_TEXTREL             = 0x00000004,
00591     DF_BIND_NOW            = 0x00000008,
00592     DF_STATIC_TLS          = 0x00000010,
00593 };
00594
00599 enum Elf_DF_1s
00600 {
00601     DF_1_NOW               = 0x00000001,
00602     DF_1_GLOBAL            = 0x00000002,
00603     DF_1_GROUP             = 0x00000004,
00604     DF_1_NODELETE          = 0x00000008,
00605     DF_1_LOADFLTR          = 0x00000010,
00606     DF_1_INITFIRST         = 0x00000020,
00607     DF_1_NOOPEN            = 0x00000040,
00608     DF_1_ORIGIN            = 0x00000080,
00609     DF_1_DIRECT            = 0x00000100,
00610     DF_1_TRANS             = 0x00000200,
00611     DF_1_INTERPOSE         = 0x00000400,
00612     DF_1_NODEFLIB         = 0x00000800,
00613     DF_1_NODUMP            = 0x00001000,
00614     DF_1_CONFALT           = 0x00002000,
00615     DF_1_ENDFILTEE         = 0x00004000,
00616     DF_1_DISPRELDNE        = 0x00008000,
00617     DF_1_DISPRELPND        = 0x00010000,
00618 };
00619
00621 enum Elf_DTF_1s
00622 {
00623     DTF_1_PARINIT          = 0x00000001,
00624     DTF_1_CONFEXP          = 0x00000002,
00625 };
00626
00628 enum Elf_DF_P1s
00629 {
00630     DF_P1_LAZYLOAD         = 0x00000001,
00631     DF_P1_GROUPPERM        = 0x00000002,
00633 };
00634
00636 typedef struct
00637 {
00638     Elf32_Addr              r_offset;
00639     Elf32_Word              r_info;
00640 } Elf32_Rel;
00641
00643 typedef struct
00644 {
00645     Elf32_Addr              r_offset;
00646     Elf32_Word              r_info;
00647     Elf32_Sword             r_addend;
00648 } Elf32_Rela;
00649
00651 typedef struct
00652 {
00653     Elf64_Addr              r_offset;
00654     Elf64_Xword             r_info;
00655 } Elf64_Rel;
00656
00658 typedef struct
00659 {
00660     Elf64_Addr              r_offset;

```

```

00661     Elf64_Xword      r_info;
00662     Elf64_Sxword      r_addend;
00663 } Elf64_Rela;
00664
00666 #define ELF32_R_SYM(i)      ((i)>>8)
00668 #define ELF32_R_TYPE(i)     ((unsigned char)(i))
00670 #define ELF32_R_INFO(s,t)   (((s)<<8)+(unsigned char)(t))
00671
00673 #define ELF64_R_SYM(i)      ((i)>>32)
00674
00676 #define ELF64_R_TYPE(i)     ((i)&0xffffffffL)
00677
00679 #define ELF64_R_INFO(s,t)   (((s)<<32)+(t)&0xffffffffL)
00680
00682 enum Elf_R_386_s
00683 {
00684     R_386_NONE                = 0,
00685     R_386_32                  = 1,
00686     R_386_PC32                = 2,
00687     R_386_GOT32               = 3,
00688     R_386_PLT32               = 4,
00689     R_386_COPY                = 5,
00690     R_386_GLOB_DAT            = 6,
00691     R_386_JMP_SLOT            = 7,
00692     R_386_RELATIVE            = 8,
00693     R_386_GOTOFF              = 9,
00694     R_386_GOTPC               = 10,
00695     R_386_32PLT              = 11,
00696     R_386_TLS_TPOFF           = 14,
00697     R_386_TLS_IE              = 15,
00699     R_386_TLS_GOTIE           = 16,
00700     R_386_TLS_LE              = 17,
00701     R_386_TLS_GD              = 18,
00703     R_386_TLS_LDM             = 19,
00705     R_386_16                  = 20,
00706     R_386_PC16                = 21,
00707     R_386_8                   = 22,
00708     R_386_PC8                 = 23,
00709     R_386_TLS_GD_32           = 24,
00711     R_386_TLS_GD_PUSH         = 25,
00712     R_386_TLS_GD_CALL         = 26,
00714     R_386_TLS_GD_POP          = 27,
00715     R_386_TLS_LDM_32          = 28,
00717     R_386_TLS_LDM_PUSH        = 29,
00718     R_386_TLS_LDM_CALL        = 30,
00720     R_386_TLS_LDM_POP         = 31,
00721     R_386_TLS_LDO_32          = 32,
00722     R_386_TLS_IE_32           = 33,
00724     R_386_TLS_LE_32           = 34,
00726     R_386_TLS_DTPMOD32        = 35,
00727     R_386_TLS_DTPOFF32        = 36,
00728     R_386_TLS_TPOFF32         = 37,
00729     R_386_NUM                 = 38,
00730 };
00731
00735 enum Elf_EF_ARM_s
00736 {
00737     EF_ARM_RELEXEC             = 0x01,
00738     EF_ARM_HASENTRY            = 0x02,
00739     EF_ARM_INTERWORK           = 0x04,
00740     EF_ARM_APCS_26             = 0x08,
00741     EF_ARM_APCS_FLOAT          = 0x10,
00742     EF_ARM_PIC                 = 0x20,
00743     EF_ARM_ALIGN8              = 0x40,
00744     EF_ARM_NEW_ABI             = 0x80,
00745     EF_ARM_OLD_ABI             = 0x100,
00746
00747     /* Other constants defined in the ARM ELF spec. version B-01. */
00748     /* NB. These conflict with values defined above. */
00749     EF_ARM_SYMSARESORTED       = 0x04,
00750     EF_ARM_DYNSYMSUSESEGIDX    = 0x08,
00751     EF_ARM_MAPSYMSFIRST        = 0x10,
00752     EF_ARM_EABIMASK            = 0xFF000000,
00753
00754     #define EF_ARM_EABI_VERSION(flags) ((flags) & EF_ARM_EABIMASK)
00755     EF_ARM_EABI_UNKNOWN        = 0x00000000,
00756     EF_ARM_EABI_VER1           = 0x01000000,
00757     EF_ARM_EABI_VER2           = 0x02000000,
00758 };
00759
00761 enum Elf_STT_ARM_s
00762 {
00763     STT_ARM_TFUNC              = 0xd,
00764 };
00765
00767 enum Elf_SHF_s_ARM
00768 {

```

```

00769 SHF_ARM_ENTRYSECT = 0x10000000,
00770 SHF_ARM_COMDEF      = 0x80000000,
00772 };
00773
00775 enum Elf_ARM_SBs
00776 {
00777     PF_ARM_SB          = 0x10000000,
00779 };
00780
00782 enum Elf_R_ARM_s
00783 {
00784     R_ARM_NONE          = 0,
00785     R_ARM_PC24          = 1,
00786     R_ARM_ABS32         = 2,
00787     R_ARM_REL32         = 3,
00788     R_ARM_PC13          = 4,
00789     R_ARM_ABS16         = 5,
00790     R_ARM_ABS12         = 6,
00791     R_ARM_THM_ABS5      = 7,
00792     R_ARM_ABS8          = 8,
00793     R_ARM_SBREL32       = 9,
00794     R_ARM_THM_PC22      = 10,
00795     R_ARM_THM_PC8       = 11,
00796     R_ARM AMP_VCALL9    = 12,
00797     R_ARM_SWI24         = 13,
00798     R_ARM_THM_SWI8      = 14,
00799     R_ARM_XPC25         = 15,
00800     R_ARM_THM_XPC22     = 16,
00801     R_ARM_COPY          = 20,
00802     R_ARM_GLOB_DAT      = 21,
00803     R_ARM_JUMP_SLOT     = 22,
00804     R_ARM_RELATIVE      = 23,
00805     R_ARM_GOTOFF        = 24,
00806     R_ARM_GOTPC         = 25,
00807     R_ARM_GOT32         = 26,
00808     R_ARM_PLT32         = 27,
00809     R_ARM_ALU_PCREL_7_0 = 32,
00810     R_ARM_ALU_PCREL_15_8 = 33,
00811     R_ARM_ALU_PCREL_23_15 = 34,
00812     R_ARM_LDR_SBREL_11_0 = 35,
00813     R_ARM_ALU_SBREL_19_12 = 36,
00814     R_ARM_ALU_SBREL_27_20 = 37,
00815     R_ARM_GNU_VTENTRY   = 100,
00816     R_ARM_GNU_VTINHERIT = 101,
00817     R_ARM_THM_PC11      = 102,
00818     R_ARM_THM_PC9       = 103,
00819     R_ARM_RXPC25        = 249,
00820     R_ARM_RSBREL32      = 250,
00821     R_ARM_THM_RPC22     = 251,
00822     R_ARM_RREL32        = 252,
00823     R_ARM_RABS22        = 253,
00824     R_ARM_RPC24         = 254,
00825     R_ARM_RBASE         = 255,
00826     R_ARM_NUM           = 256,
00827 };
00828
00830 enum Elf_R_AARCH64_s
00831 {
00832     R_AARCH64_NONE      = 0,
00833     R_AARCH64_RELATIVE  = 1027,
00834 };
00835
00837 enum Elf_R_X86_64_s
00838 {
00839     R_X86_64_NONE       = 0,
00840     R_X86_64_64         = 1,
00841     R_X86_64_PC32       = 2,
00842     R_X86_64_GOT32      = 3,
00843     R_X86_64_PLT32      = 4,
00844     R_X86_64_COPY       = 5,
00845     R_X86_64_GLOB_DAT   = 6,
00846     R_X86_64_JUMP_SLOT  = 7,
00847     R_X86_64_RELATIVE   = 8,
00848     R_X86_64_GOTPCREL   = 9,
00849     R_X86_64_32         = 10,
00850     R_X86_64_32S        = 11,
00851     R_X86_64_16         = 12,
00852     R_X86_64_PC16       = 13,
00853     R_X86_64_8          = 14,
00854     R_X86_64_PC8        = 15,
00855     R_X86_64_DTPMOD64   = 16,
00856     R_X86_64_DTPOFF64   = 17,
00857     R_X86_64_TPOFF64    = 18,
00858     R_X86_64_TLSGD      = 19,
00860     R_X86_64_TLSLD      = 20,
00862     R_X86_64_DTPOFF32   = 21,
00863     R_X86_64_GOTTPOFF   = 22,

```

```

00865 R_X86_64_TPOFF32      = 23,
00866 R_X86_64_NUM          = 24,
00867 };
00868
00870 enum Elf_STNs
00871 {
00872     STN_UNDEF           = 0,
00873 };
00874
00876 typedef struct
00877 {
00878     Elf32_Word           st_name;
00879     Elf32_Addr           st_value;
00880     Elf32_Word           st_size;
00881     unsigned char        st_info;
00882     unsigned char        st_other;
00883     Elf32_Half           st_shndx;
00884 } Elf32_Sym;
00885
00887 typedef struct
00888 {
00889     Elf64_Word           st_name;
00890     unsigned char        st_info;
00891     unsigned char        st_other;
00892     Elf64_Half           st_shndx;
00893     Elf64_Addr           st_value;
00894     Elf64_Xword          st_size;
00895 } Elf64_Sym;
00896
00898 #define ELF32_ST_BIND(i)    ((i)>4)
00899
00901 #define ELF32_ST_TYPE(i)    ((i)&0xf)
00902
00904 #define ELF32_ST_INFO(b,t)  (( (b)<<4) + ((t)&0xf) )
00905
00907 #define ELF64_ST_BIND(i)    ((i)>4)
00908
00910 #define ELF64_ST_TYPE(i)    ((i)&0xf)
00911
00913 #define ELF64_ST_INFO(b,t)  (( (b)<<4) + ((t)&0xf) )
00914
00917 enum Elf_STBs
00918 {
00919     STB_LOCAL            = 0,
00920     STB_GLOBAL           = 1,
00921     STB_WEAK             = 2,
00922     STB_LOOS            = 10,
00923     STB_HIOS            = 12,
00924     STB_LOPROC          = 13,
00925     STB_HIPROC          = 15,
00926 };
00927
00930 enum Elf_STTs
00931 {
00932     STT_NOTYPE           = 0,
00933     STT_OBJECT           = 1,
00934     STT_FUNC            = 2,
00935     STT_SECTION         = 3,
00936     STT_FILE            = 4,
00937     STT_LOOS            = 10,
00938     STT_HIOS            = 12,
00939     STT_LOPROC          = 13,
00940     STT_HIPROC          = 15,
00941 };
00942
00944 enum Elf_ATs
00945 {
00946     AT_NULL              = 0,
00947     AT_IGNORE            = 1,
00948     AT_EXECFD           = 2,
00949     AT_PHDR             = 3,
00950     AT_PHENT            = 4,
00951     AT_PHNUM            = 5,
00952     AT_PAGESZ           = 6,
00953     AT_BASE             = 7,
00954     AT_FLAGS            = 8,
00955     AT_ENTRY            = 9,
00956     AT_NOTELF           = 10,
00957     AT_UID              = 11,
00958     AT_EUID             = 12,
00959     AT_GID              = 13,
00960     AT_EGID            = 14,
00962     AT_L4_AUX           = 0xf0,
00963     AT_L4_ENV           = 0xf1,
00964 };
00965
00967 typedef struct Elf32_Auxv

```

```

00968 {
00969     Elf32_Word atype;
00970     Elf32_Word avalue;
00971 } Elf32_Auxv;
00972
00974 typedef struct Elf64_Auxv
00975 {
00976     Elf64_Word atype;
00977     Elf64_Word avalue;
00978 } Elf64_Auxv;
00979
00987 static inline int l4util_elf_check_magic(ElfW(Ehdr) const *hdr);
00988
00996 static inline int l4util_elf_check_arch(ElfW(Ehdr) const *hdr);
00997
01004 static inline ElfW(Phdr) *l4util_elf_phdr(ElfW(Ehdr) const *hdr);
01005
01006
01007 /* Implementations */
01008
01009 static inline
01010 int l4util_elf_check_magic(ElfW(Ehdr) const *hdr)
01011 {
01012     return    hdr->e_ident[EI_MAG0] == ELFMAG0
01013             && hdr->e_ident[EI_MAG1] == ELFMAG1
01014             && hdr->e_ident[EI_MAG2] == ELFMAG2
01015             && hdr->e_ident[EI_MAG3] == ELFMAG3;
01016 }
01017
01018 static inline
01019 int l4util_elf_check_arch(ElfW(Ehdr) const *hdr)
01020 {
01021     return    hdr->e_ident[EI_CLASS] == L4_ARCH_EI_CLASS
01022             && hdr->e_ident[EI_DATA]  == L4_ARCH_EI_DATA
01023             && hdr->e_machine        == L4_ARCH_E_MACHINE;
01024 }
01025
01026 static inline
01027 ElfW(Phdr) *l4util_elf_phdr(ElfW(Ehdr) const *hdr)
01028 {
01029     return (ElfW(Phdr) *) ((char *)hdr + hdr->e_phoff);
01030 }

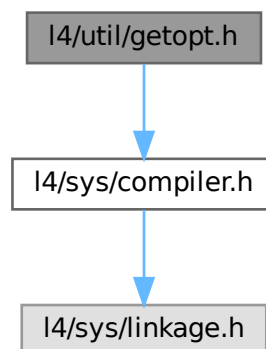
```

## 16.605 l4/util/getopt.h File Reference

getopt

```
#include <l4/sys/compiler.h>
```

Include dependency graph for getopt.h:





## 16.605.1 Detailed Description

getopt

Definition in file [getopt.h](#).

## 16.606 getopt.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU Lesser General Public License 2.1.
00011  * Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013 #ifndef _GETOPT_H
00014 #define _GETOPT_H
00015
00016 #ifndef NULL
00017 #define NULL 0
00018 #endif
00019
00020 #include <14/sys/compiler.h>
00021
00022 EXTERN_C_BEGIN
00023
00024 /* For communication from `getopt' to the caller.
00025  * When `getopt' finds an option that takes an argument,
00026  * the argument value is returned here.
00027  * Also, when `ordering' is RETURN_IN_ORDER,
00028  * each non-option ARGV-element is returned here. */
00029
00030 extern char *optarg;
00031
00032 /* Index in ARGV of the next element to be scanned.
00033  * This is used for communication to and from the caller
00034  * and for communication between successive calls to `getopt'.
00035  *
00036  * On entry to `getopt', zero means this is the first call; initialize.
00037  *
00038  * When `getopt' returns -1, this is the index of the first of the
00039  * non-option elements that the caller should itself scan.
00040  *
00041  * Otherwise, `optind' communicates from one call to the next
00042  * how much of ARGV has been scanned so far. */
00043
00044 extern int optind;
00045
00046 /* Callers store zero here to inhibit the error message `getopt' prints
00047  * for unrecognized options. */
00048
00049 extern int opterr;
00050
00051 /* Set to an option character which was unrecognized. */
00052
00053 extern int optopt;
00054
00055 /* Describe the long-named options requested by the application.
00056  * The LONG_OPTIONS argument to getopt_long or getopt_long_only is a vector
00057  * of `struct option' terminated by an element containing a name which is
00058  * zero.
00059  *
00060  * The field `has_arg' is:
00061  * no_argument      (or 0) if the option does not take an argument,
00062  * required_argument (or 1) if the option requires an argument,
00063  * optional_argument (or 2) if the option takes an optional argument.
00064  *
00065  * If the field `flag' is not NULL, it points to a variable that is set
00066  * to the value given in the field `val' when the option is found, but
00067  * left unchanged if the option is not found.
00068  *
00069  * To have a long-named option do something other than set an `int' to
00070  * a compiled-in constant, such as set a value from `optarg', set the
00071  * option's `flag' field to zero and its `val' field to a nonzero
00072  * value (the equivalent single-letter option character, if there is

```

```

00073     one).  For long options that have a zero `flag' field, `getopt'
00074     returns the contents of the `val' field.  */
00075
00076 struct option
00077 {
00078     const char *name;
00079     /* has_arg can't be an enum because some compilers complain about
00080        type mismatches in all the code that assumes it is an int.  */
00081     int has_arg;
00082     int *flag;
00083     int val;
00084 };
00085
00086 /* Names for the values of the `has_arg' field of `struct option'.  */
00087
00088 #define no_argument    0
00089 #define required_argument 1
00090 #define optional_argument 2
00091
00092 L4_CV int getopt (int argc, char *const *argv, const char *shortopts);
00093
00094 L4_CV int getopt_long (int argc, char *const *argv, const char *shortopts,
00095                       const struct option *longopts, int *longind);
00096 L4_CV int getopt_long_only (int argc, char *const *argv,
00097                             const char *shortopts,
00098                             const struct option *longopts, int *longind);
00099
00100 L4_CV int _getopt_internal (int argc, char *const *argv,
00101                             const char *shortopts,
00102                             const struct option *longopts, int *longind,
00103                             int long_only);
00104
00105 EXTERN_C_END
00106
00107 #endif /* _GETOPT_H */

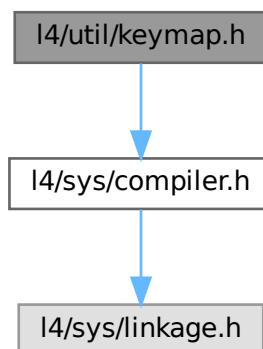
```

## 16.607 I4/util/keymap.h File Reference

Event to ASCII key mapping.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for keymap.h:



### 16.607.1 Detailed Description

Event to ASCII key mapping.

Definition in file [keymap.h](#).

## 16.608 keymap.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012 #ifndef __L4UTIL__KEYMAP_H__
00013 #define __L4UTIL__KEYMAP_H__
00014
00015 #include <l4/sys/compiler.h>
00016
00017 __BEGIN_DECLS
00018
00019 int l4util_map_event_to_keymap(unsigned value, unsigned shift);
00020
00021 __END_DECLS
00022
00023
00024 #endif /* __L4UTIL__KEYMAP_H__ */

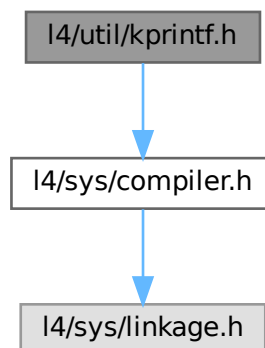
```

## 16.609 l4/util/kprintf.h File Reference

printf using the kernel debugger

```
#include <l4/sys/compiler.h>
```

Include dependency graph for kprintf.h:



### 16.609.1 Detailed Description

printf using the kernel debugger

Date

04/05/2007

Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de),

Definition in file [kprintf.h](#).

## 16.610 kprintf.h

[Go to the documentation of this file.](#)

```

00001 /*****
00009 */
00010 * (c) 2007-2009 Author(s)
00011 *     economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 #ifndef __L4UTIL__INCLUDE__KPRINTF_H__
00018 #define __L4UTIL__INCLUDE__KPRINTF_H__
00019
00020 #include <l4/sys/compiler.h>
00021
00022 EXTERN_C_BEGIN
00023
00024 L4_CV int l4_kprintf(const char *fmt, ...)
00025                 __attribute__((format (printf, 1, 2)));
00026
00027 EXTERN_C_END
00028
00029 #endif /* ! __L4UTIL__INCLUDE__KPRINTF_H__ */

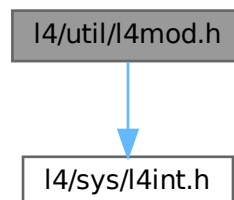
```

## 16.611 l4/util/l4mod.h File Reference

L4mod structures and constants.

```
#include <l4/sys/l4int.h>
```

Include dependency graph for l4mod.h:



### Data Structures

- struct [l4util\\_l4mod\\_mod](#)  
*A single module.*
- struct [l4util\\_l4mod\\_info](#)  
*Base module structure.*

### Enumerations

- enum [l4util\\_l4mod\\_mod\\_info\\_flag](#) {  
[L4util\\_l4mod\\_mod\\_flag\\_unspec](#) = 0 , [L4util\\_l4mod\\_mod\\_flag\\_kernel](#) = 1 , [L4util\\_l4mod\\_mod\\_flag\\_sigma0](#) = 2 , [L4util\\_l4mod\\_mod\\_flag\\_roottask](#) = 3 ,  
[L4util\\_l4mod\\_mod\\_flag\\_mask](#) = 7 << 0 }  
*Flags for l4util\_l4mod\_mod.flags.*

## 16.611.1 Detailed Description

L4mod structures and constants.

Definition in file [l4mod.h](#).

## 16.611.2 Enumeration Type Documentation

### 16.611.2.1 l4util\_l4mod\_mod\_info\_flag

enum [l4util\\_l4mod\\_mod\\_info\\_flag](#)

Flags for [l4util\\_l4mod\\_mod.flags](#).

Enumerator

L4util_l4mod_mod_flag_unspec	Flag for a generic module.
L4util_l4mod_mod_flag_kernel	Flag for the kernel module.
L4util_l4mod_mod_flag_sigma0	Flag for the sigma0 module.
L4util_l4mod_mod_flag_roottask	Flag for the root task module.
L4util_l4mod_mod_flag_mask	Mask for specified flags.

Definition at line 16 of file [l4mod.h](#).

## 16.612 l4mod.h

[Go to the documentation of this file.](#)

```

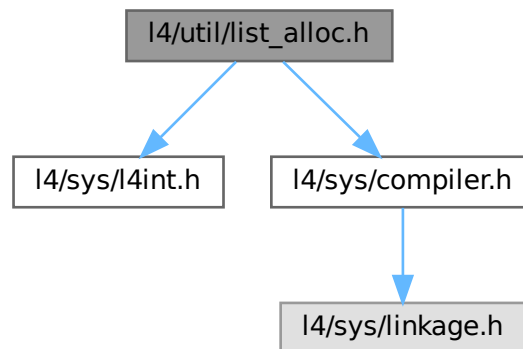
00001 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2021-2022 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@l4re.org>
00005  */
00006
00011 #pragma once
00012
00013 #include <l4/sys/l4int.h>
00014
00016 enum l4util_l4mod_mod_info_flag
00017 {
00018     L4util_l4mod_mod_flag_unspec    = 0,
00019     L4util_l4mod_mod_flag_kernel    = 1,
00020     L4util_l4mod_mod_flag_sigma0    = 2,
00021     L4util_l4mod_mod_flag_roottask  = 3,
00022     L4util_l4mod_mod_flag_mask      = 7 « 0,
00023 };
00024
00026 typedef struct
00027 {
00028     l4_uint64_t flags;
00029     l4_uint64_t mod_start;
00030     l4_uint64_t mod_end;
00031     l4_uint64_t cmdline;
00032 } l4util_l4mod_mod;
00033
00035 typedef struct
00036 {
00037     l4_uint64_t flags;
00038     l4_uint64_t cmdline;
00039     l4_uint64_t mods_addr;
00040     l4_uint32_t mods_count;
00041     l4_uint32_t _pad;
00042
00047     l4_uint64_t vbe_ctrl_info;
00048     l4_uint64_t vbe_mode_info;
00049 } l4util_l4mod_info;

```

## 16.613 l4/util/list\_alloc.h File Reference

Simple list-based allocator.

```
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
Include dependency graph for list_alloc.h:
```



### Functions

- void `l4la_free` (`l4la_free_t **first`, void `*block`, `l4_size_t` `size`)  
*Add free memory to memory pool.*
- void \* `l4la_alloc` (`l4la_free_t **first`, `l4_size_t` `size`, unsigned `align`)  
*Allocate memory from pool.*
- void `l4la_dump` (`l4la_free_t **first`)  
*Show all list members.*
- void `l4la_init` (`l4la_free_t **first`)  
*Init memory pool.*
- `l4_size_t` `l4la_avail` (`l4la_free_t **first`)  
*Show available memory in pool.*

### 16.613.1 Detailed Description

Simple list-based allocator.

Taken from the Fiasco kernel.

#### Date

Alexander Warg <aw11os.inf.tu-dresden.de> Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file `list_alloc.h`.

## 16.613.2 Function Documentation

### 16.613.2.1 l4la\_alloc()

```
void * l4la_alloc (
    l4la_free_t ** first,
    l4_size_t size,
    unsigned align )
```

Allocate memory from pool.

#### Parameters

<i>first</i>	list identifier
<i>size</i>	length of memory block to allocate
<i>align</i>	alignment

### 16.613.2.2 l4la\_avail()

```
l4_size_t l4la_avail (
    l4la_free_t ** first )
```

Show available memory in pool.

#### Parameters

<i>first</i>	list identifier
--------------	-----------------

### 16.613.2.3 l4la\_dump()

```
void l4la_dump (
    l4la_free_t ** first )
```

Show all list members.

#### Parameters

<i>first</i>	list identifier
--------------	-----------------

### 16.613.2.4 l4la\_free()

```
void l4la_free (
    l4la_free_t ** first,
    void * block,
    l4_size_t size )
```

Add free memory to memory pool.

## Parameters

<i>first</i>	list identifier
<i>block</i>	address of unused memory block
<i>size</i>	size of memory block

**16.613.2.5 l4la\_init()**

```
void l4la_init (
    l4la_free_t ** first )
```

Init memory pool.

## Parameters

<i>first</i>	list identifier
--------------	-----------------

**16.614 list\_alloc.h**

[Go to the documentation of this file.](#)

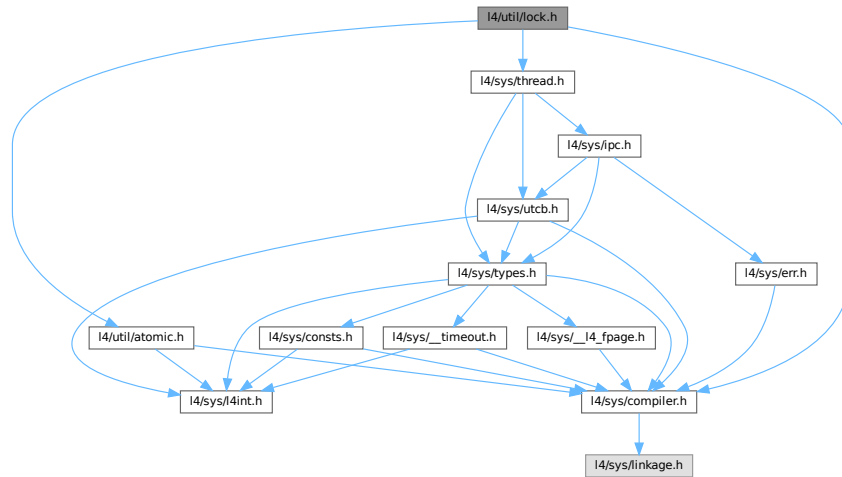
```
00001
00008 /*
00009  * (c) 2003-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00010  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU Lesser General Public License 2.1.
00014  * Please see the COPYING-LGPL-2.1 file for details.
00015  */
00016
00017 #ifndef L4UTIL_L4LA_H
00018 #define L4UTIL_L4LA_H
00019
00020 #include <l4/sys/l4int.h>
00021 #include <l4/sys/compiler.h>
00022
00023 typedef struct l4la_free_t_s
00024 {
00025     struct l4la_free_t_s *next;
00026     l4_size_t             size;
00027 } l4la_free_t;
00028
00029 #define L4LA_INITIALIZER { 0 }
00030
00031 EXTERN_C_BEGIN
00032
00037 L4_CV void      l4la_free(l4la_free_t **first, void *block, l4_size_t size);
00038
00043 L4_CV void*     l4la_alloc(l4la_free_t **first, l4_size_t size, unsigned align);
00044
00047 L4_CV void      l4la_dump(l4la_free_t **first);
00048
00051 L4_CV void      l4la_init(l4la_free_t **first);
00052
00055 L4_CV l4_size_t l4la_avail(l4la_free_t **first);
00056
00057 EXTERN_C_END
00058
00059 #endif
```

**16.615 l4/util/lock.h File Reference**

Simple lock implementation.



```
#include <l4/sys/thread.h>
#include <l4/sys/compiler.h>
#include <l4/util/atomic.h>
Include dependency graph for lock.h:
```



### 16.615.1 Detailed Description

Simple lock implementation.

Does only work if all thread have the same priority!

Date

02/1997

Author

Michael Hohmuth [hohmuth@os.inf.tu-dresden.de](mailto:hohmuth@os.inf.tu-dresden.de)

Definition in file [lock.h](#).

## 16.616 lock.h

[Go to the documentation of this file.](#)

```
00001 /*****
00009 */
00010 * (c) 2000-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 /*****
00018 #ifndef __L4UTIL_LOCK_H__
00019 #define __L4UTIL_LOCK_H__
00020
```

```

00021 #include <l4/sys/thread.h>
00022 #include <l4/sys/compiler.h>
00023 #include <l4/util/atomic.h>
00024
00025 EXTERN_C_BEGIN
00026
00027 typedef l4_uint32_t l4util_simple_lock_t;
00028
00029 L4_INLINE int l4_simple_try_lock(l4util_simple_lock_t *lock);
00030 L4_INLINE void l4_simple_unlock(l4util_simple_lock_t *lock);
00031 L4_INLINE int l4_simple_lock_locked(l4util_simple_lock_t *lock);
00032 L4_INLINE void l4_simple_lock_solid(register l4util_simple_lock_t *p);
00033 L4_INLINE void l4_simple_lock(l4util_simple_lock_t * lock);
00034
00035 L4_INLINE int
00036 l4_simple_try_lock(l4util_simple_lock_t *lock)
00037 {
00038     return l4util_xchg32(lock, 1) == 0;
00039 }
00040
00041 L4_INLINE void
00042 l4_simple_unlock(l4util_simple_lock_t *lock)
00043 {
00044     *lock = 0;
00045 }
00046
00047 L4_INLINE int
00048 l4_simple_lock_locked(l4util_simple_lock_t *lock)
00049 {
00050     return (*lock == 0) ? 0 : 1;
00051 }
00052
00053 L4_INLINE void
00054 l4_simple_lock_solid(register l4util_simple_lock_t *p)
00055 {
00056     while (l4_simple_lock_locked(p) || !l4_simple_try_lock(p))
00057         l4_thread_switch(L4_INVALID_CAP);
00058 }
00059
00060 L4_INLINE void
00061 l4_simple_lock(l4util_simple_lock_t * lock)
00062 {
00063     if (!l4_simple_try_lock(lock))
00064         l4_simple_lock_solid(lock);
00065 }
00066
00067 EXTERN_C_END
00068
00069 #endif

```

## 16.617 l4/util/mb\_info.h File Reference

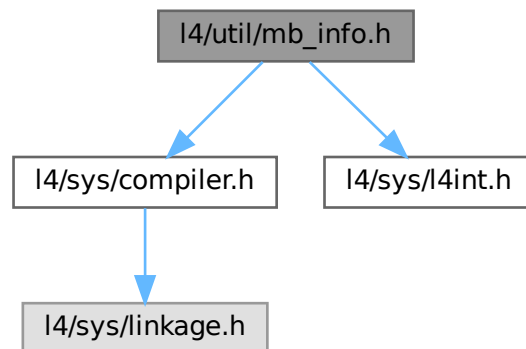
Multiboot info structure as defined by GRUB.

```

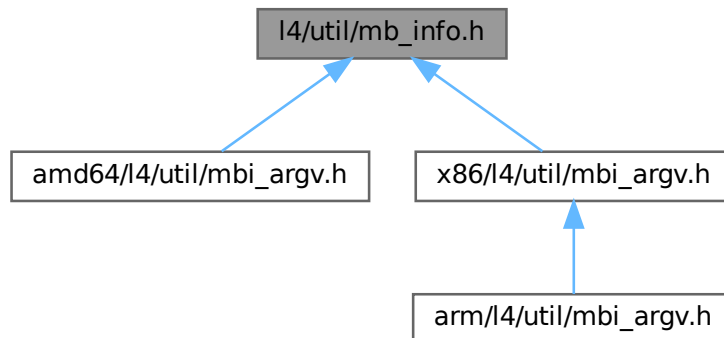
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>

```

Include dependency graph for mb\_info.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4util\\_mb\\_mod\\_t](#)  
The structure type "mod\_list" is used by the [multiboot\\_info](#) structure.
- struct [l4util\\_mb\\_addr\\_range\\_t](#)  
*INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached.*
- struct [l4util\\_mb\\_drive\\_t](#)  
*Drive Info structure.*
- struct [l4util\\_mb\\_apm\\_t](#)  
*APM BIOS info.*
- struct [l4util\\_mb\\_vbe\\_ctrl\\_t](#)

- *VBE controller information.*
- struct [l4util\\_mb\\_vbe\\_mode\\_t](#)  
*VBE mode information.*
- struct [l4util\\_mb\\_info\\_t](#)  
*MultiBoot Info description.*

## Macros

- #define [MB\\_ARD\\_MEMORY](#) 1  
*usable memory "Type", all others are reserved.*
- #define [MB\\_ART\\_MEMORY](#) 1  
*Address Range Types (ART) from "Advanced Configuration and Power Interface Specification" Rev3.0a (p.*
- #define [MB\\_ART\\_RESERVED](#) 2  
*in use or reserved by system*
- #define [MB\\_ART\\_ACPI](#) 3  
*ACPI Reclaim Memory (RAM that contains ACPI tables)*
- #define [MB\\_ART\\_NVS](#) 4  
*ACPI NVS Memory (must not be used by the OS.*
- #define [MB\\_ART\\_UNUSABLE](#) 5  
*memory in which errors have been detected*
- #define [l4util\\_mb\\_for\\_each\\_mmap\\_entry](#)(i, mbi)  
*Iterate over a memory map provided in a Multiboot info.*
- #define [L4UTIL\\_MB\\_MEMORY](#) 0x00000001  
*Flags to be set in the 'flags' parameter above.*
- #define [L4UTIL\\_MB\\_BOOTDEV](#) 0x00000002  
*is there a boot device set?*
- #define [L4UTIL\\_MB\\_CMDLINE](#) 0x00000004  
*is the command-line defined?*
- #define [L4UTIL\\_MB\\_MODS](#) 0x00000008  
*are there modules to do something with?*
- #define [L4UTIL\\_MB\\_AOUT\\_SYMS](#) 0x00000010  
*is there a symbol table loaded?*
- #define [L4UTIL\\_MB\\_ELF\\_SHDR](#) 0x00000020  
*is there an ELF section header table?*
- #define [L4UTIL\\_MB\\_MEM\\_MAP](#) 0x00000040  
*is there a full memory map?*
- #define [L4UTIL\\_MB\\_DRIVE\\_INFO](#) 0x00000080  
*Is there drive info?*
- #define [L4UTIL\\_MB\\_CONFIG\\_TABLE](#) 0x00000100  
*Is there a config table?*
- #define [L4UTIL\\_MB\\_BOOT\\_LOADER\\_NAME](#) 0x00000200  
*Is there a boot loader name?*
- #define [L4UTIL\\_MB\\_APM\\_TABLE](#) 0x00000400  
*Is there a APM table?*
- #define [L4UTIL\\_MB\\_VIDEO\\_INFO](#) 0x00000800  
*Is there video information?*
- #define [L4UTIL\\_MB\\_VALID](#) 0x2BADB002UL  
*If we are multiboot-compliant, this value is present in the eax register.*

## 16.617.1 Detailed Description

Multiboot info structure as defined by GRUB.

Definition in file [mb\\_info.h](#).

## 16.617.2 Macro Definition Documentation

### 16.617.2.1 l4util\_mb\_for\_each\_mmap\_entry

```
#define l4util_mb_for_each_mmap_entry(  
    i,  
    mbi )
```

**Value:**

```
for (i = l4util_mb_first_mmap_entry(mbi);  
     (unsigned long)i < (unsigned long)mbi->mmap_addr + mbi->mmap_length;  
     i = l4util_mb_next_mmap_entry(i)) \
```

Iterate over a memory map provided in a Multiboot info.

**Parameters**

<i>i</i>	Name of a variable of type <a href="#">l4util_mb_addr_range_t</a> * that is consecutively assigned pointers to the entries of the memory map.
<i>mbi</i>	Pointer to the <a href="#">l4util_mb_info_t</a> where the memory map can be found.

Definition at line 334 of file [mb\\_info.h](#).

### 16.617.2.2 L4UTIL\_MB\_MEMORY

```
#define L4UTIL_MB_MEMORY 0x00000001
```

Flags to be set in the 'flags' parameter above.

is there basic lower/upper memory information?

Definition at line 346 of file [mb\\_info.h](#).

### 16.617.2.3 MB\_ARD\_MEMORY

```
#define MB_ARD_MEMORY 1
```

usable memory "Type", all others are reserved.

Definition at line 60 of file [mb\\_info.h](#).

### 16.617.2.4 MB\_ART\_MEMORY

```
#define MB_ART_MEMORY 1
```

Address Range Types (ART) from "Advanced Configuration and Power Interface Specification" Rev3.0a (p.

390). Other values are undefined. available, usable RAM

Definition at line 66 of file [mb\\_info.h](#).

## 16.618 mb\_info.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU Lesser General Public License 2.1.
00011  * Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013
00014 #ifndef L4UTIL_MB_INFO_H
00015 #define L4UTIL_MB_INFO_H
00016
00017 /*****
00018  * Multiboot (v1)
00019  *****/
00020
00021 #ifndef __ASSEMBLY__
00022
00023 #include <l4/sys/compiler.h>
00024 #include <l4/sys/l4int.h>
00025
00026 /*
00027  * \defgroup l4util_mb_mod Multiboot v1
00028  * \ingroup l4util_api
00029  */
00030
00035 typedef struct
00036 {
00037     l4_uint32_t mod_start;
00038     l4_uint32_t mod_end;
00039     l4_uint32_t cmdline;
00040     l4_uint32_t pad;
00041 } l4util_mb_mod_t;
00042
00043
00050 typedef struct __attribute__((packed))
00051 {
00052     l4_uint32_t struct_size;
00053     l4_uint64_t addr;
00054     l4_uint64_t size;
00055     l4_uint32_t type;
00056     /* unspecified optional padding... */
00057 } l4util_mb_addr_range_t;
00058
00060 #define MB_ART_MEMORY 1
00061
00066 #define MB_ART_MEMORY 1
00067 #define MB_ART_RESERVED 2
00068 #define MB_ART_ACPI 3
00070 #define MB_ART_NVS 4
00071 #define MB_ART_UNUSABLE 5
00075 typedef struct
00076 {
00077     l4_uint32_t size;
00078     l4_uint8_t drive_number;
00079     l4_uint8_t drive_mode;
00080     l4_uint16_t drive_cylinders;
00081     l4_uint8_t drive_heads;
00082     l4_uint8_t drive_sectors;
00083     l4_uint16_t drive_ports[0];
00084 } l4util_mb_drive_t;
00085
```

```

00086 /* Drive Mode. */
00087 #define MB_DI_CHS_MODE 0
00088 #define MB_DI_LBA_MODE 1
00089
00090
00092 typedef struct
00093 {
00094     l4_uint16_t version;
00095     l4_uint16_t cseg;
00096     l4_uint32_t offset;
00097     l4_uint16_t cseg_l6;
00098     l4_uint16_t dseg_l6;
00099     l4_uint16_t flags;
00100     l4_uint16_t cseg_len;
00101     l4_uint16_t cseg_l6_len;
00102     l4_uint16_t dseg_l6_len;
00103 } __attribute__((packed)) l4util_mb_apm_t;
00104 static_assert(sizeof(l4util_mb_apm_t) == 20, "Check l4util_mb_apm_t");
00105
00106
00108 typedef struct
00109 {
00110     l4_uint8_t signature[4];
00111     l4_uint16_t version;
00112     l4_uint32_t oem_string;
00113     l4_uint32_t capabilities;
00114     l4_uint32_t video_mode;
00115     l4_uint16_t total_memory;
00116     l4_uint16_t oem_software_rev;
00117     l4_uint32_t oem_vendor_name;
00118     l4_uint32_t oem_product_name;
00119     l4_uint32_t oem_product_rev;
00120     l4_uint8_t reserved[222];
00121     l4_uint8_t oem_data[256];
00122 } __attribute__((packed)) l4util_mb_vbe_ctrl_t;
00123 static_assert(sizeof(l4util_mb_vbe_ctrl_t) == 512, "Check l4util_mb_vbe_ctrl_t");
00124
00125
00127 typedef struct
00128 {
00132     l4_uint16_t mode_attributes;
00134     l4_uint8_t win_a_attributes;
00136     l4_uint8_t win_b_attributes;
00138     l4_uint16_t win_granularity;
00140     l4_uint16_t win_size;
00142     l4_uint16_t win_a_segment;
00144     l4_uint16_t win_b_segment;
00146     l4_uint32_t win_func;
00148     l4_uint16_t bytes_per_scanline;
00154     l4_uint16_t x_resolution;
00156     l4_uint16_t y_resolution;
00158     l4_uint8_t x_char_size;
00160     l4_uint8_t y_char_size;
00162     l4_uint8_t number_of_planes;
00164     l4_uint8_t bits_per_pixel;
00166     l4_uint8_t number_of_banks;
00168     l4_uint8_t memory_model;
00170     l4_uint8_t bank_size;
00172     l4_uint8_t number_of_image_pages;
00174     l4_uint8_t reserved0;
00180     l4_uint8_t red_mask_size;
00182     l4_uint8_t red_field_position;
00184     l4_uint8_t green_mask_size;
00186     l4_uint8_t green_field_position;
00188     l4_uint8_t blue_mask_size;
00190     l4_uint8_t blue_field_position;
00192     l4_uint8_t reserved_mask_size;
00194     l4_uint8_t reserved_field_position;
00196     l4_uint8_t direct_color_mode_info;
00202     l4_uint32_t phys_base;
00204     l4_uint32_t reserved1;
00206     l4_uint16_t reversed2;
00212     l4_uint16_t linear_bytes_per_scanline;
00214     l4_uint8_t banked_number_of_image_pages;
00216     l4_uint8_t linear_number_of_image_pages;
00218     l4_uint8_t linear_red_mask_size;
00220     l4_uint8_t linear_red_field_position;
00222     l4_uint8_t linear_green_mask_size;
00224     l4_uint8_t linear_green_field_position;
00226     l4_uint8_t linear_blue_mask_size;
00228     l4_uint8_t linear_blue_field_position;
00230     l4_uint8_t linear_reserved_mask_size;
00232     l4_uint8_t linear_reserved_field_position;
00234     l4_uint32_t max_pixel_clock;
00236     l4_uint8_t reserved3[190];
00238 } __attribute__((packed)) l4util_mb_vbe_mode_t;
00239 static_assert(sizeof(l4util_mb_vbe_mode_t) == 256, "Check l4util_mb_vbe_mode_t");

```

```

00240
00241
00249 typedef struct
00250 {
00251     l4_uint32_t flags;
00252     l4_uint32_t mem_lower;
00253     l4_uint32_t mem_upper;
00254     l4_uint32_t boot_device;
00255     l4_uint32_t cmdline;
00256     l4_uint32_t mods_count;
00257     l4_uint32_t mods_addr;
00259     union
00260     {
00261         struct
00262         {
00264             l4_uint32_t tabsize;
00265             l4_uint32_t strsize;
00266             l4_uint32_t addr;
00267             l4_uint32_t pad;
00268         }
00269         a;
00270
00271         struct
00272         {
00274             l4_uint32_t num;
00275             l4_uint32_t size;
00276             l4_uint32_t addr;
00277             l4_uint32_t shndx;
00278         }
00279         e;
00280     }
00281     syms;
00282
00283     l4_uint32_t mmap_length;
00284     l4_uint32_t mmap_addr;
00285     l4_uint32_t drives_length;
00286     l4_uint32_t drives_addr;
00287     l4_uint32_t config_table;
00288     l4_uint32_t boot_loader_name;
00289     l4_uint32_t apm_table;
00290     l4_uint32_t vbe_ctrl_info;
00291     l4_uint32_t vbe_mode_info;
00292     l4_uint16_t vbe_mode;
00293     l4_uint16_t vbe_interface_seg;
00294     l4_uint16_t vbe_interface_off;
00295     l4_uint16_t vbe_interface_len;
00296 } l4util_mb_info_t;
00297 static_assert(sizeof(l4util_mb_info_t) == 88, "Check l4util_mb_info_t");
00298
00305 static inline l4util_mb_addr_range_t *
00306 l4util_mb_first_mmap_entry(l4util_mb_info_t *mbi)
00307 {
00308     return (l4util_mb_addr_range_t *) (l4_addr_t) mbi->mmap_addr;
00309 }
00310
00320 static inline l4util_mb_addr_range_t *
00321 l4util_mb_next_mmap_entry(l4util_mb_addr_range_t *e)
00322 {
00323     return (l4util_mb_addr_range_t *) ((l4_addr_t) e + e->struct_size
00324                                         + sizeof(e->struct_size));
00325 }
00326
00334 #define l4util_mb_for_each_mmap_entry(i, mbi)          \
00335     for (i = l4util_mb_first_mmap_entry(mbi);        \
00336          (unsigned long)i < (unsigned long)mbi->mmap_addr + mbi->mmap_length; \
00337          i = l4util_mb_next_mmap_entry(i))
00338
00339 #endif /* ! __ASSEMBLY__ */
00340
00346 #define L4UTIL_MB_MEMORY      0x00000001
00347
00349 #define L4UTIL_MB_BOOTDEV     0x00000002
00350
00352 #define L4UTIL_MB_CMDLINE     0x00000004
00353
00355 #define L4UTIL_MB_MODS        0x00000008
00356
00357 /* These next two are mutually exclusive */
00359 #define L4UTIL_MB_AOUT_SYMS    0x00000010
00360
00362 #define L4UTIL_MB_ELF_SHDR     0x00000020
00363
00365 #define L4UTIL_MB_MEM_MAP      0x00000040
00366
00368 #define L4UTIL_MB_DRIVE_INFO    0x00000080
00369
00371 #define L4UTIL_MB_CONFIG_TABLE 0x00000100

```



```

00372
00374 #define L4UTIL_MB_BOOT_LOADER_NAME 0x00000200
00375
00377 #define L4UTIL_MB_APM_TABLE 0x00000400
00378
00380 #define L4UTIL_MB_VIDEO_INFO 0x00000800
00381
00382
00384 #define L4UTIL_MB_VALID 0x2BADB002UL
00385 #define L4UTIL_MB_VALID_ASM 0x2BADB002
00386
00387
00388 /*****
00389  * Multiboot2
00390  *****/
00391
00392 #ifndef __ASSEMBLY__
00393
00394 typedef struct
00395 {
00396     l4_uint32_t total_size;
00397     l4_uint32_t reserved;
00398 } __attribute__((packed)) l4util_mb2_info_t;
00399
00400 typedef struct
00401 {
00402     char string[0];
00403 } __attribute__((packed)) l4util_mb2_cmdline_tag_t;
00404
00405 typedef struct
00406 {
00407     l4_uint32_t mod_start;
00408     l4_uint32_t mod_end;
00409     char string[];
00410 } __attribute__((packed)) l4util_mb2_module_tag_t;
00411
00412 typedef struct
00413 {
00414     l4_uint64_t base_addr;
00415     l4_uint64_t length;
00416     l4_uint32_t type;
00417     l4_uint32_t reserved;
00418 } __attribute__((packed)) l4util_mb2_memmap_entry_t;
00419
00420 typedef struct
00421 {
00422     l4_uint32_t entry_size;
00423     l4_uint32_t entry_version;
00424     l4util_mb2_memmap_entry_t entries[];
00425 } __attribute__((packed)) l4util_mb2_memmap_tag_t;
00426
00427 typedef struct
00428 {
00429     char data[0];
00430 } __attribute__((packed)) l4util_mb2_rsdp_tag_t;
00431
00432 typedef struct
00433 {
00434     l4_uint32_t type;
00435     l4_uint32_t size;
00436
00437     union
00438     {
00439         l4util_mb2_cmdline_tag_t cmdline;
00440         l4util_mb2_module_tag_t module;
00441         l4util_mb2_memmap_tag_t memmap;
00442         l4util_mb2_rsdp_tag_t rsdp;
00443     };
00444 } __attribute__((packed)) l4util_mb2_tag_t;
00445
00446 #endif /* ! __ASSEMBLY__ */
00447
00448
00449 #define L4UTIL_MB2_MAGIC 0xE85250D6
00450 #define L4UTIL_MB2_ARCH_I386 0x0
00451
00452 #define L4UTIL_MB2_TERMINATOR_HEADER_TAG 0
00453 #define L4UTIL_MB2_INFO_REQUEST_HEADER_TAG 1
00454 #define L4UTIL_MB2_ENTRY_ADDRESS_HEADER_TAG 3
00455 #define L4UTIL_MB2_RELOCATABLE_HEADER_TAG 10
00456
00457 #define L4UTIL_MB2_TAG_FLAG_REQUIRED 0
00458
00459 #define L4UTIL_MB2_TAG_ALIGN_SHIFT 3
00460 #define L4UTIL_MB2_TAG_ALIGN 8
00461
00462 #define L4UTIL_MB2_TERMINATOR_INFO_TAG 0

```

```

00463 #define L4UTIL_MB2_BOOT_CMDLINE_INFO_TAG      1
00464 #define L4UTIL_MB2_MODULE_INFO_TAG             3
00465 #define L4UTIL_MB2_MEMORY_MAP_INFO_TAG         6
00466 #define L4UTIL_MB2_RSDP_OLD_INFO_TAG           14
00467 #define L4UTIL_MB2_RSDP_NEW_INFO_TAG           15
00468 #define L4UTIL_MB2_IMAGE_LOAD_BASE_PHYS_INFO_TAG 21
00469
00470 #define L4UTIL_MB2_RELO_PREFERRED_NONE 0
00471 #define L4UTIL_MB2_RELO_PREFERRED_MIN  1
00472 #define L4UTIL_MB2_RELO_PREFERRED_MAX  2
00473
00474 #endif

```

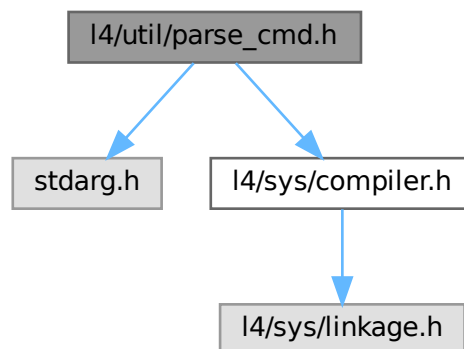
## 16.619 l4/util/parse\_cmd.h File Reference

comfortable command-line parsing

```
#include <stdarg.h>
```

```
#include <l4/sys/compiler.h>
```

Include dependency graph for parse\_cmd.h:



### Typedefs

- typedef void(\* **parse\_cmd\_fn\_t**) (int)  
*Function type for PARSE\_CMD\_FN.*
- typedef void(\* **parse\_cmd\_fn\_arg\_t**) (int, const char \*, int)  
*Function type for PARSE\_CMD\_FN\_ARG.*

### Enumerations

- enum **parse\_cmd\_type**  
*Types for parsing.*

### Functions

- int **parse\_cmdline** (int \*argc, const char \*\*\*argv, int arg0,...)  
*Parse the command-line for specified arguments and store the values into variables.*

## 16.619.1 Detailed Description

comfortable command-line parsing

Date

2002

Author

Jork Loeser [jork.loeser@inf.tu-dresden.de](mailto:jork.loeser@inf.tu-dresden.de)

Definition in file [parse\\_cmd.h](#).

## 16.620 parse\_cmd.h

[Go to the documentation of this file.](#)

```

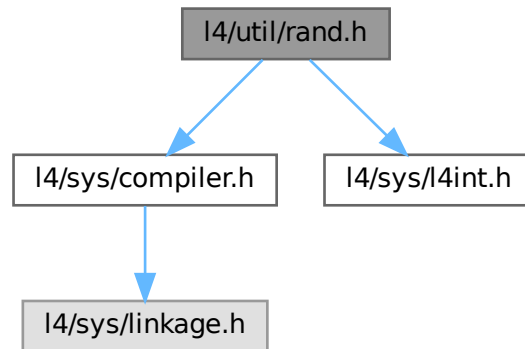
00001
00009 /*
00010  * (c) 2003-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU Lesser General Public License 2.1.
00014  * Please see the COPYING-LGPL-2.1 file for details.
00015  */
00016
00017 #ifndef __PARSE_CMD_H
00018 #define __PARSE_CMD_H
00019
00020 #include <stdarg.h>
00021 #include <14/sys/compiler.h>
00022
00032 enum parse_cmd_type {
00033     PARSE_CMD_INT,
00034     PARSE_CMD_SWITCH,
00035     PARSE_CMD_STRING,
00036     PARSE_CMD_FN,
00037     PARSE_CMD_FN_ARG,
00038     PARSE_CMD_INC,
00039     PARSE_CMD_DEC,
00040 };
00041
00045 typedef L4_CV void (*parse_cmd_fn_t)(int);
00046
00050 typedef L4_CV void (*parse_cmd_fn_arg_t)(int, const char*, int);
00051
00052 EXTERN_C_BEGIN
00053
00140 L4_CV int parse_cmdline(int *argc, const char***argv, int arg0, ...);
00141 L4_CV int parse_cmdlinev(int *argc, const char***argv, int arg0, va_list va);
00142 L4_CV int parse_cmdline_extra(const char*argv0, const char*line, char delim,
00143                             int arg0,...);
00144
00145 EXTERN_C_END
00148 #endif
00149

```

## 16.621 l4/util/rand.h File Reference

Simple Pseudo-Random Number Generator.

```
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
Include dependency graph for rand.h:
```



## Functions

- `l4_uint32_t l4util_rand` (void)  
*Deliver next random number.*
- void `l4util_srand` (`l4_uint32_t` seed)  
*Initialize random number generator.*

### 16.621.1 Detailed Description

Simple Pseudo-Random Number Generator.

#### Date

1998

#### Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de)

Definition in file [rand.h](#).

## 16.622 rand.h

[Go to the documentation of this file.](#)

```

00001
00008 /*
00009  * (c) 2008-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
00014  */
00015
00016 #ifndef __L4UTIL_RAND_H
00017 #define __L4UTIL_RAND_H
00018
00019 #define L4_RAND_MAX 65535
00020
00021 #include <l4/sys/compiler.h>
00022 #include <l4/sys/l4int.h>
00023
00024 EXTERN_C_BEGIN
00025
00037 L4_CV l4_uint32_t
00038 l4util_rand(void);
00039
00046 L4_CV void
00047 l4util_srand (l4_uint32_t seed);
00048
00049 EXTERN_C_END
00050
00051 #endif /* __L4UTIL_RAND_H */

```

## 16.623 I4/util/splitlog2.h File Reference

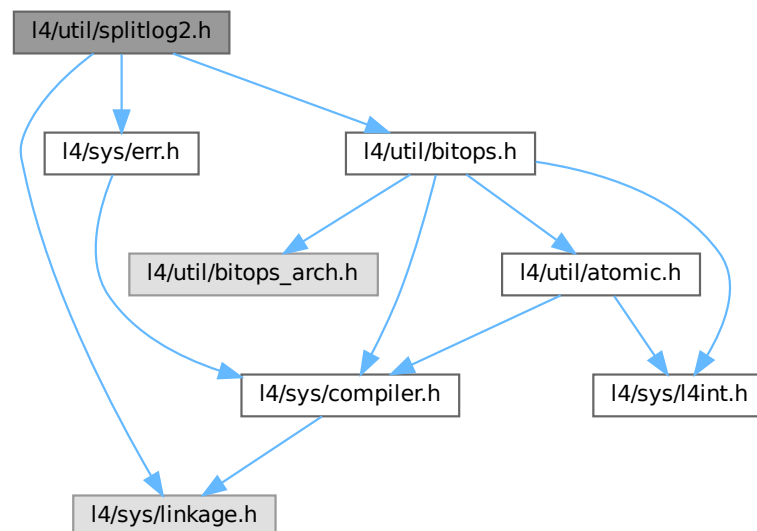
Split a range in log2 aligned and size-aligned chunks.

```

#include <l4/sys/linkage.h>
#include <l4/sys/err.h>
#include <l4/util/bitops.h>

```

Include dependency graph for splitlog2.h:



## Functions

- long `l4util_splitlog2_hdl` (`l4_addr_t` start, `l4_addr_t` end, long(\*handler)(`l4_addr_t` s, `l4_addr_t` e, int log2size))  
*Split a range into log2 base and size aligned chunks.*
- `l4_addr_t` `l4util_splitlog2_size` (`l4_addr_t` start, `l4_addr_t` end)  
*Return log2 base and size aligned length of a range.*

### 16.623.1 Detailed Description

Split a range in log2 aligned and size-aligned chunks.

Definition in file [splitlog2.h](#).

## 16.624 splitlog2.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012 #ifndef __L4UTIL__INCLUDE__SPLITLOG2_H__
00013 #define __L4UTIL__INCLUDE__SPLITLOG2_H__
00014
00015 #include <l4/sys/linkage.h>
00016 #include <l4/sys/err.h>
00017 #include <l4/util/bitops.h>
00018
00019 EXTERN_C_BEGIN
00020
00023 L4_INLINE long
00024 l4util_splitlog2_hdl(l4_addr_t start, l4_addr_t end,
00025                     long (*handler)(l4_addr_t s, l4_addr_t e, int log2size));
00026
00027 L4_INLINE l4_addr_t
00028 l4util_splitlog2_size(l4_addr_t start, l4_addr_t end);
00029
00030 EXTERN_C_END
00031
00032 /* Implementation */
00033
00034 L4_INLINE long
00035 l4util_splitlog2_hdl(l4_addr_t start, l4_addr_t end,
00036                     long (*handler)(l4_addr_t s, l4_addr_t e, int log2size))
00037 {
00038     if (end < start)
00039         return -L4_EINVAL;
00040     while (start <= end)
00041     {
00042         long retval;
00043         int len2 = l4util_splitlog2_size(start, end);
00044         l4_addr_t len = 1UL << len2;
00045         if ((retval = handler(start, start + len - 1, len2)))
00046             return retval;
00047         start += len;
00048     }
00049     return 0;
00050 }
00051
00052 L4_INLINE l4_addr_t
00053 l4util_splitlog2_size(l4_addr_t start, l4_addr_t end)
00054 {
00055     int start_bits = l4util_bsf(start);
00056     int len_bits = l4util_bsr(end - start + 1);
00057     if (start_bits != -1 && len_bits > start_bits)
00058         len_bits = start_bits;
00059     return len_bits;
00060 }
00061
00062 #endif /* ! __L4UTIL__INCLUDE__SPLITLOG2_H__ */

```

## 16.625 util.h

```

00001
00004 /*
00005  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00006  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00007  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU Lesser General Public License 2.1.
00011  * Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013 #ifndef __L4UTIL__UTIL_H__
00014 #define __L4UTIL__UTIL_H__
00015
00016 #include <l4/sys/types.h>
00017 #include <l4/sys/compiler.h>
00018 #include <l4/sys/ipc.h>
00019
00024 EXTERN_C_BEGIN
00025
00036 L4_CV l4_timeout_s l4util_micros2l4to(l4_uint64_t us) L4_NOTHROW;
00037
00043 L4_CV void l4_sleep(l4_uint32_t ms) L4_NOTHROW;
00044
00051 L4_CV void l4_usleep(l4_uint64_t us) L4_NOTHROW;
00052
00058 L4_INLINE void l4_sleep_forever(void) L4_NOTHROW L4_NORETURN;
00059
00067 L4_INLINE void
00068 l4_touch_ro(const void *addr, unsigned size) L4_NOTHROW;
00069
00077 L4_INLINE void
00078 l4_touch_rw(const void *addr, unsigned size) L4_NOTHROW;
00079
00080
00081
00082 /*
00083  * Implementations
00084  */
00085
00086 L4_INLINE void
00087 l4_sleep_forever(void) L4_NOTHROW
00088 {
00089     for (;;)
00090         l4_ipc_sleep(L4_IPC_NEVER);
00091 }
00092
00093 L4_INLINE void
00094 l4_touch_ro(const void *addr, unsigned size) L4_NOTHROW
00095 {
00096     l4_addr_t b, e;
00097
00098     b = l4_trunc_page((l4_addr_t)addr);
00099     e = l4_trunc_page((l4_addr_t)addr + size - 1);
00100
00101     for (; b <= e; b += L4_PAGESIZE)
00102         (void)(*(volatile char *)b);
00103 }
00104
00105
00106 L4_INLINE void
00107 l4_touch_rw(const void *addr, unsigned size) L4_NOTHROW
00108 {
00109     l4_addr_t b, e;
00110
00111     b = l4_trunc_page((l4_addr_t)addr);
00112     e = l4_trunc_page((l4_addr_t)addr + size - 1);
00113
00114     for (; b <= e; b += L4_PAGESIZE)
00115         *(volatile char *)b |= 0;
00116 }
00117
00118 EXTERN_C_END
00119
00120 #endif /* __L4UTIL__UTIL_H__ */

```

## 16.626 vbus

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)

```

```

00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  */
00010 #pragma once
00011
00012 #include <l4/vbus/vbus.h>
00013 #include <l4/vbus/vbus_pm.h>
00014 #include <l4/sys/icu>
00015
00016 #include <l4/re/dataspace>
00017 #include <l4/re/dma_space>
00018 #include <l4/re/event>
00019 #include <l4/re/inhibitor>
00020
00042 namespace L4vbus {
00043
00044 class Vbus;
00045
00051 template<typename DEC>
00052 class Pm
00053 {
00054 private:
00055     DEC const *self() const { return static_cast<DEC const *>(this); }
00056     DEC *self() { return static_cast<DEC *>(this); }
00057 public:
00065     int pm_suspend() const
00066     { return l4vbus_pm_suspend(self()->bus_cap().cap(), self()->dev_handle()); }
00067
00076     int pm_resume() const
00077     { return l4vbus_pm_resume(self()->bus_cap().cap(), self()->dev_handle()); }
00078 };
00079
00080
00085 class Device : public Pm<Device>
00086 {
00087 public:
00091     Device() : _dev(L4VBUS_NULL) {}
00092
00102     Device(L4::Cap<Vbus> bus, l4vbus_device_handle_t dev)
00103     : _bus(bus), _dev(dev) {}
00104
00109     L4::Cap<Vbus> bus_cap() const { return _bus; }
00110
00118     l4vbus_device_handle_t dev_handle() const { return _dev; }
00119
00120
00150     int device_by_hid(Device *child, char const *hid,
00151                     int depth = L4VBUS_MAX_DEPTH,
00152                     l4vbus_device_t *devinfo = 0) const
00153     {
00154         child->_bus = _bus;
00155         return l4vbus_get_device_by_hid(_bus.cap(), _dev, &child->_dev, hid,
00156                                       depth, devinfo);
00157     }
00158
00173     int next_device(Device *child, int depth = L4VBUS_MAX_DEPTH,
00174                   l4vbus_device_t *devinfo = 0) const
00175     {
00176         child->_bus = _bus;
00177         return l4vbus_get_next_device(_bus.cap(), _dev, &child->_dev, depth,
00178                                       devinfo);
00179     }
00180
00191     int device(l4vbus_device_t *devinfo) const
00192     { return l4vbus_get_device(_bus.cap(), _dev, devinfo); }
00193
00211     int get_resource(unsigned res_idx, l4vbus_resource_t *res) const
00212     {
00213         return l4vbus_get_resource(_bus.cap(), _dev, res_idx, res);
00214     }
00215
00225     int is_compatible(char const *cid) const
00226     { return l4vbus_is_compatible(_bus.cap(), _dev, cid); }
00227
00232     bool operator == (Device const &o) const
00233     {
00234         return _bus == o._bus && _dev == o._dev;
00235     }
00236
00241     bool operator != (Device const &o) const
00242     {
00243         return _bus != o._bus || _dev != o._dev;
00244     }
00245
00246 protected:

```



```

00247  L4::Cap<Vbus> _bus;
00249  l4vbus_device_handle_t _dev;
00250 };
00251
00262 class Icu : public Device
00263 {
00264 public:
00266  enum Src_types
00267  {
00275      Src_dev_handle = L4VBUS_ICU_SRC_DEV_HANDLE
00276  };
00277
00287  int vicu(L4::Cap<L4::Icu> icu) const
00288  {
00289      return l4vbus_vicu_get_cap(_bus.cap(), _dev, icu.cap());
00290  }
00291 };
00292
00300 class Vbus : public L4::Kobject_3t<Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event>
00301 {
00302 public:
00303
00313  int request_ioport(l4vbus_resource_t *res) const
00314  {
00315      return l4vbus_request_ioport(cap(), res);
00316  }
00317
00325  int release_ioport(l4vbus_resource_t *res) const
00326  {
00327      return l4vbus_release_ioport(cap(), res);
00328  }
00329
00338  Device root() const
00339  {
00340      return Device(L4::Cap<Vbus>(cap()), L4VBUS_ROOT_BUS);
00341  }
00342
00359  int assign_dma_domain(unsigned domain_id, unsigned flags,
00360                        L4::Cap<L4Re::Dma_space> dma_space) const
00361  {
00362      flags |= L4VBUS_DMAD_L4RE_DMA_SPACE;
00363      flags &= ~L4VBUS_DMAD_KERNEL_DMA_SPACE;
00364      return l4vbus_assign_dma_domain(cap(), domain_id, flags, dma_space.cap());
00365  }
00366
00384  int assign_dma_domain(unsigned domain_id, unsigned flags,
00385                        L4::Cap<L4::Task> dma_space) const
00386  {
00387      flags |= L4VBUS_DMAD_KERNEL_DMA_SPACE;
00388      flags &= ~L4VBUS_DMAD_L4RE_DMA_SPACE;
00389      return l4vbus_assign_dma_domain(cap(), domain_id, flags, dma_space.cap());
00390  }
00391
00392  typedef L4::Typeid::Raw_ipc<Vbus> Rpcs;
00393 };
00394
00395 } // namespace L4vbus

```

## 16.627 l4/vbus/vbus.h File Reference

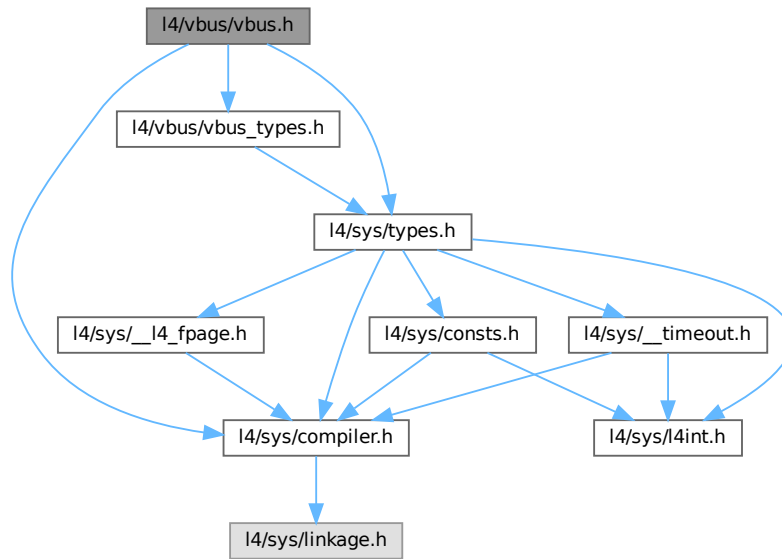
Description of the vbus C API.

```

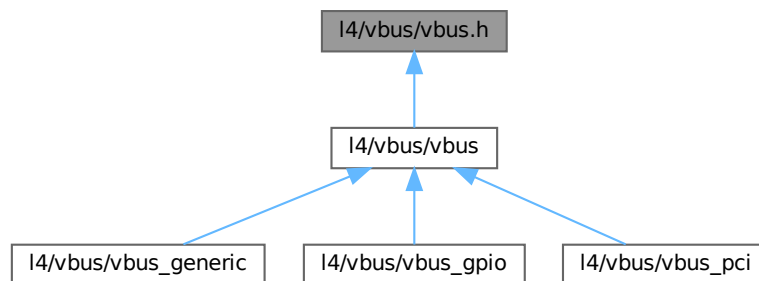
#include <l4/sys/compiler.h>
#include <l4/vbus/vbus_types.h>
#include <l4/sys/types.h>

```

Include dependency graph for vbus.h:



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum { **L4VBUS\_NULL** = -1L , **L4VBUS\_ROOT\_BUS** = 0 }  
*Constants for device nodes.*
- enum **l4vbus\_icu\_src\_types** { **L4VBUS\_ICU\_SRC\_DEV\_HANDLE** = 1ULL << 63 }  
*Flags that can be used with the ICU on a vbus device.*
- enum **L4vbus\_dma\_domain\_assign\_flags** { **L4VBUS\_DMAD\_UNBIND** = 0 , **L4VBUS\_DMAD\_BIND** = 1 , **L4VBUS\_DMAD\_L4RE\_DMA\_SPACE** = 0 , **L4VBUS\_DMAD\_KERNEL\_DMA\_SPACE** = 2 }  
*Flags for **l4vbus\_assign\_dma\_domain()**.*

## Functions

- `int l4vbus_get_device_by_hid (l4_cap_idx_t vbus, l4vbus_device_handle_t parent, l4vbus_device_handle_t *child, char const *hid, int depth, l4vbus_device_t *devinfo)`  
*Find a device by the hardware interface identifier (HID).*
- `int l4vbus_get_next_device (l4_cap_idx_t vbus, l4vbus_device_handle_t parent, l4vbus_device_handle_t *child, int depth, l4vbus_device_t *devinfo)`  
*Find next child following `child`.*
- `int l4vbus_get_device (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, l4vbus_device_t *devinfo)`  
*Obtain detailed information about a Vbus device.*
- `int l4vbus_get_resource (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, unsigned res_idx, l4vbus_resource_t *res)`  
*Obtain the resource description of an individual device resource.*
- `int l4vbus_is_compatible (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, char const *cid)`  
*Check if the given device has a compatibility ID (CID) or HID that matches `cid`.*
- `int l4vbus_get_hid (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, char *hid, unsigned long max_len)`  
*Get the HID (hardware identifier) of a device.*
- `int l4vbus_get_adr (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, l4_uint32_t *adr)`  
*Get the bus-specific address of a device.*
- `int l4vbus_request_ioport (l4_cap_idx_t vbus, l4vbus_resource_t const *res)`  
*Request an IO port resource.*
- `int l4vbus_assign_dma_domain (l4_cap_idx_t vbus, unsigned domain_id, unsigned flags, l4_cap_idx_t dma_space)`  
*Bind or unbind a kernel *DMA space* or a *L4Re::Dma\_space* to a DMA domain.*
- `int l4vbus_release_ioport (l4_cap_idx_t vbus, l4vbus_resource_t const *res)`  
*Release a previously requested IO port resource.*
- `int l4vbus_vicu_get_cap (l4_cap_idx_t vbus, l4vbus_device_handle_t icu, l4_cap_idx_t cap)`  
*Get capability of ICU.*

### 16.627.1 Detailed Description

Description of the vbus C API.

Definition in file [vbus.h](#).

### 16.627.2 Enumeration Type Documentation

#### 16.627.2.1 anonymous enum

`anonymous enum`

Constants for device nodes.

Enumerator

L4VBUS_NULL	NULL device.
L4VBUS_ROOT_BUS	Root device on the vbus.

Definition at line 22 of file [vbus.h](#).

### 16.627.2.2 l4vbus\_icu\_src\_types

enum `l4vbus_icu_src_types`

Flags that can be used with the ICU on a vbus device.

#### Enumerator

L4VBUS_ICU_SRC_DEV_HANDLE	Flag to denote that the value should be interpreted as a device handle. This flag may be used in the <code>source</code> parameter in <code>l4_icu_msi_info()</code> to denote that the ICU should interpret the source ID as a device handle.
---------------------------	--

Definition at line 28 of file `vbus.h`.

## 16.628 vbus.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  */
00015 #pragma once
00016
00017 #include <l4/sys/compiler.h>
00018 #include <l4/vbus/vbus_types.h>
00019 #include <l4/sys/types.h>
00020
00022 enum {
00023     L4VBUS_NULL = -1L,
00024     L4VBUS_ROOT_BUS = 0,
00025 };
00026
00028 enum l4vbus_icu_src_types {
00035     L4VBUS_ICU_SRC_DEV_HANDLE = 1ULL << 63
00036 };
00037
00056 __BEGIN_DECLS
00057
00065 int L4_CV
00066 l4vbus_get_device_by_hid(l4_cap_idx_t vbus, l4vbus_device_handle_t parent,
00067                         l4vbus_device_handle_t *child, char const *hid,
00068                         int depth, l4vbus_device_t *devinfo);
00069
00085 int L4_CV
00086 l4vbus_get_next_device(l4_cap_idx_t vbus, l4vbus_device_handle_t parent,
00087                       l4vbus_device_handle_t *child, int depth,
00088                       l4vbus_device_t *devinfo);
00089
00103 int L4_CV
00104 l4vbus_get_device(l4_cap_idx_t vbus, l4vbus_device_handle_t dev,
00105                  l4vbus_device_t *devinfo);
00106
00115 int L4_CV
00116 l4vbus_get_resource(l4_cap_idx_t vbus, l4vbus_device_handle_t dev,
00117                    unsigned res_idx, l4vbus_resource_t *res);
00118
00119
00126 int L4_CV
00127 l4vbus_is_compatible(l4_cap_idx_t vbus, l4vbus_device_handle_t dev,
00128                     char const *cid);
00129
00140 int L4_CV
00141 l4vbus_get_hid(l4_cap_idx_t vbus, l4vbus_device_handle_t dev, char *hid,
00142               unsigned long max_len);
00143
00154 int L4_CV

```

```

00155 l4vbus_get_adr(l4_cap_idx_t vbus, l4vbus_device_handle_t dev, l4_uint32_t *adr);
00156
00170 int L4_CV
00171 l4vbus_request_ioport(l4_cap_idx_t vbus, l4vbus_resource_t const *res);
00172
00176 enum L4vbus_dma_domain_assign_flags
00177 {
00179     L4VBUS_DMAD_UNBIND = 0,
00181     L4VBUS_DMAD_BIND   = 1,
00183     L4VBUS_DMAD_L4RE_DMA_SPACE = 0,
00185     L4VBUS_DMAD_KERNEL_DMA_SPACE = 2,
00186 };
00187
00209 int L4_CV
00210 l4vbus_assign_dma_domain(l4_cap_idx_t vbus, unsigned domain_id,
00211                          unsigned flags, l4_cap_idx_t dma_space);
00212
00221 int L4_CV
00222 l4vbus_release_ioport(l4_cap_idx_t vbus, l4vbus_resource_t const *res);
00223
00233 int L4_CV
00234 l4vbus_vicu_get_cap(l4_cap_idx_t vbus, l4vbus_device_handle_t icu,
00235                     l4_cap_idx_t cap);
00236
00237 __END_DECLS
00238

```

## 16.629 vbus\_generic

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  */
00011
00012 #pragma once
00013
00014 #include <l4/cxx/ipc_stream>
00015 #include <l4/vbus/vbus_types.h>
00016 #include <l4/vbus/vbus>
00017
00018 inline void
00019 l4vbus_device_msg(l4vbus_device_handle_t handle, l4_uint32_t op,
00020                  L4::Ipc::Iostream &s)
00021 {
00022     s « handle « op;
00023 }

```

## 16.630 vbus\_gpio

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  */
00010 #pragma once
00011
00012 #include <l4/vbus/vbus>
00013 #include <l4/vbus/vbus_gpio.h>
00014
00021 namespace L4vbus {
00022
00028 class Gpio_pin : public Device
00029 {
00030 public:
00031     Gpio_pin(Device const &dev, unsigned pin)
00032         : Device(dev), _pin(pin)
00033     {}
00034
00040     int get() const

```

```

00041 {
00042     return l4vbus_gpio_get(_bus.cap(), _dev, _pin);
00043 }
00044
00051 int set(int value) const
00052 {
00053     return l4vbus_gpio_set(_bus.cap(), _dev, _pin, value);
00054 }
00055
00066 int setup(unsigned mode, unsigned value) const
00067 {
00068     return l4vbus_gpio_setup(_bus.cap(), _dev, _pin, mode, value);
00069 }
00070
00077 int config_pull(unsigned mode) const
00078 {
00079     return l4vbus_gpio_config_pull(_bus.cap(), _dev, _pin, mode);
00080 }
00081
00091 int config_pad(unsigned func, unsigned value) const
00092 {
00093     return l4vbus_gpio_config_pad(_bus.cap(), _dev, _pin, func, value);
00094 }
00095
00104 int config_get(unsigned func, unsigned *value) const
00105 {
00106     return l4vbus_gpio_config_get(_bus.cap(), _dev, _pin, func, value);
00107 }
00108
00114 int to_irq() const
00115 {
00116     return l4vbus_gpio_to_irq(_bus.cap(), _dev, _pin);
00117 }
00118
00124 unsigned pin() const { return _pin; }
00125
00126 protected:
00127     Gpio_pin() {}
00128     unsigned _pin;
00129 };
00130
00135 class Gpio_module : public Device
00136 {
00137 public:
00138     Gpio_module(Device dev)
00139     : Device(dev)
00140     {}
00141
00148 struct Pin_slice
00149 {
00150     Pin_slice(unsigned offset, unsigned mask) : offset(offset), mask(mask) {}
00151     unsigned offset, mask;
00152 };
00153
00168 int setup(Pin_slice const &mask, unsigned mode, unsigned value) const
00169 {
00170     return l4vbus_gpio_multi_setup(_bus.cap(), _dev, mask.offset, mask.mask,
00171                                     mode, value);
00172 }
00173
00187 int config_pad(Pin_slice const &mask, unsigned func, unsigned value) const
00188 {
00189     return l4vbus_gpio_multi_config_pad(_bus.cap(), _dev, mask.offset,
00190                                         mask.mask, func, value);
00191 }
00192
00203 int get(unsigned offset, unsigned *data) const
00204 {
00205     return l4vbus_gpio_multi_get(_bus.cap(), _dev, offset, data);
00206 }
00207
00219 int set(Pin_slice const &mask, unsigned data)
00220 {
00221     return l4vbus_gpio_multi_set(_bus.cap(), _dev, mask.offset,
00222                                   mask.mask, data);
00223 }
00224
00231 Gpio_pin pin(unsigned pin) const
00232 {
00233     return Gpio_pin(*this, pin);
00234 }
00235
00236 protected:
00237     Gpio_module() {}
00238 };
00239
00240 }

```

## 16.631 vbus\_gpio-ops.h

```

00001 /*
00002  * (c) 2011 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/vbus/vbus_interfaces.h>
00013
00014 enum L4vbus_gpio_op
00015 {
00016     L4VBUS_GPIO_OP_SETUP = L4VBUS_INTERFACE_GPIO « L4VBUS_IFACE_SHIFT,
00017     L4VBUS_GPIO_OP_CONFIG_PAD,
00018     L4VBUS_GPIO_OP_CONFIG_GET,
00019     L4VBUS_GPIO_OP_GET,
00020     L4VBUS_GPIO_OP_SET,
00021     L4VBUS_GPIO_OP_MULTI_SETUP,
00022     L4VBUS_GPIO_OP_MULTI_CONFIG_PAD,
00023     L4VBUS_GPIO_OP_MULTI_GET,
00024     L4VBUS_GPIO_OP_MULTI_SET,
00025     L4VBUS_GPIO_OP_TO_IRQ,
00026     L4VBUS_GPIO_OP_CONFIG_PULL
00027 };

```

## 16.632 vbus\_gpio.h

```

00001 /*
00002  * (c) 2011 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/sys/compiler.h>
00013 #include <l4/sys/types.h>
00014 #include <l4/vbus/vbus_types.h>
00015
00016 __BEGIN_DECLS
00017
00018 enum L4vbus_gpio_generic_func
00019 {
00020     L4VBUS_GPIO_SETUP_INPUT = 0x100,
00021     L4VBUS_GPIO_SETUP_OUTPUT = 0x200,
00022     L4VBUS_GPIO_SETUP_IRQ = 0x300,
00023 };
00024
00025 enum L4vbus_gpio_pull_modes
00026 {
00027     L4VBUS_GPIO_PIN_PULL_NONE = 0x100,
00028     L4VBUS_GPIO_PIN_PULL_UP = 0x200,
00029     L4VBUS_GPIO_PIN_PULL_DOWN = 0x300,
00030 };
00031
00032 int L4_CV
00033 l4vbus_gpio_setup(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00034                  unsigned pin, unsigned mode, int value);
00035
00036 int L4_CV
00037 l4vbus_gpio_config_pull(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00038                        unsigned pin, unsigned mode);
00039
00040 int L4_CV
00041 l4vbus_gpio_config_pad(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00042                       unsigned pin, unsigned func, unsigned value);
00043
00044 int L4_CV
00045 l4vbus_gpio_config_get(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00046                       unsigned pin, unsigned func, unsigned *value);
00047
00048 int L4_CV
00049 l4vbus_gpio_get(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00050                unsigned pin);
00051
00052 int L4_CV

```

```

00106 l4vbus_gpio_set(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00107                 unsigned pin, int value);
00108
00117 int L4_CV
00118 l4vbus_gpio_multi_setup(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00119                        unsigned offset, unsigned mask,
00120                        unsigned mode, unsigned value);
00121
00130 int L4_CV
00131 l4vbus_gpio_multi_config_pad(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00132                             unsigned offset, unsigned mask,
00133                             unsigned func, unsigned value);
00134
00141 int L4_CV
00142 l4vbus_gpio_multi_get(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00143                      unsigned offset, unsigned *data);
00144
00153 int L4_CV
00154 l4vbus_gpio_multi_set(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00155                      unsigned offset, unsigned mask, unsigned data);
00156
00164 int L4_CV
00165 l4vbus_gpio_to_irq(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00166                  unsigned pin);
00167
00170 __END_DECLS

```

## 16.633 vbus\_i2c.h

```

00001 /*
00002  * (c) 2009 Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00003  *      economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  */
00009 #pragma once
00010
00011 #include <l4/sys/compiler.h>
00012 #include <l4/sys/types.h>
00013 #include <l4/vbus/vbus_types.h>
00014
00015 __BEGIN_DECLS
00016
00017 int L4_CV
00018 l4vbus_i2c_write(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00019                l4_uint16_t addr, l4_uint8_t sub_addr,
00020                l4_uint8_t *buffer, unsigned long size);
00021
00022 int L4_CV
00023 l4vbus_i2c_read(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00024               l4_uint16_t addr, l4_uint8_t sub_addr,
00025               l4_uint8_t *buffer, unsigned long *size);
00026
00027 __END_DECLS

```

## 16.634 vbus\_inhibitor.h

```

00001
00007 #pragma once
00008
00009 enum Vbus_inhibitor
00010 {
00011     L4VBUS_INHIBITOR_SUSPEND = 0,
00012     L4VBUS_INHIBITOR_SHUTDOWN = 1,
00013     L4VBUS_INHIBITOR_REBOOT = L4VBUS_INHIBITOR_SHUTDOWN,
00014     L4VBUS_INHIBITOR_WAKEUP = 2,
00015     L4VBUS_INHIBITOR_MAX
00016 };
00017

```

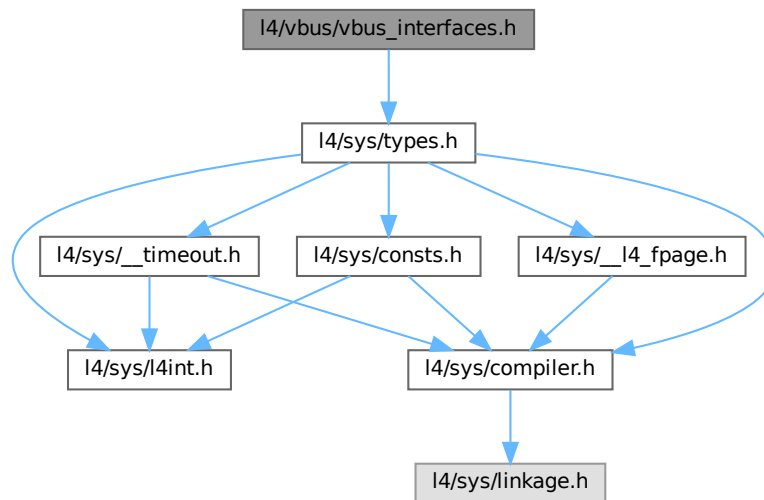
## 16.635 l4/vbus/vbus\_interfaces.h File Reference

This header contains the definition of VBUS sub-interfaces and convenience functions to work with the interface IDs.

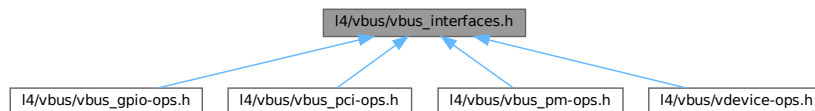


```
#include <l4/sys/types.h>
```

Include dependency graph for vbus\_interfaces.h:



This graph shows which files directly or indirectly include this file:



## Typedefs

- typedef enum `l4vbus_iface_type_t` `l4vbus_iface_type_t`

*Different sub-interfaces a vbus device may support.*

## Enumerations

- enum `l4vbus_iface_type_t` {  
`L4VBUS_INTERFACE_ICU` = 0 , `L4VBUS_INTERFACE_GPIO` , `L4VBUS_INTERFACE_PCI` , `L4VBUS_INTERFACE_PCIDEV`  
 ,  
`L4VBUS_INTERFACE_PM` , `L4VBUS_INTERFACE_BUS` , `L4VBUS_INTERFACE_GENERIC` = 0x20 }  
*Different sub-interfaces a vbus device may support.*
- enum { `L4VBUS_IFACE_SHIFT` = 26 }

## Functions

- unsigned **l4vbus\_subinterface** (unsigned opcode)  
*Return the ID of the vbus sub-interface.*
- unsigned **l4vbus\_interface\_opcode** (unsigned opcode)  
*Return the function opcode within the sub-interface of the vbus command.*
- int **l4vbus\_subinterface\_supported** (l4\_uint32\_t dev\_type, l4vbus\_iface\_type\_t iface\_type)  
*Check if a vbus device supports a given sub-interface.*

### 16.635.1 Detailed Description

This header contains the definition of VBUS sub-interfaces and convenience functions to work with the interface IDs.

Definition in file [vbus\\_interfaces.h](#).

### 16.635.2 Typedef Documentation

#### 16.635.2.1 l4vbus\_iface\_type\_t

```
typedef enum l4vbus_iface_type_t l4vbus_iface_type_t
```

Different sub-interfaces a vbus device may support.

The IPC interface of vbus devices is divided into functional groups of sub-interfaces. Every device must implement the generic interface which provides general device information. According to the type of device, additional functionality may be supported.

The sub-interface constants are first of all used to divide the function opcode space of the interface into these functional groups (see L4VBUS\_IFACE\_SHIFT). They also make up a bitmask that specify the type of the device, i.e. from the point of view of the client a device is defined by the kinds of sub-interfaces it supports.

### 16.635.3 Enumeration Type Documentation

#### 16.635.3.1 anonymous enum

```
anonymous enum
```

##### Enumerator

L4VBUS_IFACE_SHIFT	Sub-interface ID shift. Divides the function opcode sent via IPC into a sub-interface ID and the actual function opcode within the sub-interface.
--------------------	---

Definition at line 50 of file [vbus\\_interfaces.h](#).

#### 16.635.3.2 l4vbus\_iface\_type\_t

```
enum l4vbus_iface_type_t
```

Different sub-interfaces a vbus device may support.

The IPC interface of vbus devices is divided into functional groups of sub-interfaces. Every device must implement the generic interface which provides general device information. According to the type of device, additional functionality may be supported.

The sub-interface constants are first of all used to divide the function opcode space of the interface into these functional groups (see L4VBUS\_IFACE\_SHIFT). They also make up a bitmask that specify the type of the device, i.e. from the point of view of the client a device is defined by the kinds of sub-interfaces it supports.

#### Enumerator

L4VBUS_INTERFACE_ICU	Interrupt Controller.
L4VBUS_INTERFACE_GPIO	GPIO.
L4VBUS_INTERFACE_PCI	PCI.
L4VBUS_INTERFACE_PCIDEV	PCI Device.
L4VBUS_INTERFACE_PM	Power Management.
L4VBUS_INTERFACE_BUS	VBus.
L4VBUS_INTERFACE_GENERIC	No specific sub interface.

Definition at line 31 of file [vbus\\_interfaces.h](#).

## 16.635.4 Function Documentation

### 16.635.4.1 l4vbus\_subinterface\_supported()

```
int l4vbus_subinterface_supported (
    l4_uint32_t dev_type,
    l4vbus_iface_type_t iface_type ) [inline]
```

Check if a vbus device supports a given sub-interface.

#### Parameters

<i>dev_type</i>	Device type as reported in <a href="#">l4vbus_device_t</a> .
<i>iface_type</i>	Sub-interface type to check for.

#### Returns

True if the device supports the sub-interface.

Definition at line 101 of file [vbus\\_interfaces.h](#).

References [L4VBUS\\_INTERFACE\\_GENERIC](#).

## 16.636 vbus\_interfaces.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2014 Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00003  *
00004  * This file is part of TUD:OS and distributed under the terms of the
00005  * GNU General Public License 2.
00006  * Please see the COPYING-GPL-2 file for details.
00007  */
00013 #pragma once
00014
00015 #include <l4/sys/types.h>
00016
00031 typedef enum l4vbus_iface_type_t
00032 {
00034     L4VBUS_INTERFACE_ICU = 0,
00036     L4VBUS_INTERFACE_GPIO,
00038     L4VBUS_INTERFACE_PCI,
00040     L4VBUS_INTERFACE_PCIDEV,
00042     L4VBUS_INTERFACE_PM,
00044     L4VBUS_INTERFACE_BUS,
00046     L4VBUS_INTERFACE_GENERIC = 0x20
00047 } l4vbus_iface_type_t;
00048
00049
00050 enum {
00058     L4VBUS_IFACE_SHIFT = 26
00059 };
00060
00072 L4_INLINE unsigned l4vbus_subinterface(unsigned opcode)
00073 {
00074     return opcode » L4VBUS_IFACE_SHIFT;
00075 }
00076
00088 L4_INLINE unsigned l4vbus_interface_opcode(unsigned opcode)
00089 {
00090     return opcode & ((1 « L4VBUS_IFACE_SHIFT) - 1);
00091 }
00092
00101 L4_INLINE int l4vbus_subinterface_supported(l4_uint32_t dev_type,
00102                                             l4vbus_iface_type_t iface_type)
00103 {
00104     if (iface_type == L4VBUS_INTERFACE_GENERIC)
00105         return 1;
00106
00107     return (dev_type & (1 « iface_type)) ? 1 : 0;
00108 }

```

## 16.637 vbus\_mcspi.h

```

00001 /*
00002  * (c) 2009 Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  */
00009 #pragma once
00010
00011 #include <l4/sys/compiler.h>
00012 #include <l4/sys/types.h>
00013 #include <l4/vbus/vbus_types.h>
00014
00015 __BEGIN_DECLS
00016
00017 int L4_CV
00018 l4vbus_mcspi_read(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00019                  unsigned channel, l4_umword_t *value);
00020
00021 int L4_CV
00022 l4vbus_mcspi_write(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00023                   unsigned channel, l4_umword_t value);
00024
00025 __END_DECLS

```

## 16.638 vbus\_pci

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>

```

```

00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/vbus/vbus>
00013 #include <l4/vbus/vbus_pci.h>
00014
00021 namespace L4vbus {
00022
00027 class Pci_host_bridge : public Device
00028 {
00029 public:
00041     int cfg_read(l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg,
00042                 l4_uint32_t *value, l4_uint32_t width) const
00043     {
00044         return l4vbus_pci_cfg_read(bus_cap().cap(), _dev, bus,
00045                                     devfn, reg, value, width);
00046     }
00047
00048
00060     int cfg_write(l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg,
00061                  l4_uint32_t value, l4_uint32_t width) const
00062     {
00063         return l4vbus_pci_cfg_write(bus_cap().cap(), _dev, bus,
00064                                     devfn, reg, value, width);
00065     }
00066
00067
00081     int irq_enable(l4_uint32_t bus, l4_uint32_t devfn, int pin,
00082                   unsigned char *trigger, unsigned char *polarity) const
00083     {
00084         return l4vbus_pci_irq_enable(bus_cap().cap(), _dev, bus,
00085                                     devfn, pin, trigger, polarity);
00086     }
00087
00088 };
00089
00090
00095 class Pci_dev : public Device
00096 {
00097 public:
00107     int cfg_read(l4_uint32_t reg, l4_uint32_t *value,
00108                 l4_uint32_t width) const
00109     {
00110         return l4vbus_pcidev_cfg_read(bus_cap().cap(), _dev, reg, value, width);
00111     }
00112
00113
00123     int cfg_write(l4_uint32_t reg, l4_uint32_t value,
00124                  l4_uint32_t width) const
00125     {
00126         return l4vbus_pcidev_cfg_write(bus_cap().cap(), _dev, reg, value, width);
00127     }
00128
00129
00139     int irq_enable(unsigned char *trigger, unsigned char *polarity) const
00140     {
00141         return l4vbus_pcidev_irq_enable(bus_cap().cap(), _dev, trigger, polarity);
00142     }
00143
00144 };
00145
00146 }

```

## 16.639 vbus\_pci-ops.h

```

00001 /*
00002  * (c) 2014 Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00003  *
00004  * This file is part of TUD:OS and distributed under the terms of the
00005  * GNU General Public License 2.
00006  * Please see the COPYING-GPL-2 file for details.
00007  */
00008 #pragma once
00009
00010 #include <l4/vbus/vbus_interfaces.h>
00011
00012 enum
00013 {

```

```

00014     L4vbus_pciroot_cfg_read = L4VBUS_INTERFACE_PCI « L4VBUS_IFACE_SHIFT,
00015     L4vbus_pciroot_cfg_write,
00016     L4vbus_pciroot_cfg_irq_enable
00017 };
00018
00019 enum
00020 {
00021     L4vbus_pcidev_cfg_read = L4VBUS_INTERFACE_PCIDEV « L4VBUS_IFACE_SHIFT,
00022     L4vbus_pcidev_cfg_write,
00023     L4vbus_pcidev_cfg_irq_enable
00024 };

```

## 16.640 vbus\_pci.h

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  */
00010 #pragma once
00011
00012 #include <l4/sys/compiler.h>
00013 #include <l4/vbus/vbus_types.h>
00014 #include <l4/sys/types.h>
00015
00023 __BEGIN_DECLS
00024
00031 int L4_CV
00032 l4vbus_pci_cfg_read(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00033                    l4_uint32_t bus, l4_uint32_t devfn,
00034                    l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width);
00035
00042 int L4_CV
00043 l4vbus_pci_cfg_write(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00044                     l4_uint32_t bus, l4_uint32_t devfn,
00045                     l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width);
00046
00053 int L4_CV
00054 l4vbus_pci_irq_enable(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00055                      l4_uint32_t bus, l4_uint32_t devfn,
00056                      int pin, unsigned char *trigger,
00057                      unsigned char *polarity);
00058
00059
00066 int L4_CV
00067 l4vbus_pcidev_cfg_read(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00068                       l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width);
00069
00076 int L4_CV
00077 l4vbus_pcidev_cfg_write(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00078                         l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width);
00079
00086 int L4_CV
00087 l4vbus_pcidev_irq_enable(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00088                          unsigned char *trigger,
00089                          unsigned char *polarity);
00090
00091
00092
00094 __END_DECLS

```

## 16.641 vbus\_pm-ops.h

```

00001 /*
00002  * (c) 2013 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *      economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  */
00009
00010 #pragma once
00011
00012 #include "vbus_interfaces.h"

```

```

00013
00014 enum L4vbus_pm_op
00015 {
00016     L4VBUS_PM_OP_SUSPEND = L4VBUS_INTERFACE_PM « L4VBUS_IFACE_SHIFT,
00017     L4VBUS_PM_OP_RESUME,
00018 };
00019

```

## 16.642 vbus\_pm.h

```

00001 /*
00002  * (c) 2013 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  */
00009 #pragma once
00010
00011 #include <l4/sys/compiler.h>
00012 #include <l4/vbus/vbus_types.h>
00013 #include <l4/sys/types.h>
00014
00021 __BEGIN_DECLS
00022
00029 int L4_CV
00030 l4vbus_pm_suspend(l4_cap_idx_t vbus, l4vbus_device_handle_t handle);
00031
00038 int L4_CV
00039 l4vbus_pm_resume(l4_cap_idx_t vbus, l4vbus_device_handle_t handle);
00040
00043 __END_DECLS

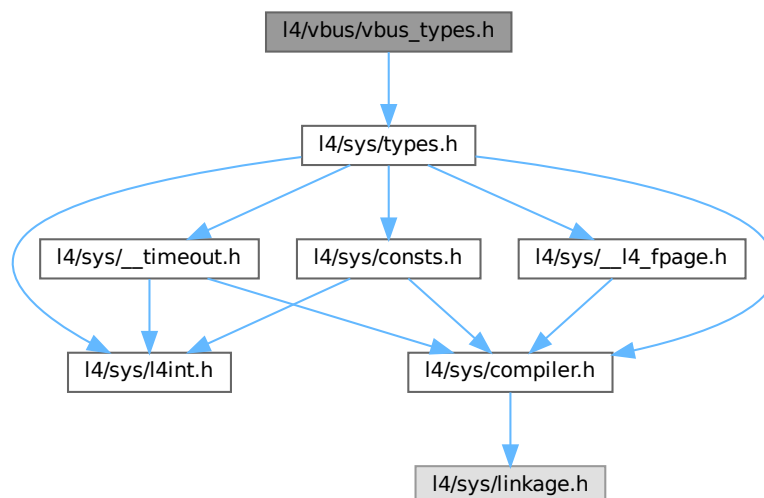
```

## 16.643 l4/vbus/vbus\_types.h File Reference

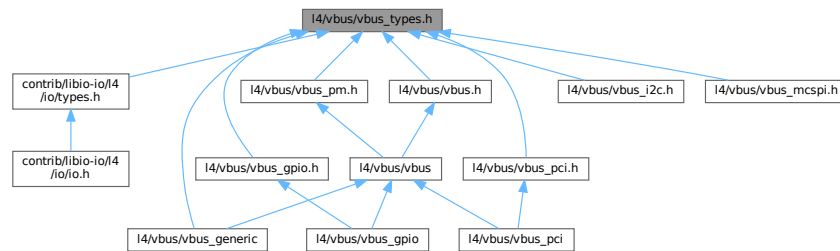
This header file contains descriptions of vbus related data types and constants.

```
#include <l4/sys/types.h>
```

Include dependency graph for vbus\_types.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4vbus\\_resource\\_t](#)  
*Description of a single vbus resource.*
- struct [l4vbus\\_device\\_t](#)  
*Detailed information about a vbus device.*

## Typedefs

- typedef [l4\\_mword\\_t](#) [l4vbus\\_device\\_handle\\_t](#)  
*Device handle for a device on the vbus.*
- typedef [l4\\_addr\\_t](#) [l4vbus\\_paddr\\_t](#)  
*Address of resources on the vbus.*

## Enumerations

- enum [l4vbus\\_resource\\_type\\_t](#) {  
[L4VBUS\\_RESOURCE\\_INVALID](#) = 0 , [L4VBUS\\_RESOURCE\\_IRQ](#) , [L4VBUS\\_RESOURCE\\_MEM](#) ,  
[L4VBUS\\_RESOURCE\\_PORT](#) ,  
[L4VBUS\\_RESOURCE\\_BUS](#) , [L4VBUS\\_RESOURCE\\_GPIO](#) , [L4VBUS\\_RESOURCE\\_DMA\\_DOMAIN](#) ,  
[L4VBUS\\_RESOURCE\\_MAX](#) }  
*Description of vbus resource types.*
- enum [l4vbus\\_resource\\_flags\\_t](#) { [L4VBUS\\_RESOURCE\\_F\\_MEM\\_R](#) = 0x1 , [L4VBUS\\_RESOURCE\\_F\\_MEM\\_W](#)  
= 0x2 , [L4VBUS\\_RESOURCE\\_F\\_MEM\\_MMIO\\_READ](#) = 0x2000 , [L4VBUS\\_RESOURCE\\_F\\_MEM\\_MMIO\\_WRITE](#)  
= 0x4000 }  
*Description of vbus resource flags.*
- enum [l4vbus\\_device\\_flags\\_t](#) { [L4VBUS\\_DEVICE\\_F\\_CHILDREN](#) = 0x10 }  
*Flags describing device properties, see [l4vbus\\_device\\_t](#).*

### 16.643.1 Detailed Description

This header file contains descriptions of vbus related data types and constants.

Definition in file [vbus\\_types.h](#).

### 16.643.2 Enumeration Type Documentation

#### 16.643.2.1 l4vbus\_device\_flags\_t

enum [l4vbus\\_device\\_flags\\_t](#)

Flags describing device properties, see [l4vbus\\_device\\_t](#).



## Enumerator

L4VBUS_DEVICE_F_CHILDREN	Device has child devices.
--------------------------	---------------------------

Definition at line 82 of file [vbus\\_types.h](#).

**16.643.2.2 l4vbus\_resource\_flags\_t**

```
enum l4vbus_resource_flags_t
```

Description of vbus resource flags.

## Enumerator

L4VBUS_RESOURCE_F_MEM_R	Memory resource is readable.
L4VBUS_RESOURCE_F_MEM_W	Memory resource is writeable.
L4VBUS_RESOURCE_F_MEM_MMIO_READ	Reading needs to be performed using the MMIO space protocol.
L4VBUS_RESOURCE_F_MEM_MMIO_WRITE	Writing needs to be performed using the MMIO space protocol.

Definition at line 53 of file [vbus\\_types.h](#).

**16.643.2.3 l4vbus\_resource\_type\_t**

```
enum l4vbus_resource_type_t
```

Description of vbus resource types.

## Enumerator

L4VBUS_RESOURCE_INVALID	Invalid type.
L4VBUS_RESOURCE_IRQ	Interrupt resource.
L4VBUS_RESOURCE_MEM	I/O memory resource.
L4VBUS_RESOURCE_PORT	I/O port resource (x86 only)
L4VBUS_RESOURCE_BUS	Bus resource.
L4VBUS_RESOURCE_GPIO	Gpio resource.
L4VBUS_RESOURCE_DMA_DOMAIN	DMA domain.
L4VBUS_RESOURCE_MAX	Maximum resource id.

Definition at line 41 of file [vbus\\_types.h](#).

**16.644 vbus\_types.h**

[Go to the documentation of this file.](#)

```
00001 /*
```

```

00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  */
00015  #pragma once
00016
00017  #include <l4/sys/types.h>
00018
00020  typedef l4_mword_t l4vbus_device_handle_t;
00022  typedef l4_addr_t l4vbus_paddr_t;
00023
00025  typedef struct {
00027      l4_uint16_t    type;
00029      l4_uint16_t    flags;
00031      l4vbus_paddr_t start;
00033      l4vbus_paddr_t end;
00035      l4vbus_device_handle_t provider;
00037      l4_uint32_t id;
00038  } l4vbus_resource_t;
00039
00041  enum l4vbus_resource_type_t {
00042      L4VBUS_RESOURCE_INVALID = 0,
00043      L4VBUS_RESOURCE_IRQ,
00044      L4VBUS_RESOURCE_MEM,
00045      L4VBUS_RESOURCE_PORT,
00046      L4VBUS_RESOURCE_BUS,
00047      L4VBUS_RESOURCE_GPIO,
00048      L4VBUS_RESOURCE_DMA_DOMAIN,
00049      L4VBUS_RESOURCE_MAX,
00050  };
00051
00053  enum l4vbus_resource_flags_t {
00055      L4VBUS_RESOURCE_F_MEM_R = 0x1,
00057      L4VBUS_RESOURCE_F_MEM_W = 0x2,
00059      L4VBUS_RESOURCE_F_MEM_MMIO_READ = 0x2000,
00061      L4VBUS_RESOURCE_F_MEM_MMIO_WRITE = 0x4000,
00062  };
00063
00064  enum l4vbus_consts_t {
00065      L4VBUS_DEV_NAME_LEN = 64,
00066      L4VBUS_MAX_DEPTH = 100,
00067  };
00068
00070  typedef struct {
00072      l4_uint32_t    type;
00074      char           name[L4VBUS_DEV_NAME_LEN];
00076      unsigned       num_resources;
00078      unsigned       flags;
00079  } l4vbus_device_t;
00080
00082  enum l4vbus_device_flags_t {
00083      L4VBUS_DEVICE_F_CHILDREN = 0x10,
00084  };

```

## 16.645 vdevice-ops.h

```

00001  /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU General Public License 2.
00009  * Please see the COPYING-GPL-2 file for details.
00010  */
00011  #pragma once
00012
00013  #include "vbus_interfaces.h"
00014
00015  enum L4vbus_vdevice_op
00016  {
00017      L4vbus_vdevice_hid = L4VBUS_INTERFACE_GENERIC « L4VBUS_IFACE_SHIFT,
00018      L4vbus_vdevice_adr,
00019      L4vbus_vdevice_get_by_hid,
00020      L4vbus_vdevice_get_next,
00021      L4vbus_vdevice_get_resource,
00022      L4vbus_vdevice_get_hid,
00023      L4vbus_vdevice_is_compatible,

```

```

00024     L4vbus_vdevice_get,
00025 };
00026
00027 enum {
00028     L4vbus_vbus_request_resource = L4VBUS_INTERFACE_BUS « L4VBUS_IFACE_SHIFT,
00029     L4vbus_vbus_release_resource,
00030     L4vbus_vbus_assign_dma_domain,
00031 };
00032
00033 enum
00034 {
00035     L4vbus_vicu_get_cap = L4VBUS_INTERFACE_ICU « L4VBUS_IFACE_SHIFT
00036 };
00037

```

## 16.646 l4/vcpu/vcpu File Reference

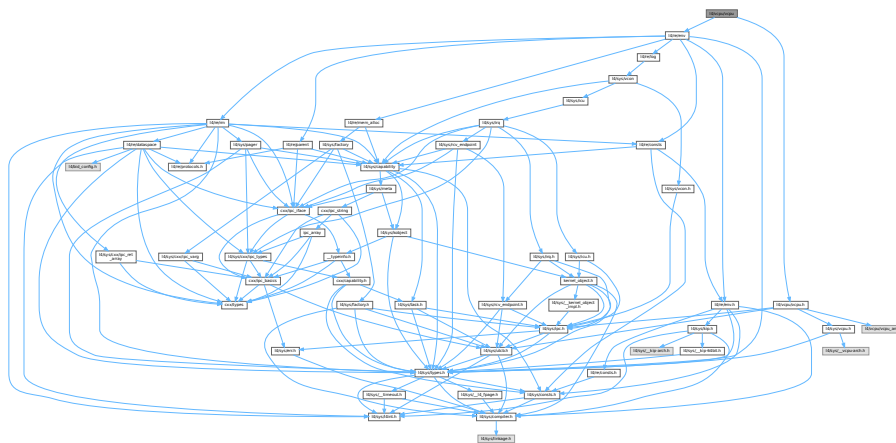
vCPU support library (C++ interface).

```

#include <l4/re/env>
#include <l4/vcpu/vcpu.h>

```

Include dependency graph for vcpu:



### Data Structures

- class [L4vcpu::State](#)  
*C++ implementation of state word in the vCPU area.*
- class [L4vcpu::Vcpu](#)  
*C++ implementation of the vCPU save state area.*

### 16.646.1 Detailed Description

vCPU support library (C++ interface).

Definition in file [vcpu](#).

## 16.647 vcpu

[Go to the documentation of this file.](#)

```

00001 // vi:se ft=cpp:
00002 /*
00003  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00024 #pragma once
00025
00026 #include <l4/re/env>
00027 #include <l4/vcpu/vcpu.h>
00028
00029 namespace L4vcpu {
00030
00035 class State
00036 {
00037 public:
00038     State() {}
00039
00045     explicit State(unsigned v) : _s(v) {}
00046
00052     void add(unsigned bits) throw() { _s |= bits; }
00053
00059     void clear(unsigned bits) throw() { _s &= ~bits; }
00060
00066     void set(unsigned v) throw() { _s = v; }
00067
00068 private:
00069     __typeof__((l4_vcpu_state_t *)0)->state _s;
00070 };
00071
00076 class Vcpu : private l4_vcpu_state_t
00077 {
00078 public:
00082     void irq_disable() throw()
00083     { l4vcpu_irq_disable(this); }
00084
00089     unsigned irq_disable_save() throw()
00090     { return l4vcpu_irq_disable_save(this); }
00091
00092     l4_vcpu_state_t *s() { return this; }
00093     l4_vcpu_state_t const *s() const { return this; }
00094
00099     State *state() throw()
00100     {
00101         static_assert(sizeof(State) == sizeof(l4_vcpu_state_t::state),
00102             "size mismatch");
00103         return reinterpret_cast<State *>(&(l4_vcpu_state_t::state));
00104     }
00105
00110     State state() const throw()
00111     { return static_cast<State>(l4_vcpu_state_t::state); }
00112
00117     State *saved_state() throw()
00118     {
00119         static_assert(sizeof(State) == sizeof(l4_vcpu_state_t::saved_state),
00120             "size mismatch");
00121         return reinterpret_cast<State *>(&(l4_vcpu_state_t::saved_state));
00122     }
00127     State saved_state() const throw()
00128     { return static_cast<State>(l4_vcpu_state_t::saved_state); }
00129
00133     l4_uint16_t sticky_flags() const throw()
00134     { return l4_vcpu_state_t::sticky_flags; }
00135
00146     void irq_enable(l4_utcb_t *utcb, l4vcpu_event_hndl_t do_event_work_cb,
00147         l4vcpu_setup_ipc_t setup_ipc) throw()
00148     { l4vcpu_irq_enable(this, utcb, do_event_work_cb, setup_ipc); }
00149
00161     void irq_restore(unsigned s, l4_utcb_t *utcb,
00162         l4vcpu_event_hndl_t do_event_work_cb,

```

```

00163         l4vcpu_setup_ipc_t setup_ipc) throw()
00164     { l4vcpu_irq_restore(this, s, utcb, do_event_work_cb, setup_ipc); }
00165
00177 void wait_for_event(l4_utcb_t *utcb, l4vcpu_event_hndl_t do_event_work_cb,
00178                    l4vcpu_setup_ipc_t setup_ipc) throw()
00179 { l4vcpu_wait_for_event(this, utcb, do_event_work_cb, setup_ipc); }
00180
00185 void task(L4::Cap<L4::Task> const task = L4::Cap<L4::Task>::Invalid) throw()
00186 { user_task = task.cap(); }
00187
00192 int is_page_fault_entry() const
00193 { return l4vcpu_is_page_fault_entry(this); }
00194
00199 int is_irq_entry() const
00200 { return l4vcpu_is_irq_entry(this); }
00201
00206 l4_vcpu_regs_t *r() throw()
00207 { return &(l4_vcpu_state_t::r); }
00208
00213 l4_vcpu_regs_t const *r() const throw()
00214 { return &(l4_vcpu_state_t::r); }
00215
00220 l4_vcpu_ipc_regs_t *i() throw()
00221 { return &(l4_vcpu_state_t::i); }
00222
00227 l4_vcpu_ipc_regs_t const *i() const throw()
00228 { return &(l4_vcpu_state_t::i); }
00229
00236 void entry_sp(l4_umword_t sp)
00237 { l4_vcpu_state_t::entry_sp = sp; }
00238
00243 void entry_ip(l4_umword_t ip)
00244 { l4_vcpu_state_t::entry_ip = ip; }
00245
00257 L4_CV static int
00258 ext_alloc(Vcpu **vcpu,
00259           l4_addr_t *ext_state,
00260           L4::Cap<L4::Task> task = L4Re::Env::env()->task(),
00261           L4::Cap<L4Re::Rm> rm = L4Re::Env::env()->rm()) throw();
00262
00270 static inline Vcpu *cast(void *x) throw()
00271 { return reinterpret_cast<Vcpu *>(x); }
00272
00280 static inline Vcpu *cast(l4_addr_t x) throw()
00281 { return reinterpret_cast<Vcpu *>(x); }
00282
00286 void print_state(const char *prefix = "") const throw()
00287 { l4vcpu_print_state(this, prefix); }
00288 };
00289
00290
00291 }

```

## 16.648 ipc-invoke.h

```

00001
00009 /*
00010  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00011  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00012  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00013  *      economic rights: Technische Universität Dresden (Germany)
00014  *
00015  * This file is part of TUD:OS and distributed under the terms of the
00016  * GNU General Public License 2.
00017  * Please see the COPYING-GPL-2 file for details.
00018  *
00019  * As a special exception, you may use this file as part of a free software
00020  * library without restriction. Specifically, if other files instantiate
00021  * templates or use macros or inline functions from this file, or you compile
00022  * this file and link it with other files to produce an executable, this
00023  * file does not by itself cause the resulting executable to be covered by
00024  * the GNU General Public License. This exception does not however
00025  * invalidate any other reasons why the executable file might be covered by
00026  * the GNU General Public License.
00027  */
00028
00029 #pragma once
00030
00031 /*
00032  * Some words about the sysenter entry frame: Since the sysenter instruction
00033  * automatically reloads the instruction pointer (eip) and the stack pointer
00034  * (esp) after kernel entry, we have to save both registers preliminary to
00035  * that instruction. We use ecx to store the user-level esp and save eip onto

```

```

00036 * the stack. The ecx register contains the IPC timeout and has to be saved
00037 * onto the stack, too. The ebp register is saved for compatibility reasons
00038 * with the Hazelnut kernel. Both the esp and the ss register are also pushed
00039 * onto the stack to be able to return using the "lret" instruction from the
00040 * sysexit trampoline page if Small Address Spaces are enabled.
00041 */
00042
00043 #ifdef __PIC__
00044 # define L4S_PIC_SAVE "push %%ebx; "
00045 # define L4S_PIC_RESTORE "pop %%ebx; "
00046 # define L4S_PIC_CLOBBER
00047 # define L4S_PIC_SYSCALL , [func] "m" (__l4sys_invoke_indirect)
00048 # if 1
00049 extern void (*__l4sys_invoke_indirect)(void);
00050 # define IPC_SYSENTER      "# indirect sys invoke \n\t" \
00051                          "call *%[func] \n\t"
00052 # else
00053 # define L4S_PIC_SYSCALL
00054 # define IPC_SYSENTER      "call __l4sys_invoke_direct@plt \n\t"
00055 # endif
00056 # define IPC_SYSENTER_ASM call __l4sys_invoke_direct@plt
00057 #else
00062 #define IPC_SYSENTER      "call __l4sys_invoke_direct \n\t"
00067 #define IPC_SYSENTER_ASM call __l4sys_invoke_direct
00072 # define L4S_PIC_SAVE
00077 # define L4S_PIC_RESTORE
00082 # define L4S_PIC_CLOBBER , "ebx"
00083 # define L4S_PIC_SYSCALL
00084
00085 #endif
00090 #define L4_ENTER_KERNEL L4S_PIC_SAVE "push %%ebp; " \
00091                          IPC_SYSENTER
00092                          " pop %%ebp; " L4S_PIC_RESTORE
00093

```

## 16.649 ipc-l42-gcc3-nopic.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *               Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00010  *               Jork Löser <jork@os.inf.tu-dresden.de>
00011  *               economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction. Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License. This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 #pragma once
00027
00028 #include <l4/sys/consts.h>
00029
00030 L4_INLINE l4_msgtag_t
00031 l4_ipc(l4_cap_idx_t dest, l4_utcb_t *u,
00032        l4_umword_t flags,
00033        l4_umword_t slabel,
00034        l4_msgtag_t tag,
00035        l4_umword_t *rlabel,
00036        l4_timeout_t timeout) L4_NOTHROW
00037 {
00038     l4_umword_t dummy, dummy1, dummy2;
00039
00040     (void)u;
00041
00042     __asm__ __volatile__
00043     (L4_ENTER_KERNEL
00044      :
00045      "=d" (dummy2),
00046      "=S" (slabel),
00047      "=c" (dummy1),
00048      "=D" (dummy),
00049      "=a" (tag.raw)
00050      :

```

```
00051     "S" (slabel),
00052     "c" (timeout),
00053     "a" (tag.raw),
00054     "d" (dest | flags)
00055     L4S_PIC_SYSCALL
00056     :
00057     "memory", "cc" L4S_PIC_CLOBBER
00058     );
00059
00060     if (rlabel)
00061         *rlabel = slabel;
00062
00063     return tag;
00064 }
```





# Chapter 17

## Examples

### 17.1 hello/server/src/main.c

This is the famous "Hello World!" program.

This is the famous "Hello World!" program.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *             Frank Mehnert <fm3@os.inf.tu-dresden.de>,
 *             Lukas Grützmacher <lg2@os.inf.tu-dresden.de>
 *             economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <stdio.h>
#include <unistd.h>

int
main(void)
{
    for (;;)
    {
        puts("Hello World!");
        sleep(1);
    }
}
```

### 17.2 examples/sys/ipc/ipc\_example.c

This example shows how two threads can exchange data using the [L4](#) IPC mechanism.

This example shows how two threads can exchange data using the [L4](#) IPC mechanism. One thread is sending an integer to the other thread which is returning the square of the integer. Both values are printed.

```
/*
 * (c) 2008-2009 Author(s)
 *             economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/ipc.h>

#include <pthread-l4.h>
#include <unistd.h>
```

```

#include <stdio.h>

static pthread_t t2;

/* Thread1 is the initiator thread, i.e. it initiates the IPC calls. In
 * other words, it takes the client role. It uses L4 IPC mechanisms to send
 * an integer value to thread2 and received a calculation result back. */
static void *thread1_fn(void *arg)
{
    l4_msgtag_t tag;
    int ipc_error;
    unsigned long value = 1;
    (void)arg;

    while (1)
    {
        printf("Sending: %ld\n", value);

        /* Store the value which we want to have squared in the first message
         * register of our UTCB. */
        l4_utcb_mr()->mr[0] = value;

        /* To an L4 IPC call, i.e. send a message to thread2 and wait for a
         * reply from thread2. The '1' in the msgtag denotes that we want to
         * transfer one word of our message registers (i.e. MR0). No timeout. */
        tag = l4_ipc_call(pthread_l4_cap(t2), l4_utcb(),
                        l4_msgtag(0, 1, 0, 0), L4_IPC_NEVER);
        /* Check for IPC error, if yes, print out the IPC error code, if not,
         * print the received result. */
        ipc_error = l4_ipc_error(tag, l4_utcb());
        if (ipc_error)
            fprintf(stderr, "thread1: IPC error: %x\n", ipc_error);
        else
            printf("Received: %ld\n", l4_utcb_mr()->mr[0]);

        /* Wait some time and increment our value. */
        sleep(1);
        value++;
    }
    return NULL;
}

/* Thread2 is in the server role, i.e. it waits for requests from others and
 * sends back the calculation results. */
static void *thread2_fn(void *arg)
{
    l4_msgtag_t tag;
    l4_umword_t label;
    int ipc_error;
    (void)arg;

    /* Wait for requests from any thread. No timeout, i.e. wait forever. */
    tag = l4_ipc_wait(l4_utcb(), &label, L4_IPC_NEVER);
    while (1)
    {
        /* Check if we had any IPC failure, if yes, print the error code
         * and just wait again. */
        ipc_error = l4_ipc_error(tag, l4_utcb());
        if (ipc_error)
        {
            fprintf(stderr, "thread2: IPC error: %x\n", ipc_error);
            tag = l4_ipc_wait(l4_utcb(), &label, L4_IPC_NEVER);
            continue;
        }

        /* So, the IPC was ok, now take the value out of message register 0
         * of the UTCB and store the square of it back to it. */
        l4_utcb_mr()->mr[0] = l4_utcb_mr()->mr[0] * l4_utcb_mr()->mr[0];

        /* Send the reply and wait again for new messages.
         * The '1' in the msgtag indicated that we want to transfer 1 word in
         * the message registers (i.e. MR0) */
        tag = l4_ipc_reply_and_wait(l4_utcb(), l4_msgtag(0, 1, 0, 0),
                                    &label, L4_IPC_NEVER);
    }
    return NULL;
}

int main(void)
{
    // We will have two threads, one is already running the main function, the
    // other (thread2) will be created using pthread_create.

    if (pthread_create(&t2, NULL, thread2_fn, NULL))
    {
        fprintf(stderr, "Thread creation failed\n");
        return 1;
    }
}

```

```

    }

    // Just run thread1 in the main thread
    thread1_fn(NULL);
    return 0;
}

```

## 17.3 examples/sys/ipc/ipc.cfg

Sample configuration file for the IPC example.

Sample configuration file for the IPC example.

```

# vim:se ft=lua:

local L4 = require("L4");

L4.default_loader:start({}, "rom/ex_ipc1");

```

## 17.4 examples/sys/start-with-exc/main.c

This example shows how to start a newly created thread with a defined set of CPU registers.

This example shows how to start a newly created thread with a defined set of CPU registers.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>,
 *               Björn Döbel <doebel@os.inf.tu-dresden.de>,
 *               Frank Mehnert <fm3@os.inf.tu-dresden.de>
 *               economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * Start a thread with an exception reply. This example does only work on
 * the x86-32 and ARM architectures.
 */

#include <l4/sys/thread.h>
#include <l4/sys/factory.h>
#include <l4/sys/ipc.h>
#include <l4/sys/utcb.h>
#include <l4/util/util.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>

#include <stdlib.h>
#include <stdio.h>

/* Stack for the thread to be created. 8kB are enough. */
static char thread_stack[8 « 10];

/* The thread to be created. For illustration it will print out its
 * register set.
 */
static void L4_STICKY(thread_func(l4_umword_t *d))
{
    while (1)
    {
        printf("hey, I'm a thread\n");
        printf("got register values: %ld %ld %ld %ld %ld %ld %ld\n",
            d[7], d[6], d[5], d[4], d[2], d[1], d[0]);
        l4_sleep(800);
    }
}

/* Startup trick for this example. Put all the CPU registers on the stack so
 * that the C function above can get it on the stack. */
asm(
    ".global thread    \n"
    "thread:           \n"
    "#ifdef ARCH_x86

```

```

" pusha      \n"
" push %esp  \n"
" call thread_func  \n"
#endif
#ifdef ARCH_arm
"      push {r0-r7}      \n"
"      mov r0, sp        \n"
"      bl thread_func    \n"
#endif
#ifdef ARCH_arm64
"      stp x0, x1, [sp, #0]! \n"
"      stp x2, x3, [sp, #0]! \n"
"      stp x4, x5, [sp, #0]! \n"
"      stp x6, x7, [sp, #0]! \n"
"      mov x0, sp         \n"
"      bl thread_func     \n"
#endif
);
extern void thread(void);

/* Our main function */
int main(void)
{
    /* Get a capability slot for our new thread. */
    l4_cap_idx_t t1 = l4re_util_cap_alloc();
    l4_utcb_t *u = l4_utcb();
    l4_exc_regs_t *e = l4_utcb_exc_u(u);
    l4_msgtag_t tag;
    int err;

    printf("Example showing how to start a thread with an exception.\n");
    /* We do not want to implement a pager here, take the shortcut. */
    printf("Make sure to start this program with ldr-flags=eager_map\n");

    if (l4_is_invalid_cap(t1))
        return 1;

    /* Create the thread using our default factory */
    tag = l4_factory_create_thread(l4re_env()->factory, t1);
    if (l4_error(tag))
        return 1;

    /* Setup the thread by setting the pager and task. */
    l4_thread_control_start();
    l4_thread_control_pager(l4re_env()->main_thread);
    l4_thread_control_exc_handler(l4re_env()->main_thread);
    l4_thread_control_bind((l4_utcb_t *)l4re_env()->first_free_utcb,
                          L4RE_THIS_TASK_CAP);
    tag = l4_thread_control_commit(t1);
    if (l4_error(tag))
        return 2;

    /* Start the thread by finally setting instruction and stack pointer */
    tag = l4_thread_ex_regs(t1,
                          (l4_umword_t)thread,
                          (l4_umword_t)thread_stack + sizeof(thread_stack),
                          L4_THREAD_EX_REGS_TRIGGER_EXCEPTION);

    if (l4_error(tag))
        return 3;

    l4_sched_param_t sp = l4_sched_param(1, 0);
    tag = l4_scheduler_run_thread(l4re_env()->scheduler, t1, &sp);
    if (l4_error(tag))
        return 4;

    /* Receive initial exception from just started thread */
    tag = l4_ipc_receive(t1, u, L4_IPC_NEVER);
    if ((err = l4_ipc_error(tag, u)))
    {
        printf("Umm, ipc error: %x\n", err);
        return 1;
    }
    /* We expect an exception IPC */
    if (!l4_msgtag_is_exception(tag))
    {
        printf("PF?: %lx %lx (not prepared to handle this) %ld\n",
              l4_utcb_mr_u(u)->mr[0], l4_utcb_mr_u(u)->mr[1], l4_msgtag_label(tag));
        return 1;
    }

    /* Fill out the complete register set of the new thread */
    e->sp = (l4_umword_t)(thread_stack + sizeof(thread_stack));
#ifdef ARCH_x86
    e->ip = (l4_umword_t)thread;
    e->edi = 0;
    e->esi = 1;

```

```

    e->ebp = 2;
    e->ebx = 4;
    e->edx = 5;
    e->ecx = 6;
    e->eax = 7;
#endif
#ifdef ARCH_arm
    e->pc = (l4_umword_t)thread;
    e->r[0] = 0;
    e->r[1] = 1;
    e->r[2] = 2;
    e->r[3] = 3;
    e->r[4] = 4;
    e->r[5] = 5;
    e->r[6] = 6;
    e->r[7] = 7;
#endif
#ifdef ARCH_arm64
    e->pc = (l4_umword_t)thread;
    e->r[0] = 0;
    e->r[1] = 1;
    e->r[2] = 2;
    e->r[3] = 3;
    e->r[4] = 4;
    e->r[5] = 5;
    e->r[6] = 6;
    e->r[7] = 7;
#endif
/* Send a complete exception */
tag = l4_msgtag(0, L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);

/* Send reply and start the thread with the defined CPU register set */
tag = l4_ipc_send(tl, u, tag, L4_IPC_NEVER);
if ((err = l4_ipc_error(tag, u)))
    printf("Error sending IPC: %x\n", err);

/* Idle around */
while (1)
    l4_sleep(10000);

return 0;
}

```

## 17.5 examples/sys/singlestep/main.c

This example shows how a thread can be single stepped on the x86 architecture.

This example shows how a thread can be single stepped on the x86 architecture.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
 *      Björn Döbel <doebel@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * Single stepping example for the x86-32 architecture.
 */
#include <l4/sys/ipc.h>
#include <l4/sys/factory.h>
#include <l4/sys/thread.h>
#include <l4/sys/utcb.h>
#include <l4/sys/kdebug.h>

#include <l4/util/util.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

static char thread_stack[8 « 10];

static void thread_func(void)
{
    while (1)

```

```

{
    unsigned long d = 0;

    /* Enable single stepping */
    asm volatile("pushf; pop %0; or $256,%0; push %0; popf\n"
        : "=r" (d) : "r" (d));

    /* Some instructions */
    asm volatile("nop");
    asm volatile("nop");
    asm volatile("nop");
    asm volatile("mov $0x12345000, %%edx" : : : "edx"); // a non-existent cap
    asm volatile("int $0x30\n");
    asm volatile("nop");
    asm volatile("nop");
    asm volatile("nop");

    /* Disabled single stepping */
    asm volatile("pushf; pop %0; and $~256,%0; push %0; popf\n"
        : "=r" (d) : "r" (d));

    /* You won't see those */
    asm volatile("nop");
    asm volatile("nop");
    asm volatile("nop");
}

int main(void)
{
    l4_msgtag_t tag;
    int ipc_stat = 0;
    l4_cap_idx_t th = l4re_util_cap_alloc();
    l4_exc_regs_t exc;
    l4_umword_t mr0, mr1;
    l4_utcb_t *u = l4_utcb();

    printf("Singlestep testing\n");

    if (l4_is_invalid_cap(th))
        return 1;

    l4_touch_rw(thread_stack, sizeof(thread_stack));
    l4_touch_ro(thread_func, 1);

    tag = l4_factory_create_thread(l4re_env()->factory, th);
    if (l4_error(tag))
        return 1;

    l4_thread_control_start();
    l4_thread_control_pager(l4re_env()->main_thread);
    l4_thread_control_exc_handler(l4re_env()->main_thread);
    l4_thread_control_bind((l4_utcb_t *)l4re_env()->first_free_utcb,
        L4RE_THIS_TASK_CAP);
    l4_thread_control_alien(1);
    tag = l4_thread_control_commit(th);
    if (l4_error(tag))
        return 2;

    tag = l4_thread_ex_regs(th, (l4_umword_t)thread_func,
        (l4_umword_t)thread_stack + sizeof(thread_stack),
        0);
    if (l4_error(tag))
        return 3;

    l4_sched_param_t sp = l4_sched_param(1, 0);
    tag = l4_scheduler_run_thread(l4re_env()->scheduler, th, &sp);
    if (l4_error(tag))
        return 4;

    /* Pager/Exception loop */
    if (l4_msgtag_has_error(tag = l4_ipc_receive(th, u, L4_IPC_NEVER)))
    {
        printf("l4_ipc_receive failed");
        return 5;
    }
    memcpy(&exc, l4_utcb_exc(), sizeof(exc));
    mr0 = l4_utcb_mr()->mr[0];
    mr1 = l4_utcb_mr()->mr[1];

    for (;;)
    {
        if (l4_msgtag_is_exception(tag))
        {
            printf("PC = %08lx Trap = %08lx Err = %08lx, SP = %08lx SC-Nr: %lx\n",
                l4_utcb_exc_pc(&exc), exc.trapno, exc.err,
                exc.sp, exc.err » 3);

```

```

        if (exc.err >> 3)
        {
            if (!(exc.err & 4))
            {
                tag = l4_msgtag(L4_PROTO_ALLOW_SYSCALL,
                                L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);

                if (ipc_stat)
                    enter_kdebug("Should not be 1");
            }
            else
            {
                tag = l4_msgtag(L4_PROTO_NONE,
                                L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);

                if (!ipc_stat)
                    enter_kdebug("Should not be 0");
            }
            ipc_stat = !ipc_stat;
        }
        l4_sleep(100);
    }
    else
        printf("Umm, non-handled request: %ld, %08lx %08lx\n",
               l4_msgtag_label(tag), mr0, mr1);

    memcpy(l4_utcb_exc(), &exc, sizeof(exc));

    /* Reply and wait */
    if (l4_msgtag_has_error(tag = l4_ipc_call(th, u, tag, L4_IPC_NEVER)))
    {
        printf("l4_ipc_call failed\n");
        return 5;
    }
    memcpy(&exc, l4_utcb_exc(), sizeof(exc));
    mr0 = l4_utcb_mr()->mr[0];
    mr1 = l4_utcb_mr()->mr[1];
}

return 0;
}

```

## 17.6 examples/sys/aliens/main.c

This example shows how system call tracing can be done.

This example shows how system call tracing can be done.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
 *      Björn Döbel <doebel@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * Example to show syscall tracing.
 */
#ifdef ARCH_x86 || defined(ARCH_amd64)
// MEASURE only works on x86/amd64
// #define MEASURE
#endif

#include <l4/sys/ipc.h>
#include <l4/sys/thread.h>
#include <l4/sys/factory.h>
#include <l4/sys/utcb.h>
#include <l4/util/util.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>
#include <l4/re/c/util/kumem_alloc.h>
#include <l4/sys/debugger.h>

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

/* Architecture specifics */
#ifdef ARCH_x86 || defined(ARCH_amd64)

static int

```

```

is_alien_after_call(l4_exc_regs_t const *exc)
{
    #if defined(ARCH_x86)
        return exc->err & 4;
    #else
        return exc->err == 1;
    #endif
}

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
    printf("PC=%08lx SP=%08lx Err=%08lx Trap=%lx, %s syscall, SC-Nr: %lx\n",
        l4_utcb_exc_pc(exc), exc->sp, exc->err,
        exc->trapno, is_alien_after_call(exc) ? " after" : "before",
        exc->err » 3);
}

#elif defined(ARCH_arm)

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{ return exc->err & 0x40; } // TODO: Should change this to (1 < 16)

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
    printf("PC=%08lx SP=%08lx ULR=%08lx CPSR=%08lx Err=%lx/%lx, %s syscall\n",
        l4_utcb_exc_pc(exc), exc->sp, exc->ulr, exc->cpsr,
        exc->err, exc->err » 26,
        is_alien_after_call(exc) ? " after" : "before");
}

#elif defined(ARCH_arm64)

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{ return exc->err & (1ul < 16); }

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
    printf("PC=%08lx SP=%08lx PSTATE=%08lx Err=%lx/%lx, %s syscall\n",
        l4_utcb_exc_pc(exc), exc->sp, exc->pstate,
        exc->err, exc->err » 26,
        is_alien_after_call(exc) ? " after" : "before");
}

#elif defined(ARCH_mips)

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{ return 0; }

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
    printf("PC=%08lx SP=%08lx Cause=%lx, %s syscall\n",
        l4_utcb_exc_pc(exc), exc->sp, exc->cause,
        is_alien_after_call(exc) ? " after" : "before");
}

#elif defined(ARCH_riscv)

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{ return exc->cause == L4_riscv_ec_l4_alien_after_syscall; }

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
    printf("PC=%08lx SP=%08lx Cause=%lx, %s syscall\n",
        l4_utcb_exc_pc(exc), exc->sp, exc->cause,
        is_alien_after_call(exc) ? " after" : "before");
}

#else

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{ return exc->err & 1; }

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
    printf("PC=%08lx SP=%08lx, %s syscall\n",
        l4_utcb_exc_pc(exc), exc->sp,

```



```

        is_alien_after_call(exc) ? " after" : "before");
    }

#endif

/* Measurement mode specifics.
 *
 * In measurement mode the code is less verbose and uses RDTSC for alien exception
 * performance measurement.
 */
#ifdef MEASURE

#include <l4/util/rdtsc.h>

static inline void
calibrate_timer(void)
{
    l4_calibrate_tsc(l4re_kip());
}

static inline void
print_timediff(l4_cpu_time_t start)
{
    e = l4_rdtsc();
    printf("time %lld\n", l4_tsc_to_ns(e - start));
}

static inline void
alien_sleep(void)
{
    l4_sleep(0);
}

static inline void
print_exc_state(l4_exc_regs_t const *exc)
{
    if (0)
        _print_exc_state(exc);
}

#else

static inline void
calibrate_timer(void)
{
}

static inline void
print_timediff(l4_cpu_time_t start)
{
    (void)start;
}

static inline l4_cpu_time_t
l4_rdtsc(void)
{
    return 0;
}

static inline void
alien_sleep(void)
{
    l4_sleep(1000);
}

static inline void
print_exc_state(l4_exc_regs_t const *exc)
{
    _print_exc_state(exc);
}

#endif

static char alien_thread_stack[8 « 10];
static l4_cap_idx_t alien;

static void alien_thread(void)
{
    while (1)
    {
        l4_ipc_call(0x1234 « L4_CAP_SHIFT, l4_utcb(),
                    l4_msgtag(0, 0, 0, 0), L4_IPC_NEVER);
        alien_sleep();
    }
}

```

```

}

int main(void)
{
    l4_msgtag_t tag;
    l4_cpu_time_t s;
    l4_utcb_t *u = l4_utcb();
    l4_exc_regs_t exc;
    l4_umword_t mr0, mr1;

    printf("Alien feature testing\n");

    l4_debugger_set_object_name(l4re_env()->main_thread, "alientest");

    /* Start alien thread */
    if (l4_is_invalid_cap(alien = l4re_util_cap_alloc()))
        return 1;

    l4_touch_rw(alien_thread_stack, sizeof(alien_thread_stack));

    tag = l4_factory_create_thread(l4re_env()->factory, alien);
    if (l4_error(tag))
        return 2;

    l4_debugger_set_object_name(alien, "alienth");

    l4_addr_t kumem;
    if (l4re_util_kumem_alloc(&kumem, 0, L4RE_THIS_TASK_CAP, l4re_env()->rm))
        return 3;

    l4_thread_control_start();
    l4_thread_control_pager(l4re_env()->main_thread);
    l4_thread_control_exc_handler(l4re_env()->main_thread);
    l4_thread_control_bind((l4_utcb_t *)kumem, L4RE_THIS_TASK_CAP);
    l4_thread_control_alien(1);
    tag = l4_thread_control_commit(alien);
    if (l4_error(tag))
        return 4;

    tag = l4_thread_ex_regs(alien,
                           (l4_umword_t)alien_thread,
                           (l4_umword_t)alien_thread_stack + sizeof(alien_thread_stack),
                           0);

    if (l4_error(tag))
        return 5;

    l4_sched_param_t sp = l4_sched_param(1, 0);
    tag = l4_scheduler_run_thread(l4re_env()->scheduler, alien, &sp);
    if (l4_error(tag))
        return 6;

    calibrate_timer();

    /* Pager/Exception loop */
    if (l4_msgtag_has_error(tag = l4_ipc_receive(alien, u, L4_IPC_NEVER)))
    {
        printf("l4_ipc_receive failed");
        return 7;
    }

    memcpy(&exc, l4_utcb_exc(), sizeof(exc));
    mr0 = l4_utcb_mr()->mr[0];
    mr1 = l4_utcb_mr()->mr[1];

    for (;;)
    {
        s = l4_rdtsc();

        if (l4_msgtag_is_exception(tag))
        {
            print_exc_state(&exc);
            tag = l4_msgtag(is_alien_after_call(&exc)
                           ? 0 : L4_PROTO_ALLOW_SYSCALL,
                           L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);
        }
        else
            printf("Umm, non-handled request (like PF): %lx %lx\n", mr0, mr1);

        memcpy(l4_utcb_exc(), &exc, sizeof(exc));

        /* Reply and wait */
        if (l4_msgtag_has_error(tag = l4_ipc_call(alien, u, tag, L4_IPC_NEVER)))
        {
            printf("l4_ipc_call failed\n");
            return 8;
        }
        memcpy(&exc, l4_utcb_exc(), sizeof(exc));
    }
}

```

```

        mr0 = l4_utcb_mr()->mr[0];
        mr1 = l4_utcb_mr()->mr[1];
        print_timediff(s);
    }

    return 0;
}

```

## 17.7 examples/sys/utcb-ipc/main.c

This example shows how to send IPC using the UTcb to store payload.

This example shows how to send IPC using the UTcb to store payload.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>,
 *               Björn Döbel <doebel@os.inf.tu-dresden.de>
 *               economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/ipc.h>
#include <l4/sys/thread.h>
#include <l4/sys/factory.h>
#include <l4/sys/utcb.h>
#include <l4/sys/task.h>
#include <l4/sys/vcon.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>
#include <l4/re/c/util/kumem_alloc.h>
#include <l4/util/thread.h>

#include <stdio.h>
#include <string.h>

static unsigned char stack2[8 « 10] __attribute__((aligned(8)));
static l4_cap_idx_t thread1_cap, thread2_cap;

static void vlogprintn(const char *s, int l)
{
    if (l > L4_VCON_WRITE_SIZE)
        l = L4_VCON_WRITE_SIZE;

    l4_vcon_send(L4_BASE_LOG_CAP, s, l);
}

static void vlogprint(const char *s)
{
    vlogprintn(s, strlen(s));
}

static void vlogprintc(const char c)
{
    vlogprintn(&c, 1);
}

static void thread1(void)
{
    l4_msg_regs_t *mr = l4_utcb_mr();
    l4_msgtag_t tag;
    int i, j;

    printf("Thread1 up (%p)\n", l4_utcb());

    for (i = 0; i < 10; i++)
    {
        for (j = 0; j < L4_UTCB_GENERIC_DATA_SIZE; j++)
            mr->mr[j] = 'A' + (i + j) % ('~' - 'A' + 1);
        tag = l4_msgtag(0, L4_UTCB_GENERIC_DATA_SIZE, 0, 0);
        if (l4_msgtag_has_error(l4_ipc_send(thread2_cap, l4_utcb(), tag, L4_IPC_NEVER)))
            printf("IPC-send error\n");
    }

    mr->mr[0] = 1;
    if (l4_msgtag_has_error(l4_ipc_send(thread2_cap, l4_utcb(), tag, L4_IPC_NEVER)))
        printf("IPC-send error\n");

    printf("Thread1 done\n");
}

```

```

}

L4UTIL_THREAD_STATIC_FUNC(thread2)
{
    l4_msgtag_t tag;
    l4_msg_regs_t mr;
    unsigned i;

    // No printf() here because this would require a working pthread environment!
    vlogprint("Thread2 up\n");

    while (1)
    {
        if (l4_msgtag_has_error(tag = l4_ipc_receive(thread1_cap, l4_utcb(), L4_IPC_NEVER)))
            vlogprint("IPC receive error\n");
        memcpy(&mr, l4_utcb_mr(), sizeof(mr));
        if (mr.mr[0] == 1) // exit notification
            break;
        vlogprint("Thread2 receive: ");
        for (i = 0; i < l4_msgtag_words(tag); i++)
            vlogprintc((char)mr.mr[i]);
        vlogprint("\n");
    }

    vlogprint("Thread2 done, switching to thread1\n");
    if (l4_msgtag_has_error(l4_ipc_send(thread1_cap, l4_utcb(),
                                      tag, L4_IPC_NEVER)))
        vlogprint("IPC-send error\n");

    // In theory this could hit if the above IPC send operation doesn't switch
    // to the other thread.
    __builtin_trap();
}

int main(void)
{
    l4_msgtag_t tag;

    thread1_cap = l4re_env()->main_thread;
    thread2_cap = l4re_util_cap_alloc();

    if (l4_is_invalid_cap(thread2_cap))
    {
        printf("Cannot allocate thread2 capability\n");
        return 1;
    }

    tag = l4_factory_create_thread(l4re_env()->factory, thread2_cap);
    if (l4_error(tag))
    {
        printf("Cannot create thread2\n");
        return 2;
    }

    l4_addr_t kumem;
    if (l4re_util_kumem_alloc(&kumem, 0, L4RE_THIS_TASK_CAP, l4re_env()->rm))
    {
        printf("Cannot allocate UTCB for thread2\n");
        return 3;
    }

    l4_thread_control_start();
    l4_thread_control_pager(l4re_env()->rm);
    l4_thread_control_exc_handler(l4re_env()->rm);
    l4_thread_control_bind((l4_utcb_t *)kumem, L4RE_THIS_TASK_CAP);
    tag = l4_thread_control_commit(thread2_cap);
    if (l4_error(tag))
    {
        printf("Cannot set thread2 thread parameters\n");
        return 4;
    }

    tag = l4_thread_ex_regs(thread2_cap,
                           (l4_umword_t)thread2,
                           (l4_umword_t)(stack2 + sizeof(stack2)), 0);
    if (l4_error(tag))
    {
        printf("Cannot set thread2 IP/SP\n");
        return 5;
    }

    l4_sched_param_t sp = l4_sched_param(1, 0);
    tag = l4_scheduler_run_thread(l4re_env()->scheduler, thread2_cap, &sp);
    if (l4_error(tag))
    {
        printf("Cannot start thread2\n");
        return 6;
    }
}

```

```

    }

    thread1();

    if (l4_msgtag_has_error(l4_ipc_receive(thread2_cap, l4_utcb(),
                                           L4_IPC_NEVER)))
        printf("IPC-receive error\n");

    l4_task_unmap(L4RE_THIS_TASK_CAP,
                  l4_obj_fpage(thread2_cap, 0, L4_FPAGE_RWX),
                  L4_FP_ALL_SPACES);

    printf("Terminated thread2. Terminating.\n");
    return 0;
}

```

## 17.8 examples/sys/ux-vhw/main.c

This example shows how to iterate the virtual hardware descriptors under Fiasco-UX.

This example shows how to iterate the virtual hardware descriptors under Fiasco-UX.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/ipc.h>
#include <l4/sys/vhw.h>
#include <l4/util/util.h>
#include <l4/util/kip.h>
#include <l4/re/env.h>

#include <stdlib.h>
#include <stdio.h>

static void print_entry(struct l4_vhw_entry *e)
{
    printf("type: %d mem start: %08lx end: %08lx\n"
           "irq: %d pid %d\n",
           e->type, e->mem_start, e->mem_size,
           e->irq_no, e->provider_pid);
}

int main(void)
{
    l4_kernel_info_t const *kip = l4re_kip();
    struct l4_vhw_descriptor *vhw;
    int i;

    if (!kip)
    {
        printf("KIP not available!\n");
        return 1;
    }

    if (!l4util_kip_kernel_is_ux(kip))
    {
        printf("This example is for Fiasco-UX only.\n");
        return 1;
    }

    vhw = l4_vhw_get(kip);

    printf("kip at %p, vhw at %p\n", kip, vhw);
    printf("magic: %08x, version: %08x, count: %02d\n",
           vhw->magic, vhw->version, vhw->count);

    for (i = 0; i < vhw->count; i++)
        print_entry(l4_vhw_get_entry(vhw, i));

    return 0;
}

```

## 17.9 examples/sys/isr/main.c

Example of an interrupt service routine.

Example of an interrupt service routine.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *           Alexander Warg <warg@os.inf.tu-dresden.de>,
 *           Björn Döbel <doebel@os.inf.tu-dresden.de>
 *           economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * This example shall show how to connect to an interrupt, receive interrupt
 * events and detach again. As the interrupt source we'll use the virtual
 * key interrupt -- pin 0 of the log capability.
 */

#include <l4/re/c/util/cap_alloc.h>
#include <l4/re/c/namespace.h>
#include <l4/sys/factory.h>
#include <l4/sys/icu.h>
#include <l4/sys/irq.h>
#include <l4/sys/vcon.h>
#include <l4/sys/utcb.h>

#include <stdio.h>

int main(void)
{
    int const irqno = 0;
    l4_cap_idx_t irqcap, icucap;
    l4_vcon_attr_t attr;
    long err;

    icucap = l4re_env()->log;

    /* Get a free capability slot for the ICU capability. */
    if (l4_is_invalid_cap(icucap))
    {
        printf("Did not find the Vlog ICU.\n");
        return 1;
    }

    /* Get another free capability slot for the corresponding IRQ object. */
    if (l4_is_invalid_cap(irqcap = l4re_util_cap_alloc()))
    {
        printf("Cannot allocate capability slot.\n");
        return 1;
    }

    /* Create IRQ object. */
    if ((err = l4_error(l4_factory_create_irq(l4re_env()->factory, irqcap))))
    {
        printf("Could not create IRQ object: %ld (%s).\n", err, l4sys_errtostr(err));
        return 1;
    }

    /*
     * Bind the recently allocated IRQ object to the IRQ number irqno as provided
     * by the ICU.
     */
    if ((err = l4_error(l4_icu_bind(icucap, irqno, irqcap))))
    {
        printf("Could not bind IRQ%d to ICU: %ld (%s).\n", irqno, err, l4sys_errtostr(err));
        return 1;
    }

    if ((err = l4_error(l4_vcon_get_attr(icucap, &attr))))
    {
        printf("Could not get Vcon attributes: %ld (%s).\n", err, l4sys_errtostr(err));
        return 1;
    }

    /* Disable echo at Vcon console. */
    attr.l_flags &= ~L4_VCON_ECHO;

    if ((err = l4_error(l4_vcon_set_attr(icucap, &attr))))
    {
        printf("Could not set Vcon attributes: %ld (%s).\n", err, l4sys_errtostr(err));
        return 1;
    }
}

```

```

    }

    printf("Vcon echo disabled.\n");

    /* Bind ourselves to the IRQ. Define the IPC label which is sent if an IRQ
     * IPC arrives. */
    if ((err = l4_error(l4_rcv_ep_bind_thread(irqcap, l4re_env()->main_thread, 0x1234)))
    {
        printf("Could not bind to IRQ%d: %ld (%s).\n", irqno, err, l4sys_errtostr(err));
        return 1;
    }

    printf("Attached to key IRQ %d.\nPress keys now, Shift-Q to exit.\n", irqno);

    /* IRQ receive loop. */
    while (1)
    {
        /* Wait for the interrupt to happen. If we received an IRQ, the label
         * return code is set to 0. If we didn't receive an IRQ, the error flag
         * in the message tag is set and l4_error() reads the IPC error code from
         * the UTCB. */
        l4_umword_t label;
        if ((err = l4_error(l4_irq_wait(irqcap, &label, L4_IPC_NEVER)))
        {
            printf("Could not receive IRQ: %ld (%s).\n", err, l4sys_errtostr(err));
        }
        else
        {
            char buf[128];
            int n;

            if (label != 0x1234)
            {
                printf("Unexpected label %0lx -- ignoring interrupt.\n", label);
                continue;
            }

            /* Process the interrupt -- may do a 'break' */
            printf("Got IRQ with expected label 0x%lX.\n", label);
            n = l4_vcon_read(icucap, buf, sizeof(buf));
            if (n < 0)
                printf("Could not read from Vcon interface: %d (%s).\n", n, l4sys_errtostr(n));
            else
            {
                unsigned i;
                int terminate = 0;
                for (i = 0; i < (unsigned)n && i < sizeof(buf); ++i)
                {
                    int c = (unsigned char)buf[i];
                    if (c >= 32 && c < 128) // Filter UTF-8 encodings.
                        printf("Got key '%c'.\n", c);
                    else
                        printf("Got keycode %d.\n", c);
                    if (buf[i] == 'Q')
                        terminate = 1;
                }

                if (terminate)
                    break;
            }
        }
    }

    /* We're done, detach from the interrupt. */
    if ((err = l4_error(l4_irq_detach(irqcap)))
    {
        printf("Could not detach from IRQ: %ld (%s).\n", err, l4sys_errtostr(err));
    }

    printf("Application terminated.\n");
    return 0;
}

```

## 17.10 examples/clntsrv/server.cc

Client/Server example using C++ infrastructure – Server implementation.

Client/Server example using C++ infrastructure – Server implementation.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the

```

```

* GNU General Public License 2.
* Please see the COPYING-GPL-2 file for details.
*/
#include <stdio.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/re/util/br_manager>
#include <l4/sys/cxx/ipc_epiface>

#include "shared.h"

static L4Re::Util::Registry_server<> server;

class Calculation_server : public L4::Epiface_t<Calculation_server, Calc>
{
public:
    int op_sub(Calc::Rights, l4_uint32_t a, l4_uint32_t b, l4_uint32_t &res)
    {
        res = a - b;
        return 0;
    }

    int op_neg(Calc::Rights, l4_uint32_t a, l4_uint32_t &res)
    {
        res = -a;
        return 0;
    }
};

int
main()
{
    static Calculation_server calc;

    // Register calculation server
    if (!server.registry()->register_obj(&calc, "calc_server").is_valid())
    {
        printf("Could not register my service, is there a 'calc_server' in the caps table?\n");
        return 1;
    }

    printf("Welcome to the calculation server!\n"
           "I can do subtractions and negations.\n");

    // Wait for client requests
    server.loop();

    return 0;
}

```

## 17.11 examples/clntsrv/client.cc

Client/Server example using C++ infrastructure – Client implementation.

Client/Server example using C++ infrastructure – Client implementation.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/err.h>
#include <l4/sys/types.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>

#include <stdio.h>
#include "shared.h"

int
main()
{
    L4::Cap<Calc> server = L4Re::Env::env()->get_cap<Calc>("calc_server");
    if (!server.is_valid())
    {

```



```

        printf("Could not get server capability!\n");
        return 1;
    }

    l4_uint32_t val1 = 8;
    l4_uint32_t val2 = 5;

    printf("Asking for %d - %d\n", val1, val2);
    if (server->sub(val1, val2, &val1))
    {
        printf("Error talking to server\n");
        return 1;
    }
    printf("Result of subtract call: %d\n", val1);

    printf("Asking for -%d\n", val1);
    if (server->neg(val1, &val1))
    {
        printf("Error talking to server\n");
        return 1;
    }
    printf("Result of negate call: %d\n", val1);

    return 0;
}

```

## 17.12 examples/clntsrv/clntsrv.cfg

Sample configuration file for the client/server example.

Sample configuration file for the client/server example.

```

-- vim:set ft=lua:

-- Include L4 functionality
local L4 = require("L4");

-- Some shortcut for less typing
local ld = L4.default_loader;

-- Channel for the two programs to talk to each other.
local calc_server = ld:new_channel();

-- The server program, getting the channel in server mode.
ld:start({ caps = { calc_server = calc_server:svr() },
        log = { "server", "blue" } },
        "rom/ex_clntsrv-server");

-- The client program, getting the 'calc_server' channel to be able to talk
-- to the server. The client will be started with a green log output.
ld:start({ caps = { calc_server = calc_server },
        log = { "client", "green" } },
        "rom/ex_clntsrv-client");

```

## 17.13 examples/libs/l4re/c/ma+rm.c

Coarse grained memory allocation, in C.

Coarse grained memory allocation, in C.

```

/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#include <l4re/c/mem_alloc.h>
#include <l4re/c/rm.h>
#include <l4re/c/util/cap_alloc.h>
#include <l4/sys/err.h>
#include <stdio.h>

```

```

#include <string.h>

static int allocate_mem(unsigned long size_in_bytes, unsigned long flags,
                        void **virt_addr)
{
    int r;
    l4re_ds_t ds;

    /* Allocate a free capability index for our data space */
    ds = l4re_util_cap_alloc();
    if (l4_is_invalid_cap(ds))
        return -L4_ENOMEM;

    size_in_bytes = l4_trunc_page(size_in_bytes);

    /* Allocate memory via a dataspace */
    if ((r = l4re_ma_alloc(size_in_bytes, ds, flags))
        return r;

    /* Make the dataspace visible in our address space */
    *virt_addr = 0;
    if ((r = l4re_rm_attach(virt_addr, size_in_bytes,
                           L4RE_RM_F_SEARCH_ADDR | L4RE_RM_F_RWX, ds, 0,
                           flags & L4RE_MA_SUPER_PAGES
                           ? L4_SUPERPAGESHIFT : L4_PAGESHIFT)))
    {
        /* Free dataspace again */
        l4re_util_cap_free_um(ds);
        return r;
    }

    /* Done, virtual address is in virt_addr */
    return 0;
}

static int free_mem(void *virt_addr)
{
    int r;
    l4re_ds_t ds;

    /* Detach memory from our address space */
    if ((r = l4re_rm_detach_ds(virt_addr, &ds)))
        return r;

    /* Free memory at our memory allocator */
    l4re_util_cap_free_um(ds);

    /* All went ok */
    return 0;
}

int main(void)
{
    void *virt;

    /* Allocate memory: 16k Bytes (usually) */
    if (allocate_mem(4 * L4_PAGESIZE, 0, &virt))
        return 1;

    printf("Allocated memory.\n");

    /* Do something with the memory */
    memset(virt, 0x12, 4 * L4_PAGESIZE);

    printf("Touched memory.\n");

    /* Free memory */
    if (free_mem(virt))
        return 2;

    printf("Freed and done. Bye.\n");

    return 0;
}

```

## 17.14 examples/libs/l4re/c++/mem\_alloc/ma+rm.cc

Coarse grained memory allocation, in C++.

Coarse grained memory allocation, in C++.

/\*

```

* (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
*   economic rights: Technische Universität Dresden (Germany)
*
* This file is part of TUD:OS and distributed under the terms of the
* GNU General Public License 2.
* Please see the COPYING-GPL-2 file for details.
*/

#include <l4/re/mem_alloc>
#include <l4/re/rm>
#include <l4/re/env>
#include <l4/re/dataspace>
#include <l4/re/util/cap_alloc>
#include <l4/sys/err.h>
#include <cstdio>
#include <cstring>

static int allocate_mem(unsigned long size_in_bytes, unsigned long flags,
                        void **virt_addr)
{
    int r;
    L4::Cap<L4Re::Dataspace> d;

    /* Allocate a free capability index for our data space */
    d = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
    if (!d.is_valid())
        return -L4_ENOMEM;

    size_in_bytes = l4_trunc_page(size_in_bytes);

    /* Allocate memory via a dataspace */
    if ((r = L4Re::Env::env()->mem_alloc()->alloc(size_in_bytes, d, flags)))
        return r;

    /* Make the dataspace visible in our address space */
    *virt_addr = 0;
    if ((r = L4Re::Env::env()->rm()->attach(virt_addr, size_in_bytes,
                                           L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
                                           L4::Ipc::make_cap_rw(d), 0,
                                           flags & L4Re::Mem_alloc::Super_pages
                                           ? L4_SUPERPAGESHIFT : L4_PAGESHIFT)))
        return r;

    /* Done, virtual address is in virt_addr */
    return 0;
}

static int free_mem(void *virt_addr)
{
    int r;
    L4::Cap<L4Re::Dataspace> ds;

    /* Detach memory from our address space */
    if ((r = L4Re::Env::env()->rm()->detach(virt_addr, &ds)))
        return r;

    /* Release and return capability slot to allocator */
    L4Re::Util::cap_alloc.free(ds, L4Re::Env::env()->task().cap());

    /* All went ok */
    return 0;
}

int main(void)
{
    void *virt;

    /* Allocate memory: 16k Bytes (usually) */
    if (allocate_mem(4 * L4_PAGE_SIZE, 0, &virt))
        return 1;

    printf("Allocated memory.\n");

    /* Do something with the memory */
    memset(virt, 0x12, 4 * L4_PAGE_SIZE);

    printf("Touched memory.\n");

    /* Free memory */
    if (free_mem(virt))
        return 2;

    printf("Freed and done. Bye.\n");

    return 0;
}

```

## 17.15 examples/libs/l4re/c++/shared\_ds/ds\_clnt.cc

Sharing memory between applications, client side.

Sharing memory between applications, client side.

```

/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#include <l4re/util/cap_alloc> // L4::Cap
#include <l4re/dataspace>     // L4Re::Dataspace
#include <l4re/rm>             // L4::Rm
#include <l4re/env>            // L4::Env
#include <l4/sys/cache.h>

#include <cstring>
#include <cstdio>
#include <unistd.h>

#include "interface.h"

int main()
{
    /*
     * Try to get server interface cap.
     */

    L4::Cap<My_interface> svr = L4Re::Env::env()->get_cap<My_interface>("shm");
    if (!svr.is_valid())
    {
        printf("Could not get the server capability\n");
        return 1;
    }

    /*
     * Alloc data space cap slot
     */
    L4::Cap<L4Re::Dataspace> ds = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
    if (!ds.is_valid())
    {
        printf("Could not get capability slot!\n");
        return 1;
    }

    /*
     * Alloc server notifier IRQ cap slot
     */
    L4::Cap<L4::Irq> irq = L4Re::Util::cap_alloc.alloc<L4::Irq>();
    if (!irq.is_valid())
    {
        printf("Could not get capability slot!\n");
        return 1;
    }

    /*
     * Request shared data-space cap.
     */
    if (svr->get_shared_buffer(ds, irq))
    {
        printf("Could not get shared memory dataspace!\n");
        return 1;
    }

    /*
     * Attach to arbitrary region
     */
    char *addr = 0;
    int err = L4Re::Env::env()->rm()->attach(&addr, ds->size(),
                                           L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
                                           L4::Ipc::make_cap_rw(ds));

    if (err < 0)
    {
        printf("Error attaching data space: %s\n", l4sys_errtostr(err));
        return 1;
    }

    printf("Content: %s\n", addr);

    // wait a bit for the demo effect

```

```

printf("Sleeping a bit...\n");
sleep(1);

/*
 * Fill in new stuff
 */
memset(addr, 0, ds->size());
char const * const msg = "Hello from client, too!";
printf("Setting new content in shared memory\n");
snprintf(addr, strlen(msg)+1, msg);
l4_cache_clean_data((unsigned long)addr,
                    (unsigned long)addr + strlen(msg) + 1);

// notify the server
irq->trigger();

/*
 * Detach region containing addr, result should be Detached_ds (other results
 * only apply if we split regions etc.).
 */
err = L4Re::Env::env()->rm()->detach(addr, 0);
if (err)
    printf("Failed to detach region\n");

/* Free objects and capabilities, just for completeness. */
L4Re::Util::cap_alloc.free(ds, L4Re::This_task);
L4Re::Util::cap_alloc.free(irq, L4Re::This_task);

return 0;
}

```

## 17.16 examples/libs/l4re/c++/shared\_ds/ds\_srv.cc

Sharing memory between applications, server/creator side.

Sharing memory between applications, server/creator side.

```

/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#include <l4/re/env>
#include <l4/re/error_helper>
#include <l4/re/namespace>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/re/dataspace>
#include <l4/cxx/ipc_server>
#include <l4/util/util.h>

#include <l4/sys/typeinfo_svr>

#include <cstring>
#include <cstdio>
#include <unistd.h>
#include <pthread.h>
#include <pthread-l4.h>
#include <thread>

#include "interface.h"

class My_server_obj : public L4::Server_object_t<L4::Kobject>
{
private:
    L4::Cap<L4Re::Dataspace> _shm;
    L4::Cap<L4::Irq> _irq;

public:
    explicit My_server_obj(L4::Cap<L4Re::Dataspace> shm, L4::Cap<L4::Irq> irq)
        : _shm(shm), _irq(irq)
    {}

    int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);

```

```

};

int My_server_obj::dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios)
{
    // we don't care about the original object reference, however
    // we could read out the access rights from the lowest 2 bits
    (void) obj;

    l4_msgtag_t t;
    ios » t; // extract the tag

    switch (t.label())
    {
        case L4::Meta::Protocol:
            // handle the meta protocol requests, implementing the
            // runtime dynamic type system for L4 objects.
            return L4::Util::handle_meta_request<My_interface>(ios);
        case 0:
            // since we have just one operation we have no opcode dispatch,
            // and just return the data-space and the notifier IRQ capabilities
            ios « _shm « _irq;
            return 0;
        default:
            // every other protocol is not supported.
            return -L4_EBADPROTO;
    }
}

class Shm_observer : public L4::Irq_handler_object
{
private:
    char *_shm;

public:
    explicit Shm_observer(char *shm)
        : _shm(shm)
    {}

    int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);
};

int Shm_observer::dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios)
{
    // We don't care about the original object reference, however
    // we could read out the access rights from the lowest 2 bits
    (void) obj;

    // Since we end up here in this function, we got a 'message' from the IRQ
    // that is bound to us. The 'ios' stream won't contain any valuable info.
    (void) ios;

    printf("Client sent us: %s\n", _shm);

    return 0;
}

enum
{
    DS_SIZE = 4 « 12,
};

static char *get_ds(L4::Cap<L4Re::Dataspace> *_ds)
{
    *_ds = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
    if (!(*_ds).is_valid())
    {
        printf("Dataspace allocation failed.\n");
        return 0;
    }

    int err = L4Re::Env::env()->mem_alloc()->alloc(DS_SIZE, *_ds, 0);
    if (err < 0)
    {
        printf("mem_alloc->alloc() failed.\n");
        L4Re::Util::cap_alloc.free(*_ds);
        return 0;
    }

    /*
     * Attach DS to local address space
     */
    char *_addr = 0;
    err = L4Re::Env::env()->rm()->attach(&_addr, (*_ds)->size(),
                                         L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
                                         L4::Ipc::make_cap_rw(*_ds));
}

```

```

if (err < 0)
{
    printf("Error attaching data space: %s\n", l4sys_errtostr(err));
    L4Re::Util::cap_alloc.free(*_ds);
    return 0;
}

/*
 * Success! Write something to DS.
 */
printf("Attached DS\n");
static char const * const msg = "[DS] Hello from server!";
snprintf(_addr, strlen(msg) + 1, msg);

return _addr;
}

static void *server_thread(void *)
{
    L4::Cap<L4::Thread> l4_thread = Pthread::L4::cap(pthread_self());
    L4Re::Util::Registry_server<> server(l4_thread, L4Re::Env::env()->factory());

    L4::Cap<L4Re::Dataspace> ds;
    char *addr;

    if (!(addr = get_ds(&ds)))
        return nullptr;

    // First the IRQ handler, because we need it in the My_server_obj object
    Shm_observer observer(addr);

    // Registering the observer as an IRQ handler, this allocates an
    // IRQ object using the factory of our server.
    L4::Cap<L4::Irq> irq = server.registry()->register_irq_obj(&observer);

    // Now the initial server object shared with the client via our parent.
    // it provides the data-space and the IRQ capabilities to a client.
    My_server_obj server_obj(ds, irq);

    // Registering the server object to the capability 'shm' in our the L4Re::Env.
    // This capability must be provided by the parent. (see the shared_ds.lua)
    server.registry()->register_obj(&server_obj, "shm");

    // Run our server loop.
    server.loop();
}

int main()
{
    pthread_attr_t pattr;

    if (pthread_attr_init(&pattr))
        L4Re::throw_error(-L4_ENOMEM, "Initialize pthread attributes");

    pthread_t thr;
    L4Re::chksys(pthread_create(&thr, &pattr, server_thread, nullptr),
        "Create server thread");
    L4Re::chksys(pthread_attr_destroy(&pattr), "Destroy pthread attributes");

    l4_sleep_forever();

    return 0;
}

```

## 17.17 examples/libs/l4re/c++/shared\_ds/shared\_ds.cfg

Sharing memory between applications, configuration file.

Sharing memory between applications, configuration file.

```

-- Include L4 functionality
local L4 = require("L4");

-- Create a channel from the client to the server
local channel = L4.default_loader:new_channel();

-- Start the server, giving the channel with full server rights.
-- The server will have a yellow log output.
L4.default_loader:start(
{

```

```

    caps = { shm = channel:svr() },
    log = { "server", "yellow" }
},
"rom/ex_l4re_ds_srv"
);

-- Start the client, giving it the channel with read only rights. The
-- log output will be green.
L4.default_loader:start(
{
    caps = { shm = channel },
    log = { "client", "green" },
    l4re_dbg = L4.Dbg.Warn
},
"rom/ex_l4re_ds_clnt"
);

```

## 17.18 examples/libs/l4re/streammap/server.cc

Client/Server example showing how to map a page to another task – Server implementation.

Client/Server example showing how to map a page to another task – Server implementation. Note that there's also a shared memory library that supplies this functionality in more convenient way.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <stdio.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/cxx/ipc_server>

#include "shared.h"

static char page_to_map[L4_PAGESIZE] __attribute__((aligned(L4_PAGESIZE)));

static L4Re::Util::Registry_server<> server;

class Smap_server : public L4::Server_object_t<Mapper>
{
public:
    int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);
};

int
Smap_server::dispatch(l4_umword_t, L4::Ipc::Iostream &ios)
{
    l4_msgtag_t t;
    ios » t;

    // We're only talking the Map_example protocol
    if (t.label() != Mapper::Protocol)
        return -L4_EBADPROTO;

    L4::Opcode opcode;
    ios » opcode;

    switch (opcode)
    {
    case Mapper::Do_map:
        l4_addr_t snd_base;
        ios » snd_base;
        // put something into the page to read it out at the other side
        sprintf(page_to_map, sizeof(page_to_map), "Hello from the server!");
        printf("Sending to client\n");
        // send page
        ios « L4::Ipc::Snd_fpage::mem((l4_addr_t)page_to_map, L4_PAGESHIFT,
                                     L4_FPAGE_RO, snd_base);

        return L4_EOK;
    default:
        return -L4_ENOSYS;
    }
}

int

```



```

main()
{
    static Smap_server smap;

    // Register server
    if (!server.registry()->register_obj(&smap, "smap").is_valid())
    {
        printf("Could not register my service, read-only namespace?\n");
        return 1;
    }

    printf("Welcome to the memory map example server!\n");

    // Wait for client requests
    server.loop();

    return 0;
}

```

## 17.19 examples/libs/l4re/streammap/client.cc

Client/Server example showing how to map a page to another task – Client implementation.

Client/Server example showing how to map a page to another task – Client implementation. Note that there's also a shared memory library that supplies this functionality in more convenient way.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/err.h>
#include <l4/sys/types.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/cxx/ipc_stream>

#include <stdio.h>

#include "shared.h"

static int
func_smap_call(L4::Cap<void> const &server)
{
    L4::Ipc::Iostream s(l4_utcb());
    l4_addr_t addr = 0;
    int err;

    if ((err = L4Re::Env::env()->rm()->reserve_area(&addr, L4_PAGESIZE,
                                                    L4Re::Rm::F::Search_addr)))
    {
        printf("The reservation of one page within our virtual memory failed with %d\n", err);
        return 1;
    }

    s << L4::Opcode(Mapper::Do_map)
      << (l4_addr_t)addr;
    s << L4::Ipc::Rcv_fpage::mem((l4_addr_t)addr, L4_PAGESHIFT, 0);
    int r = l4_error(s.call(server.cap(), Mapper::Protocol));
    if (r)
        return r; // failure

    printf("String sent by server: %s\n", (char *)addr);

    return 0; // ok
}

int
main()
{
    L4::Cap<void> server = L4Re::Env::env()->get_cap<void>("smap");
    if (!server.is_valid())
    {
        printf("Could not get capability slot!\n");
        return 1;
    }
}

```

```

    }

    printf("Asking for page from server\n");

    if (func_smap_call(server))
    {
        printf("Error talking to server\n");
        return 1;
    }
    printf("It worked!\n");

    L4Re::Util::cap_alloc.free(server, L4Re::This_task);

    return 0;
}

```

## 17.20 examples/libs/l4re/streammap/streammap.cfg

Sample configuration file for the client/server map example.

Sample configuration file for the client/server map example.

```

-- vim:set ft=lua:

-- Include L4 functionality
local L4 = require("L4");

-- Channel for the communication between the server and the client.
local smap_channel = L4.default_loader:new_channel();

-- The server program, using the 'smap' channel in server
-- mode. The log prefix will be 'server', colored yellow.
L4.default_loader:start({ caps = { smap = smap_channel:svr() },
    log = { "server", "yellow" } },
    "rom/ex_smap-server");

-- The client program.
-- It is given the 'smap' channel to be able to talk to the server.
-- The log prefix will be 'client', colored green.
L4.default_loader:start({ caps = { smap = smap_channel },
    log = { "client", "green" } },
    "rom/ex_smap-client");

```

## 17.21 examples/libs/libirq/loop.c

libirq usage example using a self-created thread.

libirq usage example using a self-created thread.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#include <l4/irq/irq.h>
#include <l4/util/util.h>
#include <stdio.h>
#include <pthread.h>

enum { IRQ_NO = 17 };

static void isr_handler(void)
{
    printf("Got IRQ %d\n", IRQ_NO);
}

static void *isr_thread(void *data)
{
    l4irq_t *irq;

```

```

(void) data;

if (! (irq = l4irq_attach (IRQ_NO)))
    return NULL;

while (1)
{
    if (l4irq_wait (irq))
        continue;
    isr_handler ();
}

return NULL;
}

int main(void)
{
    pthread_t thread;

    if (pthread_create(&thread, NULL, isr_thread, NULL))
        return 1;

    l4_sleep_forever();
    return 0;
}

```

## 17.22 examples/libs/libirq/async\_isr.c

libirq usage example using asynchronous ISR handler functionality.

libirq usage example using asynchronous ISR handler functionality.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * This example shall show how to use the libirq.
 */

#include <l4/irq/irq.h>
#include <l4/util/util.h>

#include <stdio.h>

enum { IRQ_NO = 17 };

static void isr_handler(void *data)
{
    (void) data;
    printf("Got IRQ %d\n", IRQ_NO);
}

int main(void)
{
    const int seconds = 5;
    l4irq_t *irqdesc;

    if (!(irqdesc = l4irq_request (IRQ_NO, isr_handler, 0, 0xff, 0)))
    {
        printf("Requesting IRQ %d failed\n", IRQ_NO);
        return 1;
    }

    printf("Attached to key IRQ %d\nPress keys now, will terminate in %d seconds\n",
           IRQ_NO, seconds);

    l4_sleep(seconds * 1000);

    if (l4irq_release (irqdesc))
    {
        printf("Failed to release IRQ\n");
        return 1;
    }

    printf("Bye\n");
    return 0;
}

```

## 17.23 examples/sys/migrate/thread\_migrate.cc

Thread migration example.

Thread migration example.

```

/*
 * (c) 2008-2009 Author(s)
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/scheduler>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>

#include <pthread-l4.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

enum { NR_THREADS = 12 };
static L4::Cap<L4::Thread> threads[NR_THREADS];
static l4_umword_t      cpu_map, cpu_nrs;

/* Function for the threads. The content is not really relevant, so lets
 * just sleep around a bit. */
static void *thread_fn(void *)
{
    while (1)
        sleep(1);

    return 0;
}

/* Check how many CPUs we have available.
 */
static int check_cpus(void)
{
    l4_sched_cpu_set_t cs = l4_sched_cpu_set(0, 0);

    if (l4_error(L4Re::Env::env()->scheduler()->info(&cpu_nrs, &cs)) < 0)
        return 1;

    cpu_map = cs.map;

    printf("%ld maximal supported CPUs.\n", cpu_nrs);
    if (cpu_nrs >= L4_MWORD_BITS)
    {
        printf("Will only handle %ld CPUs.\n", cpu_nrs);
        cpu_nrs = L4_MWORD_BITS;
    }
    else if (cpu_nrs == 1)
        printf("Only found 1 CPU.\n");

    return cpu_nrs < 2;
}

/* Create a couple of threads and store their capabilities in an array */
static int create_threads(void)
{
    unsigned i;

    for (i = 0; i < NR_THREADS; ++i)
    {
        pthread_t t;

        if (pthread_create(&t, NULL, thread_fn, NULL))
            return 1;

        threads[i] = L4::Cap<L4::Thread>(pthread_l4_cap(t));
    }
    printf("Created %d threads.\n", NR_THREADS);
    return 0;
}

/* Helper function to get the next CPU */
static unsigned get_next_cpu(unsigned c)
{
    unsigned x = c;
    for (;;)
    {

```

```

    x = (x + 1) % cpu_nrs;
    if (L4Re::Env::env()->scheduler()->is_online(x))
        return x;
    if (x == c)
        return c;
}

/* Function that shuffles the threads on the available CPUs */
static void shuffle(void)
{
    unsigned start = 0;
    while (1)
    {
        unsigned t;
        unsigned c = start;
        for (t = 0; t < NR_THREADS; ++t)
        {
            l4_sched_param_t sp = l4_sched_param(20);
            c = get_next_cpu(c);
            sp.affinity = l4_sched_cpu_set(c, 0);
            if (l4_error(L4Re::Env::env()->scheduler()->run_thread(threads[t], sp)))
                printf("Error migrating thread%02d to CPU%02d\n", t, c);
            printf("Migrated Thread%02d -> CPU%02d\n", t, c);
        }

        start++;
        if (start == cpu_nrs)
            start = 0;
        sleep(1);
    }
}

int main(void)
{
    if (check_cpus())
        return 1;

    if (create_threads())
        return 1;

    shuffle();

    return 0;
}

```

## 17.24 examples/sys/migrate/thread\_migrate.cfg

Sample configuration file for the thread migration example.

Sample configuration file for the thread migration example.

```

-- vim:set ft=lua:

local L4 = require("L4");

-- The log prefix will be 'migrate', colored green.
L4.default_loader:start({ log = { "migrate", "green" } },
    "rom/ex_thread_migrate");

```

## 17.25 tmpfs/lib/src/fs.cc

Example file system for [L4Re::Vfs](#).

Example file system for [L4Re::Vfs](#).

```

/*
 * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.

```

```

*/

#include <l4/l4re_vfs/backend>
#include <l4/cxx/string>
#include <l4/cxx/avl_tree>

#include <sys/stat.h>
#include <sys/ioctl.h>
#include <dirent.h>

#include <cstdio>
#include <cstdlib>
#include <cstring>

namespace {

using namespace L4Re::Vfs;
using cxx::Ref_ptr;

class File_data
{
public:
    File_data() : _buf(0), _size(0) {}

    unsigned long put(unsigned long offset,
                     unsigned long bufsize, void *srcbuf);
    unsigned long get(unsigned long offset,
                     unsigned long bufsize, void *dstbuf);

    unsigned long size(unsigned long offset);
    unsigned long size() const { return _size; }

    ~File_data() throw() { free(_buf); }

private:
    void *_buf;
    unsigned long _size;
};

unsigned long
File_data::put(unsigned long offset, unsigned long bufsize, void *srcbuf)
{
    if (offset + bufsize > _size)
        size(offset + bufsize);

    if (!_buf)
        return 0;

    memcpy((char *)_buf + offset, srcbuf, bufsize);
    return bufsize;
}

unsigned long
File_data::get(unsigned long offset, unsigned long bufsize, void *dstbuf)
{
    unsigned long s = bufsize;

    if (offset > _size)
        return 0;

    if (offset + bufsize > _size)
        s = _size - offset;

    memcpy(dstbuf, (char *)_buf + offset, s);
    return s;
}

unsigned long
File_data::size(unsigned long offset)
{
    if (offset != _size)
    {
        _size = offset;
        _buf = realloc(_buf, _size);
    }

    if (!_buf)
        return 0;
    return -ENOSPC;
}

class Node : public cxx::Avl_tree_node
{
public:
    Node(const char *path, mode_t mode)
        : _ref_cnt(0), _path(strdup(path))

```

```

{
    memset(&_info, 0, sizeof(_info));
    _info.st_mode = mode;
}

const char *path() const { return _path; }
struct stat64 *info() { return &_info; }

void add_ref() throw() { ++_ref_cnt; }
int remove_ref() throw() { return --_ref_cnt; }

bool is_dir() const { return S_ISDIR(_info.st_mode); }

virtual ~Node() { free(_path); }

private:
    int      _ref_cnt;
    char     *_path;
    struct stat64 _info;
};

struct Node_get_key
{
    typedef cxx::String Key_type;
    static Key_type key_of(Node const *n)
    { return n->path(); }
};

struct Path_avl_tree_compare
{
    bool operator () (const char *l, const char *r) const
    { return strcmp(l, r) < 0; }
    bool operator () (const cxx::String l, const cxx::String r) const
    {
        int v = strncmp(l.start(), r.start(), cxx::min(l.len(), r.len()));
        return v < 0 || (v == 0 && l.len() < r.len());
    }
};

class Pers_file : public Node
{
public:
    Pers_file(const char *name, mode_t mode)
        : Node(name, (mode & 0777) | __S_IFREG) {}
    File_data const &data() const { return _data; }
    File_data &data() { return _data; }
private:
    File_data _data;
};

class Pers_dir : public Node
{
private:
    typedef cxx::Avl_tree<Node, Node_get_key, Path_avl_tree_compare> Tree;
    Tree _tree;

public:
    Pers_dir(const char *name, mode_t mode)
        : Node(name, (mode & 0777) | __S_IFDIR) {}
    Ref_ptr<Node> find_path(cxx::String);
    bool add_node(Ref_ptr<Node> const &);

    typedef Tree::Const_iterator Const_iterator;
    Const_iterator begin() const { return _tree.begin(); }
    Const_iterator end() const { return _tree.end(); }
};

Ref_ptr<Node> Pers_dir::find_path(cxx::String path)
{
    return cxx::ref_ptr(_tree.find_node(path));
}

bool Pers_dir::add_node(Ref_ptr<Node> const &n)
{
    bool e = _tree.insert(n.ptr()).second;
    if (e)
        n->add_ref();
    return e;
}

class Tmpfs_dir : public Be_file
{
public:
    explicit Tmpfs_dir(Ref_ptr<Pers_dir> const &d) throw()
        : _dir(d), _getdents_state(false) {}
    int get_entry(const char *, int, mode_t, Ref_ptr<File> *) throw();
    ssize_t getdents(char *, size_t) throw();
};

```

```

int fstat64(struct stat64 *buf) const throw();
int utime(const struct utimbuf *) throw();
int fchmod(mode_t) throw();
int mkdir(const char *, mode_t) throw();
int unlink(const char *) throw();
int rename(const char *, const char *) throw();
int faccessat(const char *, int, int) throw();

private:
    int walk_path(cxx::String const &s,
                  Ref_ptr<Node> *ret, cxx::String *remaining = 0);

    Ref_ptr<Pers_dir> _dir;
    bool _getdents_state;
    Pers_dir::Const_iterator _getdents_iter;
};

class Tmpfs_file : public Be_file_pos
{
public:
    explicit Tmpfs_file(Ref_ptr<Pers_file> const &f) throw()
        : Be_file_pos(), _file(f) {}

    off64_t size() const throw();
    int fstat64(struct stat64 *buf) const throw();
    int ftruncate64(off64_t p) throw();
    int ioctl(unsigned long, va_list) throw();
    int utime(const struct utimbuf *) throw();
    int fchmod(mode_t) throw();

private:
    ssize_t preadv(const struct iovec *v, int iovcnt, off64_t p) throw();
    ssize_t pwritev(const struct iovec *v, int iovcnt, off64_t p) throw();
    Ref_ptr<Pers_file> _file;
};

ssize_t Tmpfs_file::preadv(const struct iovec *v, int iovcnt, off64_t p) throw()
{
    if (iovcnt < 0)
        return -EINVAL;

    ssize_t sum = 0;
    for (int i = 0; i < iovcnt; ++i)
    {
        sum += _file->data().get(p, v[i].iov_len, v[i].iov_base);
        p += v[i].iov_len;
    }
    return sum;
}

ssize_t Tmpfs_file::pwritev(const struct iovec *v, int iovcnt, off64_t p) throw()
{
    if (iovcnt < 0)
        return -EINVAL;

    ssize_t sum = 0;
    for (int i = 0; i < iovcnt; ++i)
    {
        sum += _file->data().put(p, v[i].iov_len, v[i].iov_base);
        p += v[i].iov_len;
    }
    return sum;
}

int Tmpfs_file::fstat64(struct stat64 *buf) const throw()
{
    _file->info()->st_size = _file->data().size();
    memcpy(buf, _file->info(), sizeof(*buf));
    return 0;
}

int Tmpfs_file::ftruncate64(off64_t p) throw()
{
    if (p < 0)
        return -EINVAL;

    if (_file->data().size(p) == 0)
        return 0;

    return -EIO; // most likely ENOSPC, but can't report that
}

off64_t Tmpfs_file::size() const throw()
{
    return _file->data().size();
}

int
Tmpfs_file::ioctl(unsigned long v, va_list args) throw()

```



```

{
    switch (v)
    {
        case FIONREAD: // return amount of data still available
            int *available = va_arg(args, int *);
            *available = _file->data().size() - pos();
            return 0;
    };
    return -EINVAL;
}

int
Tmpfs_file::utime(const struct utimbuf *times) throw()
{
    _file->info()->st_atime = times->actime;
    _file->info()->st_mtime = times->modtime;
    return 0;
}

int
Tmpfs_file::fchmod(mode_t m) throw()
{
    _file->info()->st_mode = m;
    return 0;
}

int
Tmpfs_dir::faccessat(const char *path, int mode, int) throw()
{
    Ref_ptr<Node> node;
    cxx::String name = path;

    int err = walk_path(name, &node, &name);
    if (err < 0)
        return err;

    if (mode == F_OK) // existence check
        return 0;

    struct stat64 *stats = node->info();

    if ((mode & R_OK) && !(stats->st_mode & S_IRUSR))
        return -EACCES;

    if ((mode & W_OK) && !(stats->st_mode & S_IWUSR))
        return -EACCES;

    if ((mode & X_OK) && !(stats->st_mode & S_IXUSR))
        return -EACCES;

    return 0;
}

int
Tmpfs_dir::get_entry(const char *name, int flags, mode_t mode,
                    Ref_ptr<File> *file) throw()
{
    Ref_ptr<Node> path;
    if (!*name)
    {
        *file = cxx::ref_ptr(this);
        return 0;
    }

    cxx::String n = name;

    int e = walk_path(n, &path, &n);

    if (e == -ENOTDIR)
        return e;

    if (!(flags & O_CREAT) && e < 0)
        return e;

    if ((flags & O_CREAT) && e == -ENOENT)
    {
        Ref_ptr<Node> node(new Pers_file(n.start(), mode));
        // when ENOENT is return, path is always a directory
        bool e = cxx::ref_ptr_static_cast<Pers_dir>(path)->add_node(node);
        if (!e)
            return -ENOMEM;
        path = node;
    }

    if (path->is_dir())
        *file = cxx::ref_ptr(new Tmpfs_dir(cxx::ref_ptr_static_cast<Pers_dir>(path)));
    else

```

```

    *file = cxx::ref_ptr(new Tmpfs_file(cxx::ref_ptr_static_cast<Pers_file>(path)));

    if (!*file)
        return -ENOMEM;

    return 0;
}

ssize_t
Tmpfs_dir::getdents(char *buf, size_t sz) throw()
{
    struct dirent64 *d = (struct dirent64 *)buf;
    ssize_t ret = 0;

    if (!_getdents_state)
    {
        _getdents_iter = _dir->begin();
        _getdents_state = true;
    }
    else if (_getdents_iter == _dir->end())
    {
        _getdents_state = false;
        return 0;
    }

    for (; _getdents_iter != _dir->end(); ++_getdents_iter)
    {
        unsigned l = strlen(_getdents_iter->path()) + 1;
        if (l > sizeof(d->d_name))
            l = sizeof(d->d_name);

        unsigned n = offsetof (struct dirent64, d_name) + l;
        n = (n + sizeof(long) - 1) & ~(sizeof(long) - 1);

        if (n > sz)
            break;

        d->d_ino = 1;
        d->d_off = 0;
        memcpy(d->d_name, _getdents_iter->path(), l);
        d->d_reclen = n;
        d->d_type = DT_REG;
        ret += n;
        sz -= n;
        d = (struct dirent64 *)((unsigned long)d + n);
    }

    return ret;
}

int
Tmpfs_dir::fstat64(struct stat64 *buf) const throw()
{
    memcpy(buf, _dir->info(), sizeof(*buf));
    return 0;
}

int
Tmpfs_dir::utime(const struct utimbuf *times) throw()
{
    _dir->info()->st_atime = times->actime;
    _dir->info()->st_mtime = times->modtime;
    return 0;
}

int
Tmpfs_dir::fchmod(mode_t m) throw()
{
    _dir->info()->st_mode = m;
    return 0;
}

int
Tmpfs_dir::walk_path(cxx::String const &s,
                    Ref_ptr<Node> *ret, cxx::String *remaining)
{
    Ref_ptr<Pers_dir> p = _dir;
    cxx::String s = s;
    Ref_ptr<Node> n;

    while (1)
    {
        if (s.len() == 0)
        {
            *ret = p;
            return 0;
        }
    }
}

```

```

    }

    cxx::String::Index sep = s.find("/");

    if (sep - s.start() == 1 && *s.start() == '.')
    {
        s = s.substr(s.start() + 2);
        continue;
    }

    n = p->find_path(s.head(sep - s.start()));

    if (!n)
    {
        *ret = p;
        if (remaining)
            *remaining = s.head(sep - s.start());
        return -ENOENT;
    }

    if (sep == s.end())
    {
        *ret = n;
        return 0;
    }

    if (!n->is_dir())
        return -ENOTDIR;

    s = s.substr(sep + 1);

    p = cxx::ref_ptr_static_cast<Pers_dir>(n);
}

*ret = n;

return 0;
}

int
Tmpfs_dir::mkdir(const char *name, mode_t mode) throw()
{
    Ref_ptr<Node> node = _dir;
    cxx::String p = cxx::String(name);
    cxx::String path, last = p;
    cxx::String::Index s = p.rfind("/");

    // trim '/'s at the end
    while (p.len() && s == p.end() - 1)
    {
        p.len(p.len() - 1);
        s = p.rfind("/");
    }

    //printf("MKDIR '%s' p=%p %p\n", name, p.start(), s);

    if (s != p.end())
    {
        path = p.head(s);
        last = p.substr(s + 1, p.end() - s);

        int e = walk_path(path, &node);
        if (e < 0)
            return e;
    }

    if (!node->is_dir())
        return -ENOTDIR;

    // due to path walking we can end up with an empty name
    if (p.len() == 0 || p == cxx::String("."))
        return 0;

    Ref_ptr<Pers_dir> dnode = cxx::ref_ptr_static_cast<Pers_dir>(node);

    Ref_ptr<Pers_dir> dir(new Pers_dir(last.start(), mode));
    return dnode->add_node(dir) ? 0 : -EEXIST;
}

int
Tmpfs_dir::unlink(const char *name) throw()
{
    cxx::Ref_ptr<Node> n;

    int e = walk_path(name, &n);
    if (e < 0)

```

```

        return -ENOENT;

    printf("Unimplemented (if file exists): %s(%s)\n", __func__, name);
    return -ENOMEM;
}

int
Tmpfs_dir::rename(const char *old, const char *newn) throw()
{
    printf("Unimplemented: %s(%s, %s)\n", __func__, old, newn);
    return -ENOMEM;
}

class Tmpfs_fs : public Be_file_system
{
public:
    Tmpfs_fs() : Be_file_system("tmpfs") {}
    int mount(char const *source, unsigned long mountflags,
              void const *data, cxx::Ref_ptr<File> *dir) throw()
    {
        (void)mountflags;
        (void)source;
        (void)data;
        *dir = cxx::ref_ptr(new Tmpfs_dir(cxx::ref_ptr(new Pers_dir("root", 0777))));
        if (!*dir)
            return -ENOMEM;
        return 0;
    }
};

static Tmpfs_fs _tmpfs L4RE_VFS_FILE_SYSTEM_ATTRIBUTE;

```

## 17.26 examples/libs/shmc/prodcons.c

Simple shared memory example.

Simple shared memory example.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

/*
 * This example uses shared memory between two threads, one producer, one
 * consumer.
 */

#include <l4/shmc/shmc.h>

#include <l4/util/util.h>

#include <stdio.h>
#include <string.h>
#include <pthread-l4.h>

#include <l4/sys/thread.h>
#include <l4/sys/debugger.h>
#include <l4/sys/kip.h>
#include <l4/re/env.h>

#define LOG(args...)      printf(NAME " : " args)
#define CHK(func)         do {
                            {
                                long r = (func);
                                if (r)
                                {
                                    printf(NAME " : Failure %ld (%s) at line %d.\n",
                                        r, l4sys_errtostr(r), __LINE__);
                                    return (void *)-1;
                                }
                            }
                        } while (0)

```

```

    } while (0)

static const char some_data[] = "Hi consumer!";

static inline l4_cap_idx_t self(void) { return pthread_l4_cap(pthread_self()); }

#define NAME "PRODUCER"
static void *thread_producer(void *d)
{
    (void)d;
    l4shmc_chunk_t p_one;
    l4shmc_signal_t s_one, s_done;
    l4shmc_area_t shmarea;
    l4_kernel_clock_t try_until;

    l4_debugger_set_object_name(self(), "producer");

    // attach this thread to the shm object
    CHK(l4shmc_attach("testshm", &shmarea));

    // add a chunk
    CHK(l4shmc_add_chunk(&shmarea, "one", 1024, &p_one));

    // add a signal
    CHK(l4shmc_add_signal(&shmarea, "testshm_prod", &s_one));

    CHK(l4shmc_attach_signal(&shmarea, "testshm_done", self(), &s_done));

    // connect chunk and signal
    CHK(l4shmc_connect_chunk_signal(&p_one, &s_one));

    CHK(l4shmc_mark_client_initialized(&shmarea));

    try_until = l4_kip_clock(l4re_kip()) + 10 * 1000000;

    for (;;)
    {
        l4_umword_t clients;
        l4shmc_get_initialized_clients(&shmarea, &clients);
        if (clients == 3UL)
            break;
        if (l4_kip_clock(l4re_kip()) >= try_until)
        {
            LOG("consumer not initialized within time\n");
            return (void *)-1;
        }
    }

    LOG("Ready.\n");

    while (1)
    {
        while (l4shmc_chunk_try_to_take(&p_one))
            printf("Uh, should not happen!\n"); //l4_thread_yield();

        memcpy(l4shmc_chunk_ptr(&p_one), some_data, sizeof(some_data));

        CHK(l4shmc_chunk_ready_sig(&p_one, sizeof(some_data)));

        LOG("Sent data.\n");

        CHK(l4shmc_wait_signal(&s_done));
    }

    l4_sleep_forever();
    return NULL;
}

#undef NAME
#define NAME "CONSUMER"
static void *thread_consumer(void *d)
{
    (void)d;
    l4shmc_area_t shmarea;
    l4shmc_chunk_t p_one;
    l4shmc_signal_t s_one, s_done;
    l4_kernel_clock_t try_until;

    l4_debugger_set_object_name(self(), "consumer");

    // attach to shared memory area
    CHK(l4shmc_attach("testshm", &shmarea));

    // get chunk 'one'

```

```

CHK(l4shmc_get_chunk(&shmarea, "one", &p_one));

// add a signal
CHK(l4shmc_add_signal(&shmarea, "testshm_done", &s_done));

// attach signal to this thread
CHK(l4shmc_attach_signal(&shmarea, "testshm_prod", self(), &s_one));

// connect chunk and signal
CHK(l4shmc_connect_chunk_signal(&p_one, &s_one));

CHK(l4shmc_mark_client_initialized(&shmarea));

try_until = l4_kip_clock(l4re_kip()) + 10 * 1000000;

for (;;)
{
    l4_umword_t clients;
    l4shmc_get_initialized_clients(&shmarea, &clients);
    if (clients == 3UL)
        break;
    if (l4_kip_clock(l4re_kip()) >= try_until)
    {
        LOG("producer not initialized within time\n");
        return (void *)-1;
    }
}

LOG("Ready.\n");

while (1)
{
    CHK(l4shmc_wait_chunk(&p_one));

    LOG("Received from chunk one: '%s'.\n",
        (char *)l4shmc_chunk_ptr(&p_one));
    memset(l4shmc_chunk_ptr(&p_one), 0, l4shmc_chunk_size(&p_one));

    CHK(l4shmc_chunk_consumed(&p_one));

    CHK(l4shmc_trigger(&s_done));
}

return NULL;
}

int main(void)
{
    pthread_t one, two;
    long r;

    // create shared memory area
    if ((r = l4shmc_create("testshm")) < 0)
    {
        printf("Error %ld (%s) creating shared memory area\n",
            r, l4sys_errtostr(r));
        return 1;
    }

    // create two threads, one for producer, one for consumer
    pthread_create(&one, 0, thread_producer, 0);
    pthread_create(&two, 0, thread_consumer, 0);

    // now sleep, the two threads are doing the work
    l4_sleep_forever();

    return 0;
}

```

# Index

%L4 Inter-Process Communication (IPC), [9](#)

\_\_iface

L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_DEMAND >, [1194](#)

L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S\_DEMAND >, [1198](#)

\_\_iface\_list

L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_DEMAND >, [1194](#)

L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S\_DEMAND >, [1198](#)

\_\_L4UTIL\_THREAD\_FUNC

thread.h, [2915](#)

\_\_check\_protocols\_\_

L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_DEMAND >, [1194](#)

L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S\_DEMAND >, [1198](#)

\_\_kdebug\_3\_text

kdebug.h, [2829](#)

\_\_kdebug\_op

kdebug.h, [2830](#)

\_\_kdebug\_op\_1

kdebug.h, [2831](#)

\_\_kdebug\_text

kdebug.h, [2833](#)

\_\_vcpu-arch.h

L4\_vcpu\_e\_field\_ids, [2097](#)

L4\_VCPU\_STATE\_VERSION, [2094](#), [2097](#), [2100](#)

~Be\_file\_system

L4Re::Vfs::Be\_file\_system, [1633](#)

~H\_list\_item\_t

cxx::H\_list\_item\_t< ELEM\_TYPE >, [858](#)

~Unique\_region

L4Re::Rm::Unique\_region< T >, [1551](#)

a

L4Re::Video::Pixel\_info, [1678](#), [1679](#)

access\_once

cxx, [665](#)

Access\_width

L4Re::Mmio\_space, [1511](#)

acquire

L4Re::Inhibitor, [1497](#)

add

L4::lpc\_svr::Timeout\_queue, [1147](#)

L4::Thread::Modify\_senders, [1308](#)

L4vcpu::State, [1779](#)

L4virtio::Svr::Driver\_mem\_list\_t< DATA >, [1930](#)

add\_block

L4virtio::Driver::Block\_device, [1803](#)

add\_image\_info

L4::Debugger, [970](#)

add\_irq\_status

L4virtio::Svr::Dev\_config, [1906](#)

add\_ku\_mem

L4::Task, [1281](#)

add\_timeout

L4::lpc\_svr::Server\_iface, [1139](#)

L4::lpc\_svr::Timeout\_queue\_hooks< HOOKS, BR\_MAN >, [1155](#)

add\_trusted\_dataspaces

L4virtio::Svr::Device\_t< DATA >, [1922](#)

AHCI driver, [59](#)

alien

L4::Thread::Attr, [1304](#)

align\_to

L4::lpc::Msg, [687](#)

all

L4::Kip::Mem\_desc, [1182](#)

alloc

cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >, [765](#)

cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >, [772](#)

cxx::List\_alloc, [866](#)

cxx::Slab< Type, Slab\_size, Max\_free, Alloc >, [893](#)

cxx::Slab\_static< Type, Slab\_size, Max\_free, Alloc >, [897](#)

L4Re::Cap\_alloc, [1441](#)

L4Re::Mem\_alloc, [1508](#)

L4Re::Util::Counting\_cap\_alloc< COUNTER-TYPE, Dbg >, [1574](#), [1575](#)

alloc\_buffer\_demand

L4::lpc\_svr::Br\_manager\_no\_buffers, [1122](#)

L4::lpc\_svr::Server\_iface, [1139](#)

L4Re::Util::Br\_manager, [1559](#)

alloc\_descriptor

L4virtio::Driver::Virtqueue, [1836](#)

alloc\_dynamic\_entry

L4Re::Util::Names::Name\_space, [1604](#)

alloc\_fd

L4Re::Vfs::Fs, [1646](#)

alloc\_max

cxx::List\_alloc, [866](#)

allocate

L4Re::Dataspace, [1450](#)

L4Re::Util::Dataspace\_svr, [1579](#)

- amd64 Virtual Registers (UTCB), [400](#)
- amd64/l4/sys/\_\_kip-arch.h, [2091](#)
- amd64/l4/sys/\_\_vcpu-arch.h, [2092](#), [2094](#)
- amd64/l4/sys/cache.h, [2697](#)
- amd64/l4/sys/consts.h, [2480](#)
- amd64/l4/sys/ktrace\_events.h, [2101](#)
- amd64/l4/sys/l4int.h, [2857](#), [2858](#)
- amd64/l4/sys/linkage.h, [2111](#)
- amd64/l4/sys/segment.h, [2069](#), [2074](#)
- amd64/l4/sys/utcb.h, [2919](#), [2920](#)
- amd64/l4/sys/vm.h, [2116](#)
- amd64/l4/util/bitops\_arch.h, [2118](#)
- amd64/l4/util/cpu.h, [2125](#), [2127](#)
- amd64/l4/util/idt.h, [2042](#), [2043](#)
- amd64/l4/util/irq.h, [2814](#), [2816](#)
- amd64/l4/util/l4\_macros.h, [2130](#), [2131](#)
- amd64/l4/util/mbi\_argv.h, [2133](#), [2134](#)
- amd64/l4/util/perform.h, [2046](#), [2047](#)
- amd64/l4/util/port\_io.h, [2083](#), [2084](#)
- amd64/l4/util/rdtsc.h, [2058](#), [2060](#)
- amd64/l4/util/spin.h, [2067](#)
- amd64/l4f/l4/sys/ipc.h, [2795](#)
- amd64/l4f/l4/sys/segment.h, [2075](#), [2078](#)
- amd64/l4f/l4/util/port\_io.h, [2084](#)
- Application and Server Building Blocks, [23](#)
- Arch
  - L4::Kip::Mem\_desc, [1181](#)
- Arch\_acpi\_nvs
  - L4::Kip::Mem\_desc, [1181](#)
- Arch\_acpi\_tables
  - L4::Kip::Mem\_desc, [1181](#)
- Arch\_sub\_type\_common
  - L4::Kip::Mem\_desc, [1180](#)
- Area
  - L4Re::Rm, [1536](#)
- ARM Virtual Registers (UTCB), [394](#)
- arm/l4/sys/\_\_kip-arch.h, [2092](#)
- arm/l4/sys/\_\_vcpu-arch.h, [2095](#), [2097](#)
- arm/l4/sys/atomic.h, [2150](#), [2151](#)
- arm/l4/sys/cache.h, [2698](#), [2699](#)
- arm/l4/sys/consts.h, [2481](#)
- arm/l4/sys/ktrace\_events.h, [2104](#)
- arm/l4/sys/l4int.h, [2858](#), [2859](#)
- arm/l4/sys/linkage.h, [2112](#)
- arm/l4/sys/mem\_op.h, [2113](#), [2115](#)
- arm/l4/sys/platform\_control.h, [2869](#)
- arm/l4/sys/task.h, [2899](#)
- arm/l4/sys/thread.h, [2903](#), [2904](#)
- arm/l4/sys/utcb.h, [2922](#), [2923](#)
- arm/l4/sys/vm, [2949](#)
- arm/l4/sys/vm.h, [2116](#), [2117](#)
- arm/l4/util/bitops\_arch.h, [2121](#), [2122](#)
- arm/l4/util/cpu.h, [2128](#)
- arm/l4/util/irq.h, [2817](#), [2818](#)
- arm/l4/util/l4\_macros.h, [2131](#)
- arm/l4/util/mbi\_argv.h, [2135](#), [2136](#)
- arm/l4f/l4/sys/ipc.h, [2796](#), [2797](#)
- arm/l4f/l4/sys/syscall\_defs.h, [2138](#)
- arm\_smccc.h
  - l4\_arm\_smccc\_call, [2689](#)
- assert.h
  - l4\_assert, [2693](#)
- assign\_dma\_domain
  - L4vbus::Vbus, [1772](#), [1773](#)
- associate
  - L4Re::Dma\_space, [1465](#)
- AT\_BASE
  - ELF binary format, [621](#)
- AT\_EGID
  - ELF binary format, [621](#)
- AT\_ENTRY
  - ELF binary format, [621](#)
- AT\_EUID
  - ELF binary format, [621](#)
- AT\_EXECFD
  - ELF binary format, [621](#)
- AT\_FLAGS
  - ELF binary format, [621](#)
- AT\_GID
  - ELF binary format, [621](#)
- AT\_IGNORE
  - ELF binary format, [621](#)
- AT\_L4\_AUX
  - ELF binary format, [621](#)
- AT\_L4\_ENV
  - ELF binary format, [621](#)
- AT\_NOTELF
  - ELF binary format, [621](#)
- AT\_NULL
  - ELF binary format, [621](#)
- AT\_PAGESZ
  - ELF binary format, [621](#)
- AT\_PHDR
  - ELF binary format, [621](#)
- AT\_PHEMT
  - ELF binary format, [621](#)
- AT\_PHNUM
  - ELF binary format, [621](#)
- AT\_UID
  - ELF binary format, [621](#)
- Atomic Instructions, [583](#)
  - l4util\_add16, [584](#)
  - l4util\_add16\_res, [584](#)
  - l4util\_add32, [585](#)
  - l4util\_add32\_res, [585](#)
  - l4util\_add8, [585](#)
  - l4util\_add8\_res, [586](#)
  - l4util\_and16, [586](#)
  - l4util\_and16\_res, [586](#)
  - l4util\_and32, [587](#)
  - l4util\_and32\_res, [587](#)
  - l4util\_and8, [587](#)
  - l4util\_and8\_res, [588](#)
  - l4util\_atomic\_add, [588](#)
  - l4util\_atomic\_inc, [588](#)
  - l4util\_cmpxchg, [589](#)



- l4util\_cmpxchg16, [589](#)
- l4util\_cmpxchg32, [590](#)
- l4util\_cmpxchg8, [590](#)
- l4util\_dec16, [591](#)
- l4util\_dec16\_res, [591](#)
- l4util\_dec32, [591](#)
- l4util\_dec32\_res, [592](#)
- l4util\_dec8, [592](#)
- l4util\_dec8\_res, [592](#)
- l4util\_inc16, [592](#)
- l4util\_inc16\_res, [593](#)
- l4util\_inc32, [593](#)
- l4util\_inc32\_res, [593](#)
- l4util\_inc8, [594](#)
- l4util\_inc8\_res, [594](#)
- l4util\_or16, [594](#)
- l4util\_or16\_res, [594](#)
- l4util\_or32, [595](#)
- l4util\_or32\_res, [595](#)
- l4util\_or8, [595](#)
- l4util\_or8\_res, [596](#)
- l4util\_sub16, [596](#)
- l4util\_sub16\_res, [596](#)
- l4util\_sub32, [597](#)
- l4util\_sub32\_res, [597](#)
- l4util\_sub8, [597](#)
- l4util\_sub8\_res, [597](#)
- l4util\_xchg, [598](#)
- l4util\_xchg16, [598](#)
- l4util\_xchg32, [599](#)
- l4util\_xchg8, [599](#)
- atomic\_clear\_bit
  - cxx::Bitmap\_base, [798](#)
- atomic\_get\_and\_clear
  - cxx::Bitmap\_base, [798](#)
- atomic\_get\_and\_set
  - cxx::Bitmap\_base, [798](#)
- atomic\_set\_bit
  - cxx::Bitmap\_base, [799](#)
- attach
  - L4Re::Rm, [1537](#), [1538](#)
  - L4Re::Util::Event\_buffer\_t< PAYLOAD >, [1592](#)
- Attach\_flags
  - L4Re::Rm::F, [1546](#)
- Attach\_mask
  - L4Re::Rm::F, [1546](#)
- Attr
  - L4::Thread::Attr, [1303](#)
- Attribute
  - L4Re::Dma\_space, [1464](#)
- Attributes
  - L4Re::Dma\_space, [1464](#)
- atype
  - Elf32\_Auxv, [916](#)
  - Elf64\_Auxv, [926](#)
- Auxiliary data, [522](#)
- avail
  - cxx::List\_alloc, [867](#)
- avail\_align
  - L4virtio::Virtqueue, [1981](#)
- avail\_size
  - L4virtio::Virtqueue, [1981](#)
- Avl\_map
  - cxx::Avl\_map< KEY\_TYPE, DATA\_TYPE, COMPARE, ALLOC >, [747](#)
- ax
  - l4\_vcpu\_regs\_t, [1411](#)
- b
  - L4Re::Video::Pixel\_info, [1679](#)
- backtrace.h
  - l4util\_backtrace, [2952](#)
- Bad\_address
  - L4virtio::Svr::Bad\_descriptor, [1847](#)
- Bad\_descriptor
  - L4virtio::Svr::Bad\_descriptor, [1847](#)
- Bad\_flags
  - L4virtio::Svr::Bad\_descriptor, [1847](#)
- Bad\_next
  - L4virtio::Svr::Bad\_descriptor, [1847](#)
- Bad\_rights
  - L4virtio::Svr::Bad\_descriptor, [1847](#)
- Bad\_size
  - L4virtio::Svr::Bad\_descriptor, [1847](#)
- Base API, [131](#)
- Base\_avl\_set
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [810](#)
- Basic Macros, [134](#)
  - l4\_align\_stack\_for\_direct\_fncall, [137](#)
  - L4\_EXPORT, [136](#)
  - L4\_HIDDEN, [136](#)
  - l4\_infinite\_loop, [137](#)
  - L4\_NOTHROW, [136](#)
- Be\_file\_system
  - L4Re::Vfs::Be\_file\_system, [1633](#)
- begin
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [811](#)
  - cxx::Bits::Bst< Node, Get\_key, Compare >, [828](#), [829](#)
- Bidirectional
  - L4Re::Dma\_space, [1465](#)
- bind
  - L4::lcu, [1014](#)
  - L4::lommu, [1028](#)
  - L4::Thread::Attr, [1304](#)
- bind\_notification\_irq
  - L4virtio::Driver::Device, [1812](#)
- bind\_thread
  - L4::Rcv\_endpoint, [1232](#)
- bit
  - cxx::Bitmap\_base, [799](#)
- Bit Manipulation, [600](#)
  - l4util\_bsf, [601](#)
  - l4util\_bsr, [601](#)
  - l4util\_btc, [601](#)

- l4util\_btr, [602](#)
- l4util\_bts, [602](#)
- l4util\_clear\_bit, [604](#)
- l4util\_complement\_bit, [605](#)
- l4util\_find\_first\_set\_bit, [605](#)
- l4util\_find\_first\_zero\_bit, [605](#)
- l4util\_next\_power2, [606](#)
- l4util\_set\_bit, [606](#)
- l4util\_test\_bit, [607](#)
- bit\_index
  - cxx::Bitmap\_base, [800](#)
- Bitmap graphics and fonts, [607](#)
- bitmap.h
  - gfxbitmap\_bmap, [2424](#)
  - gfxbitmap\_color\_pix\_t, [2424](#)
  - gfxbitmap\_color\_t, [2424](#)
  - gfxbitmap\_convert\_color, [2425](#)
  - gfxbitmap\_copy, [2425](#)
  - gfxbitmap\_fill, [2425](#)
  - gfxbitmap\_set, [2426](#)
  - pSLIM\_BMAP\_START\_LSB, [2423](#)
- Bits
  - cxx::Bitfield< T, LSB, MSB >, [777](#)
- bits\_per\_pixel
  - L4Re::Video::Pixel\_info, [1679](#)
- Bits\_type
  - cxx::Bitfield< T, LSB, MSB >, [776](#)
- Block\_dev\_base
  - L4virtio::Svr::Block\_dev\_base< Ds\_data >, [1852](#)
- Block\_device::Device\_discard\_feature, [721](#)
- Block\_device::Device\_mgr< DEV, FACTORY, SCHEDULER >, [722](#)
- Block\_device::Device\_with\_notification\_domain< DEV >, [723](#)
- Block\_device::Dma\_region\_info, [724](#)
- Block\_device::Errand::Errand, [724](#)
  - expired, [727](#)
- Block\_device::Errand::Poll\_errand, [728](#)
  - expired, [730](#)
- Block\_device::Impl::Partitioned\_device\_discard\_mixin< PART\_DEV, BASE\_DEV, bool >, [731](#)
- Block\_device::Impl::Partitioned\_device\_discard\_mixin< PART\_DEV, BASE\_DEV, true >, [732](#)
- Block\_device::Inout\_block, [733](#)
- Block\_device::Inout\_memory< DEV >, [734](#)
- Block\_device::Mem\_region\_info, [735](#)
- Block\_device::Notification\_domain, [735](#)
- Block\_device::Partition\_info, [736](#)
- Block\_device::Partition\_reader< DEV >, [737](#)
- Block\_device::Partitioned\_device< BASE\_DEV >, [738](#)
- Block\_device::Pending\_request, [739](#)
  - fail\_request, [740](#)
  - handle\_request, [740](#)
- Block\_device::Rr\_scheduler< DEV >, [740](#)
- Block\_device::Scheduler\_base< DEV >, [742](#)
- Bootloader
  - L4::Kip::Mem\_desc, [1181](#)
- Bootstrap, the %L4 kernel bootstrapper, [79](#)
- bp
  - l4\_vcpu\_regs\_t, [1411](#)
- Br\_bytes
  - L4::lpc::Msg, [687](#)
- buf
  - L4Re::Util::Event\_buffer\_t< PAYLOAD >, [1592](#)
- buf\_cp\_in
  - L4::lpc, [679](#)
- buf\_cp\_out
  - L4::lpc, [679](#)
- buf\_in
  - L4::lpc, [680](#)
- buffer
  - L4Re::Util::Event\_t< PAYLOAD >, [1596](#)
- Buffer Registers (BRs), [394](#)
  - L4\_BDR\_IO\_SHIFT, [395](#)
  - L4\_BDR\_MEM\_SHIFT, [395](#)
  - L4\_BDR\_OBJ\_SHIFT, [395](#)
  - l4\_buffer\_desc\_consts\_t, [395](#)
- Bufferable
  - L4Re::Dataspace::F, [1458](#)
- bus\_cap
  - L4vbus::Device, [1724](#)
- bx
  - l4\_vcpu\_regs\_t, [1411](#)
- bytes\_per\_pixel
  - L4Re::Video::Pixel\_info, [1680](#)
- c
  - L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_DEMAND >, [1194](#)
  - L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S\_DEMAND >, [1198](#)
- C++ Exceptions, [523](#)
- C++ IPC Interface Definition., [138](#)
- C\_bits
  - cxx::Bitmap\_base, [798](#)
- Cache Consistency, [139](#)
  - l4\_cache\_clean\_data, [139](#)
  - l4\_cache\_coherent, [140](#)
  - l4\_cache\_dma\_coherent, [140](#)
  - l4\_cache\_flush\_data, [141](#)
  - l4\_cache\_inv\_data, [141](#)
- Cache\_buffered
  - L4Re::Rm::F, [1547](#)
- Cache\_normal
  - L4Re::Rm::F, [1547](#)
- Cache\_uncached
  - L4Re::Rm::F, [1547](#)
- Cacheable
  - L4Re::Dataspace::F, [1458](#)
- Caching\_mask
  - L4Re::Dataspace::F, [1458](#)
  - L4Re::Rm::F, [1547](#)
- Caching\_shift
  - L4Re::Dataspace::F, [1457](#)
  - L4Re::Rm, [1537](#)
- call
  - L4::Arm\_smccc, [937](#)

- L4::lpc::loststream, [1051](#)
- Cap
  - L4::Cap< T >, [950](#), [951](#)
  - L4::lpc::Cap< T >, [1041](#)
- cap
  - L4::Cap\_base, [956](#)
  - L4::Invalid\_capability, [1023](#)
  - L4::Kobject, [1190](#)
- cap\_alloc
  - L4Re Capability API, [533](#)
- Cap\_base
  - L4::Cap\_base, [955](#)
- cap\_cast
  - L4, [672](#)
- cap\_dynamic\_cast
  - L4, [673](#)
- cap\_equal
  - L4::Task, [1282](#)
- Cap\_mask
  - L4::lpc::Cap< T >, [1041](#)
- cap\_received
  - L4::lpc::Gen\_fpage, [1044](#)
- cap\_reinterpret\_cast
  - L4, [674](#)
- Cap\_type
  - L4::Cap\_base, [955](#)
- cap\_valid
  - L4::Task, [1283](#)
- Capabilities, [142](#)
  - L4\_BASE\_ARM\_SMCCC\_CAP, [144](#)
  - L4\_BASE\_CAPS\_LAST, [144](#)
  - L4\_BASE\_DEBUGGER\_CAP, [144](#)
  - L4\_BASE\_FACTORY\_CAP, [144](#)
  - L4\_BASE\_ICU\_CAP, [144](#)
  - L4\_BASE\_IOMMU\_CAP, [144](#)
  - L4\_BASE\_LOG\_CAP, [144](#)
  - L4\_BASE\_PAGER\_CAP, [144](#)
  - L4\_BASE\_SCHEDULER\_CAP, [144](#)
  - L4\_BASE\_TASK\_CAP, [144](#)
  - L4\_BASE\_THREAD\_CAP, [144](#)
  - l4\_cap\_consts\_t, [143](#)
  - l4\_cap\_idx\_t, [143](#)
  - L4\_CAP\_MASK, [143](#)
  - L4\_CAP\_OFFSET, [143](#)
  - L4\_CAP\_SHIFT, [143](#)
  - L4\_CAP\_SIZE, [143](#)
  - l4\_capability\_equal, [144](#)
  - l4\_default\_caps\_t, [144](#)
  - L4\_INVALID\_CAP, [143](#)
  - l4\_is\_invalid\_cap, [145](#)
  - l4\_is\_valid\_cap, [145](#)
- Capabilities and Naming, [18](#)
- capability
  - L4\_DISABLE\_COPY, [2706](#)
- Capability allocator, [469](#)
  - l4re\_util\_cap\_last, [470](#)
- Caps
  - L4::Type\_info::Demand\_t< CAPS, FLAGS, MEM, PORTS >, [1320](#)
- caps
  - l4re\_env\_t, [1697](#)
- cast
  - L4vcpu::Vcpu, [1783](#)
- cfg\_read
  - L4vbus::Pci\_dev, [1754](#)
  - L4vbus::Pci\_host\_bridge, [1760](#)
- cfg\_write
  - L4vbus::Pci\_dev, [1754](#)
  - L4vbus::Pci\_host\_bridge, [1760](#)
- chain
  - L4::lrc\_mux, [1173](#)
- change\_mode
  - L4::Uart, [1358](#)
  - L4::Uart\_apb, [1364](#)
- change\_queue\_config
  - L4virtio::Svr::Dev\_config, [1907](#)
- char\_avail
  - L4::Uart, [1358](#)
  - L4::Uart\_apb, [1365](#)
- check\_castable\_from
  - L4::Cap< T >, [951](#)
- check\_convertible\_from
  - L4::Cap< T >, [951](#)
- check\_size
  - L4::lpc::Msg, [688](#), [689](#)
- chkcap
  - L4Re, [698](#)
- chkipc
  - L4Re, [699](#)
- chksys
  - L4Re, [700–702](#)
- Chunks, [548](#)
  - l4shmc\_add\_chunk, [548](#)
  - l4shmc\_chunk\_capacity, [549](#)
  - l4shmc\_chunk\_ptr, [549](#)
  - l4shmc\_chunk\_signal, [550](#)
  - l4shmc\_get\_chunk, [550](#)
  - l4shmc\_get\_chunk\_to, [550](#)
  - l4shmc\_iterate\_chunk, [551](#)
- clamp
  - Small C++ Template Library, [574](#)
- Class
  - L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_DEMAND >, [1194](#)
  - L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S\_DEMAND >, [1198](#)
- clear
  - cxx::Bits::Basic\_list< POLICY >, [824](#)
  - L4::Types::Flags< BITS\_ENUM, UNDERLYING >, [1345](#)
  - L4drivers::Register\_tmpl< BITS, BLOCK >, [1434](#)
  - L4Re::Dataspace, [1451](#)
  - L4Re::Util::Dataspace\_svr, [1580](#)
  - L4vcpu::State, [1779](#)
- clear\_bit

- cxx::Bitmap\_base, [800](#)
- Coherent
  - L4Re::Dma\_space, [1465](#)
- Color\_component
  - L4Re::Video::Color\_component, [1665](#)
- Com\_error
  - L4::Com\_error, [966](#)
- Comfortable Command Line Parsing, [611](#)
  - parse\_cmdline, [612](#)
- config\_get
  - L4vbus::Gpio\_pin, [1741](#)
- config\_pad
  - L4vbus::Gpio\_module, [1733](#)
  - L4vbus::Gpio\_pin, [1741](#)
- config\_pull
  - L4vbus::Gpio\_pin, [1742](#)
- config\_queue
  - L4virtio::Device, [1795](#)
  - L4virtio::Driver::Device, [1813](#)
- Cons, the Console Multiplexer, [61](#)
- Console API, [524](#)
- console\_multiport\_bfm\_t
  - L4virtio::Svr::Console::Features, [1880](#)
- Console\_port
  - L4virtio::Svr::Console::Control\_message, [1860](#)
- console\_size\_bfm\_t
  - L4virtio::Svr::Console::Features, [1880](#)
- consumed
  - L4virtio::Svr::Virtqueue, [1969](#)
- Consumer, [551](#), [562](#)
  - l4shmc\_chunk\_consumed, [552](#)
  - l4shmc\_chunk\_size, [552](#)
  - l4shmc\_chunk\_try\_to\_take\_for\_reading, [553](#)
  - l4shmc\_enable\_chunk, [553](#)
  - l4shmc\_enable\_signal, [563](#)
  - l4shmc\_is\_chunk\_ready, [553](#)
  - l4shmc\_wait\_any, [563](#)
  - l4shmc\_wait\_any\_to, [563](#)
  - l4shmc\_wait\_any\_try, [564](#)
  - l4shmc\_wait\_chunk, [554](#)
  - l4shmc\_wait\_chunk\_to, [554](#)
  - l4shmc\_wait\_chunk\_try, [555](#)
  - l4shmc\_wait\_signal, [564](#)
  - l4shmc\_wait\_signal\_to, [565](#)
  - l4shmc\_wait\_signal\_try, [565](#)
- contains
  - L4virtio::Svr::Driver\_mem\_region\_t< DATA >, [1939](#)
- Continuous
  - L4Re::Mem\_alloc, [1508](#)
- contrib/libio-io/l4/io/io.h, [2139](#), [2141](#)
- contrib/libio-io/l4/io/types.h, [2142](#)
- control
  - L4::Thread, [1292](#)
- Conventional
  - L4::Kip::Mem\_desc, [1181](#)
- copy
  - L4::Cap< T >, [952](#)
  - L4::Cap\_base, [957](#)
  - L4Re::Util::Dataspace\_svr, [1580](#)
- copy\_in
  - L4Re::Dataspace, [1451](#)
- copy\_receive\_cap
  - L4Re::Util::Names::Name\_space, [1604](#)
- copy\_to
  - L4virtio::Svr::Data\_buffer, [1901](#)
- count
  - L4::Kip::Mem\_desc, [1183](#), [1184](#)
- Counting\_cap\_alloc
  - L4Re::Util::Counting\_cap\_alloc< COUNTER-  
TYPE, Dbg >, [1574](#)
- CPU related functions, [608](#)
  - l4util\_cpu\_capabilities, [609](#)
  - l4util\_cpu\_capabilities\_nocheck, [609](#)
  - l4util\_cpu\_has\_cpuid, [610](#)
- cpu\_allow\_shutdown
  - L4::Platform\_control, [1220](#)
- cpu\_disable
  - L4::Platform\_control, [1221](#)
- cpu\_enable
  - L4::Platform\_control, [1221](#)
- create
  - L4::Factory, [999](#), [1000](#)
- create\_buffer
  - L4Re::Video::Goos, [1671](#)
- create\_factory
  - L4::Factory, [1001](#)
- create\_gate
  - L4::Factory, [1002](#)
- create\_task
  - L4::Factory, [1003](#)
- create\_view
  - L4Re::Video::Goos, [1672](#)
- current\_flags
  - L4virtio::Svr::Request\_processor, [1945](#)
- cx
  - l4\_vcpu\_regs\_t, [1411](#)
- cxx, [663](#)
  - access\_once, [665](#)
  - write\_now, [666](#)
- cxx::Avl\_map< KEY\_TYPE, DATA\_TYPE, COMPARE,  
ALLOC >, [744](#)
  - Avl\_map, [747](#)
  - insert, [748](#)
  - operator[], [749](#)
- cxx::Avl\_set< ITEM\_TYPE, COMPARE, ALLOC >, [750](#)
- cxx::Avl\_tree< Node, Get\_key, Compare >, [754](#)
  - insert, [759](#)
  - Iterator, [759](#)
  - remove, [760](#)
- cxx::Avl\_tree\_node, [761](#)
- cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc  
>, [763](#)
  - alloc, [765](#)
  - free, [766](#)
  - free\_objects, [767](#)

- max\_free\_slabs, [765](#)
- object\_size, [765](#)
- objects\_per\_slab, [765](#)
- slab\_size, [765](#)
- total\_objects, [768](#)
- cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc  
>::Slab\_i, [769](#)
- cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free,  
Alloc >, [769](#)
- alloc, [772](#)
- free, [772](#)
- free\_objects, [773](#)
- max\_free\_slabs, [771](#)
- object\_size, [771](#)
- objects\_per\_slab, [771](#)
- slab\_size, [771](#)
- total\_objects, [773](#)
- cxx::Bitfield< T, LSB, MSB >, [774](#)
- Bits, [777](#)
- Bits\_type, [776](#)
- get, [777](#)
- get\_unshifted, [778](#)
- Low\_mask, [777](#)
- Lsb, [777](#)
- Mask, [777](#)
- Masks, [777](#)
- Msb, [777](#)
- set, [779](#)
- set\_dirty, [779](#)
- set\_unshifted, [780](#)
- set\_unshifted\_dirty, [781](#)
- Shift\_type, [776](#)
- val, [782](#)
- val\_dirty, [783](#)
- val\_unshifted, [785](#)
- cxx::Bitfield< T, LSB, MSB >::Value< TT >, [786](#)
- cxx::Bitfield< T, LSB, MSB >::Value\_base< TT >, [788](#)
- cxx::Bitfield< T, LSB, MSB >::Value\_unshifted< TT >,  
[789](#)
- cxx::Bitmap< BITS >, [790](#)
- scan\_zero, [794](#)
- cxx::Bitmap\_base, [795](#)
- atomic\_clear\_bit, [798](#)
- atomic\_get\_and\_clear, [798](#)
- atomic\_get\_and\_set, [798](#)
- atomic\_set\_bit, [799](#)
- bit, [799](#)
- bit\_index, [800](#)
- C\_bits, [798](#)
- clear\_bit, [800](#)
- operator[], [801](#)
- scan\_zero, [802](#)
- set\_bit, [802](#)
- W\_bits, [798](#)
- word\_index, [803](#)
- cxx::Bitmap\_base::Bit, [803](#)
- cxx::Bitmap\_base::Char< BITS >, [804](#)
- cxx::Bitmap\_base::Word< BITS >, [805](#)
- cxx::Bits, [667](#)
- cxx::Bits::Avl\_map\_get\_key< KEY\_TYPE >, [806](#)
- cxx::Bits::Avl\_set\_get\_key< KEY\_TYPE >, [806](#)
- cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, AL-  
LOC, GET\_KEY >, [807](#)
- Base\_avl\_set, [810](#)
- begin, [811](#)
- E\_exist, [810](#)
- E\_inval, [810](#)
- E\_noent, [810](#)
- E\_nomem, [810](#)
- end, [812](#), [813](#)
- erase, [814](#)
- find\_node, [814](#)
- insert, [815](#)
- lower\_bound\_node, [816](#)
- rbegin, [817](#)
- remove, [818](#)
- rend, [819](#)
- cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, AL-  
LOC, GET\_KEY >::Node, [820](#)
- operator->, [822](#)
- operator\*, [822](#)
- valid, [822](#)
- cxx::Bits::Basic\_list< POLICY >, [823](#)
- clear, [824](#)
- iter, [824](#)
- cxx::Bits::Bst< Node, Get\_key, Compare >, [825](#)
- begin, [828](#), [829](#)
- dir, [829](#), [830](#)
- end, [831](#)
- find, [832](#)
- find\_node, [832](#)
- lower\_bound\_node, [833](#)
- rbegin, [834](#), [835](#)
- remove\_all, [836](#)
- remove\_tree, [837](#)
- rend, [838](#)
- cxx::Bits::Bst\_node, [839](#)
- cxx::Bits::Direction, [841](#)
- Direction\_e, [842](#)
- L, [842](#)
- N, [842](#)
- operator!, [842](#)
- R, [842](#)
- cxx::Bits::Smart\_ptr\_list< ITEM >, [843](#)
- pop\_front, [846](#)
- cxx::Bits::Smart\_ptr\_list\_item< T, STORE\_T >, [846](#)
- cxx::H\_list< T, POLICY >, [848](#)
- erase, [851](#)
- insert, [851](#)
- insert\_after, [852](#)
- insert\_before, [853](#)
- iter, [853](#)
- pop\_front, [854](#)
- remove, [855](#)
- replace, [855](#)
- cxx::H\_list\_item\_t< ELEM\_TYPE >, [856](#)

- ~H\_list\_item\_t, [858](#)
- H\_list\_item\_t, [858](#)
- cxx::H\_list\_t< T >, [859](#)
- cxx::List< D, Alloc >, [862](#)
  - operator[], [863](#)
- cxx::List< D, Alloc >::Iter, [864](#)
- cxx::List\_alloc, [865](#)
  - alloc, [866](#)
  - alloc\_max, [866](#)
  - avail, [867](#)
  - free, [867](#)
  - List\_alloc, [865](#)
- cxx::List\_item, [868](#)
  - push\_back, [869](#)
  - push\_front, [870](#)
  - remove, [871](#)
- cxx::List\_item::Iter, [872](#)
- cxx::List\_item::T\_iter< T, Poly >, [873](#)
- cxx::Lt\_functor< Obj >, [875](#)
- cxx::New\_allocator< \_Type >, [875](#)
- cxx::Nothrow, [876](#)
- cxx::Pair< First, Second >, [877](#)
  - Pair, [878](#)
- cxx::Pair\_first\_compare< Cmp, Typ >, [880](#)
  - operator(), [881](#)
  - Pair\_first\_compare, [881](#)
- cxx::Ref\_obj\_list\_item< T >, [882](#)
- cxx::Ref\_ptr< T, CNT >, [883](#)
  - get, [886](#)
  - ptr, [886](#)
  - Ref\_ptr, [885](#)
  - release, [886](#)
- cxx::S\_list< T, POLICY >, [887](#)
  - pop\_front, [889](#)
- cxx::Slab< Type, Slab\_size, Max\_free, Alloc >, [890](#)
  - alloc, [893](#)
  - free, [893](#)
- cxx::Slab\_static< Type, Slab\_size, Max\_free, Alloc >, [894](#)
  - alloc, [897](#)
- cxx::static\_vector< T, IDX >, [898](#)
- cxx::String, [899](#)
  - find, [902](#)
  - from\_dec, [903](#)
  - from\_hex, [903](#)
  - starts\_with, [904](#)
  - String, [902](#)
- cxx::Weak\_ref< T >, [905](#)
- cxx::Weak\_ref\_base, [908](#)
- cxx::Weak\_ref\_base::List, [911](#)
- d\_tag
  - Elf32\_Dyn, [917](#)
  - Elf64\_Dyn, [927](#)
- data
  - L4::lpc::Varg, [1105](#)
- Data\_buffer
  - L4virtio::Svr::Data\_buffer, [1901](#)
- data\_size
  - L4virtio::Svr::Block\_request< Ds\_data >, [1857](#)
- data\_space
  - L4Re::Vfs::Be\_file, [1630](#)
  - L4Re::Vfs::Regular\_file, [1659](#)
- Dataspace interface, [474](#)
  - l4re\_ds\_allocate, [475](#)
  - l4re\_ds\_clear, [476](#)
  - l4re\_ds\_copy\_in, [476](#)
  - L4RE\_DS\_F\_BUFFERABLE, [475](#)
  - L4RE\_DS\_F\_CACHEABLE, [475](#)
  - L4RE\_DS\_F\_CACHING\_MASK, [475](#)
  - L4RE\_DS\_F\_CACHING\_SHIFT, [475](#)
  - L4RE\_DS\_F\_NORMAL, [475](#)
  - L4RE\_DS\_F\_UNCACHEABLE, [475](#)
  - l4re\_ds\_flags, [477](#)
  - l4re\_ds\_info, [477](#)
  - l4re\_ds\_map\_flags, [475](#)
  - l4re\_ds\_map\_info, [478](#)
  - l4re\_ds\_size, [478](#)
- debug
  - L4Re::Debug\_obj, [1461](#)
- Debug interface, [479](#)
  - l4re\_debug\_obj\_debug, [479](#)
- Debugging API, [524](#)
- dec
  - L4Re::Util::Counter< COUNTER >, [1569](#)
  - L4Re::Util::Counter\_atomic< COUNTER >, [1572](#)
- dec\_refcnt
  - L4::Kobject, [1191](#)
- Dedicated
  - L4::Kip::Mem\_desc, [1181](#)
- delete\_buffer
  - L4Re::Video::Goos, [1672](#)
- delete\_obj
  - L4::Task, [1284](#)
- delete\_view
  - L4Re::Video::Goos, [1672](#)
- Demand
  - L4::Kobject\_typeid< T >, [1202](#)
  - L4::Type\_info::Demand, [1317](#)
- demand
  - L4::Kobject\_typeid< T >, [1203](#)
- Deprecated List, [83](#)
- desc
  - L4virtio::Driver::Virtqueue, [1836](#)
  - L4virtio::Svr::Virtqueue, [1970](#)
  - L4virtio::Svr::Virtqueue::Head\_desc, [1976](#)
- desc\_align
  - L4virtio::Virtqueue, [1982](#)
- desc\_avail
  - L4virtio::Svr::Virtqueue, [1971](#)
- desc\_size
  - L4virtio::Virtqueue, [1982](#)
- detach
  - L4::Irq, [1161](#)
  - L4Re::Rm, [1539–1541](#)
  - L4Re::Util::Event\_buffer\_t< PAYLOAD >, [1593](#)
- Detach\_again

- L4Re::Rm, [1537](#)
- Detach\_exact
  - L4Re::Rm, [1537](#)
- Detach\_flags
  - L4Re::Rm, [1536](#)
- Detach\_free
  - L4Re::Rm::F, [1547](#)
- Detach\_keep
  - L4Re::Rm, [1537](#)
- Detach\_overlap
  - L4Re::Rm, [1537](#)
- Detach\_result
  - L4Re::Rm, [1537](#)
- Detached\_ds
  - L4Re::Rm, [1537](#)
- Dev\_config
  - L4virtio::Svr::Dev\_config, [1905](#), [1906](#)
- dev\_handle
  - L4vbus::Device, [1724](#)
- Device
  - L4vbus::Device, [1723](#)
  - L4virtio::Svr::Console::Device, [1866](#), [1867](#)
- device
  - L4vbus::Device, [1725](#)
- Device\_add
  - L4virtio::Svr::Console::Control\_message, [1860](#)
- device\_by\_hid
  - L4vbus::Device, [1725](#)
- device\_config
  - L4virtio::Device, [1796](#)
- device\_error
  - L4virtio::Svr::Device\_t< DATA >, [1923](#)
- device\_notification\_irq
  - L4virtio::Device, [1796](#)
- device\_notify\_irq
  - L4virtio::Svr::Device\_t< DATA >, [1923](#)
- Device\_ready
  - L4virtio::Svr::Console::Control\_message, [1860](#)
- Device\_remove
  - L4virtio::Svr::Console::Control\_message, [1860](#)
- DF\_1\_CONFALT
  - ELF binary format, [623](#)
- DF\_1\_DIRECT
  - ELF binary format, [622](#)
- DF\_1\_DISPRELDNE
  - ELF binary format, [623](#)
- DF\_1\_DISPRELPND
  - ELF binary format, [623](#)
- DF\_1\_ENDFILTEE
  - ELF binary format, [623](#)
- DF\_1\_GLOBAL
  - ELF binary format, [622](#)
- DF\_1\_GROUP
  - ELF binary format, [622](#)
- DF\_1\_INITFIRST
  - ELF binary format, [622](#)
- DF\_1\_INTERPOSE
  - ELF binary format, [622](#)
- DF\_1\_LOADFLTR
  - ELF binary format, [622](#)
- DF\_1\_NODEFLIB
  - ELF binary format, [623](#)
- DF\_1\_NODELETE
  - ELF binary format, [622](#)
- DF\_1\_NODUMP
  - ELF binary format, [623](#)
- DF\_1\_NOOPEN
  - ELF binary format, [622](#)
- DF\_1\_NOW
  - ELF binary format, [622](#)
- DF\_1\_ORIGIN
  - ELF binary format, [622](#)
- DF\_BIND\_NOW
  - ELF binary format, [623](#)
- DF\_ORIGIN
  - ELF binary format, [623](#)
- DF\_P1\_GROUPPERM
  - ELF binary format, [623](#)
- DF\_P1\_LAZYLOAD
  - ELF binary format, [623](#)
- DF\_STATIC\_TLS
  - ELF binary format, [623](#)
- DF\_SYMBOLIC
  - ELF binary format, [623](#)
- DF\_TEXTREL
  - ELF binary format, [623](#)
- di
  - l4\_vcpu\_regs\_t, [1411](#)
- dir
  - cxx::Bits::Bst< Node, Get\_key, Compare >, [829](#), [830](#)
- Direction
  - L4Re::Dma\_space, [1465](#)
- Direction\_e
  - cxx::Bits::Direction, [842](#)
- disable
  - L4virtio::Virtqueue, [1983](#)
- disable\_notify
  - L4virtio::Svr::Virtqueue, [1971](#)
- disassociate
  - L4Re::Dma\_space, [1466](#)
- dispatch
  - L4::Basic\_registry, [942](#)
  - L4::Epiface, [984](#)
  - L4::Epiface\_t< Derived, IFACE, BASE, bool >, [990](#)
  - L4::Irqp\_t< Derived, BASE, bool >, [1178](#)
  - L4::Server\_object, [1261](#), [1262](#)
- dispatch\_meta\_request
  - L4::Server\_object\_t< IFACE, BASE >, [1267](#)
- DMA space, [195](#)
- DMA Space Interface, [470](#)
  - l4re\_dma\_space\_associate, [471](#)
  - l4re\_dma\_space\_disassociate, [472](#)
  - l4re\_dma\_space\_map, [472](#)
  - l4re\_dma\_space\_t, [471](#)
  - l4re\_dma\_space\_unmap, [473](#)



dma\_space.h  
     L4RE\_DMA\_SPACE\_BIDIRECTIONAL, 2440  
     L4RE\_DMA\_SPACE\_COHERENT, 2441  
     l4re\_dma\_space\_direction, 2440  
     L4RE\_DMA\_SPACE\_FROM\_DEVICE, 2440  
     L4RE\_DMA\_SPACE\_NONE, 2440  
     L4RE\_DMA\_SPACE\_PHYS\_SPACE, 2441  
     l4re\_dma\_space\_space\_attribs, 2440  
     L4RE\_DMA\_SPACE\_TO\_DEVICE, 2440  
 do\_control\_work  
     L4virtio::Svr::Console::Device, 1868  
 done  
     L4virtio::Svr::Data\_buffer, 1902  
 down  
     L4::Semaphore, 1252  
 drive\_cylinders  
     l4util\_mb\_drive\_t, 1713  
 drive\_mode  
     l4util\_mb\_drive\_t, 1713  
 drive\_number  
     l4util\_mb\_drive\_t, 1713  
 driver\_acknowledge  
     L4virtio::Driver::Device, 1814  
 driver\_connect  
     L4virtio::Driver::Device, 1815  
 Driver\_mem\_region\_t  
     L4virtio::Svr::Driver\_mem\_region\_t< DATA >, 1938  
 drv\_base  
     L4virtio::Svr::Driver\_mem\_region\_t< DATA >, 1940  
 ds  
     L4virtio::Svr::Dev\_config, 1907  
     L4virtio::Svr::Driver\_mem\_region\_t< DATA >, 1940  
 Ds\_map\_mask  
     L4Re::Rm::F, 1547  
 ds\_offset  
     L4virtio::Svr::Driver\_mem\_region\_t< DATA >, 1940  
 DT\_BIND\_NOW  
     ELF binary format, 624  
 DT\_DEBUG  
     ELF binary format, 624  
 DT\_ENCODING  
     ELF binary format, 624  
 DT\_FINI  
     ELF binary format, 624  
 DT\_FINI\_ARRAY  
     ELF binary format, 624  
 DT\_FINI\_ARRAYSZ  
     ELF binary format, 624  
 DT\_FLAGS  
     ELF binary format, 624  
 DT\_HASH  
     ELF binary format, 624  
 DT\_HIOS  
     ELF binary format, 624  
 DT\_HIPROC  
     ELF binary format, 624  
 DT\_INIT  
     ELF binary format, 624  
 DT\_INIT\_ARRAY  
     ELF binary format, 624  
 DT\_INIT\_ARRAYSZ  
     ELF binary format, 624  
 DT\_JMPREL  
     ELF binary format, 624  
 DT\_LOOS  
     ELF binary format, 624  
 DT\_LOPROC  
     ELF binary format, 624  
 DT\_NEEDED  
     ELF binary format, 624  
 DT\_NULL  
     ELF binary format, 624  
 DT\_NUM  
     ELF binary format, 624  
 DT\_PLTGOT  
     ELF binary format, 624  
 DT\_PLTRELSZ  
     ELF binary format, 624  
 DT\_PREINIT\_ARRAY  
     ELF binary format, 624  
 DT\_PREINIT\_ARRAYSZ  
     ELF binary format, 624  
 DT\_PTRREL  
     ELF binary format, 624  
 DT\_REL  
     ELF binary format, 624  
 DT\_RELA  
     ELF binary format, 624  
 DT\_RELAENT  
     ELF binary format, 624  
 DT\_RELASZ  
     ELF binary format, 624  
 DT\_RELENT  
     ELF binary format, 624  
 DT\_RELSZ  
     ELF binary format, 624  
 DT\_RPATH  
     ELF binary format, 624  
 DT\_RUNPATH  
     ELF binary format, 624  
 DT\_SONAME  
     ELF binary format, 624  
 DT\_STRSZ  
     ELF binary format, 624  
 DT\_STRTAB  
     ELF binary format, 624  
 DT\_SYMBOLIC  
     ELF binary format, 624  
 DT\_SYMENT  
     ELF binary format, 624  
 DT\_SYMTAB  
     ELF binary format, 624



- DT\_TEXTREL
  - ELF binary format, [624](#)
- dump
  - L4Re::Video::Color\_component, [1665](#)
  - L4Re::Video::Pixel\_info, [1681](#)
  - L4virtio::Virtqueue, [1984](#)
- dx
  - l4\_vcpu\_regs\_t, [1412](#)
- E\_exist
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [810](#)
- e\_flags
  - Elf32\_Ehdr, [918](#)
  - Elf64\_Ehdr, [928](#)
- E\_inval
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [810](#)
- e\_machine
  - Elf32\_Ehdr, [918](#)
  - Elf64\_Ehdr, [928](#)
- E\_noent
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [810](#)
- E\_nomem
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [810](#)
- e\_type
  - Elf32\_Ehdr, [919](#)
  - Elf64\_Ehdr, [929](#)
- e\_version
  - Elf32\_Ehdr, [919](#)
  - Elf64\_Ehdr, [929](#)
- Eager\_map
  - L4Re::Rm::F, [1546](#)
- EDID parsing functionality, [401](#)
  - libedid\_block\_size, [402](#)
  - libedid\_check\_header, [402](#)
  - libedid\_checksum, [403](#)
  - libedid\_consts, [402](#)
  - libedid\_dump, [403](#)
  - libedid\_dump\_standard\_timings, [403](#)
  - libedid\_num\_ext\_blocks, [404](#)
  - libedid\_pnp\_id, [404](#)
  - libedid\_preferred\_resolution, [404](#)
  - libedid\_revision, [404](#)
  - libedid\_version, [405](#)
- EF\_ARM\_ALIGN8
  - ELF binary format, [625](#)
- EI\_ABIVERSION
  - ELF binary format, [625](#)
- EI\_CLASS
  - ELF binary format, [625](#)
- EI\_DATA
  - ELF binary format, [625](#)
- EI\_MAG0
  - ELF binary format, [625](#)
- EI\_MAG1
  - ELF binary format, [625](#)
- EI\_MAG2
  - ELF binary format, [625](#)
- EI\_MAG3
  - ELF binary format, [625](#)
- EI\_NIDENT
  - ELF binary format, [620](#)
- EI\_OSABI
  - ELF binary format, [625](#)
- EI\_PAD
  - ELF binary format, [625](#)
- EI\_VERSION
  - ELF binary format, [625](#)
- ELF binary format, [613](#)
  - AT\_BASE, [621](#)
  - AT\_EGID, [621](#)
  - AT\_ENTRY, [621](#)
  - AT\_EUID, [621](#)
  - AT\_EXECFD, [621](#)
  - AT\_FLAGS, [621](#)
  - AT\_GID, [621](#)
  - AT\_IGNORE, [621](#)
  - AT\_L4\_AUX, [621](#)
  - AT\_L4\_ENV, [621](#)
  - AT\_NOTELF, [621](#)
  - AT\_NULL, [621](#)
  - AT\_PAGESZ, [621](#)
  - AT\_PHDR, [621](#)
  - AT\_PHEMT, [621](#)
  - AT\_PHNUM, [621](#)
  - AT\_UID, [621](#)
  - DF\_1\_CONFALT, [623](#)
  - DF\_1\_DIRECT, [622](#)
  - DF\_1\_DISPRELDNE, [623](#)
  - DF\_1\_DISPRELPND, [623](#)
  - DF\_1\_ENDFILTEE, [623](#)
  - DF\_1\_GLOBAL, [622](#)
  - DF\_1\_GROUP, [622](#)
  - DF\_1\_INITFIRST, [622](#)
  - DF\_1\_INTERPOSE, [622](#)
  - DF\_1\_LOADFLTR, [622](#)
  - DF\_1\_NODEFLIB, [623](#)
  - DF\_1\_NODELETE, [622](#)
  - DF\_1\_NODUMP, [623](#)
  - DF\_1\_NOOPEN, [622](#)
  - DF\_1\_NOW, [622](#)
  - DF\_1\_ORIGIN, [622](#)
  - DF\_BIND\_NOW, [623](#)
  - DF\_ORIGIN, [623](#)
  - DF\_P1\_GROUPPERM, [623](#)
  - DF\_P1\_LAZYLOAD, [623](#)
  - DF\_STATIC\_TLS, [623](#)
  - DF\_SYMBOLIC, [623](#)
  - DF\_TEXTREL, [623](#)
  - DT\_BIND\_NOW, [624](#)
  - DT\_DEBUG, [624](#)
  - DT\_ENCODING, [624](#)
  - DT\_FINI, [624](#)
  - DT\_FINI\_ARRAY, [624](#)

DT\_FINI\_ARRAYSZ, [624](#)  
DT\_FLAGS, [624](#)  
DT\_HASH, [624](#)  
DT\_HIOS, [624](#)  
DT\_HIPROC, [624](#)  
DT\_INIT, [624](#)  
DT\_INIT\_ARRAY, [624](#)  
DT\_INIT\_ARRAYSZ, [624](#)  
DT\_JMPREL, [624](#)  
DT\_LOOS, [624](#)  
DT\_LOPROC, [624](#)  
DT\_NEEDED, [624](#)  
DT\_NULL, [624](#)  
DT\_NUM, [624](#)  
DT\_PLTGOT, [624](#)  
DT\_PLTRELSZ, [624](#)  
DT\_PREINIT\_ARRAY, [624](#)  
DT\_PREINIT\_ARRAYSZ, [624](#)  
DT\_PTRREL, [624](#)  
DT\_REL, [624](#)  
DT\_RELA, [624](#)  
DT\_RELAENT, [624](#)  
DT\_RELASZ, [624](#)  
DT\_RELENT, [624](#)  
DT\_RELSZ, [624](#)  
DT\_RPATH, [624](#)  
DT\_RUNPATH, [624](#)  
DT\_SONAME, [624](#)  
DT\_STRSZ, [624](#)  
DT\_STRTAB, [624](#)  
DT\_SYMBOLIC, [624](#)  
DT\_SYMENT, [624](#)  
DT\_SYMTAB, [624](#)  
DT\_TEXTREL, [624](#)  
EF\_ARM\_ALIGN8, [625](#)  
EI\_ABIVERSION, [625](#)  
EI\_CLASS, [625](#)  
EI\_DATA, [625](#)  
EI\_MAG0, [625](#)  
EI\_MAG1, [625](#)  
EI\_MAG2, [625](#)  
EI\_MAG3, [625](#)  
EI\_NIDENT, [620](#)  
EI\_OSABI, [625](#)  
EI\_PAD, [625](#)  
EI\_VERSION, [625](#)  
ELF32\_R\_TYPE, [619](#)  
ELF32\_ST\_BIND, [619](#)  
ELF32\_ST\_TYPE, [619](#)  
ELF64\_R\_TYPE, [619](#)  
ELF64\_ST\_BIND, [620](#)  
ELF64\_ST\_TYPE, [620](#)  
Elf\_ARM\_SBs, [620](#)  
Elf\_ATs, [621](#)  
Elf\_Classes, [621](#)  
Elf\_DATAs, [622](#)  
Elf\_DF\_1s, [622](#)  
Elf\_DF\_P1s, [623](#)  
Elf\_DFs, [623](#)  
Elf\_DTs, [623](#)  
Elf\_EF\_ARM\_s, [624](#)  
Elf\_EIs, [625](#)  
Elf\_EMs, [625](#)  
Elf\_ETs, [627](#)  
Elf\_EVs, [627](#)  
Elf\_MAGs, [628](#)  
Elf\_NT\_core, [628](#)  
Elf\_NT\_obj, [629](#)  
Elf\_OSABIs, [629](#)  
ELF\_PFs, [630](#)  
Elf\_PTs, [630](#)  
Elf\_R\_386\_s, [631](#)  
Elf\_R\_AARCH64\_s, [632](#)  
Elf\_R\_ARM\_s, [632](#)  
Elf\_R\_X86\_64\_s, [633](#)  
Elf\_SHF\_s\_ARM, [633](#)  
Elf\_SHFs, [634](#)  
Elf\_SHNs, [634](#)  
Elf\_SHTs, [634](#)  
Elf\_STBs, [635](#)  
Elf\_STTs, [636](#)  
ELFCLASS32, [622](#)  
ELFCLASS64, [622](#)  
ELFCLASSNONE, [622](#)  
ELFCLASSNUM, [622](#)  
ELFDATA2LSB, [622](#)  
ELFDATA2MSB, [622](#)  
ELFDATANONE, [622](#)  
ELFDATANUM, [622](#)  
ELFMAG0, [628](#)  
ELFMAG1, [628](#)  
ELFMAG2, [628](#)  
ELFMAG3, [628](#)  
ELFOSABI\_AIX, [629](#)  
ELFOSABI\_ARM, [630](#)  
ELFOSABI\_FREEBSD, [630](#)  
ELFOSABI\_HPUX, [629](#)  
ELFOSABI\_IRIX, [629](#)  
ELFOSABI\_LINUX, [629](#)  
ELFOSABI\_MODESTO, [630](#)  
ELFOSABI\_NETBSD, [629](#)  
ELFOSABI\_NONE, [629](#)  
ELFOSABI\_OPENBSD, [630](#)  
ELFOSABI\_SOLARIS, [629](#)  
ELFOSABI\_STANDALONE, [630](#)  
ELFOSABI\_SYSV, [629](#)  
ELFOSABI\_TRU64, [630](#)  
EM\_386, [626](#)  
EM\_68HC05, [626](#)  
EM\_68HC08, [626](#)  
EM\_68HC11, [626](#)  
EM\_68HC12, [626](#)  
EM\_68HC16, [626](#)  
EM\_68K, [626](#)  
EM\_860, [626](#)  
EM\_88K, [626](#)

EM\_960, [626](#)  
EM\_AARCH64, [627](#)  
EM\_ALPHA, [626](#)  
EM\_ALTERA\_NIOS2, [627](#)  
EM\_ARC, [626](#)  
EM\_ARC\_A5, [627](#)  
EM\_ARM, [626](#)  
EM\_AVR, [627](#)  
EM\_COLDFIRE, [626](#)  
EM\_CRIS, [626](#)  
EM\_D10V, [627](#)  
EM\_D30V, [627](#)  
EM\_FIREPATH, [626](#)  
EM\_FR20, [626](#)  
EM\_FR30, [627](#)  
EM\_FX66, [626](#)  
EM\_H8\_300, [626](#)  
EM\_H8\_300H, [626](#)  
EM\_H8\_500, [626](#)  
EM\_H8S, [626](#)  
EM\_HUANY, [627](#)  
EM\_IA\_64, [626](#)  
EM\_JAVELIN, [626](#)  
EM\_M32, [626](#)  
EM\_M32R, [627](#)  
EM\_MICROBLAZE, [627](#)  
EM\_MIPS, [626](#)  
EM\_MIPS\_RS4\_BE, [626](#)  
EM\_MIPS\_X, [626](#)  
EM\_MMIX, [627](#)  
EM\_MN10200, [627](#)  
EM\_MN10300, [627](#)  
EM\_NONE, [626](#)  
EM\_OPENRISC, [627](#)  
EM\_PARISC, [626](#)  
EM\_PDSP, [626](#)  
EM\_PJ, [627](#)  
EM\_PPC, [626](#)  
EM\_PRISM, [627](#)  
EM\_RCE, [626](#)  
EM\_RH32, [626](#)  
EM\_RISCV, [627](#)  
EM\_SH, [626](#)  
EM\_SPARC, [626](#)  
EM\_SPARC32PLUS, [626](#)  
EM\_SPARC64, [626](#)  
EM\_SPARCV9, [626](#)  
EM\_ST19, [626](#)  
EM\_ST7, [626](#)  
EM\_ST9PLUS, [626](#)  
EM\_SVX, [626](#)  
EM\_TILEGX, [627](#)  
EM\_TILEPRO, [627](#)  
EM\_TRICORE, [626](#)  
EM\_V800, [626](#)  
EM\_V850, [627](#)  
EM\_VAX, [626](#)  
EM\_VPP500, [626](#)  
EM\_X86\_64, [626](#)  
EM\_XTENSA, [627](#)  
EM\_ZSP, [627](#)  
ET\_CORE, [627](#)  
ET\_DYN, [627](#)  
ET\_EXEC, [627](#)  
ET\_HIPROC, [627](#)  
ET\_LOPROC, [627](#)  
ET\_NONE, [627](#)  
ET\_REL, [627](#)  
EV\_CURRENT, [628](#)  
EV\_NONE, [628](#)  
NT\_ASRS, [629](#)  
NT\_AUXV, [628](#)  
NT\_FPREGSET, [628](#)  
NT\_GWINDOWS, [628](#)  
NT\_LWPSINFO, [629](#)  
NT\_LWPSTATUS, [629](#)  
NT\_PLATFORM, [628](#)  
NT\_PRURED, [629](#)  
NT\_PRFPXREG, [629](#)  
NT\_PRPSINFO, [628](#)  
NT\_PRSTATUS, [628](#)  
NT\_PRXREG, [628](#)  
NT\_PSINFO, [629](#)  
NT\_PSTATUS, [629](#)  
NT\_TASKSTRUCT, [628](#)  
NT\_UTSNAME, [629](#)  
NT\_VERSION, [629](#)  
PF\_ARM\_SB, [621](#)  
PF\_MASKOS, [630](#)  
PF\_MASKPROC, [630](#)  
PF\_R, [630](#)  
PF\_W, [630](#)  
PF\_X, [630](#)  
PT\_DYNAMIC, [630](#)  
PT\_GNU\_EH\_FRAME, [631](#)  
PT\_GNU\_RELRO, [631](#)  
PT\_GNU\_STACK, [631](#)  
PT\_HIOS, [631](#)  
PT\_HIPROC, [631](#)  
PT\_INTERP, [630](#)  
PT\_L4\_AUX, [631](#)  
PT\_L4\_KIP, [631](#)  
PT\_L4\_STACK, [631](#)  
PT\_LOAD, [630](#)  
PT\_LOOS, [630](#)  
PT\_LOPROC, [631](#)  
PT\_NOTE, [630](#)  
PT\_NULL, [630](#)  
PT\_NUM, [630](#)  
PT\_PHDR, [630](#)  
PT\_SHLIB, [630](#)  
PT\_TLS, [630](#)  
R\_386\_32, [631](#)  
R\_386\_COPY, [631](#)  
R\_386\_GLOB\_DAT, [631](#)  
R\_386\_GOT32, [631](#)

R\_386\_GOTOFF, [631](#)  
R\_386\_GOTPC, [631](#)  
R\_386\_JMP\_SLOT, [631](#)  
R\_386\_NONE, [631](#)  
R\_386\_NUM, [632](#)  
R\_386\_PC32, [631](#)  
R\_386\_PLT32, [631](#)  
R\_386\_RELATIVE, [631](#)  
R\_386\_TLS\_DTPMOD32, [632](#)  
R\_386\_TLS\_DTPOFF32, [632](#)  
R\_386\_TLS\_GD, [631](#)  
R\_386\_TLS\_GD\_32, [631](#)  
R\_386\_TLS\_GD\_CALL, [631](#)  
R\_386\_TLS\_GD\_POP, [631](#)  
R\_386\_TLS\_GD\_PUSH, [631](#)  
R\_386\_TLS\_GOTIE, [631](#)  
R\_386\_TLS\_IE, [631](#)  
R\_386\_TLS\_IE\_32, [631](#)  
R\_386\_TLS\_LDM, [631](#)  
R\_386\_TLS\_LDM\_32, [631](#)  
R\_386\_TLS\_LDM\_CALL, [631](#)  
R\_386\_TLS\_LDM\_POP, [631](#)  
R\_386\_TLS\_LDM\_PUSH, [631](#)  
R\_386\_TLS\_LDO\_32, [631](#)  
R\_386\_TLS\_LE, [631](#)  
R\_386\_TLS\_LE\_32, [632](#)  
R\_386\_TLS\_TPOFF, [631](#)  
R\_386\_TLS\_TPOFF32, [632](#)  
R\_AARCH64\_NONE, [632](#)  
R\_ARM\_ABS12, [632](#)  
R\_ARM\_ABS16, [632](#)  
R\_ARM\_ABS32, [632](#)  
R\_ARM\_ABS8, [632](#)  
R\_ARM\_COPY, [632](#)  
R\_ARM\_GLOB\_DAT, [632](#)  
R\_ARM\_GOT32, [632](#)  
R\_ARM\_GOTOFF, [632](#)  
R\_ARM\_GOTPC, [632](#)  
R\_ARM\_JUMP\_SLOT, [632](#)  
R\_ARM\_NONE, [632](#)  
R\_ARM\_NUM, [632](#)  
R\_ARM\_PC24, [632](#)  
R\_ARM\_PLT32, [632](#)  
R\_ARM\_REL32, [632](#)  
R\_ARM\_RELATIVE, [632](#)  
R\_ARM\_THM\_PC11, [632](#)  
R\_ARM\_THM\_PC9, [632](#)  
R\_X86\_64\_16, [633](#)  
R\_X86\_64\_32, [633](#)  
R\_X86\_64\_32S, [633](#)  
R\_X86\_64\_64, [633](#)  
R\_X86\_64\_8, [633](#)  
R\_X86\_64\_COPY, [633](#)  
R\_X86\_64\_DTPMOD64, [633](#)  
R\_X86\_64\_DTPOFF32, [633](#)  
R\_X86\_64\_DTPOFF64, [633](#)  
R\_X86\_64\_GLOB\_DAT, [633](#)  
R\_X86\_64\_GOT32, [633](#)  
R\_X86\_64\_GOTPCREL, [633](#)  
R\_X86\_64\_GOTTPOFF, [633](#)  
R\_X86\_64\_JUMP\_SLOT, [633](#)  
R\_X86\_64\_NONE, [633](#)  
R\_X86\_64\_PC16, [633](#)  
R\_X86\_64\_PC32, [633](#)  
R\_X86\_64\_PC8, [633](#)  
R\_X86\_64\_PLT32, [633](#)  
R\_X86\_64\_RELATIVE, [633](#)  
R\_X86\_64\_TLSGD, [633](#)  
R\_X86\_64\_TLSLD, [633](#)  
R\_X86\_64\_TPOFF32, [633](#)  
R\_X86\_64\_TPOFF64, [633](#)  
SHF\_ALLOC, [634](#)  
SHF\_ARM\_COMDEF, [633](#)  
SHF\_ARM\_ENTRYSECT, [633](#)  
SHF\_EXECINSTR, [634](#)  
SHF\_GROUP, [634](#)  
SHF\_INFO\_LINK, [634](#)  
SHF\_LINK\_ORDER, [634](#)  
SHF\_MASKOS, [634](#)  
SHF\_MASKPROC, [634](#)  
SHF\_MERGE, [634](#)  
SHF\_OS\_NONCONFORMING, [634](#)  
SHF\_STRINGS, [634](#)  
SHF\_TLS, [634](#)  
SHF\_WRITE, [634](#)  
SHN\_ABS, [634](#)  
SHN\_COMMON, [634](#)  
SHN\_HIPROC, [634](#)  
SHN\_HIRESERVE, [634](#)  
SHN\_LOPROC, [634](#)  
SHN\_LORESERVE, [634](#)  
SHN\_UNDEF, [634](#)  
SHT\_DYNAMIC, [635](#)  
SHT\_DYNSYM, [635](#)  
SHT\_FINI\_ARRAY, [635](#)  
SHT\_GROUP, [635](#)  
SHT\_HASH, [635](#)  
SHT\_HIOS, [635](#)  
SHT\_HIPROC, [635](#)  
SHT\_HIUSER, [635](#)  
SHT\_INIT\_ARRAY, [635](#)  
SHT\_LOOS, [635](#)  
SHT\_LOPROC, [635](#)  
SHT\_LOUSER, [635](#)  
SHT\_NOBITS, [635](#)  
SHT\_NOTE, [635](#)  
SHT\_NULL, [635](#)  
SHT\_NUM, [635](#)  
SHT\_PREINIT\_ARRAY, [635](#)  
SHT\_PROGBITS, [635](#)  
SHT\_REL, [635](#)  
SHT\_RELA, [635](#)  
SHT\_SHLIB, [635](#)  
SHT\_STRTAB, [635](#)  
SHT\_SYMTAB, [635](#)  
SHT\_SYMTAB\_SHNDX, [635](#)

- STB\_GLOBAL, [635](#)
- STB\_HIOS, [635](#)
- STB\_HIPROC, [635](#)
- STB\_LOCAL, [635](#)
- STB\_LOOS, [635](#)
- STB\_LOPROC, [635](#)
- STB\_WEAK, [635](#)
- STT\_FILE, [636](#)
- STT\_FUNC, [636](#)
- STT\_HIOS, [636](#)
- STT\_HIPROC, [636](#)
- STT\_LOOS, [636](#)
- STT\_LOPROC, [636](#)
- STT\_NOTYPE, [636](#)
- STT\_OBJECT, [636](#)
- STT\_SECTION, [636](#)
- Elf32\_Auxv, [915](#)
  - atype, [916](#)
- Elf32\_Dyn, [916](#)
  - d\_tag, [917](#)
- Elf32\_Ehdr, [917](#)
  - e\_flags, [918](#)
  - e\_machine, [918](#)
  - e\_type, [919](#)
  - e\_version, [919](#)
- Elf32\_Phdr, [920](#)
  - p\_flags, [921](#)
  - p\_type, [921](#)
- ELF32\_R\_TYPE
  - ELF binary format, [619](#)
- Elf32\_Rel, [921](#)
- Elf32\_Rela, [922](#)
- Elf32\_Shdr, [923](#)
  - sh\_flags, [924](#)
  - sh\_type, [924](#)
- ELF32\_ST\_BIND
  - ELF binary format, [619](#)
- ELF32\_ST\_TYPE
  - ELF binary format, [619](#)
- Elf32\_Sym, [924](#)
- Elf64\_Auxv, [925](#)
  - atype, [926](#)
- Elf64\_Dyn, [926](#)
  - d\_tag, [927](#)
- Elf64\_Ehdr, [927](#)
  - e\_flags, [928](#)
  - e\_machine, [928](#)
  - e\_type, [929](#)
  - e\_version, [929](#)
- Elf64\_Phdr, [930](#)
  - p\_flags, [931](#)
  - p\_type, [931](#)
- ELF64\_R\_TYPE
  - ELF binary format, [619](#)
- Elf64\_Rel, [931](#)
- Elf64\_Rela, [932](#)
- Elf64\_Shdr, [933](#)
  - sh\_flags, [934](#)
  - sh\_type, [934](#)
- ELF64\_ST\_BIND
  - ELF binary format, [620](#)
- ELF64\_ST\_TYPE
  - ELF binary format, [620](#)
- Elf64\_Sym, [934](#)
- Elf\_ARM\_SBs
  - ELF binary format, [620](#)
- Elf\_ATs
  - ELF binary format, [621](#)
- Elf\_CIASs
  - ELF binary format, [621](#)
- Elf\_DATAs
  - ELF binary format, [622](#)
- Elf\_DF\_1s
  - ELF binary format, [622](#)
- Elf\_DF\_P1s
  - ELF binary format, [623](#)
- Elf\_DFs
  - ELF binary format, [623](#)
- Elf\_DTs
  - ELF binary format, [623](#)
- Elf\_EF\_ARM\_s
  - ELF binary format, [624](#)
- Elf\_EIs
  - ELF binary format, [625](#)
- Elf\_EMs
  - ELF binary format, [625](#)
- Elf\_ETs
  - ELF binary format, [627](#)
- Elf\_EVs
  - ELF binary format, [627](#)
- Elf\_MAGs
  - ELF binary format, [628](#)
- Elf\_NT\_s\_core
  - ELF binary format, [628](#)
- Elf\_NT\_s\_obj
  - ELF binary format, [629](#)
- Elf\_OSABIs
  - ELF binary format, [629](#)
- ELF\_PFs
  - ELF binary format, [630](#)
- Elf\_PT\_s
  - ELF binary format, [630](#)
- Elf\_R\_386\_s
  - ELF binary format, [631](#)
- Elf\_R\_AARCH64\_s
  - ELF binary format, [632](#)
- Elf\_R\_ARM\_s
  - ELF binary format, [632](#)
- Elf\_R\_X86\_64\_s
  - ELF binary format, [633](#)
- Elf\_SHF\_s\_ARM
  - ELF binary format, [633](#)
- Elf\_SHFs
  - ELF binary format, [634](#)
- Elf\_SHNs
  - ELF binary format, [634](#)

- Elf\_SHTs
  - ELF binary format, [634](#)
- Elf\_STBs
  - ELF binary format, [635](#)
- Elf\_STTs
  - ELF binary format, [636](#)
- ELFCLASS32
  - ELF binary format, [622](#)
- ELFCLASS64
  - ELF binary format, [622](#)
- ELFCLASSNONE
  - ELF binary format, [622](#)
- ELFCLASSNUM
  - ELF binary format, [622](#)
- ELFDATA2LSB
  - ELF binary format, [622](#)
- ELFDATA2MSB
  - ELF binary format, [622](#)
- ELFDATANONE
  - ELF binary format, [622](#)
- ELFDATANUM
  - ELF binary format, [622](#)
- ELFMAG0
  - ELF binary format, [628](#)
- ELFMAG1
  - ELF binary format, [628](#)
- ELFMAG2
  - ELF binary format, [628](#)
- ELFMAG3
  - ELF binary format, [628](#)
- ELFOSABI\_AIX
  - ELF binary format, [629](#)
- ELFOSABI\_ARM
  - ELF binary format, [630](#)
- ELFOSABI\_FREEBSD
  - ELF binary format, [630](#)
- ELFOSABI\_HPUX
  - ELF binary format, [629](#)
- ELFOSABI\_IRIX
  - ELF binary format, [629](#)
- ELFOSABI\_LINUX
  - ELF binary format, [629](#)
- ELFOSABI\_MODESTO
  - ELF binary format, [630](#)
- ELFOSABI\_NETBSD
  - ELF binary format, [629](#)
- ELFOSABI\_NONE
  - ELF binary format, [629](#)
- ELFOSABI\_OPENBSD
  - ELF binary format, [630](#)
- ELFOSABI\_SOLARIS
  - ELF binary format, [629](#)
- ELFOSABI\_STANDALONE
  - ELF binary format, [630](#)
- ELFOSABI\_SYSV
  - ELF binary format, [629](#)
- ELFOSABI\_TRU64
  - ELF binary format, [630](#)
- EM\_386
  - ELF binary format, [626](#)
- EM\_68HC05
  - ELF binary format, [626](#)
- EM\_68HC08
  - ELF binary format, [626](#)
- EM\_68HC11
  - ELF binary format, [626](#)
- EM\_68HC12
  - ELF binary format, [626](#)
- EM\_68HC16
  - ELF binary format, [626](#)
- EM\_68K
  - ELF binary format, [626](#)
- EM\_860
  - ELF binary format, [626](#)
- EM\_88K
  - ELF binary format, [626](#)
- EM\_960
  - ELF binary format, [626](#)
- EM\_AARCH64
  - ELF binary format, [627](#)
- EM\_ALPHA
  - ELF binary format, [626](#)
- EM\_ALTERA\_NIOS2
  - ELF binary format, [627](#)
- EM\_ARC
  - ELF binary format, [626](#)
- EM\_ARC\_A5
  - ELF binary format, [627](#)
- EM\_ARM
  - ELF binary format, [626](#)
- EM\_AVR
  - ELF binary format, [627](#)
- EM\_COLDFIRE
  - ELF binary format, [626](#)
- EM\_CRIS
  - ELF binary format, [626](#)
- EM\_D10V
  - ELF binary format, [627](#)
- EM\_D30V
  - ELF binary format, [627](#)
- EM\_FIREPATH
  - ELF binary format, [626](#)
- EM\_FR20
  - ELF binary format, [626](#)
- EM\_FR30
  - ELF binary format, [627](#)
- EM\_FX66
  - ELF binary format, [626](#)
- EM\_H8\_300
  - ELF binary format, [626](#)
- EM\_H8\_300H
  - ELF binary format, [626](#)
- EM\_H8\_500
  - ELF binary format, [626](#)
- EM\_H8S
  - ELF binary format, [626](#)

- EM\_HUANY
  - ELF binary format, [627](#)
- EM\_IA\_64
  - ELF binary format, [626](#)
- EM\_JAVELIN
  - ELF binary format, [626](#)
- EM\_M32
  - ELF binary format, [626](#)
- EM\_M32R
  - ELF binary format, [627](#)
- EM\_MICROBLAZE
  - ELF binary format, [627](#)
- EM\_MIPS
  - ELF binary format, [626](#)
- EM\_MIPS\_RS4\_BE
  - ELF binary format, [626](#)
- EM\_MIPS\_X
  - ELF binary format, [626](#)
- EM\_MMX
  - ELF binary format, [627](#)
- EM\_MN10200
  - ELF binary format, [627](#)
- EM\_MN10300
  - ELF binary format, [627](#)
- EM\_NONE
  - ELF binary format, [626](#)
- EM\_OPENRISC
  - ELF binary format, [627](#)
- EM\_PARISC
  - ELF binary format, [626](#)
- EM\_PDSP
  - ELF binary format, [626](#)
- EM\_PJ
  - ELF binary format, [627](#)
- EM\_PPC
  - ELF binary format, [626](#)
- EM\_PRISM
  - ELF binary format, [627](#)
- EM\_RCE
  - ELF binary format, [626](#)
- EM\_RH32
  - ELF binary format, [626](#)
- EM\_RISCV
  - ELF binary format, [627](#)
- EM\_SH
  - ELF binary format, [626](#)
- EM\_SPARC
  - ELF binary format, [626](#)
- EM\_SPARC32PLUS
  - ELF binary format, [626](#)
- EM\_SPARC64
  - ELF binary format, [626](#)
- EM\_SPARCV9
  - ELF binary format, [626](#)
- EM\_ST19
  - ELF binary format, [626](#)
- EM\_ST7
  - ELF binary format, [626](#)
- EM\_ST9PLUS
  - ELF binary format, [626](#)
- EM\_SVX
  - ELF binary format, [626](#)
- EM\_TILEGX
  - ELF binary format, [627](#)
- EM\_TILEPRO
  - ELF binary format, [627](#)
- EM\_TRICORE
  - ELF binary format, [626](#)
- EM\_V800
  - ELF binary format, [626](#)
- EM\_V850
  - ELF binary format, [627](#)
- EM\_VAX
  - ELF binary format, [626](#)
- EM\_VPP500
  - ELF binary format, [626](#)
- EM\_X86\_64
  - ELF binary format, [626](#)
- EM\_XTENSA
  - ELF binary format, [627](#)
- EM\_ZSP
  - ELF binary format, [627](#)
- emerg\_write\_bfm\_t
  - L4virtio::Svr::Console::Features, [1880](#)
- empty
  - L4virtio::Svr::Driver\_mem\_region\_t< DATA >, [1941](#)
- enable\_notify
  - L4virtio::Svr::Virtqueue, [1971](#)
- enable\_rx\_irq
  - L4::Uart, [1358](#)
  - L4::Uart\_apb, [1365](#)
- end
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [812](#), [813](#)
  - cxx::Bits::Bst< Node, Get\_key, Compare >, [831](#)
  - L4::Kip::Mem\_desc, [1184](#)
- enqueue\_descriptor
  - L4virtio::Driver::Virtqueue, [1837](#)
- enter\_kdebug
  - kdebug.h, [2834](#)
- entry\_ip
  - L4vcpu::Vcpu, [1783](#)
- entry\_sp
  - L4vcpu::Vcpu, [1784](#)
- env
  - L4Re::Env, [1472](#)
- env.h
  - l4re\_env\_t, [2505](#)
- erase
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [814](#)
  - cxx::H\_list< T, POLICY >, [851](#)
- err\_no
  - L4::Runtime\_error, [1241](#)
- Error

- L4virtio::Svr::Bad\_descriptor, 1847
- Error codes, 146
  - L4\_E2BIG, 147
  - L4\_EACCESS, 147
  - L4\_EADDRNOTAVAIL, 147
  - L4\_EAGAIN, 147
  - L4\_EBADPROTO, 147
  - L4\_EBUSY, 147
  - L4\_EEXIST, 147
  - L4\_EFAULT, 147
  - L4\_EINVAL, 147
  - L4\_EIO, 147
  - L4\_EIPC\_HI, 148
  - L4\_EIPC\_LO, 148
  - L4\_EMSGMISSARG, 148
  - L4\_EMSGTOOLONG, 148
  - L4\_EMSGTOOSHORT, 148
  - L4\_ENAMETOOLONG, 147
  - L4\_ENODEV, 147
  - L4\_ENOENT, 147
  - L4\_ENOMEM, 147
  - L4\_ENOREPLY, 148
  - L4\_ENOSPC, 147
  - L4\_ENOSYS, 147
  - L4\_ENOTDIR, 147
  - L4\_ENXIO, 147
  - L4\_EOK, 147
  - L4\_EPERM, 147
  - L4\_ERANGE, 147
  - L4\_ERRNOMAX, 147
  - l4\_error\_code\_t, 147
- Error Handling, 358
  - l4\_error, 360
  - L4\_IPC\_ENOT\_EXISTENT, 359
  - l4\_ipc\_error, 361
  - l4\_ipc\_error\_code, 362
  - L4\_IPC\_ERROR\_MASK, 359
  - l4\_ipc\_is\_rcv\_error, 363
  - l4\_ipc\_is\_snd\_error, 363
  - L4\_IPC\_REABORTED, 360
  - L4\_IPC\_RECANCELED, 360
  - L4\_IPC\_REMAPFAILED, 360
  - L4\_IPC\_REMSGCUT, 360
  - L4\_IPC\_RERCVPFTO, 360
  - L4\_IPC\_RESNDPFTO, 360
  - L4\_IPC\_RETIMEOUT, 359
  - L4\_IPC\_SEABORTED, 360
  - L4\_IPC\_SECANCELED, 360
  - L4\_IPC\_SEMAPFAILED, 360
  - L4\_IPC\_SEMSGCUT, 360
  - L4\_IPC\_SERCVPFTO, 360
  - L4\_IPC\_SESNDPFTO, 360
  - L4\_IPC\_SETIMEOUT, 360
  - L4\_IPC\_SND\_ERR\_MASK, 359
  - l4\_ipc\_tcr\_error\_t, 359
- ET\_CORE
  - ELF binary format, 627
- ET\_DYN
  - ELF binary format, 627
- ET\_EXEC
  - ELF binary format, 627
- ET\_HIPROC
  - ELF binary format, 627
- ET\_LOPROC
  - ELF binary format, 627
- ET\_NONE
  - ELF binary format, 627
- ET\_REL
  - ELF binary format, 627
- EV\_CURRENT
  - ELF binary format, 628
- EV\_NONE
  - ELF binary format, 628
- Event API, 525
- Event interface, 480
  - l4re\_event\_get\_axis\_info, 480
  - l4re\_event\_get\_buffer, 481
  - l4re\_event\_get\_num\_streams, 481
  - l4re\_event\_get\_stream\_info, 482
  - l4re\_event\_get\_stream\_info\_for\_id, 482
- Event\_buffer\_t
  - L4Re::Event\_buffer\_t< PAYLOAD >, 1490
- Events
  - L4virtio::Svr::Console::Control\_message, 1860
- ex\_regs
  - L4::Thread, 1293, 1294
- exc\_handler
  - L4::Thread::Attr, 1305
- exception
  - L4::Exception, 993
- Exception registers, 396
  - l4\_utcb\_exc, 397
  - l4\_utcb\_exc\_is\_ex\_regs\_exception, 397
  - l4\_utcb\_exc\_is\_pf, 398
  - l4\_utcb\_exc\_pc, 398
  - l4\_utcb\_exc\_pc\_set, 399
- execute
  - L4Re::Ned::Cmd\_control, 1521, 1522
- expired
  - Block\_device::Errand::Errand, 727
  - Block\_device::Errand::Poll\_errand, 730
  - L4::lpc\_svr::Timeout, 1145
- ext\_alloc
  - L4vcpu::Vcpu, 1784
- Extended vCPU support, 662
  - l4vcpu\_ext\_alloc, 662
- extra\_str
  - L4::Runtime\_error, 1241
- F\_above
  - L4Re::Video::View, 1686
- F\_auto\_refresh
  - L4Re::Video::Goos, 1671
- F\_dyn\_allocated
  - L4Re::Video::View, 1685
- F\_dynamic\_buffers
  - L4Re::Video::Goos, 1671



- F\_dynamic\_views
  - L4Re::Video::Goos, [1671](#)
- F\_flags\_mask
  - L4Re::Video::View, [1686](#)
- F\_fully\_dynamic
  - L4Re::Video::View, [1685](#)
- F\_l4re\_video\_goos\_auto\_refresh
  - Video API, [515](#)
- F\_l4re\_video\_goos\_dynamic\_buffers
  - Video API, [515](#)
- F\_l4re\_video\_goos\_dynamic\_views
  - Video API, [515](#)
- F\_l4re\_video\_goos\_pointer
  - Video API, [515](#)
- F\_l4re\_video\_view\_above
  - Video API, [515](#)
- F\_l4re\_video\_view\_dyn\_allocated
  - Video API, [515](#)
- F\_l4re\_video\_view\_flags\_mask
  - Video API, [515](#)
- F\_l4re\_video\_view\_none
  - Video API, [515](#)
- F\_l4re\_video\_view\_set\_background
  - Video API, [515](#)
- F\_l4re\_video\_view\_set\_buffer
  - Video API, [515](#)
- F\_l4re\_video\_view\_set\_buffer\_offset
  - Video API, [515](#)
- F\_l4re\_video\_view\_set\_bytes\_per\_line
  - Video API, [515](#)
- F\_l4re\_video\_view\_set\_flags
  - Video API, [515](#)
- F\_l4re\_video\_view\_set\_pixel
  - Video API, [515](#)
- F\_l4re\_video\_view\_set\_position
  - Video API, [515](#)
- F\_none
  - L4Re::Video::View, [1685](#)
- F\_pointer
  - L4Re::Video::Goos, [1671](#)
- F\_set\_background
  - L4Re::Video::View, [1685](#)
- F\_set\_buffer
  - L4Re::Video::View, [1685](#)
- F\_set\_buffer\_offset
  - L4Re::Video::View, [1685](#)
- F\_set\_bytes\_per\_line
  - L4Re::Video::View, [1685](#)
- F\_set\_flags
  - L4Re::Video::View, [1685](#)
- F\_set\_pixel
  - L4Re::Video::View, [1685](#)
- F\_set\_position
  - L4Re::Video::View, [1685](#)
- faccessat
  - L4Re::Vfs::Directory, [1636](#)
- Factory, [196](#)
  - l4\_factory\_create, [197](#)
  - l4\_factory\_create\_factory, [198](#)
  - l4\_factory\_create\_gate, [199](#)
  - l4\_factory\_create\_irq, [200](#)
  - l4\_factory\_create\_task, [201](#)
  - l4\_factory\_create\_thread, [202](#)
  - l4\_factory\_create\_vcpu\_context, [203](#)
  - l4\_factory\_create\_vm, [204](#)
- factory
  - L4Re::Env, [1472](#)
- fail\_request
  - Block\_device::Pending\_request, [740](#)
- fchmod
  - L4Re::Vfs::Generic\_file, [1649](#)
- fdatasync
  - L4Re::Vfs::Regular\_file, [1659](#)
- feature\_negotiated
  - L4virtio::Driver::Device, [1816](#)
- features
  - l4\_icu\_info\_t, [1390](#)
- Fiasco extensions, [148](#)
  - fiasco\_gdt\_get\_entry\_offset, [149](#)
  - fiasco\_gdt\_set, [149](#)
  - fiasco\_ldt\_set, [150](#)
- Fiasco real time scheduling extensions, [151](#)
- Fiasco-UX Virtual devices, [188](#)
  - L4\_TYPE\_VHW\_FRAMEBUFFER, [189](#)
  - L4\_TYPE\_VHW\_INPUT, [189](#)
  - L4\_TYPE\_VHW\_NET, [189](#)
  - L4\_TYPE\_VHW\_NONE, [189](#)
  - l4\_vhw\_entry\_type, [189](#)
- fiasco\_amd64\_segment\_info
  - segment.h, [2071](#)
- fiasco\_amd64\_set\_fs
  - segment.h, [2072](#), [2076](#)
- fiasco\_amd64\_set\_segment\_base
  - segment.h, [2073](#), [2077](#)
- fiasco\_gdt\_get\_entry\_offset
  - Fiasco extensions, [149](#)
- fiasco\_gdt\_set
  - Fiasco extensions, [149](#)
- fiasco\_ldt\_set
  - Fiasco extensions, [150](#)
- fiasco\_tbuf\_log
  - Kernel Tracing, [160](#)
- fiasco\_tbuf\_log\_3val
  - Kernel Tracing, [160](#)
- fiasco\_tbuf\_log\_binary
  - Kernel Tracing, [161](#)
- finalize\_request
  - L4virtio::Svr::Block\_dev\_base< Ds\_data >, [1853](#)
- find
  - cxx::Bits::Bst< Node, Get\_key, Compare >, [832](#)
  - cxx::String, [902](#)
  - L4::Basic\_registry, [943](#)
  - L4Re::Rm, [1541](#)
  - L4virtio::Svr::Driver\_mem\_list\_t< DATA >, [1930](#)
- find\_next\_used
  - L4virtio::Driver::Virtqueue, [1838](#)

- find\_node
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, 814
  - cxx::Bits::Bst< Node, Get\_key, Compare >, 832
- finish
  - L4virtio::Svr::Virtqueue, 1972, 1973
- finish\_rx
  - L4virtio::Driver::Virtio\_net\_device, 1826
- first
  - L4::Kip::Mem\_desc, 1185
- first\_free\_cap
  - L4Re::Env, 1473
- first\_free\_utcb
  - L4Re::Env, 1473
- Fixed\_paddr
  - L4Re::Mem\_alloc, 1508
- Flags
  - L4::Type\_info::Demand\_t< CAPS, FLAGS, MEM, PORTS >, 1320
  - L4::Types::Flags< BITS\_ENUM, UNDERLYING >, 1344
  - L4Re::Dataspace::F, 1457
  - L4Re::Video::Goos, 1671
  - L4Re::Video::View, 1685
- flags
  - l4\_exc\_regs\_t, 1386
  - l4\_msgtag\_t, 1396
  - L4Re::Dataspace, 1452
  - l4re\_env\_cap\_entry\_t, 1695
  - L4virtio::Svr::Driver\_mem\_region\_t< DATA >, 1941
- Flex pages, 162
  - L4\_CAP\_FPAGE\_D, 166
  - L4\_CAP\_FPAGE\_R, 166
  - L4\_cap\_fpage\_rights, 165
  - L4\_CAP\_FPAGE\_RO, 166
  - L4\_CAP\_FPAGE\_RS, 166
  - L4\_CAP\_FPAGE\_RSD, 166
  - L4\_CAP\_FPAGE\_RW, 166
  - L4\_CAP\_FPAGE\_RWD, 166
  - L4\_CAP\_FPAGE\_RWS, 166
  - L4\_CAP\_FPAGE\_RWSD, 166
  - L4\_CAP\_FPAGE\_S, 165
  - L4\_CAP\_FPAGE\_W, 165
  - l4\_fpage, 169
  - L4\_FPAGE\_ADDR\_BITS, 167
  - L4\_FPAGE\_ADDR\_SHIFT, 167
  - l4\_fpage\_all, 169
  - L4\_FPAGE\_BUFFERABLE, 167
  - L4\_FPAGE\_C\_IPCGATE\_SVR, 169
  - L4\_FPAGE\_C\_NO\_REF\_CNT, 169
  - L4\_FPAGE\_C\_OBJ\_RIGHT1, 169
  - L4\_FPAGE\_C\_OBJ\_RIGHT2, 169
  - L4\_FPAGE\_C\_OBJ\_RIGHT3, 169
  - L4\_FPAGE\_C\_OBJ\_RIGHTS, 169
  - L4\_FPAGE\_C\_REF\_CNT, 168
  - L4\_FPAGE\_CACHE\_OPT, 167
  - l4\_fpage\_cacheability\_opt\_t, 166
  - L4\_FPAGE\_CACHEABLE, 167
  - L4\_fpage\_consts, 167
  - l4\_fpage\_contains, 170
  - L4\_fpage\_control, 167
  - L4\_FPAGE\_CONTROL\_MASK, 167
  - L4\_FPAGE\_CONTROL\_OFFSET\_SHIFT, 167
  - l4\_fpage\_invalid, 171
  - L4\_FPAGE\_IO, 168
  - l4\_fpage\_ioport, 171
  - l4\_fpage\_max\_order, 172
  - l4\_fpage\_memaddr, 172
  - L4\_FPAGE\_MEMORY, 168
  - L4\_FPAGE\_OBJ, 168
  - l4\_fpage\_obj, 174
  - l4\_fpage\_page, 175
  - L4\_fpage\_rights, 167
  - l4\_fpage\_rights, 175
  - L4\_FPAGE\_RIGHTS\_ALL, 167
  - L4\_FPAGE\_RIGHTS\_BITS, 167
  - L4\_FPAGE\_RIGHTS\_MASK, 167
  - L4\_FPAGE\_RIGHTS\_SHIFT, 167
  - L4\_FPAGE\_RO, 168
  - L4\_FPAGE\_RW, 168
  - L4\_FPAGE\_RWX, 168
  - L4\_FPAGE\_RX, 168
  - l4\_fpage\_set\_rights, 176
  - l4\_fpage\_size, 176
  - L4\_FPAGE\_SIZE\_BITS, 167
  - L4\_FPAGE\_SIZE\_SHIFT, 167
  - L4\_FPAGE\_SPECIAL, 168
  - L4\_fpage\_type, 168
  - l4\_fpage\_type, 177
  - L4\_FPAGE\_TYPE\_BITS, 167
  - L4\_FPAGE\_TYPE\_SHIFT, 167
  - L4\_FPAGE\_UNCACHEABLE, 167
  - L4\_FPAGE\_W, 168
  - L4\_FPAGE\_X, 168
  - l4\_iofpage, 177
  - L4\_IOPORT\_MAX, 165
  - l4\_is\_fpage\_writable, 178
  - l4\_obj\_fpage, 179
  - L4\_obj\_fpage\_ctl, 168
  - L4\_WHOLE\_ADDRESS\_SPACE, 165
  - L4\_WHOLE\_IOADDRESS\_SPACE, 165
- font.h
  - gfxbitmap\_font\_data, 2429
  - gfxbitmap\_font\_get, 2430
  - gfxbitmap\_font\_height, 2430
  - gfxbitmap\_font\_init, 2430
  - gfxbitmap\_font\_text, 2430
  - gfxbitmap\_font\_text\_scale, 2431
  - gfxbitmap\_font\_width, 2432
- foreach\_available\_event
  - L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >, 1587
- fpage
  - L4::Cap\_base, 958
- free

- cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >, [766](#)
  - cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >, [772](#)
  - cxx::List\_alloc, [867](#)
  - cxx::Slab< Type, Slab\_size, Max\_free, Alloc >, [893](#)
  - L4Re::Cap\_alloc, [1442](#)
  - L4Re::Util::Counting\_cap\_alloc< COUNTER-TYPE, Dbg >, [1575](#)
- free\_area
  - L4Re::Rm, [1542](#)
- free\_capability
  - L4Re::Util::Names::Name\_space, [1605](#)
- free\_descriptor
  - L4virtio::Driver::Virtqueue, [1838](#)
- free\_dynamic\_entry
  - L4Re::Util::Names::Name\_space, [1605](#)
- free\_epiface
  - L4Re::Util::Names::Name\_space, [1606](#)
- free\_fd
  - L4Re::Vfs::Fs, [1646](#)
- free\_objects
  - cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >, [767](#)
  - cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >, [773](#)
- from\_ci
  - L4::lpc::Cap< T >, [1042](#)
- from\_dec
  - cxx::String, [903](#)
- From\_device
  - L4Re::Dma\_space, [1465](#)
- from\_hex
  - cxx::String, [903](#)
- from\_raw
  - L4::Types::Flags< BITS\_ENUM, UNDERLYING >, [1345](#)
- fstat64
  - L4Re::Vfs::Be\_file, [1630](#)
  - L4Re::Vfs::Generic\_file, [1649](#)
- fsync
  - L4Re::Vfs::Regular\_file, [1659](#)
- ftruncate64
  - L4Re::Vfs::Regular\_file, [1659](#)
- full
  - L4virtio::Svr::Driver\_mem\_list\_t< DATA >, [1931](#)
- Functions for rendering bitmap data in frame buffers, [608](#)
- Functions for rendering bitmap fonts to frame buffers, [608](#)
- Functions to manipulate the local IDT, [636](#)
- g
  - L4Re::Video::Pixel\_info, [1681](#)
- generic\_write
  - L4::Uart, [1359](#)
- get
  - cxx::Bitfield< T, LSB, MSB >, [777](#)
  - cxx::Ref\_ptr< T, CNT >, [886](#)
  - L4::lpc::lostream, [1051](#), [1052](#)
  - L4::lpc::lstream, [1060](#), [1061](#)
  - L4Re::Env, [1474](#)
  - L4Re::Rm::Unique\_region< T >, [1551](#)
  - L4Re::Video::Color\_component, [1665](#)
  - L4vbus::Gpio\_module, [1733](#)
  - L4vbus::Gpio\_pin, [1743](#)
  - L4virtio::Ptr< T >, [1844](#)
- get\_areas
  - L4Re::Rm, [1543](#)
- get\_attr
  - L4::Vcon, [1373](#)
- get\_avail\_idx
  - L4virtio::Virtqueue, [1984](#)
- get\_axis\_info
  - L4Re::Event, [1485](#)
- get\_buffer
  - L4Re::Event, [1486](#)
- get\_buffer\_demand
  - L4::Epiface, [985](#)
  - L4::Server\_object\_t< IFACE, BASE >, [1267](#)
- get\_cap
  - L4Re::Env, [1474](#), [1476](#)
- get\_cap\_alloc
  - L4Re::Cap\_alloc, [1442](#)
- get\_char
  - L4::Uart, [1359](#)
  - L4::Uart\_apb, [1365](#)
- get\_cmd
  - L4virtio::Svr::Dev\_config, [1908](#)
- get\_epiface
  - L4Re::Util::Names::Name\_space, [1607](#)
- get\_fb
  - L4Re::Util::Video::Goos\_svr, [1625](#)
- get\_file
  - L4Re::Vfs::Fs, [1646](#)
- get\_infos
  - L4::lpc\_gate, [1119](#)
- get\_lock
  - L4Re::Vfs::Regular\_file, [1660](#)
- get\_num\_streams
  - L4Re::Event, [1486](#)
- get\_object\_name
  - L4::Debugger, [971](#)
- get\_random
  - L4Re::Random, [1530](#)
- get\_rcv\_cap
  - L4::lpc\_svr::Server\_iface, [1140](#)
  - L4Re::Util::Br\_manager, [1560](#)
- get\_regions
  - L4Re::Rm, [1543](#)
- get\_resource
  - L4vbus::Device, [1726](#)
- get\_static\_buffer
  - L4Re::Video::Goos, [1673](#)
- get\_status\_flags
  - L4Re::Vfs::Generic\_file, [1650](#)
- get\_stream\_info

- L4Re::Event, [1486](#)
- get\_stream\_info\_for\_id
  - L4Re::Event, [1487](#)
- get\_stream\_state\_for\_id
  - L4Re::Event, [1487](#)
- get\_tail\_avail\_idx
  - L4virtio::Virtqueue, [1985](#)
- get\_unshifted
  - cxx::Bitfield< T, LSB, MSB >, [778](#)
- get\_value
  - L4::lpc::Varg, [1105](#)
- get\_writeback
  - L4virtio::Svr::Block\_dev\_base< Ds\_data >, [1854](#)
- gfxbitmap\_bmap
  - bitmap.h, [2424](#)
- gfxbitmap\_color\_pix\_t
  - bitmap.h, [2424](#)
- gfxbitmap\_color\_t
  - bitmap.h, [2424](#)
- gfxbitmap\_convert\_color
  - bitmap.h, [2425](#)
- gfxbitmap\_copy
  - bitmap.h, [2425](#)
- gfxbitmap\_fill
  - bitmap.h, [2425](#)
- gfxbitmap\_font\_data
  - font.h, [2429](#)
- gfxbitmap\_font\_get
  - font.h, [2430](#)
- gfxbitmap\_font\_height
  - font.h, [2430](#)
- gfxbitmap\_font\_init
  - font.h, [2430](#)
- gfxbitmap\_font\_text
  - font.h, [2430](#)
- gfxbitmap\_font\_text\_scale
  - font.h, [2431](#)
- gfxbitmap\_font\_width
  - font.h, [2432](#)
- gfxbitmap\_offset, [935](#)
- gfxbitmap\_set
  - bitmap.h, [2426](#)
- global\_id
  - L4::Debugger, [971](#)
- gran\_offset
  - l4\_sched\_cpu\_set\_t, [1400](#)
- granularity
  - l4\_sched\_cpu\_set\_t, [1398](#)
- guest\_features
  - L4virtio::Svr::Dev\_config, [1908](#)
- H\_list\_item\_t
  - cxx::H\_list\_item\_t< ELEM\_TYPE >, [858](#)
- handle\_control\_message
  - L4virtio::Svr::Console::Virtio\_con, [1889](#)
- handle\_expired\_timeouts
  - L4::lpc\_svr::Timeout\_queue, [1147](#)
- handle\_mem\_cmd\_write
  - L4virtio::Svr::Device\_t< DATA >, [1923](#)
- handle\_request
  - Block\_device::Pending\_request, [740](#)
- has\_alpha
  - L4Re::Video::Pixel\_info, [1682](#)
- has\_more
  - L4virtio::Svr::Request\_processor, [1945](#)
- hdr
  - L4virtio::Svr::Dev\_config, [1909](#)
- i
  - L4vcpu::Vcpu, [1785](#)
- IA32 Port I/O API, [637](#)
- l4util\_in16, [637](#)
- l4util\_in32, [638](#)
- l4util\_in8, [638](#)
- l4util\_ins16, [639](#)
- l4util\_ins32, [639](#)
- l4util\_ins8, [639](#)
- l4util\_out16, [640](#)
- l4util\_out32, [640](#)
- l4util\_out8, [640](#)
- l4util\_outs16, [642](#)
- l4util\_outs32, [642](#)
- l4util\_outs8, [643](#)
- id
  - L4::Kobject\_typeid< T >, [1203](#)
- id\_received
  - L4::lpc::Gen\_fpage, [1044](#)
- idle\_time
  - L4::Scheduler, [1245](#)
- In\_area
  - L4Re::Rm::F, [1546](#)
- inc
  - L4Re::Util::Counter< COUNTER >, [1569](#)
  - L4Re::Util::Counter\_atomic< COUNTER >, [1572](#)
- include.mk - Header File Role, [32](#)
- indirect\_bfm\_t
  - L4virtio::Virtqueue::Desc::Flags, [2000](#)
- Info
  - L4::Kip::Mem\_desc, [1181](#)
- info
  - L4::lcu, [1015](#)
  - L4::Scheduler, [1246](#)
  - L4Re::Dataspace, [1453](#)
  - L4Re::Video::Goos, [1673](#)
  - L4Re::Video::View, [1686](#)
- Info\_acpi\_rsdp
  - L4::Kip::Mem\_desc, [1181](#)
- Info\_sub\_type
  - L4::Kip::Mem\_desc, [1181](#)
- inhibitor.h
  - l4re\_inhibitor\_acquire, [2443](#)
  - l4re\_inhibitor\_next\_lock\_info, [2444](#)
  - l4re\_inhibitor\_release, [2444](#)
- init
  - L4Re::Util::Event\_t< PAYLOAD >, [1597](#)
  - L4virtio::Svr::Driver\_mem\_list\_t< DATA >, [1932](#)
- init\_infos
  - L4Re::Util::Video::Goos\_svr, [1625](#)

- init\_mem\_info
  - L4virtio::Svr::Device\_t< DATA >, 1924
- init\_poll
  - L4Re::Util::Event\_t< PAYLOAD >, 1598
- init\_queue
  - L4virtio::Driver::Virtqueue, 1839, 1840
- Initial Environment, 483
  - l4re\_env, 484
  - l4re\_env\_get\_cap, 484
  - l4re\_env\_get\_cap\_e, 485
  - l4re\_env\_get\_cap\_l, 486
  - l4re\_kip, 487
- Initial Environment and Application Bootstrapping, 20
- Initial Memory Allocator and Factory, 23
- initial\_caps
  - L4Re::Env, 1476
- initialize\_rings
  - L4virtio::Driver::Virtqueue, 1841
- insert
  - cxx::Avl\_map< KEY\_TYPE, DATA\_TYPE, COMPARE, ALLOC >, 748
  - cxx::Avl\_tree< Node, Get\_key, Compare >, 759
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, 815
  - cxx::H\_list< T, POLICY >, 851
- insert\_after
  - cxx::H\_list< T, POLICY >, 852
- insert\_before
  - cxx::H\_list< T, POLICY >, 853
- Integer Types, 180
- interface
  - L4::Meta, 1209
- Interface Definition Language, 25
- Interface for asynchronous ISR handlers with a given IRQ capability., 416
  - l4irq\_request\_cap, 416
- Interface for asynchronous ISR handlers., 414
  - l4irq\_release, 415
  - l4irq\_request, 415
- Interface using direct functionality., 417, 421
  - l4irq\_attach, 418
  - l4irq\_attach\_cap, 422
  - l4irq\_attach\_cap\_ft, 422
  - l4irq\_attach\_ft, 418
  - l4irq\_attach\_thread, 418
  - l4irq\_attach\_thread\_cap, 422
  - l4irq\_attach\_thread\_cap\_ft, 423
  - l4irq\_attach\_thread\_ft, 419
  - l4irq\_detach, 419
  - l4irq\_unmask, 420
  - l4irq\_unmask\_and\_wait\_any, 420
  - l4irq\_wait, 420
  - l4irq\_wait\_any, 421
- Internal, 538
  - L4SHMC\_RINGBUF\_DATA, 539
  - L4SHMC\_RINGBUF\_DATA\_SIZE, 539
  - L4SHMC\_RINGBUF\_HEAD, 539
- Internal constants, 571
- Internal functions, 643
- Internal Helpers, 138
- internal\_loop
  - L4::Server< LOOP\_HOOKS >, 1257
- Interrupt controller, 222
  - l4\_icu\_bind, 224
  - l4\_icu\_bind\_u, 225
  - L4\_ICU\_FLAG\_MSI, 224
  - l4\_icu\_flags, 224
  - l4\_icu\_info, 226
  - l4\_icu\_info\_t, 224
  - l4\_icu\_info\_u, 227
  - l4\_icu\_mask, 228
  - l4\_icu\_mask\_u, 229
  - l4\_icu\_msi\_info, 230
  - l4\_icu\_msi\_info\_u, 231
  - l4\_icu\_set\_mode, 232
  - l4\_icu\_set\_mode\_u, 233
  - l4\_icu\_unbind, 234
  - l4\_icu\_unbind\_u, 235
  - l4\_icu\_unmask, 236
  - l4\_icu\_unmask\_u, 237
- Introduction, 3
- Invalid
  - L4::Cap\_base, 955
  - L4virtio::Ptr< T >, 1844
- Invalid\_capability
  - L4::Invalid\_capability, 1023
- Invalid\_type
  - L4virtio::Ptr< T >, 1844
- IO interface, 405
  - L4IO\_DEVICE\_ANY, 407
  - L4IO\_DEVICE\_INVALID, 407
  - L4IO\_DEVICE\_OTHER, 407
  - L4IO\_DEVICE\_PCI, 407
  - l4io\_device\_types\_t, 406
  - L4IO\_DEVICE\_USB, 407
  - l4io\_has\_resource, 407
  - l4io\_iomem\_flags\_t, 407
  - l4io\_lookup\_device, 408
  - l4io\_lookup\_resource, 408
  - L4IO\_MEM\_CACHED, 407
  - L4IO\_MEM\_EAGER\_MAP, 407
  - L4IO\_MEM\_NONCACHED, 407
  - L4IO\_MEM\_USE\_MTRR, 407
  - L4IO\_MEM\_USE\_RESERVED\_AREA, 407
  - l4io\_release\_iomem, 409
  - l4io\_release\_ioport, 409
  - l4io\_request\_iomem, 409
  - l4io\_request\_iomem\_region, 410
  - l4io\_request\_ioport, 411
  - l4io\_request\_resource\_iomem, 411
  - L4IO\_RESOURCE\_ANY, 407
  - L4IO\_RESOURCE\_INVALID, 407
  - L4IO\_RESOURCE\_IRQ, 407
  - L4IO\_RESOURCE\_MEM, 407
  - L4IO\_RESOURCE\_PORT, 407
  - l4io\_resource\_t, 406

- l4io\_resource\_types\_t, [407](#)
- Io, the Io Server, [53](#)
- io.h
  - l4io\_get\_root\_device, [2140](#)
  - l4io\_iterate\_devices, [2140](#)
  - l4io\_request\_all\_ioports, [2140](#)
  - l4io\_request\_icu, [2141](#)
- io\_page\_fault
  - L4::Io\_pager, [1025](#)
- ioctl
  - L4Re::Vfs::Special\_file, [1663](#)
- lostream
  - L4::lpc::lostream, [1050](#)
- IPC Helpers, [412](#)
  - throw\_ipc\_exception, [412](#), [413](#)
- IPC-Gate API, [205](#)
  - l4\_ipc\_gate\_get\_infos, [206](#)
  - l4\_rcv\_ep\_bind\_thread, [207](#)
- ipc.h
  - l4\_ipc\_to\_errno, [2800](#)
- ipc\_client
  - L4\_RPC\_DEF, [2723](#)
- ipc\_iface
  - L4\_INLINE\_RPC, [2730](#)
  - L4\_INLINE\_RPC\_NF, [2731](#)
  - L4\_INLINE\_RPC\_NF\_OP, [2731](#)
  - L4\_INLINE\_RPC\_OP, [2732](#)
  - L4\_RPC, [2732](#)
  - L4\_RPC\_NF, [2733](#)
  - L4\_RPC\_NF\_OP, [2733](#)
  - L4\_RPC\_OP, [2733](#)
- ipc\_stream
  - operator<<, [2226–2228](#)
  - operator>>, [2228–2232](#)
- irq
  - L4Re::Util::Event\_t< PAYLOAD >, [1598](#)
- IRQ handling library, [414](#)
- irq.h
  - l4util\_irq\_acknowledge, [2815](#), [2826](#)
- irq\_disable\_save
  - L4vcpu::Vcpu, [1785](#)
- irq\_enable
  - L4vbus::Pci\_dev, [1755](#)
  - L4vbus::Pci\_host\_bridge, [1761](#)
  - L4vcpu::Vcpu, [1785](#)
- irq\_restore
  - L4vcpu::Vcpu, [1786](#)
- IRQs, [208](#)
  - l4\_irq\_detach, [210](#)
  - l4\_irq\_detach\_u, [211](#)
  - L4\_IRQ\_F\_BOTH, [210](#)
  - L4\_IRQ\_F\_BOTH\_EDGE, [210](#)
  - L4\_IRQ\_F\_CLEAR\_WAKEUP, [210](#)
  - L4\_IRQ\_F\_EDGE, [210](#)
  - L4\_IRQ\_F\_LEVEL, [210](#)
  - L4\_IRQ\_F\_LEVEL\_HIGH, [210](#)
  - L4\_IRQ\_F\_LEVEL\_LOW, [210](#)
  - L4\_IRQ\_F\_MASK, [210](#)
  - L4\_IRQ\_F\_NEG, [210](#)
  - L4\_IRQ\_F\_NEG\_EDGE, [210](#)
  - L4\_IRQ\_F\_NONE, [210](#)
  - L4\_IRQ\_F\_POS, [210](#)
  - L4\_IRQ\_F\_POS\_EDGE, [210](#)
  - L4\_IRQ\_F\_SET\_WAKEUP, [210](#)
  - L4\_irq\_mode, [209](#)
  - l4\_irq\_mux\_chain, [212](#)
  - l4\_irq\_mux\_chain\_u, [213](#)
  - l4\_irq\_receive, [214](#)
  - l4\_irq\_receive\_u, [215](#)
  - l4\_irq\_trigger, [216](#)
  - l4\_irq\_trigger\_u, [217](#)
  - l4\_irq\_unmask, [218](#)
  - l4\_irq\_unmask\_u, [219](#)
  - l4\_irq\_wait, [220](#)
  - l4\_irq\_wait\_u, [220](#)
- is\_compatible
  - L4vbus::Device, [1727](#)
- is\_compound
  - L4::lpc::Gen\_fpage, [1045](#)
- is\_irq\_entry
  - L4vcpu::Vcpu, [1787](#)
- is\_nil
  - L4::lpc::Varg, [1106](#)
- is\_of
  - L4::lpc::Varg, [1106](#)
- is\_of\_int
  - L4::lpc::Varg, [1107](#)
- is\_online
  - L4::Scheduler, [1247](#)
- is\_page\_fault\_entry
  - L4vcpu::Vcpu, [1787](#)
- is\_static
  - L4Re::Util::Dataspace\_svr, [1581](#)
- is\_valid
  - L4::Cap\_base, [959](#)
  - L4Re::Rm::Unique\_region< T >, [1552](#)
  - L4virtio::Ptr< T >, [1845](#)
- is\_virtual
  - L4::Kip::Mem\_desc, [1186](#)
- is\_writable
  - L4virtio::Svr::Driver\_mem\_region\_t< DATA >, [1941](#)
- Istream
  - L4::lpc::Istream, [1059](#)
- Item\_bytes
  - L4::lpc::Msg, [687](#)
- Item\_words
  - L4::lpc::Msg, [687](#)
- iter
  - cxx::Bits::Basic\_list< POLICY >, [824](#)
  - cxx::H\_list< T, POLICY >, [853](#)
- Iterator
  - cxx::Avl\_tree< Node, Get\_key, Compare >, [759](#)
- kdebug.h
  - \_\_kdebug\_3\_text, [2829](#)
  - \_\_kdebug\_op, [2830](#)

- \_\_kdebug\_op\_1, [2831](#)
- \_\_kdebug\_text, [2833](#)
- enter\_kdebug, [2834](#)
- l4\_kdebug\_ops\_t, [2829](#)
- outchar, [2835](#)
- outdec, [2836](#)
- outhex12, [2836](#)
- outhex16, [2837](#)
- outhex20, [2837](#)
- outhex32, [2838](#)
- outhex64, [2839](#)
- outhex8, [2839](#)
- outnstring, [2840](#)
- outstring, [2841](#)
- outumword, [2842](#)
- Kept\_ds
  - L4Re::Rm, [1537](#)
- Kernel Debugger, [152](#)
  - l4\_debugger\_add\_image\_info, [153](#)
  - l4\_debugger\_get\_object\_name, [153](#)
  - l4\_debugger\_global\_id, [154](#)
  - l4\_debugger\_kobj\_to\_id, [155](#)
  - l4\_debugger\_query\_log\_name, [156](#)
  - l4\_debugger\_query\_log\_typeid, [157](#)
  - l4\_debugger\_set\_object\_name, [157](#)
  - l4\_debugger\_switch\_log, [158](#)
- Kernel Factory, [40](#)
- Kernel Interface Page, [182](#)
  - l4\_kernel\_info\_version\_offset, [184](#)
  - l4\_kip, [184](#)
  - l4\_kip\_clock, [184](#)
  - l4\_kip\_clock\_lw, [185](#)
  - l4\_kip\_clock\_ns, [186](#)
  - L4\_KIP\_OFFS\_READ\_NS, [183](#)
  - L4\_KIP\_OFFS\_READ\_US, [183](#)
  - l4\_kip\_version, [187](#)
  - l4\_kip\_version\_string, [187](#)
- Kernel Interface Page API, [644](#)
  - l4util\_kip\_for\_each\_feature, [644](#)
  - l4util\_kip\_kernel\_abi\_version, [645](#)
  - l4util\_kip\_kernel\_has\_feature, [645](#)
  - l4util\_kip\_kernel\_is\_ux, [645](#)
- Kernel Objects, [194](#)
- Kernel Tracing, [159](#)
  - fiasco\_tbuf\_log, [160](#)
  - fiasco\_tbuf\_log\_3val, [160](#)
  - fiasco\_tbuf\_log\_binary, [161](#)
- Kernel-provided semaphore, [238](#)
  - l4\_semaphore\_down, [238](#)
  - l4\_semaphore\_up, [239](#)
- kip.h
  - l4\_kip\_for\_each\_feature, [2851](#)
  - l4\_kip\_kernel\_has\_feature, [2851](#)
- kobj\_to\_id
  - L4::Debugger, [972](#)
- kobject\_typeid
  - L4 kernel object type information, [241](#)
- Kumem allocator utility, [488](#)
- Kumem utilities, [530](#)
  - kumem\_alloc, [530](#)
- kumem\_alloc
  - Kumem utilities, [530](#)
- kumem\_alloc.h
  - l4re\_util\_kumem\_alloc, [2459](#)
- L
  - cxx::Bits::Direction, [842](#)
- L4, [667](#)
  - cap\_cast, [672](#)
  - cap\_dynamic\_cast, [673](#)
  - cap\_reinterpret\_cast, [674](#)
  - PROTO\_ANY, [671](#)
  - PROTO\_EMPTY, [671](#)
  - round\_order, [675](#)
  - trunc\_order, [676](#)
- L4 IPC Opcodes, [423](#)
  - L4\_ICU\_OP\_BIND, [424](#)
  - L4\_ICU\_OP\_INFO, [425](#)
  - L4\_ICU\_OP\_MASK, [425](#)
  - L4\_ICU\_OP\_MSI\_INFO, [425](#)
  - L4\_ICU\_OP\_SET\_MODE, [425](#)
  - L4\_ICU\_OP\_UNBIND, [424](#)
  - L4\_ICU\_OP\_UNMASK, [425](#)
  - L4\_icu\_opcode, [424](#)
  - L4\_IPC\_GATE\_BIND\_OP, [425](#)
  - L4\_IPC\_GATE\_GET\_INFO\_OP, [425](#)
  - L4\_ipc\_gate\_ops, [425](#)
  - L4\_PLATFORM\_CTL\_CPU\_ALLOW\_SHUTDOWN\_OP, [426](#)
  - L4\_PLATFORM\_CTL\_CPU\_DISABLE\_OP, [426](#)
  - L4\_PLATFORM\_CTL\_CPU\_ENABLE\_OP, [426](#)
  - L4\_platform\_ctl\_ops, [425](#)
  - L4\_PLATFORM\_CTL\_SET\_TASK\_ASID\_OP, [426](#)
  - L4\_PLATFORM\_CTL\_SYS\_SHUTDOWN\_OP, [426](#)
  - L4\_PLATFORM\_CTL\_SYS\_SUSPEND\_OP, [426](#)
  - L4\_TASK\_ADD\_KU\_MEM\_OP, [426](#)
  - L4\_TASK\_CAP\_INFO\_OP, [426](#)
  - L4\_TASK\_LDT\_SET\_X86\_OP, [426](#)
  - L4\_TASK\_MAP\_OP, [426](#)
  - L4\_TASK\_MAP\_VGICC\_ARM\_OP, [426](#)
  - L4\_task\_ops, [426](#)
  - L4\_TASK\_UNMAP\_OP, [426](#)
  - L4\_THREAD\_AMD64\_GET\_SEGMENT\_INFO\_OP, [426](#)
  - L4\_THREAD\_AMD64\_SET\_SEGMENT\_BASE\_OP, [426](#)
  - L4\_THREAD\_ARM\_TPIDRURO\_OP, [426](#)
  - L4\_THREAD\_CONTROL\_OP, [426](#)
  - L4\_THREAD\_EX\_REGS\_OP, [426](#)
  - L4\_THREAD\_MODIFY\_SENDER\_OP, [426](#)
  - L4\_THREAD\_OPCODE\_MASK, [426](#)
  - L4\_thread\_ops, [426](#)
  - L4\_THREAD\_REGISTER\_DELETE\_IRQ\_OP, [426](#)
  - L4\_THREAD\_STATS\_OP, [426](#)
  - L4\_THREAD\_SWITCH\_OP, [426](#)
  - L4\_THREAD\_VCPU\_CONTROL\_OP, [426](#)



- L4\_THREAD\_VCPU\_RESUME\_OP, [426](#)
- L4\_THREAD\_X86\_GDT\_OP, [426](#)
- L4\_VCON\_GET\_ATTR\_OP, [427](#)
- L4\_vcon\_ops, [427](#)
- L4\_VCON\_READ\_OP, [427](#)
- L4\_VCON\_SET\_ATTR\_OP, [427](#)
- L4\_VCON\_WRITE\_OP, [427](#)
- L4 kernel object type information, [240](#)
  - kobject\_typeid, [241](#)
- L4 Vbus functions, [439](#)
  - l4vbus\_assign\_dma\_domain, [441](#)
  - l4vbus\_dma\_domain\_assign\_flags, [440](#)
  - L4VBUS\_DMAD\_BIND, [441](#)
  - L4VBUS\_DMAD\_KERNEL\_DMA\_SPACE, [441](#)
  - L4VBUS\_DMAD\_L4RE\_DMA\_SPACE, [441](#)
  - L4VBUS\_DMAD\_UNBIND, [441](#)
  - l4vbus\_get\_adr, [442](#)
  - l4vbus\_get\_device, [442](#)
  - l4vbus\_get\_device\_by\_hid, [443](#)
  - l4vbus\_get\_hid, [444](#)
  - l4vbus\_get\_next\_device, [444](#)
  - l4vbus\_get\_resource, [446](#)
  - l4vbus\_is\_compatible, [447](#)
  - l4vbus\_release\_ioport, [447](#)
  - l4vbus\_request\_ioport, [448](#)
  - l4vbus\_vicu\_get\_cap, [449](#)
- L4 VIRTIO Block Device, [428](#)
  - L4virtio\_block\_operations, [428](#)
  - L4VIRTIO\_BLOCK\_S\_IOERR, [429](#)
  - L4VIRTIO\_BLOCK\_S\_OK, [429](#)
  - L4VIRTIO\_BLOCK\_S\_UNSUPP, [429](#)
  - L4virtio\_block\_status, [429](#)
  - L4VIRTIO\_BLOCK\_T\_DISCARD, [429](#)
  - L4VIRTIO\_BLOCK\_T\_FLUSH, [429](#)
  - L4VIRTIO\_BLOCK\_T\_GET\_ID, [429](#)
  - L4VIRTIO\_BLOCK\_T\_IN, [429](#)
  - L4VIRTIO\_BLOCK\_T\_OUT, [429](#)
  - L4VIRTIO\_BLOCK\_T\_WRITE\_ZEROES, [429](#)
- L4 VIRTIO Input Device, [429](#)
- L4 VIRTIO Interface, [427](#)
- L4 VIRTIO Network Device, [430](#)
- L4 VIRTIO Transport Layer, [431](#)
  - L4\_virtio\_cmd, [433](#)
  - L4\_virtio\_irq\_status, [433](#)
  - L4\_virtio\_opcodes, [434](#)
  - L4VIRTIO\_CMD\_CFG\_CHANGED, [433](#)
  - L4VIRTIO\_CMD\_CFG\_QUEUE, [433](#)
  - L4VIRTIO\_CMD\_MASK, [433](#)
  - L4VIRTIO\_CMD\_NONE, [433](#)
  - L4VIRTIO\_CMD\_NOTIFY\_QUEUE, [433](#)
  - L4VIRTIO\_CMD\_SET\_STATUS, [433](#)
  - l4virtio\_config\_queue, [435](#)
  - l4virtio\_config\_queue\_t, [433](#)
  - l4virtio\_config\_queues, [436](#)
  - l4virtio\_device\_config, [436](#)
  - l4virtio\_device\_config\_ds, [436](#)
  - L4virtio\_device\_ids, [434](#)
  - l4virtio\_device\_notification\_irq, [437](#)
  - L4virtio\_device\_status, [435](#)
  - L4virtio\_feature\_bits, [435](#)
  - L4VIRTIO\_FEATURE\_CMD\_CONFIG, [435](#)
  - L4VIRTIO\_FEATURE\_VERSION\_1, [435](#)
  - L4VIRTIO\_ID\_9P, [434](#)
  - L4VIRTIO\_ID\_BALLOON, [434](#)
  - L4VIRTIO\_ID\_BLOCK, [434](#)
  - L4VIRTIO\_ID\_CAIF, [434](#)
  - L4VIRTIO\_ID\_CONSOLE, [434](#)
  - L4VIRTIO\_ID\_CRYPTOD, [434](#)
  - L4VIRTIO\_ID\_FS, [434](#)
  - L4VIRTIO\_ID\_GPU, [434](#)
  - L4VIRTIO\_ID\_INPUT, [434](#)
  - L4VIRTIO\_ID\_NET, [434](#)
  - L4VIRTIO\_ID\_RNG, [434](#)
  - L4VIRTIO\_ID\_RPMSG, [434](#)
  - L4VIRTIO\_ID\_RPROC\_SERIAL, [434](#)
  - L4VIRTIO\_ID\_SCMI, [434](#)
  - L4VIRTIO\_ID\_SCSI, [434](#)
  - L4VIRTIO\_ID\_SOCKET, [434](#)
  - L4VIRTIO\_ID\_VSOCKET, [434](#)
  - L4VIRTIO\_IRQ\_STATUS\_CONFIG, [434](#)
  - L4VIRTIO\_IRQ\_STATUS\_VRING, [434](#)
  - L4VIRTIO\_OP\_CONFIG\_QUEUE, [434](#)
  - L4VIRTIO\_OP\_DEVICE\_CONFIG, [434](#)
  - L4VIRTIO\_OP\_GET\_DEVICE\_IRQ, [434](#)
  - L4VIRTIO\_OP\_REGISTER\_DS, [434](#)
  - L4VIRTIO\_OP\_SET\_STATUS, [434](#)
  - l4virtio\_register\_ds, [437](#)
  - l4virtio\_set\_status, [438](#)
  - L4VIRTIO\_STATUS\_ACKNOWLEDGE, [435](#)
  - L4VIRTIO\_STATUS\_DEVICE\_NEEDS\_RESET, [435](#)
  - L4VIRTIO\_STATUS\_DRIVER, [435](#)
  - L4VIRTIO\_STATUS\_DRIVER\_OK, [435](#)
  - L4VIRTIO\_STATUS\_FAILED, [435](#)
  - L4VIRTIO\_STATUS\_FEATURES\_OK, [435](#)
- l4/cxx/alloc.h, [2149](#)
- l4/cxx/arith, [2150](#)
- l4/cxx/atomic.h, [2152](#)
- l4/cxx/avl\_map, [2160](#), [2161](#)
- l4/cxx/avl\_set, [2163](#), [2165](#)
- l4/cxx/avl\_tree, [2168](#), [2170](#)
- l4/cxx/basic\_ostream, [2174](#), [2175](#)
- l4/cxx/basic\_vector.h, [2178](#), [2179](#)
- l4/cxx/bitfield, [2179](#)
- l4/cxx/bitmap, [2182](#)
- l4/cxx/bits/bst.h, [2185](#), [2187](#)
- l4/cxx/bits/bst\_base.h, [2189](#), [2191](#)
- l4/cxx/bits/bst\_iter.h, [2192](#), [2194](#)
- l4/cxx/bits/list\_basics.h, [2195](#)
- l4/cxx/bits/smart\_ptr\_list.h, [2197](#), [2200](#)
- l4/cxx/bits/type\_traits.h, [2201](#)
- l4/cxx/dlist, [2204](#)
- l4/cxx/exceptions, [2207](#), [2209](#)
- l4/cxx/hlist, [2211](#)
- l4/cxx/iostream, [2214](#), [2215](#)
- l4/cxx/ipc\_helper, [2215](#), [2217](#)



l4/cxx/ipc\_server, 2217, 2219  
l4/cxx/ipc\_stream, 2223, 2233  
l4/cxx/ipc\_timeout\_queue, 2241  
l4/cxx/l4iostream, 2243, 2244  
l4/cxx/l4types.h, 2245, 2246  
l4/cxx/list, 2246  
l4/cxx/list\_alloc, 2250  
l4/cxx/main\_thread, 2256, 2257  
l4/cxx/minmax, 2258  
l4/cxx/observer, 2259  
l4/cxx/pair, 2259, 2261  
l4/cxx/ref\_ptr, 2262  
l4/cxx/ref\_ptr\_list, 2265, 2267  
l4/cxx/slab\_alloc, 2267  
l4/cxx/slist, 2271  
l4/cxx/static\_container, 2274  
l4/cxx/static\_vector, 2274  
l4/cxx/std\_alloc, 2275  
l4/cxx/std\_exc\_io, 2276, 2277  
l4/cxx/std\_ops, 2277  
l4/cxx/string, 2278  
l4/cxx/string.h, 2281, 2282  
l4/cxx/thread, 2283, 2284  
l4/cxx/type\_list, 2288  
l4/cxx/type\_traits, 2288  
l4/cxx/unique\_ptr, 2293  
l4/cxx/unique\_ptr\_list, 2295, 2297  
l4/cxx/utils, 2297  
l4/cxx/weak\_ref, 2297  
l4/irq/irq.h, 2818, 2820  
l4/l4re\_vfs/backend, 2299  
l4/l4re\_vfs/impl/default\_ops\_impl.h, 2302  
l4/l4re\_vfs/impl/fd\_store.h, 2303  
l4/l4re\_vfs/impl/fd\_store\_impl.h, 2304  
l4/l4re\_vfs/impl/ns\_fs.h, 2304  
l4/l4re\_vfs/impl/ns\_fs\_impl.h, 2305  
l4/l4re\_vfs/impl/ro\_file.h, 2310  
l4/l4re\_vfs/impl/ro\_file\_impl.h, 2311  
l4/l4re\_vfs/impl/vcon\_stream.h, 2312  
l4/l4re\_vfs/impl/vcon\_stream\_impl.h, 2313  
l4/l4re\_vfs/impl/vfs\_impl.h, 2315  
l4/l4re\_vfs/vfs.h, 2328  
l4/l4virtio/client/l4virtio, 2349  
l4/l4virtio/client/virtio-block, 2337  
l4/l4virtio/client/virtio-net, 2346  
l4/l4virtio/l4virtio, 2351  
l4/l4virtio/server/l4virtio, 2353  
l4/l4virtio/server/virtio, 2364  
l4/l4virtio/server/virtio-block, 2340  
l4/l4virtio/server/virtio-console, 2368  
l4/l4virtio/server/virtio-console-device, 2373  
l4/l4virtio/server/virtio-scmi-device, 2377  
l4/l4virtio/virtio.h, 2386  
l4/l4virtio/virtio\_block.h, 2389  
l4/l4virtio/virtio\_input.h, 2390  
l4/l4virtio/virtio\_net.h, 2391  
l4/l4virtio/virtqueue, 2392  
l4/libblock-device/block\_device\_mgr.h, 2396  
l4/libblock-device/debug.h, 2437  
l4/libblock-device/device.h, 2401  
l4/libblock-device/errand.h, 2402  
l4/libblock-device/gpt.h, 2404  
l4/libblock-device/inout\_memory.h, 2405  
l4/libblock-device/part\_device.h, 2406  
l4/libblock-device/partition.h, 2409  
l4/libblock-device/request.h, 2412  
l4/libblock-device/scheduler.h, 2880  
l4/libblock-device/types.h, 2143  
l4/libblock-device/virtio\_client.h, 2412  
l4/libedid/edid.h, 2420, 2421  
l4/libgfxbitmap/bitmap.h, 2422, 2426  
l4/libgfxbitmap/font.h, 2427, 2432  
l4/libgfxbitmap/support, 2433, 2434  
l4/re/c/dataspace.h, 2434, 2436  
l4/re/c/debug.h, 2437, 2438  
l4/re/c/dma\_space.h, 2439, 2441  
l4/re/c/event.h, 2516, 2518  
l4/re/c/event\_buffer.h, 2442  
l4/re/c/inhibitor.h, 2442, 2445  
l4/re/c/log.h, 2445, 2446  
l4/re/c/mem\_alloc.h, 2447, 2449  
l4/re/c/namespace.h, 2449, 2451  
l4/re/c/parent.h, 2451, 2453  
l4/re/c/rm.h, 2453, 2455  
l4/re/c/util/cap\_alloc.h, 2457, 2458  
l4/re/c/util/kumem\_alloc.h, 2458, 2460  
l4/re/c/util/video/goos\_fb.h, 2460, 2461  
l4/re/c/video/colors.h, 2462, 2464  
l4/re/c/video/goos.h, 2464, 2466  
l4/re/c/video/view.h, 2467, 2469  
l4/re/cap\_alloc, 2470, 2472  
l4/re/console, 2477  
l4/re/consts, 2477, 2479  
l4/re/consts.h, 2482, 2483  
l4/re/dataspace, 2489, 2490  
l4/re/dataspace-sys.h, 2492  
l4/re/debug, 2493, 2494  
l4/re/dma\_space, 2496, 2498  
l4/re/elf\_aux.h, 2499, 2500  
l4/re/env, 2501, 2502  
l4/re/env.h, 2504, 2506  
l4/re/error\_helper, 2507, 2509  
l4/re/event, 2510  
l4/re/event-sys.h, 2515  
l4/re/event.h, 2518, 2519  
l4/re/event\_enums.h, 2520  
l4/re/impl/dataspace\_impl.h, 2527, 2528  
l4/re/impl/mem\_alloc\_impl.h, 2529, 2530  
l4/re/impl/namespace\_impl.h, 2531, 2532  
l4/re/impl/rm\_impl.h, 2533, 2534  
l4/re/inhibitor, 2536  
l4/re/inhibitor-sys.h, 2536  
l4/re/l4aux.h, 2536, 2538  
l4/re/log, 2538, 2540  
l4/re/log-sys.h, 2540, 2541  
l4/re/mem\_alloc, 2541, 2543

l4/re/mem\_alloc-sys.h, 2543, 2544  
l4/re/mmio\_space, 2545, 2546  
l4/re/namespace, 2546, 2548  
l4/re/namespace-sys.h, 2549, 2550  
l4/re/parent, 2550, 2552  
l4/re/parent-sys.h, 2552, 2553  
l4/re/protocols.h, 2553, 2554  
l4/re/random, 2555, 2556  
l4/re/rm, 2557, 2558  
l4/re/rm-sys.h, 2562, 2563  
l4/re/shared\_cap, 2563, 2565  
l4/re/unique\_cap, 2569, 2570  
l4/re/util/bitmap\_cap\_alloc, 2573, 2575  
l4/re/util/br\_manager, 2576  
l4/re/util/cap, 2578, 2580  
l4/re/util/cap\_alloc, 2474, 2476  
l4/re/util/cap\_alloc\_impl.h, 2580, 2581  
l4/re/util/counting\_cap\_alloc, 2582, 2583  
l4/re/util/dataspace\_svr, 2587  
l4/re/util/debug, 2494  
l4/re/util/env\_ns, 2590  
l4/re/util/event, 2513, 2514  
l4/re/util/event\_buffer, 2591  
l4/re/util/event\_svr, 2592  
l4/re/util/icu\_svr, 2594  
l4/re/util/item\_alloc, 2596, 2597  
l4/re/util/kumem\_alloc, 2599, 2600  
l4/re/util/meta, 2600  
l4/re/util/name\_space\_svr, 2602  
l4/re/util/object\_registry, 2608  
l4/re/util/poll\_timeout\_kipclock, 2611  
l4/re/util/region\_mapping, 2611, 2612  
l4/re/util/region\_mapping\_svr\_2, 2618  
l4/re/util/shared\_cap, 2566, 2568  
l4/re/util/unique\_cap, 2571, 2573  
l4/re/util/vcon\_svr, 2621  
l4/re/util/video/goos\_fb, 2622  
l4/re/util/video/goos\_svr, 2624  
l4/re/video/colors, 2626  
l4/re/video/goos, 2627  
l4/re/video/goos-sys.h, 2630, 2631  
l4/re/video/view, 2632  
l4/shmc/ringbuf.h, 2632, 2639  
l4/shmc/shmc.h, 2640, 2643  
l4/sigma0/sigma0.h, 2645, 2647  
l4/sys/\_\_kernel\_object\_impl.h, 2649  
l4/sys/\_\_kip-32bit.h, 2650  
l4/sys/\_\_kip-64bit.h, 2651  
l4/sys/\_\_ktrace-impl.h, 2652, 2654  
l4/sys/\_\_l4\_fpage.h, 2655  
l4/sys/\_\_platform\_control-arm.h, 2659  
l4/sys/\_\_task-arm.h, 2659  
l4/sys/\_\_timeout.h, 2660  
l4/sys/\_\_typeinfo.h, 2662, 2665  
l4/sys/\_\_vcpu-arm.h, 2675  
l4/sys/\_\_vm-arm.h, 2676, 2678  
l4/sys/\_\_vm-svm.h, 2678  
l4/sys/\_\_vm-vmx.h, 2680  
l4/sys/arm\_smccc, 2686, 2688  
l4/sys/arm\_smccc.h, 2688, 2690  
l4/sys/assert.h, 2691, 2693  
l4/sys/cache.h, 2701, 2702  
l4/sys/capability, 2704, 2706  
l4/sys/compiler.h, 2708, 2710  
l4/sys/consts.h, 2484, 2486  
l4/sys/cxx/capability.h, 2712  
l4/sys/cxx/consts, 2479  
l4/sys/cxx/ipc\_array, 2714  
l4/sys/cxx/ipc\_basics, 2718  
l4/sys/cxx/ipc\_client, 2722, 2724  
l4/sys/cxx/ipc\_epiface, 2724  
l4/sys/cxx/ipc\_iface, 2728, 2734  
l4/sys/cxx/ipc\_legacy, 2739  
l4/sys/cxx/ipc\_ret\_array, 2740  
l4/sys/cxx/ipc\_server, 2220  
l4/sys/cxx/ipc\_server\_loop, 2741  
l4/sys/cxx/ipc\_string, 2744  
l4/sys/cxx/ipc\_types, 2745, 2747  
l4/sys/cxx/ipc\_varg, 2754  
l4/sys/cxx/smart\_capability\_1x, 2760, 2761  
l4/sys/cxx/types, 2763, 2764  
l4/sys/debugger, 2767, 2768  
l4/sys/debugger.h, 2769, 2770  
l4/sys/err.h, 2773, 2775  
l4/sys/exception, 2775, 2777  
l4/sys/factory, 2777, 2779  
l4/sys/factory.h, 2781, 2783  
l4/sys/icu, 2787, 2788  
l4/sys/icu.h, 2789, 2792  
l4/sys/iommu, 2795  
l4/sys/ipc.h, 2798, 2800  
l4/sys/ipc\_gate, 2805, 2806  
l4/sys/ipc\_gate.h, 2807, 2810  
l4/sys/irq, 2811, 2812  
l4/sys/irq.h, 2821, 2823  
l4/sys/kdebug.h, 2827, 2842  
l4/sys/kernel\_object.h, 2845, 2846  
l4/sys/kip, 2847, 2848  
l4/sys/kip.h, 2850, 2852  
l4/sys/kobject, 2855  
l4/sys/ktrace.h, 2856, 2857  
l4/sys/l4int.h, 2859, 2860  
l4/sys/memdesc.h, 2862, 2863  
l4/sys/meta, 2600, 2602  
l4/sys/pager, 2865, 2867  
l4/sys/platform\_control, 2867, 2869  
l4/sys/platform\_control.h, 2869, 2871  
l4/sys/rcv\_endpoint, 2873, 2875  
l4/sys/rcv\_endpoint.h, 2875, 2877  
l4/sys/scheduler, 2878, 2879  
l4/sys/scheduler.h, 2883, 2885  
l4/sys/semaphore, 2888, 2889  
l4/sys/semaphore.h, 2890, 2891  
l4/sys/smart\_capability, 2892, 2894  
l4/sys/task, 2896, 2898  
l4/sys/task.h, 2899, 2901

- [l4/sys/thread](#), 2285, 2286
- [l4/sys/thread.h](#), 2905, 2907
- [l4/sys/typeinfo\\_svr](#), 2916, 2918
- [l4/sys/types.h](#), 2144, 2146
- [l4/sys/utcb.h](#), 2924, 2926
- [l4/sys/vcon](#), 2931, 2933
- [l4/sys/vcon.h](#), 2933, 2936
- [l4/sys/vcpu](#), 2939, 2941
- [l4/sys/vcpu\\_context](#), 2946, 2947
- [l4/sys/vcpu\\_context.h](#), 2947
- [l4/sys/vhw.h](#), 2947, 2948
- [l4/sys/vm](#), 2949, 2951
- [l4/util/assert.h](#), 2694, 2695
- [l4/util/atomic.h](#), 2153, 2156
- [l4/util/backtrace.h](#), 2951, 2952
- [l4/util/base64.h](#), 2953, 2954
- [l4/util/bitops.h](#), 2954, 2957
- [l4/util/elf.h](#), 2960, 2966
- [l4/util/getopt.h](#), 2976, 2977
- [l4/util/keymap.h](#), 2978, 2979
- [l4/util/kip.h](#), 2854, 2855
- [l4/util/kprintf.h](#), 2979, 2980
- [l4/util/l4\\_macros.h](#), 2132
- [l4/util/l4mod.h](#), 2980, 2981
- [l4/util/list\\_alloc.h](#), 2982, 2984
- [l4/util/lock.h](#), 2984, 2985
- [l4/util/mb\\_info.h](#), 2986, 2990
- [l4/util/parse\\_cmd.h](#), 2994, 2995
- [l4/util/rand.h](#), 2995, 2997
- [l4/util/splitlog2.h](#), 2997, 2998
- [l4/util/thread.h](#), 2914, 2916
- [l4/util/util.h](#), 2999
- [l4/vbus/vbus](#), 2999
- [l4/vbus/vbus.h](#), 3001, 3004
- [l4/vbus/vbus\\_generic](#), 3005
- [l4/vbus/vbus\\_gpio](#), 3005
- [l4/vbus/vbus\\_gpio-ops.h](#), 3007
- [l4/vbus/vbus\\_gpio.h](#), 3007
- [l4/vbus/vbus\\_i2c.h](#), 3008
- [l4/vbus/vbus\\_inhibitor.h](#), 3008
- [l4/vbus/vbus\\_interfaces.h](#), 3008, 3011
- [l4/vbus/vbus\\_mcspi.h](#), 3012
- [l4/vbus/vbus\\_pci](#), 3012
- [l4/vbus/vbus\\_pci-ops.h](#), 3013
- [l4/vbus/vbus\\_pci.h](#), 3014
- [l4/vbus/vbus\\_pm-ops.h](#), 3014
- [l4/vbus/vbus\\_pm.h](#), 3015
- [l4/vbus/vbus\\_types.h](#), 3015, 3017
- [l4/vbus/vdevice-ops.h](#), 3018
- [l4/vcpu/vcpu](#), 3019, 3020
- [l4/vcpu/vcpu.h](#), 2942, 2944
- [L4::Alloc\\_list](#), 936
- [L4::Arm\\_smccc](#), 937
  - [call](#), 937
- [L4::Base\\_exception](#), 938
- [L4::Basic\\_registry](#), 941
  - [dispatch](#), 942
  - [find](#), 943
- [L4::Bounds\\_error](#), 944
- [L4::Cap< T >](#), 946
  - [Cap](#), 950, 951
  - [check\\_castable\\_from](#), 951
  - [check\\_convertible\\_from](#), 951
  - [copy](#), 952
  - [move](#), 952
- [L4::Cap\\_base](#), 952
  - [cap](#), 956
  - [Cap\\_base](#), 955
  - [Cap\\_type](#), 955
  - [copy](#), 957
  - [fpage](#), 958
  - [Invalid](#), 955
  - [is\\_valid](#), 959
  - [move](#), 960
  - [No\\_init](#), 955
  - [No\\_init\\_type](#), 955
  - [snd\\_base](#), 961
  - [validate](#), 961, 962
- [L4::Com\\_error](#), 963
  - [Com\\_error](#), 966
- [L4::Debugger](#), 967
  - [add\\_image\\_info](#), 970
  - [get\\_object\\_name](#), 971
  - [global\\_id](#), 971
  - [kobj\\_to\\_id](#), 972
  - [query\\_log\\_name](#), 973
  - [query\\_log\\_typeid](#), 974
  - [set\\_object\\_name](#), 974
  - [switch\\_log](#), 975
- [L4::Element\\_already\\_exists](#), 976
- [L4::Element\\_not\\_found](#), 979
- [L4::Epiface](#), 982
  - [dispatch](#), 984
  - [get\\_buffer\\_demand](#), 985
  - [obj\\_cap](#), 985
  - [server\\_iface](#), 985
  - [set\\_server](#), 986
- [L4::Epiface\\_t< Derived, IFACE, BASE, bool >](#), 987
  - [dispatch](#), 990
- [L4::Epiface\\_t0< RPC\\_IFACE, BASE >](#), 990
  - [obj\\_cap](#), 992
- [L4::Exception](#), 993
  - [exception](#), 993
- [L4::Exception\\_tracer](#), 994
- [L4::Factory](#), 996
  - [create](#), 999, 1000
  - [create\\_factory](#), 1001
  - [create\\_gate](#), 1002
  - [create\\_task](#), 1003
- [L4::Factory::Lstr](#), 1005
  - [Lstr](#), 1005
- [L4::Factory::Nil](#), 1006
- [L4::Factory::S](#), 1007
  - [operator l4\\_msgtag\\_t](#), 1009
  - [put](#), 1009, 1010
  - [S](#), 1008

- L4::lcu, 1011
  - bind, 1014
  - info, 1015
  - mask, 1016
  - msi\_info, 1017
  - set\_mode, 1017
  - unbind, 1018
- L4::lcu::Info, 1019
- L4::Invalid\_capability, 1021
  - cap, 1023
  - Invalid\_capability, 1023
- L4::lo\_pager, 1024
  - io\_page\_fault, 1025
- L4::lommu, 1026
  - bind, 1028
  - unbind, 1028
- L4::LOModifier, 1028
- L4::lpc, 677
  - buf\_cp\_in, 679
  - buf\_cp\_out, 679
  - buf\_in, 680
  - make\_cap, 680
  - make\_cap\_full, 681
  - make\_cap\_rw, 681
  - make\_cap\_rws, 683
  - msg\_ptr, 684
  - read, 684
  - str\_cp\_in, 684
- L4::lpc::Array< ELEM\_TYPE, LEN\_TYPE >, 1029
- L4::lpc::Array\_in\_buf< ELEM\_TYPE, LEN\_TYPE, MAX >, 1032
- L4::lpc::Array\_ref< ELEM\_TYPE, LEN\_TYPE >, 1033
- L4::lpc::As\_value< T >, 1034
- L4::lpc::Call, 1035
- L4::lpc::Call\_t< RIGHTS >, 1037
- L4::lpc::Call\_zero\_send\_timeout, 1038
- L4::lpc::Cap< T >, 1039
  - Cap, 1041
  - Cap\_mask, 1041
  - from\_ci, 1042
  - Rights\_mask, 1041
- L4::lpc::Gen\_fpage, 1042
  - cap\_received, 1044
  - id\_received, 1044
  - is\_compound, 1045
  - local\_id\_received, 1045
- L4::lpc::In\_out< T >, 1045
- L4::lpc::lostream, 1046
  - call, 1051
  - get, 1051, 1052
  - lostream, 1050
  - put, 1053
  - reply\_and\_wait, 1053, 1054
  - reset, 1055
- L4::lpc::lstream, 1056
  - get, 1060, 1061
  - lstream, 1059
  - receive, 1062
  - reset, 1063
  - skip, 1063
  - tag, 1064
  - wait, 1065, 1066
- L4::lpc::Msg, 685
  - align\_to, 687
  - Br\_bytes, 687
  - check\_size, 688, 689
  - Item\_bytes, 687
  - Item\_words, 687
  - Mr\_bytes, 687
  - Mr\_words, 687
  - msg\_add, 689
  - msg\_get, 690
  - Word\_bytes, 687
- L4::lpc::Msg::Cnt\_val\_ops< MTYPE, DIR, CLASS >, 1067
- L4::lpc::Msg::Cls\_buffer, 1068
- L4::lpc::Msg::Cls\_data, 1069
- L4::lpc::Msg::Cls\_item, 1070
- L4::lpc::Msg::Dir\_in, 1071
- L4::lpc::Msg::Dir\_out, 1072
- L4::lpc::Msg::Do\_in\_data, 1073
- L4::lpc::Msg::Do\_in\_items, 1074
- L4::lpc::Msg::Do\_out\_data, 1075
- L4::lpc::Msg::Do\_out\_items, 1076
- L4::lpc::Msg::Do\_rcv\_buffers, 1077
- L4::lpc::Msg::Elem< Array< A, LEN > >, 1080
- L4::lpc::Msg::Elem< Array< A, LEN > & >, 1079
- L4::lpc::Msg::Elem< Array\_ref< A, LEN > & >, 1081
- L4::lpc::Msg::Is\_valid\_rpc\_type< T >, 1082
- L4::lpc::Msg::Svr\_arg\_pack< IPC\_TYPE >, 1084
- L4::lpc::Msg::Svr\_val\_ops< MTYPE, DIR, CLASS >, 1084
- L4::lpc::Msg\_ptr< T >, 1085
  - Msg\_ptr, 1086
- L4::lpc::Opt< T >, 1086
- L4::lpc::Ostream, 1088
  - put, 1091, 1092
  - send, 1092
  - tag, 1093
- L4::lpc::Out< T >, 1094
- L4::lpc::Rcv\_fpage, 1095
- L4::lpc::Ret\_array< T >, 1097
- L4::lpc::Send\_only, 1098
- L4::lpc::Small\_buf, 1098
  - Small\_buf, 1099
- L4::lpc::Snd\_fpage, 1100
- L4::lpc::Str\_cp\_in< T >, 1102
  - Str\_cp\_in, 1103
- L4::lpc::Varg, 1103
  - data, 1105
  - get\_value, 1105
  - is\_nil, 1106
  - is\_of, 1106
  - is\_of\_int, 1107
  - length, 1107
  - tag, 1107

- type, 1108
- value, 1108
- L4::lpc::Varg\_list< MAX >, 1109
- L4::lpc::Varg\_list\_ref, 1111
  - Varg\_list\_ref, 1113
- L4::lpc::Varg\_list\_ref::iterator, 1113
- L4::lpc\_gate, 1114
  - get\_infos, 1119
- L4::lpc\_svr, 690
- L4::lpc\_svr::Br\_manager\_no\_buffers, 1119
  - alloc\_buffer\_demand, 1122
- L4::lpc\_svr::Compound\_reply, 1123
- L4::lpc\_svr::Default\_loop\_hooks, 1125
- L4::lpc\_svr::Default\_setup\_wait, 1128
- L4::lpc\_svr::Default\_timeout, 1129
- L4::lpc\_svr::Direct\_dispatch< R >, 1131
- L4::lpc\_svr::Direct\_dispatch< R \* >, 1133
- L4::lpc\_svr::Exc\_dispatch< R, Exc >, 1134
- L4::lpc\_svr::Ignore\_errors, 1136
- L4::lpc\_svr::Server\_iface, 1137
  - add\_timeout, 1139
  - alloc\_buffer\_demand, 1139
  - get\_rcv\_cap, 1140
  - rcv\_cap, 1140, 1141
  - realloc\_rcv\_cap, 1142
  - remove\_timeout, 1142
- L4::lpc\_svr::Timeout, 1143
  - expired, 1145
  - timeout, 1145
- L4::lpc\_svr::Timeout\_queue, 1146
  - add, 1147
  - handle\_expired\_timeouts, 1147
  - next\_timeout, 1148
  - remove, 1149
  - timeout\_expired, 1150
- L4::lpc\_svr::Timeout\_queue\_hooks< HOOKS, BR\_MAN >, 1151
  - add\_timeout, 1155
  - remove\_timeout, 1156
- L4::lrq, 1157
  - detach, 1161
  - receive, 1162
  - unmask, 1163
  - wait, 1164
- L4::lrq\_eoi, 1165
  - unmask, 1166
- L4::lrq\_handler\_object, 1167
- L4::lrq\_mux, 1170
  - chain, 1173
- L4::lrqep\_t< Derived, BASE, bool >, 1174
  - dispatch, 1178
  - obj\_cap, 1178
- L4::Kip::Mem\_desc, 1179
  - all, 1182
  - Arch, 1181
  - Arch\_acpi\_nvs, 1181
  - Arch\_acpi\_tables, 1181
  - Arch\_sub\_type\_common, 1180
  - Bootloader, 1181
  - Conventional, 1181
  - count, 1183, 1184
  - Dedicated, 1181
  - end, 1184
  - first, 1185
  - Info, 1181
  - Info\_acpi\_rsd, 1181
  - Info\_sub\_type, 1181
  - is\_virtual, 1186
  - Mem\_desc, 1181
  - Mem\_type, 1181
  - Reserved, 1181
  - set, 1186
  - Shared, 1181
  - size, 1187
  - start, 1187
  - sub\_type, 1188
  - type, 1188
  - Undefined, 1181
- L4::Kobject, 1189
  - cap, 1190
  - dec\_refcnt, 1191
- L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_DEMAND >, 1192
  - \_\_iface, 1194
  - \_\_iface\_list, 1194
  - \_\_check\_protocols\_\_, 1194
  - c, 1194
  - Class, 1194
- L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S\_DEMAND >, 1195
  - \_\_iface, 1198
  - \_\_iface\_list, 1198
  - \_\_check\_protocols\_\_, 1198
  - c, 1198
  - Class, 1198
- L4::Kobject\_demand< T >, 1199
- L4::Kobject\_t< Derived, Base, PROTO, S\_DEMAND >, 1200
- L4::Kobject\_typeid< T >, 1201
  - Demand, 1202
  - demand, 1203
  - id, 1203
  - proto\_dispatch, 1203
- L4::Kobject\_typeid< void >, 1204
- L4::Kobject\_x< Derived, ARGS >, 1205
- L4::Meta, 1206
  - interface, 1209
  - num\_interfaces, 1210
  - supports, 1210
- L4::Out\_of\_memory, 1211
- L4::Pager, 1214
  - page\_fault, 1216
- L4::Platform\_control, 1218
  - cpu\_allow\_shutdown, 1220
  - cpu\_disable, 1221
  - cpu\_enable, 1221

- system\_shutdown, 1221
- system\_suspend, 1222
- L4::Poll\_timeout\_counter, 1222
  - Poll\_timeout\_counter, 1223
  - set, 1224
  - timed\_out, 1224
- L4::Poll\_timeout\_kipclock, 1225
  - Poll\_timeout\_kipclock, 1226
  - set, 1226
  - test, 1227
  - timed\_out, 1228
- L4::Proto\_t< P >, 1228
- L4::Rcv\_endpoint, 1229
  - bind\_thread, 1232
- L4::Registry\_iface, 1233
  - register\_irq\_obj, 1235
  - register\_obj, 1235, 1236
  - unregister\_obj, 1237
- L4::Runtime\_error, 1237
  - err\_no, 1241
  - extra\_str, 1241
  - Runtime\_error, 1240
- L4::Scheduler, 1241
  - idle\_time, 1245
  - info, 1246
  - is\_online, 1247
  - run\_thread, 1247
- L4::Semaphore, 1248
  - down, 1252
  - up, 1253
- L4::Server< LOOP\_HOOKS >, 1254
  - internal\_loop, 1257
  - loop, 1258
  - Server, 1257
- L4::Server\_object, 1259
  - dispatch, 1261, 1262
- L4::Server\_object\_t< IFACE, BASE >, 1263
  - dispatch\_meta\_request, 1267
  - get\_buffer\_demand, 1267
  - proto\_dispatch, 1267
- L4::Server\_object\_x< Derived, IFACE, BASE >, 1269
- L4::Smart\_cap< T, SMART >, 1272
  - Smart\_cap, 1276
- L4::String, 1276
- L4::Task, 1277
  - add\_ku\_mem, 1281
  - cap\_equal, 1282
  - cap\_valid, 1283
  - delete\_obj, 1284
  - map, 1285
  - release\_cap, 1286
  - unmap, 1287
  - unmap\_batch, 1288
- L4::Thread, 1289
  - control, 1292
  - ex\_regs, 1293, 1294
  - modify\_senders, 1295
  - register\_del\_irq, 1296
  - stats\_time, 1297
  - switch\_to, 1298
  - vcpu\_control, 1298
  - vcpu\_control\_ext, 1299
  - vcpu\_resume\_commit, 1300
  - vcpu\_resume\_start, 1301
- L4::Thread::Attr, 1302
  - alien, 1304
  - Attr, 1303
  - bind, 1304
  - exc\_handler, 1305
  - pager, 1306
  - ux\_host\_syscall, 1307
- L4::Thread::Modify\_senders, 1307
  - add, 1308
- L4::Triggerable, 1309
  - trigger, 1312
- L4::Type\_info, 1313
- L4::Type\_info::Demand, 1314
  - Demand, 1317
  - no\_demand, 1317
- L4::Type\_info::Demand\_t< CAPS, FLAGS, MEM, PORTS >, 1318
  - Caps, 1320
  - Flags, 1320
  - Mem, 1320
  - Ports, 1320
- L4::Type\_info::Demand\_union\_t< D1, D2 >, 1320
- L4::Typeid, 691
- L4::Typeid::Detail::\_Rpc< OPCODE, O, Default\_op< R > >::Rpc< Y >, 1325
- L4::Typeid::Detail::\_Rpc< OPCODE, O, R, X... >, 1325
- L4::Typeid::Detail::\_Rpc< OPCODE, O, R, X... >::Rpc< Y >, 1327
- L4::Typeid::Detail::\_Rpc< OPCODE, O, X >, 1323
- L4::Typeid::Detail::Rpc\_end, 1327
- L4::Typeid::P\_dispatch< LIST >, 1328
- L4::Typeid::Raw\_ipc< CLASS >, 1329
- L4::Typeid::Rpc\_nocode< OPERATION >, 1330
- L4::Typeid::Rpc< RPCS >, 1332
- L4::Typeid::Rpc\_code< OPCODE\_TYPE >, 1334
- L4::Typeid::Rpc\_code< OPCODE\_TYPE >::F< RPCS >, 1335
- L4::Typeid::Rpc\_sys< ARG >, 1337
- L4::Types, 692
- L4::Types::Bool< V >, 1339
- L4::Types::False, 1340
- L4::Types::Flags< BITS\_ENUM, UNDERLYING >, 1342
  - clear, 1345
  - Flags, 1344
  - from\_raw, 1345
  - None, 1344
  - None\_type, 1344
- L4::Types::Flags\_ops\_t< DT >, 1346
- L4::Types::Flags\_t< DT, T >, 1348
- L4::Types::Int\_for\_size< SIZE, bool >, 1350

- L4::Types::Int\_for\_type< T >, [1351](#)
- L4::Types::Same< A, B >, [1352](#)
- L4::Types::True, [1354](#)
- L4::Uart, [1356](#)
  - change\_mode, [1358](#)
  - char\_avail, [1358](#)
  - enable\_rx\_irq, [1358](#)
  - generic\_write, [1359](#)
  - get\_char, [1359](#)
  - mode, [1360](#)
  - rate, [1360](#)
  - shutdown, [1360](#)
  - startup, [1360](#)
  - write, [1361](#)
- L4::Uart\_apb, [1361](#)
  - change\_mode, [1364](#)
  - char\_avail, [1365](#)
  - enable\_rx\_irq, [1365](#)
  - get\_char, [1365](#)
  - shutdown, [1366](#)
  - startup, [1366](#)
  - write, [1366](#)
- L4::Unknown\_error, [1367](#)
- L4::Vcon, [1369](#)
  - get\_attr, [1373](#)
  - read, [1374](#)
  - read\_with\_flags, [1375](#)
  - send, [1375](#)
  - set\_attr, [1376](#)
  - write, [1377](#)
- L4::Vm, [1378](#)
  - vgicc\_map, [1382](#)
- l4\_addr\_consts\_t
  - Memory related, [335](#)
- l4\_align\_stack\_for\_direct\_fncall
  - Basic Macros, [137](#)
- L4\_AMD64\_SEGMENT\_FS
  - segment.h, [2071](#)
- L4\_AMD64\_SEGMENT\_GS
  - segment.h, [2071](#)
- l4\_arm\_smccc\_call
  - arm\_smccc.h, [2689](#)
- l4\_assert
  - assert.h, [2693](#)
- L4\_BASE\_ARM\_SMCCC\_CAP
  - Capabilities, [144](#)
- L4\_BASE\_CAPS\_LAST
  - Capabilities, [144](#)
- L4\_BASE\_DEBUGGER\_CAP
  - Capabilities, [144](#)
- L4\_BASE\_FACTORY\_CAP
  - Capabilities, [144](#)
- L4\_BASE\_ICU\_CAP
  - Capabilities, [144](#)
- L4\_BASE\_IOMMU\_CAP
  - Capabilities, [144](#)
- L4\_BASE\_LOG\_CAP
  - Capabilities, [144](#)
- L4\_BASE\_PAGER\_CAP
  - Capabilities, [144](#)
- L4\_BASE\_SCHEDULER\_CAP
  - Capabilities, [144](#)
- L4\_BASE\_TASK\_CAP
  - Capabilities, [144](#)
- L4\_BASE\_THREAD\_CAP
  - Capabilities, [144](#)
- L4\_BDR\_IO\_SHIFT
  - Buffer Registers (BRs), [395](#)
- L4\_BDR\_MEM\_SHIFT
  - Buffer Registers (BRs), [395](#)
- L4\_BDR\_OBJ\_SHIFT
  - Buffer Registers (BRs), [395](#)
- l4\_buf\_regs\_t, [1383](#)
- l4\_buffer\_desc\_consts\_t
  - Buffer Registers (BRs), [395](#)
- l4\_busy\_wait\_ns
  - Timestamp Counter, [648](#)
- l4\_busy\_wait\_us
  - Timestamp Counter, [649](#)
- l4\_bytes\_to\_mwords
  - Memory related, [336](#)
- l4\_cache\_clean\_data
  - Cache Consistency, [139](#)
- l4\_cache\_coherent
  - Cache Consistency, [140](#)
- l4\_cache\_dma\_coherent
  - Cache Consistency, [140](#)
- l4\_cache\_flush\_data
  - Cache Consistency, [141](#)
- l4\_cache\_inv\_data
  - Cache Consistency, [141](#)
- l4\_calibrate\_tsc
  - Timestamp Counter, [649](#)
- l4\_cap\_consts\_t
  - Capabilities, [143](#)
- L4\_CAP\_FPAGE\_D
  - Flex pages, [166](#)
- L4\_CAP\_FPAGE\_R
  - Flex pages, [166](#)
- L4\_cap\_fpage\_rights
  - Flex pages, [165](#)
- L4\_CAP\_FPAGE\_RO
  - Flex pages, [166](#)
- L4\_CAP\_FPAGE\_RS
  - Flex pages, [166](#)
- L4\_CAP\_FPAGE\_RSD
  - Flex pages, [166](#)
- L4\_CAP\_FPAGE\_RW
  - Flex pages, [166](#)
- L4\_CAP\_FPAGE\_RWD
  - Flex pages, [166](#)
- L4\_CAP\_FPAGE\_RWS
  - Flex pages, [166](#)
- L4\_CAP\_FPAGE\_RWSD
  - Flex pages, [166](#)
- L4\_CAP\_FPAGE\_S



- Flex pages, [165](#)
- L4\_CAP\_FPAGE\_W
  - Flex pages, [165](#)
- l4\_cap\_idx\_t
  - Capabilities, [143](#)
- L4\_CAP\_MASK
  - Capabilities, [143](#)
- L4\_CAP\_OFFSET
  - Capabilities, [143](#)
- L4\_CAP\_SHIFT
  - Capabilities, [143](#)
- L4\_CAP\_SIZE
  - Capabilities, [143](#)
- l4\_capability\_equal
  - Capabilities, [144](#)
- l4\_capability\_next
  - types.h, [2146](#)
- l4\_debugger\_add\_image\_info
  - Kernel Debugger, [153](#)
- l4\_debugger\_get\_object\_name
  - Kernel Debugger, [153](#)
- l4\_debugger\_global\_id
  - Kernel Debugger, [154](#)
- l4\_debugger\_kobj\_to\_id
  - Kernel Debugger, [155](#)
- l4\_debugger\_query\_log\_name
  - Kernel Debugger, [156](#)
- l4\_debugger\_query\_log\_typeid
  - Kernel Debugger, [157](#)
- l4\_debugger\_set\_object\_name
  - Kernel Debugger, [157](#)
- l4\_debugger\_switch\_log
  - Kernel Debugger, [158](#)
- l4\_default\_caps\_t
  - Capabilities, [144](#)
- L4\_DISABLE\_COPY
  - capability, [2706](#)
- L4\_E2BIG
  - Error codes, [147](#)
- L4\_EACCESS
  - Error codes, [147](#)
- L4\_EADDRNOTAVAIL
  - Error codes, [147](#)
- L4\_EAGAIN
  - Error codes, [147](#)
- L4\_EBADPROTO
  - Error codes, [147](#)
- L4\_EBUSY
  - Error codes, [147](#)
- L4\_EEXIST
  - Error codes, [147](#)
- L4\_EFAULT
  - Error codes, [147](#)
- L4\_EINVAL
  - Error codes, [147](#)
- L4\_EIO
  - Error codes, [147](#)
- L4\_EIPC\_HI
  - Error codes, [148](#)
- L4\_EIPC\_LO
  - Error codes, [148](#)
- L4\_EMMSGMISSARG
  - Error codes, [148](#)
- L4\_EMMSGTOOLONG
  - Error codes, [148](#)
- L4\_EMMSGTOOSHORT
  - Error codes, [148](#)
- L4\_ENAMETOOLONG
  - Error codes, [147](#)
- L4\_ENODEV
  - Error codes, [147](#)
- L4\_ENOENT
  - Error codes, [147](#)
- L4\_ENOMEM
  - Error codes, [147](#)
- L4\_ENOREPLY
  - Error codes, [148](#)
- L4\_ENOSPC
  - Error codes, [147](#)
- L4\_ENOSYS
  - Error codes, [147](#)
- L4\_ENOTDIR
  - Error codes, [147](#)
- L4\_ENXIO
  - Error codes, [147](#)
- L4\_EOK
  - Error codes, [147](#)
- L4\_EPERM
  - Error codes, [147](#)
- L4\_ERANGE
  - Error codes, [147](#)
- L4\_ERRNOMAX
  - Error codes, [147](#)
- l4\_error
  - Error Handling, [360](#)
- l4\_error\_code\_t
  - Error codes, [147](#)
- l4\_exc\_regs\_t, [1384](#)
  - flags, [1386](#)
  - ss, [1386](#)
- L4\_EXPORT
  - Basic Macros, [136](#)
- l4\_ext\_vcpu\_state\_vmx\_t, [1387](#)
  - VM API for VMX, [315](#)
- l4\_factory\_create
  - Factory, [197](#)
- l4\_factory\_create\_factory
  - Factory, [198](#)
- l4\_factory\_create\_gate
  - Factory, [199](#)
- l4\_factory\_create\_irq
  - Factory, [200](#)
- l4\_factory\_create\_task
  - Factory, [201](#)
- l4\_factory\_create\_thread
  - Factory, [202](#)



- [l4\\_factory\\_create\\_vcpu\\_context](#)
  - [Factory, 203](#)
- [l4\\_factory\\_create\\_vm](#)
  - [Factory, 204](#)
- [L4\\_FP\\_ALL\\_SPACES](#)
  - [Task, 256](#)
- [L4\\_FP\\_DELETE\\_OBJ](#)
  - [Task, 256](#)
- [L4\\_FP\\_OTHER\\_SPACES](#)
  - [Task, 256](#)
- [l4\\_fpage](#)
  - [Flex pages, 169](#)
- [L4\\_FPAGE\\_ADDR\\_BITS](#)
  - [Flex pages, 167](#)
- [L4\\_FPAGE\\_ADDR\\_SHIFT](#)
  - [Flex pages, 167](#)
- [l4\\_fpage\\_all](#)
  - [Flex pages, 169](#)
- [L4\\_FPAGE\\_BUFFERABLE](#)
  - [Flex pages, 167](#)
- [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#)
  - [Flex pages, 169](#)
- [L4\\_FPAGE\\_C\\_NO\\_REF\\_CNT](#)
  - [Flex pages, 169](#)
- [L4\\_FPAGE\\_C\\_OBJ\\_RIGHT1](#)
  - [Flex pages, 169](#)
- [L4\\_FPAGE\\_C\\_OBJ\\_RIGHT2](#)
  - [Flex pages, 169](#)
- [L4\\_FPAGE\\_C\\_OBJ\\_RIGHT3](#)
  - [Flex pages, 169](#)
- [L4\\_FPAGE\\_C\\_OBJ\\_RIGHTS](#)
  - [Flex pages, 169](#)
- [L4\\_FPAGE\\_C\\_REF\\_CNT](#)
  - [Flex pages, 168](#)
- [L4\\_FPAGE\\_CACHE\\_OPT](#)
  - [Flex pages, 167](#)
- [l4\\_fpage\\_cacheability\\_opt\\_t](#)
  - [Flex pages, 166](#)
- [L4\\_FPAGE\\_CACHEABLE](#)
  - [Flex pages, 167](#)
- [L4\\_fpage\\_consts](#)
  - [Flex pages, 167](#)
- [l4\\_fpage\\_contains](#)
  - [Flex pages, 170](#)
- [L4\\_fpage\\_control](#)
  - [Flex pages, 167](#)
- [L4\\_FPAGE\\_CONTROL\\_MASK](#)
  - [Flex pages, 167](#)
- [L4\\_FPAGE\\_CONTROL\\_OFFSET\\_SHIFT](#)
  - [Flex pages, 167](#)
- [l4\\_fpage\\_invalid](#)
  - [Flex pages, 171](#)
- [L4\\_FPAGE\\_IO](#)
  - [Flex pages, 168](#)
- [l4\\_fpage\\_ioport](#)
  - [Flex pages, 171](#)
- [l4\\_fpage\\_max\\_order](#)
  - [Flex pages, 172](#)
- [l4\\_fpage\\_memaddr](#)
  - [Flex pages, 172](#)
- [L4\\_FPAGE\\_MEMORY](#)
  - [Flex pages, 168](#)
- [L4\\_FPAGE\\_OBJ](#)
  - [Flex pages, 168](#)
- [l4\\_fpage\\_obj](#)
  - [Flex pages, 174](#)
- [l4\\_fpage\\_page](#)
  - [Flex pages, 175](#)
- [L4\\_fpage\\_rights](#)
  - [Flex pages, 167](#)
- [l4\\_fpage\\_rights](#)
  - [Flex pages, 175](#)
- [L4\\_FPAGE\\_RIGHTS\\_ALL](#)
  - [Flex pages, 167](#)
- [L4\\_FPAGE\\_RIGHTS\\_BITS](#)
  - [Flex pages, 167](#)
- [L4\\_FPAGE\\_RIGHTS\\_MASK](#)
  - [Flex pages, 167](#)
- [L4\\_FPAGE\\_RIGHTS\\_SHIFT](#)
  - [Flex pages, 167](#)
- [L4\\_FPAGE\\_RO](#)
  - [Flex pages, 168](#)
- [L4\\_FPAGE\\_RW](#)
  - [Flex pages, 168](#)
- [L4\\_FPAGE\\_RWX](#)
  - [Flex pages, 168](#)
- [L4\\_FPAGE\\_RX](#)
  - [Flex pages, 168](#)
- [l4\\_fpage\\_set\\_rights](#)
  - [Flex pages, 176](#)
- [l4\\_fpage\\_size](#)
  - [Flex pages, 176](#)
- [L4\\_FPAGE\\_SIZE\\_BITS](#)
  - [Flex pages, 167](#)
- [L4\\_FPAGE\\_SIZE\\_SHIFT](#)
  - [Flex pages, 167](#)
- [L4\\_FPAGE\\_SPECIAL](#)
  - [Flex pages, 168](#)
- [l4\\_fpage\\_t, 1388](#)
- [L4\\_fpage\\_type](#)
  - [Flex pages, 168](#)
- [l4\\_fpage\\_type](#)
  - [Flex pages, 177](#)
- [L4\\_FPAGE\\_TYPE\\_BITS](#)
  - [Flex pages, 167](#)
- [L4\\_FPAGE\\_TYPE\\_SHIFT](#)
  - [Flex pages, 167](#)
- [L4\\_FPAGE\\_UNCACHEABLE](#)
  - [Flex pages, 167](#)
- [L4\\_FPAGE\\_W](#)
  - [Flex pages, 168](#)
- [L4\\_FPAGE\\_X](#)
  - [Flex pages, 168](#)
- [l4\\_get\\_hz](#)
  - [Timestamp Counter, 650](#)
- [L4\\_HIDDEN](#)

- Basic Macros, [136](#)
- I4\_icu\_bind
  - Interrupt controller, [224](#)
- I4\_icu\_bind\_u
  - Interrupt controller, [225](#)
- L4\_ICU\_FLAG\_MSI
  - Interrupt controller, [224](#)
- L4\_icu\_flags
  - Interrupt controller, [224](#)
- I4\_icu\_info
  - Interrupt controller, [226](#)
- I4\_icu\_info\_t, [1389](#)
  - features, [1390](#)
  - Interrupt controller, [224](#)
- I4\_icu\_info\_u
  - Interrupt controller, [227](#)
- I4\_icu\_mask
  - Interrupt controller, [228](#)
- I4\_icu\_mask\_u
  - Interrupt controller, [229](#)
- I4\_icu\_msi\_info
  - Interrupt controller, [230](#)
- I4\_icu\_msi\_info\_t, [1390](#)
- I4\_icu\_msi\_info\_u
  - Interrupt controller, [231](#)
- L4\_ICU\_OP\_BIND
  - L4 IPC Opcodes, [424](#)
- L4\_ICU\_OP\_INFO
  - L4 IPC Opcodes, [425](#)
- L4\_ICU\_OP\_MASK
  - L4 IPC Opcodes, [425](#)
- L4\_ICU\_OP\_MSI\_INFO
  - L4 IPC Opcodes, [425](#)
- L4\_ICU\_OP\_SET\_MODE
  - L4 IPC Opcodes, [425](#)
- L4\_ICU\_OP\_UNBIND
  - L4 IPC Opcodes, [424](#)
- L4\_ICU\_OP\_UNMASK
  - L4 IPC Opcodes, [425](#)
- L4\_icu\_opcode
  - L4 IPC Opcodes, [424](#)
- I4\_icu\_set\_mode
  - Interrupt controller, [232](#)
- I4\_icu\_set\_mode\_u
  - Interrupt controller, [233](#)
- I4\_icu\_unbind
  - Interrupt controller, [234](#)
- I4\_icu\_unbind\_u
  - Interrupt controller, [235](#)
- I4\_icu\_unmask
  - Interrupt controller, [236](#)
- I4\_icu\_unmask\_u
  - Interrupt controller, [237](#)
- I4\_infinite\_loop
  - Basic Macros, [137](#)
- L4\_INLINE\_RPC
  - ipc\_iface, [2730](#)
- L4\_INLINE\_RPC\_NF
  - ipc\_iface, [2731](#)
- L4\_INLINE\_RPC\_NF\_OP
  - ipc\_iface, [2731](#)
- L4\_INLINE\_RPC\_OP
  - ipc\_iface, [2732](#)
- L4\_INVALID\_ADDR
  - Memory related, [336](#)
- L4\_INVALID\_CAP
  - Capabilities, [143](#)
- I4\_iofpage
  - Flex pages, [177](#)
- L4\_IOPORT\_MAX
  - Flex pages, [165](#)
- I4\_ipc
  - Object Invocation, [343](#)
- I4\_ipc\_call
  - Object Invocation, [345](#)
- L4\_IPC\_ENOT\_EXISTENT
  - Error Handling, [359](#)
- I4\_ipc\_error
  - Error Handling, [361](#)
- I4\_ipc\_error\_code
  - Error Handling, [362](#)
- L4\_IPC\_ERROR\_MASK
  - Error Handling, [359](#)
- L4\_IPC\_GATE\_BIND\_OP
  - L4 IPC Opcodes, [425](#)
- L4\_IPC\_GATE\_GET\_INFO\_OP
  - L4 IPC Opcodes, [425](#)
- I4\_ipc\_gate\_get\_infos
  - IPC-Gate API, [206](#)
- L4\_ipc\_gate\_ops
  - L4 IPC Opcodes, [425](#)
- I4\_ipc\_is\_rcv\_error
  - Error Handling, [363](#)
- I4\_ipc\_is\_snd\_error
  - Error Handling, [363](#)
- L4\_IPC\_REABORTED
  - Error Handling, [360](#)
- L4\_IPC\_RECANCELED
  - Error Handling, [360](#)
- I4\_ipc\_receive
  - Object Invocation, [347](#)
- L4\_IPC\_REMAPFAILED
  - Error Handling, [360](#)
- L4\_IPC\_REMSGCUT
  - Error Handling, [360](#)
- I4\_ipc\_reply\_and\_wait
  - Object Invocation, [349](#)
- L4\_IPC\_RERCVPFTO
  - Error Handling, [360](#)
- L4\_IPC\_RESNDPFTO
  - Error Handling, [360](#)
- L4\_IPC\_RETIMEOUT
  - Error Handling, [359](#)
- L4\_IPC\_SEABORTED
  - Error Handling, [360](#)
- L4\_IPC\_SECANCELED

- Error Handling, [360](#)
- L4\_IPC\_SEMAPFAILED
  - Error Handling, [360](#)
- L4\_IPC\_SEMSGCUT
  - Error Handling, [360](#)
- l4\_ipc\_send
  - Object Invocation, [351](#)
- l4\_ipc\_send\_and\_wait
  - Object Invocation, [352](#)
- L4\_IPC\_SERCVPFTO
  - Error Handling, [360](#)
- L4\_IPC\_SESNDFPTO
  - Error Handling, [360](#)
- L4\_IPC\_SETIMEOUT
  - Error Handling, [360](#)
- l4\_ipc\_sleep
  - Object Invocation, [353](#)
- l4\_ipc\_sleep\_ms
  - Object Invocation, [354](#)
- l4\_ipc\_sleep\_us
  - Object Invocation, [355](#)
- L4\_IPC\_SND\_ERR\_MASK
  - Error Handling, [359](#)
- l4\_ipc\_tcr\_error\_t
  - Error Handling, [359](#)
- l4\_ipc\_timeout
  - Timeouts, [383](#)
- L4\_IPC\_TIMEOUT\_0
  - Timeouts, [383](#)
- l4\_ipc\_to\_errno
  - ipc.h, [2800](#)
- l4\_ipc\_wait
  - Object Invocation, [356](#)
- l4\_irq\_detach
  - IRQs, [210](#)
- l4\_irq\_detach\_u
  - IRQs, [211](#)
- L4\_IRQ\_F\_BOTH
  - IRQs, [210](#)
- L4\_IRQ\_F\_BOTH\_EDGE
  - IRQs, [210](#)
- L4\_IRQ\_F\_CLEAR\_WAKEUP
  - IRQs, [210](#)
- L4\_IRQ\_F\_EDGE
  - IRQs, [210](#)
- L4\_IRQ\_F\_LEVEL
  - IRQs, [210](#)
- L4\_IRQ\_F\_LEVEL\_HIGH
  - IRQs, [210](#)
- L4\_IRQ\_F\_LEVEL\_LOW
  - IRQs, [210](#)
- L4\_IRQ\_F\_MASK
  - IRQs, [210](#)
- L4\_IRQ\_F\_NEG
  - IRQs, [210](#)
- L4\_IRQ\_F\_NEG\_EDGE
  - IRQs, [210](#)
- L4\_IRQ\_F\_NONE
  - IRQs, [210](#)
- L4\_IRQ\_F\_POS
  - IRQs, [210](#)
- L4\_IRQ\_F\_POS\_EDGE
  - IRQs, [210](#)
- L4\_IRQ\_F\_SET\_WAKEUP
  - IRQs, [210](#)
- L4\_irq\_mode
  - IRQs, [209](#)
- l4\_irq\_mux\_chain
  - IRQs, [212](#)
- l4\_irq\_mux\_chain\_u
  - IRQs, [213](#)
- l4\_irq\_receive
  - IRQs, [214](#)
- l4\_irq\_receive\_u
  - IRQs, [215](#)
- l4\_irq\_trigger
  - IRQs, [216](#)
- l4\_irq\_trigger\_u
  - IRQs, [217](#)
- l4\_irq\_unmask
  - IRQs, [218](#)
- l4\_irq\_unmask\_u
  - IRQs, [219](#)
- l4\_irq\_wait
  - IRQs, [220](#)
- l4\_irq\_wait\_u
  - IRQs, [220](#)
- l4\_is\_fpage\_writable
  - Flex pages, [178](#)
- l4\_is\_invalid\_cap
  - Capabilities, [145](#)
- l4\_is\_valid\_cap
  - Capabilities, [145](#)
- L4\_ITEM\_CONT
  - Message Items, [365](#)
- L4\_ITEM\_MAP
  - Message Items, [365](#)
- l4\_kdebug\_ops\_t
  - kdebug.h, [2829](#)
- l4\_kernel\_info\_get\_mem\_desc\_end
  - Memory descriptors (C version), [192](#)
- l4\_kernel\_info\_get\_mem\_desc\_is\_virtual
  - Memory descriptors (C version), [192](#)
- l4\_kernel\_info\_get\_mem\_desc\_start
  - Memory descriptors (C version), [192](#)
- l4\_kernel\_info\_get\_mem\_desc\_subtype
  - Memory descriptors (C version), [192](#)
- l4\_kernel\_info\_get\_mem\_desc\_type
  - Memory descriptors (C version), [192](#)
- l4\_kernel\_info\_get\_num\_mem\_descs
  - Memory descriptors (C version), [193](#)
- l4\_kernel\_info\_mem\_desc\_t, [1391](#)
  - Memory descriptors (C version), [190](#)
- l4\_kernel\_info\_set\_mem\_desc
  - Memory descriptors (C version), [193](#)
- l4\_kernel\_info\_t, [1392](#)

- `I4_kernel_info_version_offset`  
Kernel Interface Page, [184](#)
- `I4_kip`  
Kernel Interface Page, [184](#)
- `I4_kip_clock`  
Kernel Interface Page, [184](#)
- `I4_kip_clock_lw`  
Kernel Interface Page, [185](#)
- `I4_kip_clock_ns`  
Kernel Interface Page, [186](#)
- `I4_kip_for_each_feature`  
kip.h, [2851](#)
- `I4_kip_kernel_has_feature`  
kip.h, [2851](#)
- `L4_KIP_OFFS_READ_NS`  
Kernel Interface Page, [183](#)
- `L4_KIP_OFFS_READ_US`  
Kernel Interface Page, [183](#)
- `I4_kip_version`  
Kernel Interface Page, [187](#)
- `I4_kip_version_string`  
Kernel Interface Page, [187](#)
- `L4_LOG2_PAGESIZE`  
Memory related, [333](#)
- `L4_LOG2_SUPERPAGESIZE`  
Memory related, [333](#)
- `I4_map_control`  
Message Items, [366](#)
- `L4_MAP_ITEM_GRANT`  
Message Items, [365](#)
- `L4_MAP_ITEM_MAP`  
Message Items, [365](#)
- `I4_map_obj_control`  
Message Items, [366](#)
- `I4_mem_archspecific_acpi_nvs`  
Memory descriptors (C version), [191](#)
- `I4_mem_archspecific_acpi_tables`  
Memory descriptors (C version), [191](#)
- `I4_mem_archspecific_sub_type_common_t`  
Memory descriptors (C version), [191](#)
- `I4_mem_info_acpi_rsdp`  
Memory descriptors (C version), [191](#)
- `I4_mem_info_sub_type_t`  
Memory descriptors (C version), [191](#)
- `L4_mem_op_widths`  
Memory operations., [330](#)
- `I4_mem_read`  
Memory operations., [331](#)
- `I4_mem_type_archspecific`  
Memory descriptors (C version), [191](#)
- `I4_mem_type_bootloader`  
Memory descriptors (C version), [191](#)
- `I4_mem_type_conventional`  
Memory descriptors (C version), [191](#)
- `I4_mem_type_dedicated`  
Memory descriptors (C version), [191](#)
- `I4_mem_type_info`  
Memory descriptors (C version), [191](#)
- `I4_mem_type_reserved`  
Memory descriptors (C version), [191](#)
- `I4_mem_type_shared`  
Memory descriptors (C version), [191](#)
- `I4_mem_type_t`  
Memory descriptors (C version), [191](#)
- `I4_mem_type_undefined`  
Memory descriptors (C version), [191](#)
- `L4_MEM_WIDTH_1BYTE`  
Memory operations., [330](#)
- `L4_MEM_WIDTH_2BYTE`  
Memory operations., [330](#)
- `L4_MEM_WIDTH_4BYTE`  
Memory operations., [330](#)
- `I4_mem_write`  
Memory operations., [331](#)
- `I4_msg_item_consts_t`  
Message Items, [365](#)
- `I4_msg_regs_t`, [1394](#)
- `I4_msgtag`  
Message Tag, [371](#)
- `L4_MSGTAG_ERROR`  
Message Tag, [369](#)
- `L4_MSGTAG_FLAGS`  
Message Tag, [369](#)
- `L4_msgtag_flags`  
Message Tag, [369](#)
- `I4_msgtag_flags`  
Message Tag, [372](#)
- `I4_msgtag_has_error`  
Message Tag, [373](#)
- `I4_msgtag_is_exception`  
Message Tag, [373](#)
- `I4_msgtag_is_io_page_fault`  
Message Tag, [375](#)
- `I4_msgtag_is_page_fault`  
Message Tag, [376](#)
- `I4_msgtag_is_preemption`  
Message Tag, [376](#)
- `I4_msgtag_is_sigma0`  
Message Tag, [377](#)
- `I4_msgtag_is_sys_exception`  
Message Tag, [378](#)
- `I4_msgtag_items`  
Message Tag, [378](#)
- `I4_msgtag_label`  
Message Tag, [379](#)
- `L4_msgtag_protocol`  
Message Tag, [370](#)
- `L4_MSGTAG_SCHEDULE`  
Message Tag, [369](#)
- `I4_msgtag_t`, [1395](#)  
flags, [1396](#)  
Message Tag, [369](#)
- `L4_MSGTAG_TRANSFER_FPU`  
Message Tag, [369](#)
- `I4_msgtag_words`  
Message Tag, [380](#)

- L4\_NOTHROW
  - Basic Macros, [136](#)
- l4\_ns\_to\_tsc
  - Timestamp Counter, [650](#)
- l4\_obj\_fpage
  - Flex pages, [179](#)
- L4\_obj\_fpage\_ctl
  - Flex pages, [168](#)
- L4\_PAGEMASK
  - Memory related, [334](#)
- L4\_PAGESHIFT
  - Memory related, [334](#)
- l4\_platform\_ctl\_cpu\_allow\_shutdown
  - Platform Control C API, [242](#)
- L4\_PLATFORM\_CTL\_CPU\_ALLOW\_SHUTDOWN\_OP
  - L4 IPC Opcodes, [426](#)
- l4\_platform\_ctl\_cpu\_disable
  - Platform Control C API, [243](#)
- L4\_PLATFORM\_CTL\_CPU\_DISABLE\_OP
  - L4 IPC Opcodes, [426](#)
- l4\_platform\_ctl\_cpu\_enable
  - Platform Control C API, [244](#)
- L4\_PLATFORM\_CTL\_CPU\_ENABLE\_OP
  - L4 IPC Opcodes, [426](#)
- l4\_platform\_ctl\_ops
  - L4 IPC Opcodes, [425](#)
- l4\_platform\_ctl\_proto
  - Message Tag, [370](#)
- l4\_platform\_ctl\_set\_task\_asid
  - Platform Control C API, [244](#)
- L4\_PLATFORM\_CTL\_SET\_TASK\_ASID\_OP
  - L4 IPC Opcodes, [426](#)
- L4\_PLATFORM\_CTL\_SYS\_SHUTDOWN\_OP
  - L4 IPC Opcodes, [426](#)
- L4\_PLATFORM\_CTL\_SYS\_SUSPEND\_OP
  - L4 IPC Opcodes, [426](#)
- l4\_platform\_ctl\_system\_shutdown
  - Platform Control C API, [245](#)
- l4\_platform\_ctl\_system\_suspend
  - Platform Control C API, [246](#)
- L4\_PROTO\_ALLOW\_SYSCALL
  - Message Tag, [370](#)
- L4\_PROTO\_DEBUGGER
  - Message Tag, [370](#)
- L4\_PROTO\_DMA\_SPACE
  - Message Tag, [370](#)
- L4\_PROTO\_EXCEPTION
  - Message Tag, [370](#)
- L4\_PROTO\_FACTORY
  - Message Tag, [370](#)
- L4\_PROTO\_IO\_PAGE\_FAULT
  - Message Tag, [370](#)
- L4\_PROTO\_IOMMU
  - Message Tag, [370](#)
- L4\_PROTO\_IRQ
  - Message Tag, [370](#)
- L4\_PROTO\_IRQ\_MUX
  - Message Tag, [370](#)
- L4\_PROTO\_IRQ\_SENDER
  - Message Tag, [370](#)
- L4\_PROTO\_KOBJECT
  - Message Tag, [370](#)
- L4\_PROTO\_LOG
  - Message Tag, [370](#)
- L4\_PROTO\_META
  - Message Tag, [370](#)
- L4\_PROTO\_NONE
  - Message Tag, [370](#)
- L4\_PROTO\_PAGE\_FAULT
  - Message Tag, [370](#)
- L4\_PROTO\_PF\_EXCEPTION
  - Message Tag, [370](#)
- L4\_PROTO\_PLATFORM\_CTL
  - Message Tag, [371](#)
- L4\_PROTO\_PREEMPTION
  - Message Tag, [370](#)
- L4\_PROTO\_SCHEDULER
  - Message Tag, [370](#)
- L4\_PROTO\_SEMAPHORE
  - Message Tag, [370](#)
- L4\_PROTO\_SIGMA0
  - Message Tag, [370](#)
- L4\_PROTO\_SMCCC
  - Message Tag, [370](#)
- L4\_PROTO\_SYS\_EXCEPTION
  - Message Tag, [370](#)
- L4\_PROTO\_TASK
  - Message Tag, [370](#)
- L4\_PROTO\_THREAD
  - Message Tag, [370](#)
- L4\_PROTO\_VCPU\_CONTEXT
  - Message Tag, [370](#)
- L4\_PROTO\_VM
  - Message Tag, [370](#)
- L4\_RCV\_EP\_BIND\_OP
  - rcv\_endpoint.h, [2877](#)
- l4\_rcv\_ep\_bind\_thread
  - IPC-Gate API, [207](#)
- L4\_rcv\_ep\_ops
  - rcv\_endpoint.h, [2877](#)
- L4\_RCV\_ITEM\_LOCAL\_ID
  - Message Items, [365](#)
- L4\_RCV\_ITEM\_SINGLE\_CAP
  - Message Items, [365](#)
- l4\_rcv\_timeout
  - Timeouts, [384](#)
- l4\_rdpmc
  - Timestamp Counter, [651](#)
- l4\_rdpmc\_32
  - Timestamp Counter, [651](#)
- l4\_rdtsc
  - Timestamp Counter, [651](#)
- l4\_rdtsc\_32
  - Timestamp Counter, [652](#)
- l4\_round\_page
  - Memory related, [336](#)

- l4\_round\_size
  - Memory related, [337](#)
- L4\_RPC
  - ipc\_iface, [2732](#)
- L4\_RPC\_DEF
  - ipc\_client, [2723](#)
- L4\_RPC\_NF
  - ipc\_iface, [2733](#)
- L4\_RPC\_NF\_OP
  - ipc\_iface, [2733](#)
- L4\_RPC\_OP
  - ipc\_iface, [2733](#)
- l4\_sched\_cpu\_set
  - Scheduler, [249](#)
- l4\_sched\_cpu\_set\_t, [1397](#)
  - gran\_offset, [1400](#)
  - granularity, [1398](#)
  - offset, [1398](#)
  - set, [1398](#)
- l4\_sched\_param
  - Scheduler, [249](#)
- l4\_sched\_param\_t, [1401](#)
- L4\_SCHEDULER\_CLASS\_FIXED\_PRIO
  - Scheduler, [248](#)
- L4\_SCHEDULER\_CLASS\_WFQ
  - Scheduler, [248](#)
- L4\_scheduler\_classes
  - Scheduler, [248](#)
- l4\_scheduler\_idle\_time
  - Scheduler, [250](#)
- L4\_SCHEDULER\_IDLE\_TIME\_OP
  - Scheduler, [249](#)
- l4\_scheduler\_info
  - Scheduler, [251](#)
- L4\_SCHEDULER\_INFO\_OP
  - Scheduler, [249](#)
- l4\_scheduler\_info\_with\_classes
  - Scheduler, [252](#)
- l4\_scheduler\_is\_online
  - Scheduler, [253](#)
- L4\_scheduler\_ops
  - Scheduler, [248](#)
- l4\_scheduler\_run\_thread
  - Scheduler, [253](#)
- L4\_SCHEDULER\_RUN\_THREAD\_OP
  - Scheduler, [249](#)
- l4\_semaphore\_down
  - Kernel-provided semaphore, [238](#)
- l4\_semaphore\_up
  - Kernel-provided semaphore, [239](#)
- l4\_sleep
  - Utility Functions, [578](#)
- l4\_snd\_fpage\_t, [1402](#)
- l4\_snd\_timeout
  - Timeouts, [384](#)
- l4\_sndfpage\_add
  - Object Invocation, [357](#)
- L4\_SUPERPAGEMASK
  - Memory related, [334](#)
- L4\_SUPERPAGESHIFT
  - Memory related, [335](#)
- L4\_SUPERPAGESIZE
  - Memory related, [335](#)
- L4\_sys\_segment
  - segment.h, [2071](#)
- l4\_syscall\_flags\_t
  - Object Invocation, [342](#)
- L4\_SYSF\_CALL
  - Object Invocation, [343](#)
- L4\_SYSF\_NONE
  - Object Invocation, [342](#)
- L4\_SYSF\_OPEN\_WAIT
  - Object Invocation, [343](#)
- L4\_SYSF\_RECV
  - Object Invocation, [343](#)
- L4\_SYSF\_REPLY
  - Object Invocation, [343](#)
- L4\_SYSF\_REPLY\_AND\_WAIT
  - Object Invocation, [343](#)
- L4\_SYSF\_SEND
  - Object Invocation, [343](#)
- L4\_SYSF\_SEND\_AND\_WAIT
  - Object Invocation, [343](#)
- L4\_SYSF\_WAIT
  - Object Invocation, [343](#)
- l4\_task\_add\_ku\_mem
  - Task, [256](#)
- L4\_TASK\_ADD\_KU\_MEM\_OP
  - L4 IPC Opcodes, [426](#)
- l4\_task\_cap\_equal
  - Task, [257](#)
- L4\_TASK\_CAP\_INFO\_OP
  - L4 IPC Opcodes, [426](#)
- l4\_task\_cap\_valid
  - Task, [258](#)
- l4\_task\_delete\_obj
  - Task, [259](#)
- L4\_TASK\_LDT\_SET\_X86\_OP
  - L4 IPC Opcodes, [426](#)
- L4\_task\_ldt\_x86\_consts
  - segment.h, [2071](#), [2080](#)
- L4\_TASK\_LDT\_X86\_ENTRY\_SIZE
  - segment.h, [2071](#), [2080](#)
- L4\_TASK\_LDT\_X86\_MAX\_ENTRIES
  - segment.h, [2071](#), [2080](#)
- l4\_task\_map
  - Task, [259](#)
- L4\_TASK\_MAP\_OP
  - L4 IPC Opcodes, [426](#)
- L4\_TASK\_MAP\_VGICC\_ARM\_OP
  - L4 IPC Opcodes, [426](#)
- L4\_task\_ops
  - L4 IPC Opcodes, [426](#)
- l4\_task\_release\_cap
  - Task, [261](#)
- l4\_task\_unmap

- Task, [262](#)
- I4\_task\_unmap\_batch
  - Task, [263](#)
- L4\_TASK\_UNMAP\_OP
  - L4 IPC Opcodes, [426](#)
- I4\_task\_vgicc\_map
  - Task, [264](#)
- L4\_THREAD\_AMD64\_GET\_SEGMENT\_INFO\_OP
  - L4 IPC Opcodes, [426](#)
- L4\_THREAD\_AMD64\_SET\_SEGMENT\_BASE\_OP
  - L4 IPC Opcodes, [426](#)
- I4\_thread\_arm\_set\_tpidruro
  - Thread, [269](#)
- L4\_THREAD\_ARM\_TPIDRURO\_OP
  - L4 IPC Opcodes, [426](#)
- L4\_THREAD\_CONTROL\_ALIEN
  - Thread, [268](#)
- I4\_thread\_control\_alien
  - Thread control, [286](#)
- I4\_thread\_control\_bind
  - Thread control, [287](#)
- L4\_THREAD\_CONTROL\_BIND\_TASK
  - Thread, [268](#)
- I4\_thread\_control\_commit
  - Thread control, [288](#)
- I4\_thread\_control\_exc\_handler
  - Thread control, [289](#)
- L4\_thread\_control\_flags
  - Thread, [267](#)
- L4\_THREAD\_CONTROL\_MR\_IDX\_BIND\_TASK
  - Thread, [268](#)
- L4\_THREAD\_CONTROL\_MR\_IDX\_BIND\_UTCB
  - Thread, [268](#)
- L4\_THREAD\_CONTROL\_MR\_IDX\_EXC\_HANDLER
  - Thread, [268](#)
- L4\_THREAD\_CONTROL\_MR\_IDX\_FLAG\_VALS
  - Thread, [268](#)
- L4\_THREAD\_CONTROL\_MR\_IDX\_FLAGS
  - Thread, [268](#)
- L4\_THREAD\_CONTROL\_MR\_IDX\_PAGER
  - Thread, [268](#)
- L4\_thread\_control\_mr\_indices
  - Thread, [268](#)
- L4\_THREAD\_CONTROL\_OP
  - L4 IPC Opcodes, [426](#)
- I4\_thread\_control\_pager
  - Thread control, [289](#)
- L4\_THREAD\_CONTROL\_SET\_EXC\_HANDLER
  - Thread, [268](#)
- L4\_THREAD\_CONTROL\_SET\_PAGER
  - Thread, [268](#)
- I4\_thread\_control\_start
  - Thread control, [290](#)
- I4\_thread\_control\_ux\_host\_syscall
  - Thread control, [291](#)
- L4\_THREAD\_CONTROL\_UX\_NATIVE
  - Thread, [268](#)
- I4\_thread\_ex\_regs
  - Thread, [269](#)
- L4\_THREAD\_EX\_REGS\_ARCH\_MASK
  - Thread, [268](#)
- L4\_THREAD\_EX\_REGS\_ARM\_SET\_EL\_EL0
  - Thread, [269](#)
- L4\_THREAD\_EX\_REGS\_ARM\_SET\_EL\_EL1
  - Thread, [269](#)
- L4\_THREAD\_EX\_REGS\_ARM\_SET\_EL\_KEEP
  - Thread, [269](#)
- L4\_THREAD\_EX\_REGS\_ARM\_SET\_EL\_MASK
  - Thread, [269](#)
- L4\_THREAD\_EX\_REGS\_CANCEL
  - Thread, [268](#)
- L4\_thread\_ex\_regs\_flags
  - Thread, [268](#)
- L4\_thread\_ex\_regs\_flags\_arm
  - Thread, [268](#)
- L4\_THREAD\_EX\_REGS\_OP
  - L4 IPC Opcodes, [426](#)
- I4\_thread\_ex\_regs\_ret
  - Thread, [270](#)
- I4\_thread\_ex\_regs\_ret\_u
  - Thread, [271](#)
- L4\_THREAD\_EX\_REGS\_TRIGGER\_EXCEPTION
  - Thread, [268](#)
- I4\_thread\_ex\_regs\_u
  - Thread, [272](#)
- I4\_thread\_modify\_sender\_add
  - Thread, [274](#)
- I4\_thread\_modify\_sender\_commit
  - Thread, [275](#)
- L4\_THREAD\_MODIFY\_SENDER\_OP
  - L4 IPC Opcodes, [426](#)
- I4\_thread\_modify\_sender\_start
  - Thread, [275](#)
- L4\_THREAD\_OPCODE\_MASK
  - L4 IPC Opcodes, [426](#)
- L4\_thread\_ops
  - L4 IPC Opcodes, [426](#)
- I4\_thread\_register\_del\_irq
  - Thread, [276](#)
- L4\_THREAD\_REGISTER\_DELETE\_IRQ\_OP
  - L4 IPC Opcodes, [426](#)
- I4\_thread\_regs\_t, [1403](#)
- L4\_THREAD\_STATS\_OP
  - L4 IPC Opcodes, [426](#)
- I4\_thread\_stats\_time
  - Thread, [277](#)
- I4\_thread\_switch
  - Thread, [278](#)
- L4\_THREAD\_SWITCH\_OP
  - L4 IPC Opcodes, [426](#)
- I4\_thread\_vcpu\_control
  - Thread, [278](#)
- I4\_thread\_vcpu\_control\_ext
  - Thread, [279](#)
- I4\_thread\_vcpu\_control\_ext\_u
  - Thread, [280](#)



- L4\_THREAD\_VCPU\_CONTROL\_OP
  - L4 IPC Opcodes, [426](#)
- l4\_thread\_vcpu\_control\_u
  - Thread, [281](#)
- l4\_thread\_vcpu\_resume\_commit
  - Thread, [283](#)
- L4\_THREAD\_VCPU\_RESUME\_OP
  - L4 IPC Opcodes, [426](#)
- l4\_thread\_vcpu\_resume\_start
  - Thread, [284](#)
- L4\_THREAD\_X86\_GDT\_OP
  - L4 IPC Opcodes, [426](#)
- l4\_thread\_yield
  - Thread, [284](#)
- l4\_timeout
  - Timeouts, [385](#)
- l4\_timeout\_abs
  - Timeouts, [385](#)
- l4\_timeout\_get
  - Timeouts, [386](#)
- l4\_timeout\_is\_absolute
  - Timeouts, [387](#)
- l4\_timeout\_rel
  - Timeouts, [388](#)
- l4\_timeout\_rel\_get
  - Timeouts, [388](#)
- l4\_timeout\_s, [1404](#)
  - Timeouts, [383](#)
- l4\_timeout\_t, [1405](#)
  - Timeouts, [383](#)
- L4\_TIMEOUT\_US\_MAX
  - Timeouts, [383](#)
- l4\_touch\_ro
  - Utility Functions, [579](#)
- l4\_touch\_rw
  - Utility Functions, [580](#)
- l4\_trunc\_page
  - Memory related, [338](#)
- l4\_trunc\_size
  - Memory related, [339](#)
- l4\_tsc\_init
  - Timestamp Counter, [652](#)
- l4\_tsc\_to\_ns
  - Timestamp Counter, [653](#)
- l4\_tsc\_to\_s\_and\_ns
  - Timestamp Counter, [653](#)
- l4\_tsc\_to\_us
  - Timestamp Counter, [654](#)
- L4\_TYPE\_VHW\_FRAMEBUFFER
  - Fiasco-UX Virtual devices, [189](#)
- L4\_TYPE\_VHW\_INPUT
  - Fiasco-UX Virtual devices, [189](#)
- L4\_TYPE\_VHW\_NET
  - Fiasco-UX Virtual devices, [189](#)
- L4\_TYPE\_VHW\_NONE
  - Fiasco-UX Virtual devices, [189](#)
- L4\_TYPES\_FLAGS\_OPS\_DEF
  - types, [2764](#)
- l4\_unmap\_flags\_t
  - Task, [256](#)
- l4\_usleep
  - Utility Functions, [580](#)
- l4\_utcb\_br
  - Virtual Registers (UTCBs), [391](#)
- L4\_UTCB\_BUF\_REGS\_OFFSET
  - x86 Virtual Registers (UTCB), [401](#)
- L4\_utcb\_consts\_x86
  - x86 Virtual Registers (UTCB), [401](#)
- l4\_utcb\_exc
  - Exception registers, [397](#)
- l4\_utcb\_exc\_is\_ex\_regs\_exception
  - Exception registers, [397](#)
- l4\_utcb\_exc\_is\_pf
  - Exception registers, [398](#)
- l4\_utcb\_exc\_pc
  - Exception registers, [398](#)
- l4\_utcb\_exc\_pc\_set
  - Exception registers, [399](#)
- L4\_UTCB\_EXCEPTION\_REGS\_SIZE
  - x86 Virtual Registers (UTCB), [401](#)
- L4\_UTCB\_GENERIC\_BUFFERS\_SIZE
  - x86 Virtual Registers (UTCB), [401](#)
- L4\_UTCB\_GENERIC\_DATA\_SIZE
  - x86 Virtual Registers (UTCB), [401](#)
- L4\_UTCB\_INHERIT\_FPU
  - x86 Virtual Registers (UTCB), [401](#)
- l4\_utcb\_mr
  - Virtual Registers (UTCBs), [392](#)
- l4\_utcb\_mr64\_idx
  - Timeouts, [389](#)
- L4\_UTCB\_MSG\_REGS\_OFFSET
  - x86 Virtual Registers (UTCB), [401](#)
- L4\_UTCB\_OFFSET
  - x86 Virtual Registers (UTCB), [401](#)
- l4\_utcb\_t
  - Virtual Registers (UTCBs), [391](#)
- l4\_utcb\_tcr
  - Virtual Registers (UTCBs), [393](#)
- L4\_UTCB\_THREAD\_REGS\_OFFSET
  - x86 Virtual Registers (UTCB), [401](#)
- l4\_vcon\_attr\_t, [1406](#)
  - set\_raw, [1407](#)
  - Virtual Console, [298](#)
- L4\_VCON\_ECHO
  - Virtual Console, [299](#)
- l4\_vcon\_get\_attr
  - Virtual Console, [299](#)
- L4\_VCON\_GET\_ATTR\_OP
  - L4 IPC Opcodes, [427](#)
- l4\_vcon\_get\_attr\_u
  - Virtual Console, [300](#)
- L4\_vcon\_i\_flags
  - Virtual Console, [298](#)
- L4\_VCON\_ICANON
  - Virtual Console, [299](#)
- L4\_VCON\_ICRNL



- Virtual Console, [298](#)
- L4\_VCON\_IGNCR
  - Virtual Console, [298](#)
- L4\_VCON\_INLCR
  - Virtual Console, [298](#)
- L4\_vcon\_l\_flags
  - Virtual Console, [299](#)
- L4\_vcon\_o\_flags
  - Virtual Console, [299](#)
- L4\_VCON\_OCRNL
  - Virtual Console, [299](#)
- L4\_VCON\_ONLCR
  - Virtual Console, [299](#)
- L4\_VCON\_ONLRET
  - Virtual Console, [299](#)
- L4\_vcon\_ops
  - L4 IPC Opcodes, [427](#)
- l4\_vcon\_read
  - Virtual Console, [301](#)
- L4\_vcon\_read\_flags
  - vcon.h, [2936](#)
- L4\_VCON\_READ\_OP
  - L4 IPC Opcodes, [427](#)
- L4\_VCON\_READ\_SIZE
  - Virtual Console, [299](#)
- L4\_VCON\_READ\_SIZE\_MASK
  - vcon.h, [2936](#)
- L4\_VCON\_READ\_STAT\_BREAK
  - vcon.h, [2936](#)
- L4\_VCON\_READ\_STAT\_DONE
  - vcon.h, [2936](#)
- l4\_vcon\_read\_u
  - Virtual Console, [302](#)
- l4\_vcon\_read\_with\_flags
  - Virtual Console, [303](#)
- l4\_vcon\_send
  - Virtual Console, [304](#)
- l4\_vcon\_send\_u
  - Virtual Console, [305](#)
- l4\_vcon\_set\_attr
  - Virtual Console, [306](#)
- L4\_VCON\_SET\_ATTR\_OP
  - L4 IPC Opcodes, [427](#)
- l4\_vcon\_set\_attr\_raw
  - Virtual Console, [307](#)
- l4\_vcon\_set\_attr\_u
  - Virtual Console, [308](#)
- L4\_vcon\_size\_consts
  - Virtual Console, [299](#)
- l4\_vcon\_write
  - Virtual Console, [309](#)
- L4\_VCON\_WRITE\_OP
  - L4 IPC Opcodes, [427](#)
- L4\_VCON\_WRITE\_SIZE
  - Virtual Console, [299](#)
- l4\_vcon\_write\_u
  - Virtual Console, [309](#)
- l4\_vcpu\_arch\_state\_t, [1407](#)
- l4\_vcpu\_check\_version
  - vcpu.h, [2941](#)
- L4\_vcpu\_e\_field\_ids
  - \_\_vcpu-arch.h, [2097](#)
- L4\_VCPU\_F\_EXCEPTIONS
  - vCPU API, [295](#)
- L4\_VCPU\_F\_FPU\_ENABLED
  - vCPU API, [295](#)
- L4\_VCPU\_F\_IRQ
  - vCPU API, [294](#)
- L4\_VCPU\_F\_PAGE\_FAULTS
  - vCPU API, [294](#)
- L4\_VCPU\_F\_USER\_MODE
  - vCPU API, [295](#)
- l4\_vcpu\_ipc\_regs\_t, [1408](#)
- L4\_VCPU\_OFFSET\_EXT\_INFOS
  - vCPU API, [295](#), [296](#)
- L4\_VCPU\_OFFSET\_EXT\_STATE
  - vCPU API, [295](#), [296](#)
- l4\_vcpu\_regs\_t, [1409](#)
  - ax, [1411](#)
  - bp, [1411](#)
  - bx, [1411](#)
  - cx, [1411](#)
  - di, [1411](#)
  - dx, [1412](#)
  - si, [1412](#)
- L4\_VCPU\_SF\_IRQ\_PENDING
  - vCPU API, [296](#)
- L4\_vcpu\_state\_flags
  - vCPU API, [294](#)
- L4\_vcpu\_state\_offset
  - vCPU API, [295](#), [296](#)
- l4\_vcpu\_state\_t, [1413](#)
  - version, [1415](#)
- L4\_VCPU\_STATE\_VERSION
  - \_\_vcpu-arch.h, [2094](#), [2097](#), [2100](#)
- L4\_vcpu\_sticky\_flags
  - vCPU API, [296](#)
- l4\_vhw\_descriptor, [1416](#)
- l4\_vhw\_entry, [1417](#)
- l4\_vhw\_entry\_type
  - Fiasco-UX Virtual devices, [189](#)
- L4\_virtio\_cmd
  - L4 VIRTIO Transport Layer, [433](#)
- L4\_virtio\_irq\_status
  - L4 VIRTIO Transport Layer, [433](#)
- L4\_virtio\_opcodes
  - L4 VIRTIO Transport Layer, [434](#)
- l4\_vm\_svm\_vmcb\_control\_area, [1418](#)
- l4\_vm\_svm\_vmcb\_state\_save\_area, [1419](#)
- l4\_vm\_svm\_vmcb\_state\_save\_area\_seg, [1420](#)
- l4\_vm\_svm\_vmcb\_t, [1420](#)
- l4\_vm\_tz\_state, [1421](#)
- L4\_VM\_VMX\_BASIC\_REG
  - VM API for VMX, [316](#)
- L4\_vm\_vmx\_caps\_regs
  - VM API for VMX, [316](#)

- `l4_vm_vmx_clear`  
VM API for VMX, [317](#)
- `L4_VM_VMX_CR0_FIXED0_REG`  
VM API for VMX, [316](#)
- `L4_VM_VMX_CR0_FIXED1_REG`  
VM API for VMX, [316](#)
- `L4_VM_VMX_CR4_FIXED0_REG`  
VM API for VMX, [316](#)
- `L4_VM_VMX_CR4_FIXED1_REG`  
VM API for VMX, [316](#)
- `L4_vm_vmx_dfl1_regs`  
VM API for VMX, [316](#)
- `L4_VM_VMX_ENTRY_CTL5_DFL1_REG`  
VM API for VMX, [316](#)
- `L4_VM_VMX_EPT_VPID_CAP_REG`  
VM API for VMX, [316](#)
- `L4_VM_VMX_EXIT_CTL5_DFL1_REG`  
VM API for VMX, [316](#)
- `l4_vm_vmx_field_len`  
VM API for VMX, [318](#)
- `l4_vm_vmx_field_order`  
VM API for VMX, [319](#)
- `l4_vm_vmx_get_caps`  
VM API for VMX, [319](#)
- `l4_vm_vmx_get_caps_default1`  
VM API for VMX, [320](#)
- `l4_vm_vmx_get_cr2_index`  
VM API for VMX, [320](#)
- `l4_vm_vmx_get_hw_vmcs`  
VM API for VMX, [321](#)
- `L4_VM_VMX_MISC_REG`  
VM API for VMX, [316](#)
- `L4_VM_VMX_NESTED_REVISION`  
VM API for VMX, [316](#)
- `L4_VM_VMX_NUM_CAPS_REGS`  
VM API for VMX, [316](#)
- `L4_VM_VMX_NUM_DFL1_REGS`  
VM API for VMX, [316](#)
- `L4_VM_VMX_PINBASED_CTL5_DFL1_REG`  
VM API for VMX, [316](#)
- `L4_VM_VMX_PROCBASED_CTL5_2_REG`  
VM API for VMX, [316](#)
- `L4_VM_VMX_PROCBASED_CTL5_DFL1_REG`  
VM API for VMX, [316](#)
- `l4_vm_vmx_ptr_load`  
VM API for VMX, [321](#)
- `l4_vm_vmx_read`  
VM API for VMX, [322](#)
- `l4_vm_vmx_read_16`  
VM API for VMX, [323](#)
- `l4_vm_vmx_read_32`  
VM API for VMX, [323](#)
- `l4_vm_vmx_read_64`  
VM API for VMX, [324](#)
- `l4_vm_vmx_read_nat`  
VM API for VMX, [325](#)
- `l4_vm_vmx_set_hw_vmcs`  
VM API for VMX, [325](#)
- `L4_vm_vmx_sw_fields`  
VM API for VMX, [316](#)
- `L4_VM_VMX_TRUE_ENTRY_CTL5_REG`  
VM API for VMX, [316](#)
- `L4_VM_VMX_TRUE_EXIT_CTL5_REG`  
VM API for VMX, [316](#)
- `L4_VM_VMX_TRUE_PINBASED_CTL5_REG`  
VM API for VMX, [316](#)
- `L4_VM_VMX_TRUE_PROCBASED_CTL5_REG`  
VM API for VMX, [316](#)
- `L4_VM_VMX_VMCS_CR2`  
VM API for VMX, [317](#)
- `L4_VM_VMX_VMCS_ENUM_REG`  
VM API for VMX, [316](#)
- `L4_VM_VMX_VMCS_MSR_CSTAR`  
VM API for VMX, [317](#)
- `L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE`  
VM API for VMX, [317](#)
- `L4_VM_VMX_VMCS_MSR_LSTAR`  
VM API for VMX, [317](#)
- `L4_VM_VMX_VMCS_MSR_STAR`  
VM API for VMX, [317](#)
- `L4_VM_VMX_VMCS_MSR_SYSCALL_MASK`  
VM API for VMX, [317](#)
- `L4_VM_VMX_VMCS_MSR_TSC_AUX`  
VM API for VMX, [317](#)
- `L4_VM_VMX_VMCS_XCR0`  
VM API for VMX, [317](#)
- `l4_vm_vmx_write`  
VM API for VMX, [326](#)
- `l4_vm_vmx_write_16`  
VM API for VMX, [327](#)
- `l4_vm_vmx_write_32`  
VM API for VMX, [328](#)
- `l4_vm_vmx_write_64`  
VM API for VMX, [328](#)
- `l4_vm_vmx_write_nat`  
VM API for VMX, [329](#)
- `l4_vmx_offset_table_t`, [1422](#)  
VM API for VMX, [315](#)
- `L4_WHOLE_ADDRESS_SPACE`  
Flex pages, [165](#)
- `L4_WHOLE_IOADDRESS_SPACE`  
Flex pages, [165](#)
- `L4drivers::Mmio_register_block< MAX_BITS >`, [1424](#)
- `L4drivers::Register_block< MAX_BITS, BLOCK >`,  
[1425](#)  
operator[], [1426](#), [1427](#)  
r, [1427](#), [1428](#)
- `L4drivers::Register_block_base< MAX_BITS >`, [1428](#)
- `L4drivers::Register_block_impl< BASE, MAX_BITS >`,  
[1429](#)
- `L4drivers::Register_block_tmpl< BLOCK >`, [1431](#)
- `L4drivers::Register_tmpl< BITS, BLOCK >`, [1432](#)  
clear, [1434](#)  
modify, [1434](#)  
operator=, [1435](#)  
set, [1435](#)

- write, [1435](#)
- L4drivers::Ro\_register\_block< MAX\_BITS, BLOCK >, [1436](#)
  - operator[], [1437](#)
  - r, [1437](#)
- L4drivers::Ro\_register\_tmpl< BITS, BLOCK >, [1438](#)
  - operator value\_type, [1439](#)
  - read, [1439](#)
- L4IO\_DEVICE\_ANY
  - IO interface, [407](#)
- L4IO\_DEVICE\_INVALID
  - IO interface, [407](#)
- L4IO\_DEVICE\_OTHER
  - IO interface, [407](#)
- L4IO\_DEVICE\_PCI
  - IO interface, [407](#)
- l4io\_device\_types\_t
  - IO interface, [406](#)
- L4IO\_DEVICE\_USB
  - IO interface, [407](#)
- l4io\_get\_root\_device
  - io.h, [2140](#)
- l4io\_has\_resource
  - IO interface, [407](#)
- l4io\_iomem\_flags\_t
  - IO interface, [407](#)
- l4io\_iterate\_devices
  - io.h, [2140](#)
- l4io\_lookup\_device
  - IO interface, [408](#)
- l4io\_lookup\_resource
  - IO interface, [408](#)
- L4IO\_MEM\_CACHED
  - IO interface, [407](#)
- L4IO\_MEM\_EAGER\_MAP
  - IO interface, [407](#)
- L4IO\_MEM\_NONCACHED
  - IO interface, [407](#)
- L4IO\_MEM\_USE\_MTRR
  - IO interface, [407](#)
- L4IO\_MEM\_USE\_RESERVED\_AREA
  - IO interface, [407](#)
- l4io\_release\_iomem
  - IO interface, [409](#)
- l4io\_release\_ioport
  - IO interface, [409](#)
- l4io\_request\_all\_ioports
  - io.h, [2140](#)
- l4io\_request\_icu
  - io.h, [2141](#)
- l4io\_request\_iomem
  - IO interface, [409](#)
- l4io\_request\_iomem\_region
  - IO interface, [410](#)
- l4io\_request\_ioport
  - IO interface, [411](#)
- l4io\_request\_resource\_iomem
  - IO interface, [411](#)
- L4IO\_RESOURCE\_ANY
  - IO interface, [407](#)
- L4IO\_RESOURCE\_INVALID
  - IO interface, [407](#)
- L4IO\_RESOURCE\_IRQ
  - IO interface, [407](#)
- L4IO\_RESOURCE\_MEM
  - IO interface, [407](#)
- L4IO\_RESOURCE\_PORT
  - IO interface, [407](#)
- l4io\_resource\_t
  - IO interface, [406](#)
- l4io\_resource\_types\_t
  - IO interface, [407](#)
- l4irq\_attach
  - Interface using direct functionality., [418](#)
- l4irq\_attach\_cap
  - Interface using direct functionality., [422](#)
- l4irq\_attach\_cap\_ft
  - Interface using direct functionality., [422](#)
- l4irq\_attach\_ft
  - Interface using direct functionality., [418](#)
- l4irq\_attach\_thread
  - Interface using direct functionality., [418](#)
- l4irq\_attach\_thread\_cap
  - Interface using direct functionality., [422](#)
- l4irq\_attach\_thread\_cap\_ft
  - Interface using direct functionality., [423](#)
- l4irq\_attach\_thread\_ft
  - Interface using direct functionality., [419](#)
- l4irq\_detach
  - Interface using direct functionality., [419](#)
- l4irq\_release
  - Interface for asynchronous ISR handlers., [415](#)
- l4irq\_request
  - Interface for asynchronous ISR handlers., [415](#)
- l4irq\_request\_cap
  - Interface for asynchronous ISR handlers with a given IRQ capability., [416](#)
- l4irq\_unmask
  - Interface using direct functionality., [420](#)
- l4irq\_unmask\_and\_wait\_any
  - Interface using direct functionality., [420](#)
- l4irq\_wait
  - Interface using direct functionality., [420](#)
- l4irq\_wait\_any
  - Interface using direct functionality., [421](#)
- l4la\_alloc
  - list\_alloc.h, [2983](#)
- l4la\_avail
  - list\_alloc.h, [2983](#)
- l4la\_dump
  - list\_alloc.h, [2983](#)
- l4la\_free
  - list\_alloc.h, [2983](#)
- l4la\_init
  - list\_alloc.h, [2984](#)
- l4mod.h

- L4util\_l4mod\_mod\_flag\_kernel, 2981
- L4util\_l4mod\_mod\_flag\_mask, 2981
- L4util\_l4mod\_mod\_flag\_roottask, 2981
- L4util\_l4mod\_mod\_flag\_sigma0, 2981
- L4util\_l4mod\_mod\_flag\_unspec, 2981
- L4util\_l4mod\_mod\_info\_flag, 2981
- L4Re, 693
  - chkcapp, 698
  - chkipc, 699
  - chksys, 700–702
  - make\_shared\_cap, 704
  - make\_shared\_del\_cap, 705
  - make\_unique\_cap, 706
  - make\_unique\_del\_cap, 706
  - Shared\_cap, 695
  - shared\_cap, 695
  - Shared\_del\_cap, 696
  - shared\_del\_cap, 696
  - throw\_error, 707
  - Unique\_cap, 697
  - unique\_cap, 697
  - Unique\_del\_cap, 697
  - unique\_del\_cap, 698
- L4Re Build System, 28
- L4Re C Interface, 467
- L4Re C++ Interface, 520
- L4Re Capability API, 531
  - cap\_alloc, 533
  - make\_ref\_cap, 532
  - make\_ref\_del\_cap, 533
- L4Re ELF Auxiliary Information, 526
  - L4RE\_ELF\_AUX\_ELEM, 527
  - L4RE\_ELF\_AUX\_ELEM\_T, 527
  - L4RE\_ELF\_AUX\_T\_EX\_REGS\_FLAGS, 528
  - L4RE\_ELF\_AUX\_T\_KIP\_ADDR, 528
  - L4RE\_ELF\_AUX\_T\_NONE, 528
  - L4RE\_ELF\_AUX\_T\_STACK\_ADDR, 528
  - L4RE\_ELF\_AUX\_T\_STACK\_SIZE, 528
  - L4RE\_ELF\_AUX\_T\_VMA, 528
- L4Re Protocol identifiers, 528
- L4Re Servers, 43
- L4Re Util C Interface, 488
- L4Re Util C++ Interface, 529
- L4Re::Cap\_alloc, 1440
  - alloc, 1441
  - free, 1442
  - get\_cap\_alloc, 1442
- L4Re::Console, 1443
- L4Re::Dataspace, 1446
  - allocate, 1450
  - clear, 1451
  - copy\_in, 1451
  - flags, 1452
  - info, 1453
  - map, 1453
  - map\_info, 1454
  - map\_region, 1455
  - size, 1456
- L4Re::Dataspace::F, 1457
  - Bufferable, 1458
  - Cacheable, 1458
  - Caching\_mask, 1458
  - Caching\_shift, 1457
  - Flags, 1457
  - Normal, 1458
  - R, 1458
  - Rights\_mask, 1458
  - Ro, 1458
  - RW, 1458
  - RWX, 1458
  - RX, 1458
  - Uncacheable, 1458
  - W, 1458
  - X, 1458
- L4Re::Dataspace::Stats, 1458
- L4Re::Debug\_obj, 1459
  - debug, 1461
- L4Re::Default\_event\_payload, 1462
- L4Re::Dma\_space, 1463
  - associate, 1465
  - Attribute, 1464
  - Attributes, 1464
  - Bidirectional, 1465
  - Coherent, 1465
  - Direction, 1465
  - disassociate, 1466
  - From\_device, 1465
  - map, 1466
  - No\_sync, 1465
  - None, 1465
  - Phys\_space, 1465
  - Space\_attr, 1465
  - To\_device, 1465
  - unmap, 1468
- L4Re::Env, 1469
  - env, 1472
  - factory, 1472
  - first\_free\_cap, 1473
  - first\_free\_utcb, 1473
  - get, 1474
  - get\_cap, 1474, 1476
  - initial\_caps, 1476
  - log, 1477
  - main\_thread, 1477
  - mem\_alloc, 1478
  - parent, 1478, 1479
  - rm, 1479
  - scheduler, 1480
  - task, 1480
  - utcb\_area, 1480, 1481
- L4Re::Event, 1481
  - get\_axis\_info, 1485
  - get\_buffer, 1486
  - get\_num\_streams, 1486
  - get\_stream\_info, 1486
  - get\_stream\_info\_for\_id, 1487

- get\_stream\_state\_for\_id, 1487
- L4Re::Event\_buffer\_t< PAYLOAD >, 1488
  - Event\_buffer\_t, 1490
  - next, 1491
  - put, 1491
- L4Re::Event\_buffer\_t< PAYLOAD >::Event, 1492
- L4Re::Inhibitor, 1493
  - acquire, 1497
  - Name\_max, 1497
  - next\_lock\_info, 1497
  - release, 1498
- L4Re::Log, 1498
  - print, 1503
  - println, 1503
- L4Re::Mem\_alloc, 1503
  - alloc, 1508
  - Continuous, 1508
  - Fixed\_paddr, 1508
  - Mem\_alloc\_flags, 1507
  - Pinned, 1508
  - Super\_pages, 1508
- L4Re::Mmio\_space, 1509
  - Access\_width, 1511
  - mmio\_read, 1512
  - mmio\_write, 1512
  - Wd\_16bit, 1512
  - Wd\_32bit, 1512
  - Wd\_64bit, 1512
  - Wd\_8bit, 1512
- L4Re::Namespace, 1513
  - Link, 1517
  - Overwrite, 1517
  - Partly\_resolved, 1516
  - query, 1517
  - Query\_result\_flags, 1516
  - Query\_timeout, 1516
  - Register\_flags, 1516
  - register\_obj, 1518
  - Ro, 1516
  - Rs, 1517
  - Rw, 1517
  - Rws, 1517
  - Strong, 1517
  - To\_default, 1516
  - To\_non\_blocking, 1516
  - Trusted, 1517
  - unlink, 1519
- L4Re::Ned::Cmd\_control, 1520
  - execute, 1521, 1522
- L4Re::Parent, 1522
  - signal, 1525
- L4Re::Random, 1526
  - get\_random, 1530
- L4Re::Rm, 1531
  - Area, 1536
  - attach, 1537, 1538
  - Caching\_shift, 1537
  - detach, 1539–1541
  - Detach\_again, 1537
  - Detach\_exact, 1537
  - Detach\_flags, 1536
  - Detach\_keep, 1537
  - Detach\_overlap, 1537
  - Detach\_result, 1537
  - Detached\_ds, 1537
  - find, 1541
  - free\_area, 1542
  - get\_areas, 1543
  - get\_regions, 1543
  - Kept\_ds, 1537
  - Region, 1536
  - Region\_flag\_shifts, 1537
  - reserve\_area, 1544
  - Split\_ds, 1537
- L4Re::Rm::F, 1545
  - Attach\_flags, 1546
  - Attach\_mask, 1546
  - Cache\_buffered, 1547
  - Cache\_normal, 1547
  - Cache\_uncached, 1547
  - Caching\_mask, 1547
  - Detach\_free, 1547
  - Ds\_map\_mask, 1547
  - Eager\_map, 1546
  - In\_area, 1546
  - No\_eager\_map, 1546
  - Pager, 1547
  - R, 1546
  - Region\_flags, 1546
  - Region\_flags\_mask, 1547
  - Reserved, 1547
  - Rights\_mask, 1546
  - RW, 1547
  - RWX, 1547
  - RX, 1547
  - Search\_addr, 1546
  - W, 1547
  - X, 1547
- L4Re::Rm::Range, 1547
- L4Re::Rm::Unique\_region< T >, 1548
  - ~Unique\_region, 1551
  - get, 1551
  - is\_valid, 1552
  - operator=, 1552
  - release, 1553
  - reset, 1553
  - Unique\_region, 1550, 1551
- L4Re::Smart\_cap\_auto< Unmap\_flags >, 1554
- L4Re::Smart\_count\_cap< Unmap\_flags >, 1555
- L4Re::Util, 709
  - make\_shared\_cap, 716
  - make\_shared\_del\_cap, 716
  - make\_unique\_cap, 716
  - make\_unique\_del\_cap, 718
  - Shared\_cap, 711
  - shared\_cap, 711

- Shared\_del\_cap, 712
- shared\_del\_cap, 713
- Unique\_cap, 713
- unique\_cap, 714
- Unique\_del\_cap, 714
- unique\_del\_cap, 715
- L4Re::Util::Br\_manager, 1556
  - alloc\_buffer\_demand, 1559
  - get\_rcv\_cap, 1560
  - realloc\_rcv\_cap, 1560
  - set\_rcv\_cap\_flags, 1561
- L4Re::Util::Br\_manager\_hooks, 1562
- L4Re::Util::Br\_manager\_timeout\_hooks, 1564
- L4Re::Util::Cap\_alloc\_base, 1568
- L4Re::Util::Counter< COUNTER >, 1569
  - dec, 1569
  - inc, 1569
- L4Re::Util::Counter\_atomic< COUNTER >, 1571
  - dec, 1572
  - inc, 1572
- L4Re::Util::Counting\_cap\_alloc< COUNTERTYPE, Dbg >, 1572
  - alloc, 1574, 1575
  - Counting\_cap\_alloc, 1574
  - free, 1575
  - release, 1576
  - setup, 1577
  - take, 1577
- L4Re::Util::Dataspace\_svr, 1578
  - allocate, 1579
  - clear, 1580
  - copy, 1580
  - is\_static, 1581
  - map, 1581
  - map\_hook, 1582
  - map\_info, 1583
  - page\_shift, 1583
  - release, 1583
  - take, 1584
- L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >, 1584
  - foreach\_available\_event, 1587
  - process, 1588
- L4Re::Util::Event\_buffer\_t< PAYLOAD >, 1589
  - attach, 1592
  - buf, 1592
  - detach, 1593
- L4Re::Util::Event\_svr< SVR >, 1593
- L4Re::Util::Event\_t< PAYLOAD >, 1595
  - buffer, 1596
  - init, 1597
  - init\_poll, 1598
  - irq, 1598
  - Mode, 1596
  - Mode\_irq, 1596
  - Mode\_polling, 1596
- L4Re::Util::Item\_alloc\_base, 1599
- L4Re::Util::Names::Name, 1599
- L4Re::Util::Names::Name\_space, 1603
  - alloc\_dynamic\_entry, 1604
  - copy\_receive\_cap, 1604
  - free\_capability, 1605
  - free\_dynamic\_entry, 1605
  - free\_epiface, 1606
  - get\_epiface, 1607
- L4Re::Util::Object\_registry, 1607
  - Object\_registry, 1610
  - register\_irq\_obj, 1611
  - register\_obj, 1611, 1612
  - unregister\_obj, 1613
- L4Re::Util::Ref\_cap< T >, 1614
- L4Re::Util::Ref\_del\_cap< T >, 1615
- L4Re::Util::Registry\_server< LOOP\_HOOKS >, 1616
  - loop, 1620
  - Registry\_server, 1619, 1620
- L4Re::Util::Smart\_cap\_auto< Unmap\_flags >, 1621
- L4Re::Util::Smart\_count\_cap< Unmap\_flags >, 1622
- L4Re::Util::Vcon\_svr< SVR >, 1623
- L4Re::Util::Video::Goos\_svr, 1624
  - get\_fb, 1625
  - init\_infos, 1625
  - refresh, 1626
  - screen\_info, 1626
  - view\_info, 1626
- L4Re::Vfs, 718
- L4Re::Vfs::Be\_file, 1627
  - data\_space, 1630
  - fstat64, 1630
  - unlock\_all\_locks, 1630
- L4Re::Vfs::Be\_file\_system, 1631
  - ~Be\_file\_system, 1633
  - Be\_file\_system, 1633
  - type, 1633
- L4Re::Vfs::Directory, 1634
  - faccessat, 1636
  - link, 1636
  - mkdir, 1636
  - rename, 1637
  - rmdir, 1637
  - symlink, 1638
  - unlink, 1638
- L4Re::Vfs::File, 1639
- L4Re::Vfs::File\_system, 1642
  - mount, 1643
  - type, 1643
- L4Re::Vfs::Fs, 1644
  - alloc\_fd, 1646
  - free\_fd, 1646
  - get\_file, 1646
  - mount, 1647
  - set\_fd, 1647
- L4Re::Vfs::Generic\_file, 1648
  - fchmod, 1649
  - fstat64, 1649
  - get\_status\_flags, 1650
  - set\_status\_flags, 1650

- unlock\_all\_locks, 1651
- L4Re::Vfs::Mman, 1651
- L4Re::Vfs::Ops, 1653
- L4Re::Vfs::Regular\_file, 1656
  - data\_space, 1659
  - fdatsync, 1659
  - fsync, 1659
  - ftruncate64, 1659
  - get\_lock, 1660
  - lseek64, 1660
  - readv, 1660
  - set\_lock, 1661
  - writev, 1661
- L4Re::Vfs::Special\_file, 1662
  - ioctl, 1663
- L4Re::Video::Color\_component, 1664
  - Color\_component, 1665
  - dump, 1665
  - get, 1665
  - operator==, 1666
  - set, 1666
  - shift, 1666
  - size, 1667
- L4Re::Video::Goos, 1668
  - create\_buffer, 1671
  - create\_view, 1672
  - delete\_buffer, 1672
  - delete\_view, 1672
  - F\_auto\_refresh, 1671
  - F\_dynamic\_buffers, 1671
  - F\_dynamic\_views, 1671
  - F\_pointer, 1671
  - Flags, 1671
  - get\_static\_buffer, 1673
  - info, 1673
  - view, 1674
- L4Re::Video::Goos::Info, 1674
- L4Re::Video::Pixel\_info, 1676
  - a, 1678, 1679
  - b, 1679
  - bits\_per\_pixel, 1679
  - bytes\_per\_pixel, 1680
  - dump, 1681
  - g, 1681
  - has\_alpha, 1682
  - operator==, 1682
  - padding, 1682
  - Pixel\_info, 1678
  - r, 1683
- L4Re::Video::View, 1684
  - F\_above, 1686
  - F\_dyn\_allocated, 1685
  - F\_flags\_mask, 1686
  - F\_fully\_dynamic, 1685
  - F\_none, 1685
  - F\_set\_background, 1685
  - F\_set\_buffer, 1685
  - F\_set\_buffer\_offset, 1685
  - F\_set\_bytes\_per\_line, 1685
  - F\_set\_flags, 1685
  - F\_set\_pixel, 1685
  - F\_set\_position, 1685
  - Flags, 1685
  - info, 1686
  - refresh, 1686
  - set\_info, 1687
  - set\_viewport, 1687
  - stack, 1688
  - V\_flags, 1686
- L4Re::Video::View::Info, 1688
- l4re\_aux\_t, 1691
- l4re\_debug\_obj\_debug
  - Debug interface, 479
- l4re\_dma\_space\_associate
  - DMA Space Interface, 471
- L4RE\_DMA\_SPACE\_BIDIRECTIONAL
  - dma\_space.h, 2440
- L4RE\_DMA\_SPACE\_COHERENT
  - dma\_space.h, 2441
- l4re\_dma\_space\_direction
  - dma\_space.h, 2440
- l4re\_dma\_space\_disassociate
  - DMA Space Interface, 472
- L4RE\_DMA\_SPACE\_FROM\_DEVICE
  - dma\_space.h, 2440
- l4re\_dma\_space\_map
  - DMA Space Interface, 472
- L4RE\_DMA\_SPACE\_NONE
  - dma\_space.h, 2440
- L4RE\_DMA\_SPACE\_PHYS\_SPACE
  - dma\_space.h, 2441
- l4re\_dma\_space\_space\_attribs
  - dma\_space.h, 2440
- l4re\_dma\_space\_t
  - DMA Space Interface, 471
- L4RE\_DMA\_SPACE\_TO\_DEVICE
  - dma\_space.h, 2440
- l4re\_dma\_space\_unmap
  - DMA Space Interface, 473
- l4re\_ds\_allocate
  - Dataspace interface, 475
- l4re\_ds\_clear
  - Dataspace interface, 476
- l4re\_ds\_copy\_in
  - Dataspace interface, 476
- L4RE\_DS\_F\_BUFFERABLE
  - Dataspace interface, 475
- L4RE\_DS\_F\_CACHEABLE
  - Dataspace interface, 475
- L4RE\_DS\_F\_CACHING\_MASK
  - Dataspace interface, 475
- L4RE\_DS\_F\_CACHING\_SHIFT
  - Dataspace interface, 475
- L4RE\_DS\_F\_NORMAL
  - Dataspace interface, 475
- L4RE\_DS\_F\_UNCACHEABLE



- Dataspace interface, [475](#)
- l4re\_ds\_flags
  - Dataspace interface, [477](#)
- l4re\_ds\_info
  - Dataspace interface, [477](#)
- l4re\_ds\_map\_flags
  - Dataspace interface, [475](#)
- l4re\_ds\_map\_info
  - Dataspace interface, [478](#)
- l4re\_ds\_size
  - Dataspace interface, [478](#)
- l4re\_ds\_stats\_t, [1692](#)
- L4RE\_ELF\_AUX\_ELEM
  - L4Re ELF Auxiliary Information, [527](#)
- L4RE\_ELF\_AUX\_ELEM\_T
  - L4Re ELF Auxiliary Information, [527](#)
- l4re\_elf\_aux\_mword\_t, [1692](#)
- l4re\_elf\_aux\_t, [1693](#)
- L4RE\_ELF\_AUX\_T\_EX\_REGS\_FLAGS
  - L4Re ELF Auxiliary Information, [528](#)
- L4RE\_ELF\_AUX\_T\_KIP\_ADDR
  - L4Re ELF Auxiliary Information, [528](#)
- L4RE\_ELF\_AUX\_T\_NONE
  - L4Re ELF Auxiliary Information, [528](#)
- L4RE\_ELF\_AUX\_T\_STACK\_ADDR
  - L4Re ELF Auxiliary Information, [528](#)
- L4RE\_ELF\_AUX\_T\_STACK\_SIZE
  - L4Re ELF Auxiliary Information, [528](#)
- L4RE\_ELF\_AUX\_T\_VMA
  - L4Re ELF Auxiliary Information, [528](#)
- l4re\_elf\_aux\_vma\_t, [1693](#)
- l4re\_env
  - Initial Environment, [484](#)
- l4re\_env\_cap\_entry\_t, [1694](#)
  - flags, [1695](#)
  - l4re\_env\_cap\_entry\_t, [1695](#)
- l4re\_env\_get\_cap
  - Initial Environment, [484](#)
- l4re\_env\_get\_cap\_e
  - Initial Environment, [485](#)
- l4re\_env\_get\_cap\_l
  - Initial Environment, [486](#)
- l4re\_env\_t, [1696](#)
  - caps, [1697](#)
  - env.h, [2505](#)
- l4re\_event\_get\_axis\_info
  - Event interface, [480](#)
- l4re\_event\_get\_buffer
  - Event interface, [481](#)
- l4re\_event\_get\_num\_streams
  - Event interface, [481](#)
- l4re\_event\_get\_stream\_info
  - Event interface, [482](#)
- l4re\_event\_get\_stream\_info\_for\_id
  - Event interface, [482](#)
- l4re\_event\_t, [1698](#)
- l4re\_inhibitor\_acquire
  - inhibitor.h, [2443](#)
- l4re\_inhibitor\_next\_lock\_info
  - inhibitor.h, [2444](#)
- l4re\_inhibitor\_release
  - inhibitor.h, [2444](#)
- l4re\_kip
  - Initial Environment, [487](#)
- l4re\_log\_print
  - Log interface, [489](#)
- l4re\_log\_print\_srv
  - Log interface, [489](#)
- l4re\_log\_printn
  - Log interface, [490](#)
- l4re\_log\_printn\_srv
  - Log interface, [491](#)
- l4re\_ma\_alloc
  - Memory allocator, [492](#)
- l4re\_ma\_alloc\_align
  - Memory allocator, [493](#)
- l4re\_ma\_alloc\_align\_srv
  - Memory allocator, [494](#)
- l4re\_ma\_flags
  - Memory allocator, [492](#)
- l4re\_ns\_query\_srv
  - Namespace interface, [496](#)
- l4re\_ns\_query\_to\_srv
  - Namespace interface, [497](#)
- l4re\_ns\_register\_flags
  - Namespace interface, [496](#)
- l4re\_ns\_register\_obj\_srv
  - Namespace interface, [498](#)
- l4re\_parent\_signal
  - parent.h, [2452](#)
- L4RE\_PROTO\_DATASPACE
  - protocols.h, [2554](#)
- L4RE\_PROTO\_DEBUG
  - protocols.h, [2554](#)
- L4RE\_PROTO\_DMA\_SPACE
  - protocols.h, [2554](#)
- L4RE\_PROTO\_EVENT
  - protocols.h, [2554](#)
- L4RE\_PROTO\_GOOS
  - protocols.h, [2554](#)
- L4RE\_PROTO\_INHIBITOR
  - protocols.h, [2554](#)
- L4RE\_PROTO\_MMIO\_SPACE
  - protocols.h, [2554](#)
- L4RE\_PROTO\_NAMESPACE
  - protocols.h, [2554](#)
- L4RE\_PROTO\_PARENT
  - protocols.h, [2554](#)
- L4RE\_PROTO\_RM
  - protocols.h, [2554](#)
- L4RE\_PROTO\_RSVD\_1
  - protocols.h, [2554](#)
- l4re\_protocols
  - protocols.h, [2554](#)
- l4re\_rm\_attach
  - Region map interface, [501](#)



- [l4re\\_rm\\_attach\\_srv](#)
  - Region map interface, [502](#)
- [L4RE\\_RM\\_CACHING\\_SHIFT](#)
  - Region map interface, [501](#)
- [l4re\\_rm\\_detach](#)
  - Region map interface, [503](#)
- [l4re\\_rm\\_detach\\_ds](#)
  - Region map interface, [504](#)
- [l4re\\_rm\\_detach\\_ds\\_unmap](#)
  - Region map interface, [505](#)
- [l4re\\_rm\\_detach\\_srv](#)
  - Region map interface, [506](#)
- [l4re\\_rm\\_detach\\_unmap](#)
  - Region map interface, [506](#)
- [L4RE\\_RM\\_F\\_ATTACH\\_FLAGS](#)
  - Region map interface, [501](#)
- [L4RE\\_RM\\_F\\_CACHE\\_BUFFERED](#)
  - Region map interface, [501](#)
- [L4RE\\_RM\\_F\\_CACHE\\_NORMAL](#)
  - Region map interface, [501](#)
- [L4RE\\_RM\\_F\\_CACHE\\_UNCACHED](#)
  - Region map interface, [501](#)
- [L4RE\\_RM\\_F\\_CACHING](#)
  - Region map interface, [501](#)
- [L4RE\\_RM\\_F\\_EAGER\\_MAP](#)
  - Region map interface, [501](#)
- [L4RE\\_RM\\_F\\_IN\\_AREA](#)
  - Region map interface, [501](#)
- [L4RE\\_RM\\_F\\_NO\\_ALIAS](#)
  - Region map interface, [500](#)
- [L4RE\\_RM\\_F\\_NO\\_EAGER\\_MAP](#)
  - Region map interface, [501](#)
- [L4RE\\_RM\\_F\\_PAGER](#)
  - Region map interface, [500](#)
- [L4RE\\_RM\\_F\\_R](#)
  - Region map interface, [500](#)
- [L4RE\\_RM\\_F\\_RESERVED](#)
  - Region map interface, [501](#)
- [L4RE\\_RM\\_F\\_SEARCH\\_ADDR](#)
  - Region map interface, [501](#)
- [l4re\\_rm\\_find](#)
  - Region map interface, [507](#)
- [l4re\\_rm\\_find\\_srv](#)
  - Region map interface, [508](#)
- [l4re\\_rm\\_flags\\_values](#)
  - Region map interface, [500](#)
- [l4re\\_rm\\_free\\_area](#)
  - Region map interface, [509](#)
- [l4re\\_rm\\_free\\_area\\_srv](#)
  - Region map interface, [510](#)
- [L4RE\\_RM\\_REGION\\_FLAGS](#)
  - Region map interface, [501](#)
- [l4re\\_rm\\_reserve\\_area](#)
  - Region map interface, [510](#)
- [l4re\\_rm\\_reserve\\_area\\_srv](#)
  - Region map interface, [511](#)
- [l4re\\_rm\\_show\\_lists](#)
  - Region map interface, [512](#)
- [l4re\\_util\\_cap\\_last](#)
  - Capability allocator, [470](#)
- [l4re\\_util\\_kumem\\_alloc](#)
  - kumem\_alloc.h, [2459](#)
- [l4re\\_video\\_color\\_component\\_t](#), [1699](#)
- [l4re\\_video\\_goos\\_create\\_buffer](#)
  - Video API, [515](#)
- [l4re\\_video\\_goos\\_create\\_view](#)
  - Video API, [516](#)
- [l4re\\_video\\_goos\\_delete\\_buffer](#)
  - Video API, [516](#)
- [l4re\\_video\\_goos\\_delete\\_view](#)
  - Video API, [516](#)
- [l4re\\_video\\_goos\\_get\\_static\\_buffer](#)
  - Video API, [516](#)
- [l4re\\_video\\_goos\\_get\\_view](#)
  - Video API, [517](#)
- [l4re\\_video\\_goos\\_info](#)
  - Video API, [517](#)
- [l4re\\_video\\_goos\\_info\\_flags\\_t](#)
  - Video API, [514](#)
- [l4re\\_video\\_goos\\_info\\_t](#), [1699](#)
- [l4re\\_video\\_goos\\_refresh](#)
  - Video API, [518](#)
- [l4re\\_video\\_pixel\\_info\\_t](#), [1701](#)
- [l4re\\_video\\_view\\_get\\_info](#)
  - Video API, [518](#)
- [l4re\\_video\\_view\\_info\\_flags\\_t](#)
  - Video API, [515](#)
- [l4re\\_video\\_view\\_info\\_t](#), [1702](#)
- [l4re\\_video\\_view\\_refresh](#)
  - Video API, [518](#)
- [l4re\\_video\\_view\\_set\\_info](#)
  - Video API, [519](#)
- [l4re\\_video\\_view\\_set\\_viewport](#)
  - Video API, [519](#)
- [l4re\\_video\\_view\\_stack](#)
  - Video API, [519](#)
- [l4re\\_video\\_view\\_t](#), [1703](#)
  - Video API, [514](#)
- [L4SHM-based ring buffer implementation](#), [538](#)
- [l4shmc\\_add\\_chunk](#)
  - Chunks, [548](#)
- [l4shmc\\_add\\_signal](#)
  - Signals, [559](#)
- [l4shmc\\_area\\_overhead](#)
  - Shared Memory Library, [544](#)
- [l4shmc\\_area\\_size](#)
  - Shared Memory Library, [544](#)
- [l4shmc\\_area\\_size\\_free](#)
  - Shared Memory Library, [544](#)
- [l4shmc\\_attach](#)
  - Shared Memory Library, [544](#)
- [l4shmc\\_attach\\_signal](#)
  - Signals, [559](#)
- [l4shmc\\_check\\_magic](#)
  - Signals, [561](#)
- [l4shmc\\_chunk\\_capacity](#)

- Chunks, [549](#)
- l4shmc\_chunk\_consumed
  - Consumer, [552](#)
- l4shmc\_chunk\_overhead
  - Shared Memory Library, [545](#)
- l4shmc\_chunk\_ptr
  - Chunks, [549](#)
- l4shmc\_chunk\_ready
  - Producer, [556](#)
- l4shmc\_chunk\_ready\_sig
  - Producer, [556](#)
- l4shmc\_chunk\_signal
  - Chunks, [550](#)
- l4shmc\_chunk\_size
  - Consumer, [552](#)
- l4shmc\_chunk\_try\_to\_take
  - Producer, [557](#)
- l4shmc\_chunk\_try\_to\_take\_for\_overwriting
  - Producer, [557](#)
- l4shmc\_chunk\_try\_to\_take\_for\_reading
  - Consumer, [553](#)
- l4shmc\_chunk\_try\_to\_take\_for\_writing
  - Producer, [557](#)
- l4shmc\_connect\_chunk\_signal
  - Shared Memory Library, [545](#)
- l4shmc\_create
  - Shared Memory Library, [546](#)
- l4shmc\_enable\_chunk
  - Consumer, [553](#)
- l4shmc\_enable\_signal
  - Consumer, [563](#)
- l4shmc\_get\_chunk
  - Chunks, [550](#)
- l4shmc\_get\_chunk\_to
  - Chunks, [550](#)
- l4shmc\_get\_client\_nr
  - Shared Memory Library, [546](#)
- l4shmc\_get\_initialized\_clients
  - Shared Memory Library, [547](#)
- l4shmc\_get\_signal
  - Signals, [561](#)
- l4shmc\_is\_chunk\_clear
  - Producer, [558](#)
- l4shmc\_is\_chunk\_ready
  - Consumer, [553](#)
- l4shmc\_iterate\_chunk
  - Chunks, [551](#)
- l4shmc\_mark\_client\_initialized
  - Shared Memory Library, [547](#)
- l4shmc\_rb\_attach\_receiver
  - ringbuf.h, [2633](#)
- l4shmc\_rb\_attach\_sender
  - ringbuf.h, [2634](#)
- l4shmc\_rb\_deinit\_buffer
  - ringbuf.h, [2634](#)
- l4shmc\_rb\_init\_buffer
  - ringbuf.h, [2634](#)
- l4shmc\_rb\_init\_receiver
  - ringbuf.h, [2635](#)
- l4shmc\_rb\_receiver\_copy\_out
  - ringbuf.h, [2636](#)
- l4shmc\_rb\_receiver\_notify\_done
  - ringbuf.h, [2636](#)
- l4shmc\_rb\_receiver\_read\_next\_size
  - ringbuf.h, [2636](#)
- l4shmc\_rb\_receiver\_wait\_for\_data
  - ringbuf.h, [2636](#)
- l4shmc\_rb\_sender\_alloc\_packet
  - ringbuf.h, [2637](#)
- l4shmc\_rb\_sender\_commit\_packet
  - ringbuf.h, [2637](#)
- l4shmc\_rb\_sender\_next\_copy\_in
  - ringbuf.h, [2638](#)
- l4shmc\_rb\_sender\_put\_data
  - ringbuf.h, [2638](#)
- L4SHMC\_RINGBUF\_DATA
  - Internal, [539](#)
- L4SHMC\_RINGBUF\_DATA\_SIZE
  - Internal, [539](#)
- L4SHMC\_RINGBUF\_HEAD
  - Internal, [539](#)
- l4shmc\_ringbuf\_head\_t, [1704](#)
- l4shmc\_ringbuf\_t, [1705](#)
- l4shmc\_signal\_cap
  - Signals, [562](#)
- l4shmc\_trigger
  - Producer, [566](#)
- l4shmc\_wait\_any
  - Consumer, [563](#)
- l4shmc\_wait\_any\_to
  - Consumer, [563](#)
- l4shmc\_wait\_any\_try
  - Consumer, [564](#)
- l4shmc\_wait\_chunk
  - Consumer, [554](#)
- l4shmc\_wait\_chunk\_to
  - Consumer, [554](#)
- l4shmc\_wait\_chunk\_try
  - Consumer, [555](#)
- l4shmc\_wait\_signal
  - Consumer, [564](#)
- l4shmc\_wait\_signal\_to
  - Consumer, [565](#)
- l4shmc\_wait\_signal\_try
  - Consumer, [565](#)
- l4sigma0\_debug\_dump
  - Sigma0 API, [568](#)
- L4SIGMA0\_IPCERROR
  - Sigma0 API, [568](#)
- l4sigma0\_map\_anypage
  - Sigma0 API, [568](#)
- l4sigma0\_map\_errstr
  - Sigma0 API, [569](#)
- l4sigma0\_map\_iomem
  - Sigma0 API, [569](#)
- l4sigma0\_map\_kip

- Sigma0 API, [570](#)
- l4sigma0\_map\_mem
  - Sigma0 API, [570](#)
- L4SIGMA0\_NOFPAGE
  - Sigma0 API, [568](#)
- L4SIGMA0\_NOTALIGNED
  - Sigma0 API, [568](#)
- L4SIGMA0\_OK
  - Sigma0 API, [568](#)
- l4sigma0\_return\_flags\_t
  - Sigma0 API, [568](#)
- L4SIGMA0\_SMALLERFPAGE
  - Sigma0 API, [568](#)
- l4util\_add16
  - Atomic Instructions, [584](#)
- l4util\_add16\_res
  - Atomic Instructions, [584](#)
- l4util\_add32
  - Atomic Instructions, [585](#)
- l4util\_add32\_res
  - Atomic Instructions, [585](#)
- l4util\_add8
  - Atomic Instructions, [585](#)
- l4util\_add8\_res
  - Atomic Instructions, [586](#)
- l4util\_and16
  - Atomic Instructions, [586](#)
- l4util\_and16\_res
  - Atomic Instructions, [586](#)
- l4util\_and32
  - Atomic Instructions, [587](#)
- l4util\_and32\_res
  - Atomic Instructions, [587](#)
- l4util\_and8
  - Atomic Instructions, [587](#)
- l4util\_and8\_res
  - Atomic Instructions, [588](#)
- l4util\_atomic\_add
  - Atomic Instructions, [588](#)
- l4util\_atomic\_inc
  - Atomic Instructions, [588](#)
- l4util\_backtrace
  - backtrace.h, [2952](#)
- l4util\_bsf
  - Bit Manipulation, [601](#)
- l4util\_bsr
  - Bit Manipulation, [601](#)
- l4util\_btc
  - Bit Manipulation, [601](#)
- l4util\_btr
  - Bit Manipulation, [602](#)
- l4util\_bts
  - Bit Manipulation, [602](#)
- l4util\_clear\_bit
  - Bit Manipulation, [604](#)
- l4util\_cmpxchg
  - Atomic Instructions, [589](#)
- l4util\_cmpxchg16
  - Atomic Instructions, [589](#)
- l4util\_cmpxchg32
  - Atomic Instructions, [590](#)
- l4util\_cmpxchg8
  - Atomic Instructions, [590](#)
- l4util\_complement\_bit
  - Bit Manipulation, [605](#)
- l4util\_cpu\_capabilities
  - CPU related functions, [609](#)
- l4util\_cpu\_capabilities\_nocheck
  - CPU related functions, [609](#)
- l4util\_cpu\_has\_cpuid
  - CPU related functions, [610](#)
- l4util\_dec16
  - Atomic Instructions, [591](#)
- l4util\_dec16\_res
  - Atomic Instructions, [591](#)
- l4util\_dec32
  - Atomic Instructions, [591](#)
- l4util\_dec32\_res
  - Atomic Instructions, [592](#)
- l4util\_dec8
  - Atomic Instructions, [592](#)
- l4util\_dec8\_res
  - Atomic Instructions, [592](#)
- l4util\_find\_first\_set\_bit
  - Bit Manipulation, [605](#)
- l4util\_find\_first\_zero\_bit
  - Bit Manipulation, [605](#)
- l4util\_idt\_desc\_t, [1706](#)
- l4util\_idt\_header\_t, [1707](#)
- l4util\_in16
  - IA32 Port I/O API, [637](#)
- l4util\_in32
  - IA32 Port I/O API, [638](#)
- l4util\_in8
  - IA32 Port I/O API, [638](#)
- l4util\_inc16
  - Atomic Instructions, [592](#)
- l4util\_inc16\_res
  - Atomic Instructions, [593](#)
- l4util\_inc32
  - Atomic Instructions, [593](#)
- l4util\_inc32\_res
  - Atomic Instructions, [593](#)
- l4util\_inc8
  - Atomic Instructions, [594](#)
- l4util\_inc8\_res
  - Atomic Instructions, [594](#)
- l4util\_ins16
  - IA32 Port I/O API, [639](#)
- l4util\_ins32
  - IA32 Port I/O API, [639](#)
- l4util\_ins8
  - IA32 Port I/O API, [639](#)
- l4util\_ioport\_map
  - port\_io.h, [2090](#)
- l4util\_irq\_acknowledge

- irq.h, [2815](#), [2826](#)
- l4util\_kip\_for\_each\_feature
  - Kernel Interface Page API, [644](#)
- l4util\_kip\_kernel\_abi\_version
  - Kernel Interface Page API, [645](#)
- l4util\_kip\_kernel\_has\_feature
  - Kernel Interface Page API, [645](#)
- l4util\_kip\_kernel\_is\_ux
  - Kernel Interface Page API, [645](#)
- l4util\_l4mod\_info, [1708](#)
  - vbe\_ctrl\_info, [1709](#)
- l4util\_l4mod\_mod, [1709](#)
- L4util\_l4mod\_mod\_flag\_kernel
  - l4mod.h, [2981](#)
- L4util\_l4mod\_mod\_flag\_mask
  - l4mod.h, [2981](#)
- L4util\_l4mod\_mod\_flag\_roottask
  - l4mod.h, [2981](#)
- L4util\_l4mod\_mod\_flag\_sigma0
  - l4mod.h, [2981](#)
- L4util\_l4mod\_mod\_flag\_unspec
  - l4mod.h, [2981](#)
- l4util\_l4mod\_mod\_info\_flag
  - l4mod.h, [2981](#)
- l4util\_mb\_addr\_range\_t, [1710](#)
- l4util\_mb\_apm\_t, [1711](#)
- l4util\_mb\_drive\_t, [1712](#)
  - drive\_cylinders, [1713](#)
  - drive\_mode, [1713](#)
  - drive\_number, [1713](#)
- l4util\_mb\_for\_each\_mmap\_entry
  - mb\_info.h, [2989](#)
- l4util\_mb\_info\_t, [1714](#)
- L4UTIL\_MB\_MEMORY
  - mb\_info.h, [2989](#)
- l4util\_mb\_mod\_t, [1715](#)
- l4util\_mb\_vbe\_ctrl\_t, [1716](#)
- l4util\_mb\_vbe\_mode\_t, [1717](#)
- l4util\_micros2l4to
  - Utility Functions, [580](#)
- l4util\_next\_power2
  - Bit Manipulation, [606](#)
- l4util\_or16
  - Atomic Instructions, [594](#)
- l4util\_or16\_res
  - Atomic Instructions, [594](#)
- l4util\_or32
  - Atomic Instructions, [595](#)
- l4util\_or32\_res
  - Atomic Instructions, [595](#)
- l4util\_or8
  - Atomic Instructions, [595](#)
- l4util\_or8\_res
  - Atomic Instructions, [596](#)
- l4util\_out16
  - IA32 Port I/O API, [640](#)
- l4util\_out32
  - IA32 Port I/O API, [640](#)
- l4util\_out8
  - IA32 Port I/O API, [640](#)
- l4util\_outs16
  - IA32 Port I/O API, [642](#)
- l4util\_outs32
  - IA32 Port I/O API, [642](#)
- l4util\_outs8
  - IA32 Port I/O API, [643](#)
- l4util\_rand
  - Random number support, [647](#)
- l4util\_set\_bit
  - Bit Manipulation, [606](#)
- l4util\_splitlog2\_hdl
  - Utility Functions, [581](#)
- l4util\_splitlog2\_size
  - Utility Functions, [582](#)
- l4util\_srand
  - Random number support, [647](#)
- l4util\_sub16
  - Atomic Instructions, [596](#)
- l4util\_sub16\_res
  - Atomic Instructions, [596](#)
- l4util\_sub32
  - Atomic Instructions, [597](#)
- l4util\_sub32\_res
  - Atomic Instructions, [597](#)
- l4util\_sub8
  - Atomic Instructions, [597](#)
- l4util\_sub8\_res
  - Atomic Instructions, [597](#)
- l4util\_test\_bit
  - Bit Manipulation, [607](#)
- l4util\_xchg
  - Atomic Instructions, [598](#)
- l4util\_xchg16
  - Atomic Instructions, [598](#)
- l4util\_xchg32
  - Atomic Instructions, [599](#)
- l4util\_xchg8
  - Atomic Instructions, [599](#)
- L4vbus, [719](#)
- L4vbus GPIO functions, [450](#)
  - l4vbus\_gpio\_config\_get, [451](#)
  - l4vbus\_gpio\_config\_pad, [452](#)
  - l4vbus\_gpio\_config\_pull, [453](#)
  - L4vbus\_gpio\_generic\_func, [451](#)
  - l4vbus\_gpio\_get, [453](#)
  - l4vbus\_gpio\_multi\_config\_pad, [454](#)
  - l4vbus\_gpio\_multi\_get, [455](#)
  - l4vbus\_gpio\_multi\_set, [455](#)
  - l4vbus\_gpio\_multi\_setup, [456](#)
  - L4VBUS\_GPIO\_PIN\_PULL\_DOWN, [451](#)
  - L4VBUS\_GPIO\_PIN\_PULL\_NONE, [451](#)
  - L4VBUS\_GPIO\_PIN\_PULL\_UP, [451](#)
  - L4vbus\_gpio\_pull\_modes, [451](#)
  - l4vbus\_gpio\_set, [457](#)
  - l4vbus\_gpio\_setup, [458](#)
  - L4VBUS\_GPIO\_SETUP\_INPUT, [451](#)

- L4VBUS\_GPIO\_SETUP\_IRQ, [451](#)
  - L4VBUS\_GPIO\_SETUP\_OUTPUT, [451](#)
  - l4vbus\_gpio\_to\_irq, [458](#)
- L4vbus PCI functions, [459](#)
  - l4vbus\_pci\_cfg\_read, [460](#)
  - l4vbus\_pci\_cfg\_write, [461](#)
  - l4vbus\_pci\_irq\_enable, [462](#)
  - l4vbus\_pciddev\_cfg\_read, [463](#)
  - l4vbus\_pciddev\_cfg\_write, [463](#)
  - l4vbus\_pciddev\_irq\_enable, [464](#)
- L4vbus power management functions, [465](#)
  - l4vbus\_pm\_resume, [465](#)
  - l4vbus\_pm\_suspend, [466](#)
- L4vbus::Device, [1721](#)
  - bus\_cap, [1724](#)
  - dev\_handle, [1724](#)
  - Device, [1723](#)
  - device, [1725](#)
  - device\_by\_hid, [1725](#)
  - get\_resource, [1726](#)
  - is\_compatible, [1727](#)
  - next\_device, [1728](#)
  - operator!=, [1729](#)
  - operator==, [1729](#)
- L4vbus::Gpio\_module, [1730](#)
  - config\_pad, [1733](#)
  - get, [1733](#)
  - pin, [1734](#)
  - set, [1735](#)
  - setup, [1736](#)
- L4vbus::Gpio\_module::Pin\_slice, [1737](#)
- L4vbus::Gpio\_pin, [1737](#)
  - config\_get, [1741](#)
  - config\_pad, [1741](#)
  - config\_pull, [1742](#)
  - get, [1743](#)
  - pin, [1743](#)
  - set, [1743](#)
  - setup, [1744](#)
  - to\_irq, [1745](#)
- L4vbus::Icu, [1746](#)
  - Src\_dev\_handle, [1749](#)
  - Src\_types, [1749](#)
  - vicu, [1749](#)
- L4vbus::Pci\_dev, [1750](#)
  - cfg\_read, [1754](#)
  - cfg\_write, [1754](#)
  - irq\_enable, [1755](#)
- L4vbus::Pci\_host\_bridge, [1756](#)
  - cfg\_read, [1760](#)
  - cfg\_write, [1760](#)
  - irq\_enable, [1761](#)
- L4vbus::Pm< DEC >, [1762](#)
  - pm\_resume, [1764](#)
  - pm\_suspend, [1764](#)
- L4vbus::Vbus, [1765](#)
  - assign\_dma\_domain, [1772](#), [1773](#)
  - release\_ioport, [1774](#)
  - request\_ioport, [1774](#)
  - root, [1775](#)
- l4vbus\_assign\_dma\_domain
  - L4 Vbus functions, [441](#)
- L4VBUS\_DEVICE\_F\_CHILDREN
  - vbus\_types.h, [3017](#)
- l4vbus\_device\_flags\_t
  - vbus\_types.h, [3016](#)
- l4vbus\_device\_t, [1776](#)
- L4vbus\_dma\_domain\_assign\_flags
  - L4 Vbus functions, [440](#)
- L4VBUS\_DMAD\_BIND
  - L4 Vbus functions, [441](#)
- L4VBUS\_DMAD\_KERNEL\_DMA\_SPACE
  - L4 Vbus functions, [441](#)
- L4VBUS\_DMAD\_L4RE\_DMA\_SPACE
  - L4 Vbus functions, [441](#)
- L4VBUS\_DMAD\_UNBIND
  - L4 Vbus functions, [441](#)
- l4vbus\_get\_adr
  - L4 Vbus functions, [442](#)
- l4vbus\_get\_device
  - L4 Vbus functions, [442](#)
- l4vbus\_get\_device\_by\_hid
  - L4 Vbus functions, [443](#)
- l4vbus\_get\_hid
  - L4 Vbus functions, [444](#)
- l4vbus\_get\_next\_device
  - L4 Vbus functions, [444](#)
- l4vbus\_get\_resource
  - L4 Vbus functions, [446](#)
- l4vbus\_gpio\_config\_get
  - L4vbus GPIO functions, [451](#)
- l4vbus\_gpio\_config\_pad
  - L4vbus GPIO functions, [452](#)
- l4vbus\_gpio\_config\_pull
  - L4vbus GPIO functions, [453](#)
- L4vbus\_gpio\_generic\_func
  - L4vbus GPIO functions, [451](#)
- l4vbus\_gpio\_get
  - L4vbus GPIO functions, [453](#)
- l4vbus\_gpio\_multi\_config\_pad
  - L4vbus GPIO functions, [454](#)
- l4vbus\_gpio\_multi\_get
  - L4vbus GPIO functions, [455](#)
- l4vbus\_gpio\_multi\_set
  - L4vbus GPIO functions, [455](#)
- l4vbus\_gpio\_multi\_setup
  - L4vbus GPIO functions, [456](#)
- L4VBUS\_GPIO\_PIN\_PULL\_DOWN
  - L4vbus GPIO functions, [451](#)
- L4VBUS\_GPIO\_PIN\_PULL\_NONE
  - L4vbus GPIO functions, [451](#)
- L4VBUS\_GPIO\_PIN\_PULL\_UP
  - L4vbus GPIO functions, [451](#)
- L4vbus\_gpio\_pull\_modes
  - L4vbus GPIO functions, [451](#)
- l4vbus\_gpio\_set

- L4vbus GPIO functions, [457](#)
- l4vbus\_gpio\_setup
  - L4vbus GPIO functions, [458](#)
- L4VBUS\_GPIO\_SETUP\_INPUT
  - L4vbus GPIO functions, [451](#)
- L4VBUS\_GPIO\_SETUP\_IRQ
  - L4vbus GPIO functions, [451](#)
- L4VBUS\_GPIO\_SETUP\_OUTPUT
  - L4vbus GPIO functions, [451](#)
- l4vbus\_gpio\_to\_irq
  - L4vbus GPIO functions, [458](#)
- L4VBUS\_ICU\_SRC\_DEV\_HANDLE
  - vbus.h, [3004](#)
- l4vbus\_icu\_src\_types
  - vbus.h, [3003](#)
- L4VBUS\_IFACE\_SHIFT
  - vbus\_interfaces.h, [3010](#)
- l4vbus\_iface\_type\_t
  - vbus\_interfaces.h, [3010](#)
- L4VBUS\_INTERFACE\_BUS
  - vbus\_interfaces.h, [3011](#)
- L4VBUS\_INTERFACE\_GENERIC
  - vbus\_interfaces.h, [3011](#)
- L4VBUS\_INTERFACE\_GPIO
  - vbus\_interfaces.h, [3011](#)
- L4VBUS\_INTERFACE\_ICU
  - vbus\_interfaces.h, [3011](#)
- L4VBUS\_INTERFACE\_PCI
  - vbus\_interfaces.h, [3011](#)
- L4VBUS\_INTERFACE\_PCIDEV
  - vbus\_interfaces.h, [3011](#)
- L4VBUS\_INTERFACE\_PM
  - vbus\_interfaces.h, [3011](#)
- l4vbus\_is\_compatible
  - L4 Vbus functions, [447](#)
- L4VBUS\_NULL
  - vbus.h, [3003](#)
- l4vbus\_pci\_cfg\_read
  - L4vbus PCI functions, [460](#)
- l4vbus\_pci\_cfg\_write
  - L4vbus PCI functions, [461](#)
- l4vbus\_pci\_irq\_enable
  - L4vbus PCI functions, [462](#)
- l4vbus\_pciddev\_cfg\_read
  - L4vbus PCI functions, [463](#)
- l4vbus\_pciddev\_cfg\_write
  - L4vbus PCI functions, [463](#)
- l4vbus\_pciddev\_irq\_enable
  - L4vbus PCI functions, [464](#)
- l4vbus\_pm\_resume
  - L4vbus power management functions, [465](#)
- l4vbus\_pm\_suspend
  - L4vbus power management functions, [466](#)
- l4vbus\_release\_ioport
  - L4 Vbus functions, [447](#)
- l4vbus\_request\_ioport
  - L4 Vbus functions, [448](#)
- L4VBUS\_RESOURCE\_BUS
  - vbus\_types.h, [3017](#)
- L4VBUS\_RESOURCE\_DMA\_DOMAIN
  - vbus\_types.h, [3017](#)
- L4VBUS\_RESOURCE\_F\_MEM\_MMIO\_READ
  - vbus\_types.h, [3017](#)
- L4VBUS\_RESOURCE\_F\_MEM\_MMIO\_WRITE
  - vbus\_types.h, [3017](#)
- L4VBUS\_RESOURCE\_F\_MEM\_R
  - vbus\_types.h, [3017](#)
- L4VBUS\_RESOURCE\_F\_MEM\_W
  - vbus\_types.h, [3017](#)
- l4vbus\_resource\_flags\_t
  - vbus\_types.h, [3017](#)
- L4VBUS\_RESOURCE\_GPIO
  - vbus\_types.h, [3017](#)
- L4VBUS\_RESOURCE\_INVALID
  - vbus\_types.h, [3017](#)
- L4VBUS\_RESOURCE\_IRQ
  - vbus\_types.h, [3017](#)
- L4VBUS\_RESOURCE\_MAX
  - vbus\_types.h, [3017](#)
- L4VBUS\_RESOURCE\_MEM
  - vbus\_types.h, [3017](#)
- L4VBUS\_RESOURCE\_PORT
  - vbus\_types.h, [3017](#)
- l4vbus\_resource\_t, [1777](#)
- l4vbus\_resource\_type\_t
  - vbus\_types.h, [3017](#)
- L4VBUS\_ROOT\_BUS
  - vbus.h, [3003](#)
- l4vbus\_subinterface\_supported
  - vbus\_interfaces.h, [3011](#)
- l4vbus\_vicu\_get\_cap
  - L4 Vbus functions, [449](#)
- L4vcpu::State, [1778](#)
  - add, [1779](#)
  - clear, [1779](#)
  - set, [1779](#)
  - State, [1778](#)
- L4vcpu::Vcpu, [1780](#)
  - cast, [1783](#)
  - entry\_ip, [1783](#)
  - entry\_sp, [1784](#)
  - ext\_alloc, [1784](#)
  - i, [1785](#)
  - irq\_disable\_save, [1785](#)
  - irq\_enable, [1785](#)
  - irq\_restore, [1786](#)
  - is\_irq\_entry, [1787](#)
  - is\_page\_fault\_entry, [1787](#)
  - r, [1787](#), [1788](#)
  - saved\_state, [1788](#)
  - state, [1788](#), [1789](#)
  - task, [1789](#)
  - wait\_for\_event, [1790](#)
- l4vcpu\_ext\_alloc
  - Extended vCPU support, [662](#)
- l4vcpu\_irq\_disable

- vCPU Support Library, 655
- l4vcpu\_irq\_disable\_save
  - vCPU Support Library, 656
- l4vcpu\_irq\_enable
  - vCPU Support Library, 657
- l4vcpu\_irq\_restore
  - vCPU Support Library, 658
- l4vcpu\_is\_irq\_entry
  - vCPU Support Library, 659
- l4vcpu\_is\_page\_fault\_entry
  - vCPU Support Library, 659
- l4vcpu\_print\_state
  - vCPU Support Library, 660
- l4vcpu\_wait\_for\_event
  - vCPU Support Library, 661
- l4vio\_net\_p2p, a virtual network point-to-point link, 76
- L4virtio, 720
- L4virtio::Device, 1791
  - config\_queue, 1795
  - device\_config, 1796
  - device\_notification\_irq, 1796
  - register\_ds, 1797
  - set\_status, 1798
- L4virtio::Driver::Block\_device, 1799
  - add\_block, 1803
  - process\_request, 1804
  - process\_used\_queue, 1805
  - send\_request, 1805
  - setup\_device, 1806
  - start\_request, 1807
- L4virtio::Driver::Block\_device::Handle, 1808
- L4virtio::Driver::Device, 1809
  - bind\_notification\_irq, 1812
  - config\_queue, 1813
  - driver\_acknowledge, 1814
  - driver\_connect, 1815
  - feature\_negotiated, 1816
  - max\_queue\_size, 1817
  - register\_ds, 1817
  - send, 1818
  - send\_and\_wait, 1819
  - wait, 1820
  - wait\_for\_next\_used, 1821
- L4virtio::Driver::Virtio\_net\_device, 1822
  - finish\_rx, 1826
  - rx\_pkt, 1826
  - rx\_queue\_size, 1827
  - setup\_device, 1827
  - tx, 1828
  - tx\_queue\_size, 1829
  - wait\_rx, 1830
- L4virtio::Driver::Virtio\_net\_device::Packet, 1831
- L4virtio::Driver::Virtqueue, 1831
  - alloc\_descriptor, 1836
  - desc, 1836
  - enqueue\_descriptor, 1837
  - find\_next\_used, 1838
  - free\_descriptor, 1838
  - init\_queue, 1839, 1840
  - initialize\_rings, 1841
- L4virtio::Ptr< T >, 1842
  - get, 1844
  - Invalid, 1844
  - Invalid\_type, 1844
  - is\_valid, 1845
- L4virtio::Svr::Bad\_descriptor, 1846
  - Bad\_address, 1847
  - Bad\_descriptor, 1847
  - Bad\_flags, 1847
  - Bad\_next, 1847
  - Bad\_rights, 1847
  - Bad\_size, 1847
  - Error, 1847
  - message, 1847
- L4virtio::Svr::Block\_dev\_base< Ds\_data >, 1848
  - Block\_dev\_base, 1852
  - finalize\_request, 1853
  - get\_writeback, 1854
  - set\_blk\_size, 1854
  - set\_config\_wce, 1854
  - set\_discard, 1855
  - set\_size\_max, 1855
  - set\_topology, 1855
  - set\_write\_zeroes, 1856
- L4virtio::Svr::Block\_request< Ds\_data >, 1856
  - data\_size, 1857
  - next\_block, 1858
- L4virtio::Svr::Console::Control\_message, 1859
  - Console\_port, 1860
  - Device\_add, 1860
  - Device\_ready, 1860
  - Device\_remove, 1860
  - Events, 1860
  - Port\_name, 1860
  - Port\_open, 1860
  - Port\_ready, 1860
  - Resize, 1860
- L4virtio::Svr::Console::Control\_request, 1860
- L4virtio::Svr::Console::Device, 1862
  - Device, 1866, 1867
  - do\_control\_work, 1868
  - notify\_queue, 1868
  - port, 1869
  - port\_read, 1870
  - port\_write, 1871
  - process\_device\_ready, 1872
  - process\_port\_open, 1873
  - process\_port\_ready, 1873
- L4virtio::Svr::Console::Device\_port, 1874
- L4virtio::Svr::Console::Features, 1878
  - console\_multiport\_bfm\_t, 1880
  - console\_size\_bfm\_t, 1880
  - emerg\_write\_bfm\_t, 1880
- L4virtio::Svr::Console::Port, 1881
  - Port\_added, 1884
  - Port\_disabled, 1884



- Port\_failed, 1884
- Port\_open, 1884
- Port\_ready, 1884
- Port\_status, 1883
- L4virtio::Svr::Console::Virtio\_con, 1884
  - handle\_control\_message, 1889
  - notify\_queue, 1890
  - port, 1891
  - port\_add, 1891
  - port\_open, 1892
  - port\_remove, 1894
  - process\_device\_ready, 1895
  - process\_port\_open, 1896
  - process\_port\_ready, 1896
  - reset\_device, 1897
  - send\_control\_message, 1897
  - Virtio\_con, 1888
- L4virtio::Svr::Data\_buffer, 1899
  - copy\_to, 1901
  - Data\_buffer, 1901
  - done, 1902
  - set, 1902
  - skip, 1902
- L4virtio::Svr::Dev\_config, 1903
  - add\_irq\_status, 1906
  - change\_queue\_config, 1907
  - Dev\_config, 1905, 1906
  - ds, 1907
  - get\_cmd, 1908
  - guest\_features, 1908
  - hdr, 1909
  - negotiated\_features, 1910
  - qconfig, 1910
  - reset\_cmd, 1911
  - reset\_queue, 1912
  - set\_device\_needs\_reset, 1913
  - set\_status, 1913
  - status, 1915
- L4virtio::Svr::Dev\_features, 1916
- L4virtio::Svr::Dev\_status, 1918
  - running, 1919
- L4virtio::Svr::Device\_t< DATA >, 1919
  - add\_trusted\_dataspaces, 1922
  - device\_error, 1923
  - device\_notify\_irq, 1923
  - handle\_mem\_cmd\_write, 1923
  - init\_mem\_info, 1924
  - register\_driver\_irq, 1924
  - reset\_queue\_config, 1925
  - setup\_queue, 1925
- L4virtio::Svr::Driver\_mem\_list\_t< DATA >, 1927
  - add, 1930
  - find, 1930
  - full, 1931
  - init, 1932
  - load\_desc, 1932–1934
  - remove, 1935
- L4virtio::Svr::Driver\_mem\_region\_t< DATA >, 1936
  - contains, 1939
  - Driver\_mem\_region\_t, 1938
  - drv\_base, 1940
  - ds, 1940
  - ds\_offset, 1940
  - empty, 1941
  - flags, 1941
  - is\_writable, 1941
  - local, 1941
  - local\_base, 1942
  - size, 1942
- L4virtio::Svr::Request\_processor, 1943
  - current\_flags, 1945
  - has\_more, 1945
  - next, 1946
  - start, 1947, 1949
- L4virtio::Svr::Scmi::Base\_attr\_t, 1950
- L4virtio::Svr::Scmi::Base\_proto, 1951
- L4virtio::Svr::Scmi::Perf\_proto, 1953
- L4virtio::Svr::Scmi::Performance\_attr\_t, 1956
- L4virtio::Svr::Scmi::Performance\_describe\_level\_t, 1956
- L4virtio::Svr::Scmi::Performance\_describe\_levels\_n\_t, 1957
- L4virtio::Svr::Scmi::Performance\_domain\_attr\_t, 1958
- L4virtio::Svr::Scmi::Proto< OBSERV >, 1959
- L4virtio::Svr::Scmi::Scmi\_dev, 1960
- L4virtio::Svr::Scmi::Scmi\_hdr\_t, 1964
- L4virtio::Svr::Virtqueue, 1964
  - consumed, 1969
  - desc, 1970
  - desc\_avail, 1971
  - disable\_notify, 1971
  - enable\_notify, 1971
  - finish, 1972, 1973
  - next\_avail, 1974
- L4virtio::Svr::Virtqueue::Head\_desc, 1975
  - desc, 1976
  - operator Null\_ptr\_check const \*, 1976
  - valid, 1976
- L4virtio::Virtqueue, 1977
  - avail\_align, 1981
  - avail\_size, 1981
  - desc\_align, 1982
  - desc\_size, 1982
  - disable, 1983
  - dump, 1984
  - get\_avail\_idx, 1984
  - get\_tail\_avail\_idx, 1985
  - no\_notify\_guest, 1985
  - no\_notify\_host, 1986, 1987
  - num, 1987
  - ready, 1988
  - setup, 1989
  - setup\_simple, 1990
  - total\_size, 1991, 1992
  - used\_align, 1993
  - used\_size, 1993



- L4virtio::Virtqueue::Avail, [1995](#)
- L4virtio::Virtqueue::Avail::Flags, [1996](#)
  - no\_irq\_bfm\_t, [1997](#)
- L4virtio::Virtqueue::Desc, [1997](#)
- L4virtio::Virtqueue::Desc::Flags, [1999](#)
  - indirect\_bfm\_t, [2000](#)
  - next\_bfm\_t, [2000](#)
  - write\_bfm\_t, [2000](#)
- L4virtio::Virtqueue::Used, [2001](#)
- L4virtio::Virtqueue::Used::Flags, [2002](#)
  - no\_notify\_bfm\_t, [2002](#)
- L4virtio::Virtqueue::Used\_elem, [2003](#)
  - Used\_elem, [2003](#)
- l4virtio\_block\_config\_t, [2004](#)
- l4virtio\_block\_discard\_t, [2005](#)
- l4virtio\_block\_header\_t, [2006](#)
- L4virtio\_block\_operations
  - L4 VIRTIO Block Device, [428](#)
- L4VIRTIO\_BLOCK\_S\_IOERR
  - L4 VIRTIO Block Device, [429](#)
- L4VIRTIO\_BLOCK\_S\_OK
  - L4 VIRTIO Block Device, [429](#)
- L4VIRTIO\_BLOCK\_S\_UNSUPP
  - L4 VIRTIO Block Device, [429](#)
- L4virtio\_block\_status
  - L4 VIRTIO Block Device, [429](#)
- L4VIRTIO\_BLOCK\_T\_DISCARD
  - L4 VIRTIO Block Device, [429](#)
- L4VIRTIO\_BLOCK\_T\_FLUSH
  - L4 VIRTIO Block Device, [429](#)
- L4VIRTIO\_BLOCK\_T\_GET\_ID
  - L4 VIRTIO Block Device, [429](#)
- L4VIRTIO\_BLOCK\_T\_IN
  - L4 VIRTIO Block Device, [429](#)
- L4VIRTIO\_BLOCK\_T\_OUT
  - L4 VIRTIO Block Device, [429](#)
- L4VIRTIO\_BLOCK\_T\_WRITE\_ZEROES
  - L4 VIRTIO Block Device, [429](#)
- L4VIRTIO\_CMD\_CFG\_CHANGED
  - L4 VIRTIO Transport Layer, [433](#)
- L4VIRTIO\_CMD\_CFG\_QUEUE
  - L4 VIRTIO Transport Layer, [433](#)
- L4VIRTIO\_CMD\_MASK
  - L4 VIRTIO Transport Layer, [433](#)
- L4VIRTIO\_CMD\_NONE
  - L4 VIRTIO Transport Layer, [433](#)
- L4VIRTIO\_CMD\_NOTIFY\_QUEUE
  - L4 VIRTIO Transport Layer, [433](#)
- L4VIRTIO\_CMD\_SET\_STATUS
  - L4 VIRTIO Transport Layer, [433](#)
- l4virtio\_config\_hdr\_t, [2007](#)
- l4virtio\_config\_queue
  - L4 VIRTIO Transport Layer, [435](#)
- l4virtio\_config\_queue\_t, [2008](#)
  - L4 VIRTIO Transport Layer, [433](#)
- l4virtio\_config\_queues
  - L4 VIRTIO Transport Layer, [436](#)
- l4virtio\_device\_config
  - L4 VIRTIO Transport Layer, [436](#)
- l4virtio\_device\_config\_ds
  - L4 VIRTIO Transport Layer, [436](#)
- l4virtio\_device\_ids
  - L4 VIRTIO Transport Layer, [434](#)
- l4virtio\_device\_notification\_irq
  - L4 VIRTIO Transport Layer, [437](#)
- L4virtio\_device\_status
  - L4 VIRTIO Transport Layer, [435](#)
- L4virtio\_feature\_bits
  - L4 VIRTIO Transport Layer, [435](#)
- L4VIRTIO\_FEATURE\_CMD\_CONFIG
  - L4 VIRTIO Transport Layer, [435](#)
- L4VIRTIO\_FEATURE\_VERSION\_1
  - L4 VIRTIO Transport Layer, [435](#)
- L4VIRTIO\_ID\_9P
  - L4 VIRTIO Transport Layer, [434](#)
- L4VIRTIO\_ID\_BALLOON
  - L4 VIRTIO Transport Layer, [434](#)
- L4VIRTIO\_ID\_BLOCK
  - L4 VIRTIO Transport Layer, [434](#)
- L4VIRTIO\_ID\_CAIF
  - L4 VIRTIO Transport Layer, [434](#)
- L4VIRTIO\_ID\_CONSOLE
  - L4 VIRTIO Transport Layer, [434](#)
- L4VIRTIO\_ID\_CRYPTOP
  - L4 VIRTIO Transport Layer, [434](#)
- L4VIRTIO\_ID\_FS
  - L4 VIRTIO Transport Layer, [434](#)
- L4VIRTIO\_ID\_GPU
  - L4 VIRTIO Transport Layer, [434](#)
- L4VIRTIO\_ID\_INPUT
  - L4 VIRTIO Transport Layer, [434](#)
- L4VIRTIO\_ID\_NET
  - L4 VIRTIO Transport Layer, [434](#)
- L4VIRTIO\_ID\_RNG
  - L4 VIRTIO Transport Layer, [434](#)
- L4VIRTIO\_ID\_RPMMSG
  - L4 VIRTIO Transport Layer, [434](#)
- L4VIRTIO\_ID\_RPROC\_SERIAL
  - L4 VIRTIO Transport Layer, [434](#)
- L4VIRTIO\_ID\_SCM
  - L4 VIRTIO Transport Layer, [434](#)
- L4VIRTIO\_ID\_SCSI
  - L4 VIRTIO Transport Layer, [434](#)
- L4VIRTIO\_ID\_SOCKET
  - L4 VIRTIO Transport Layer, [434](#)
- L4VIRTIO\_ID\_VSOCKET
  - L4 VIRTIO Transport Layer, [434](#)
- l4virtio\_input\_absinfo\_t, [2009](#)
- l4virtio\_input\_config\_t, [2010](#)
- l4virtio\_input\_devids\_t, [2010](#)
- l4virtio\_input\_event\_t, [2011](#)
- L4VIRTIO\_IRQ\_STATUS\_CONFIG
  - L4 VIRTIO Transport Layer, [434](#)
- L4VIRTIO\_IRQ\_STATUS\_VRING
  - L4 VIRTIO Transport Layer, [434](#)
- l4virtio\_net\_config\_t, [2011](#)

- [l4virtio\\_net\\_header\\_t](#), [2012](#)
- [L4VIRTIO\\_OP\\_CONFIG\\_QUEUE](#)
  - [L4 VIRTIO Transport Layer](#), [434](#)
- [L4VIRTIO\\_OP\\_DEVICE\\_CONFIG](#)
  - [L4 VIRTIO Transport Layer](#), [434](#)
- [L4VIRTIO\\_OP\\_GET\\_DEVICE\\_IRQ](#)
  - [L4 VIRTIO Transport Layer](#), [434](#)
- [L4VIRTIO\\_OP\\_REGISTER\\_DS](#)
  - [L4 VIRTIO Transport Layer](#), [434](#)
- [L4VIRTIO\\_OP\\_SET\\_STATUS](#)
  - [L4 VIRTIO Transport Layer](#), [434](#)
- [l4virtio\\_register\\_ds](#)
  - [L4 VIRTIO Transport Layer](#), [437](#)
- [l4virtio\\_set\\_status](#)
  - [L4 VIRTIO Transport Layer](#), [438](#)
- [L4VIRTIO\\_STATUS\\_ACKNOWLEDGE](#)
  - [L4 VIRTIO Transport Layer](#), [435](#)
- [L4VIRTIO\\_STATUS\\_DEVICE\\_NEEDS\\_RESET](#)
  - [L4 VIRTIO Transport Layer](#), [435](#)
- [L4VIRTIO\\_STATUS\\_DRIVER](#)
  - [L4 VIRTIO Transport Layer](#), [435](#)
- [L4VIRTIO\\_STATUS\\_DRIVER\\_OK](#)
  - [L4 VIRTIO Transport Layer](#), [435](#)
- [L4VIRTIO\\_STATUS\\_FAILED](#)
  - [L4 VIRTIO Transport Layer](#), [435](#)
- [L4VIRTIO\\_STATUS\\_FEATURES\\_OK](#)
  - [L4 VIRTIO Transport Layer](#), [435](#)
- [length](#)
  - [L4::lpc::Varg](#), [1107](#)
- [Libedid\\_block\\_size](#)
  - [EDID parsing functionality](#), [402](#)
- [libedid\\_check\\_header](#)
  - [EDID parsing functionality](#), [402](#)
- [libedid\\_checksum](#)
  - [EDID parsing functionality](#), [403](#)
- [Libedid\\_consts](#)
  - [EDID parsing functionality](#), [402](#)
- [libedid\\_dump](#)
  - [EDID parsing functionality](#), [403](#)
- [libedid\\_dump\\_standard\\_timings](#)
  - [EDID parsing functionality](#), [403](#)
- [libedid\\_num\\_ext\\_blocks](#)
  - [EDID parsing functionality](#), [404](#)
- [libedid\\_pnp\\_id](#)
  - [EDID parsing functionality](#), [404](#)
- [libedid\\_prefered\\_resolution](#)
  - [EDID parsing functionality](#), [404](#)
- [libedid\\_revision](#)
  - [EDID parsing functionality](#), [404](#)
- [libedid\\_version](#)
  - [EDID parsing functionality](#), [405](#)
- [Link](#)
  - [L4Re::Namespace](#), [1517](#)
- [link](#)
  - [L4Re::Vfs::Directory](#), [1636](#)
- [List\\_alloc](#)
  - [cxx::List\\_alloc](#), [865](#)
- [list\\_alloc.h](#)
  - [l4la\\_alloc](#), [2983](#)
  - [l4la\\_avail](#), [2983](#)
  - [l4la\\_dump](#), [2983](#)
  - [l4la\\_free](#), [2983](#)
  - [l4la\\_init](#), [2984](#)
- [load\\_desc](#)
  - [L4virtio::Svr::Driver\\_mem\\_list\\_t< DATA >](#), [1932–1934](#)
- [local](#)
  - [L4virtio::Svr::Driver\\_mem\\_region\\_t< DATA >](#), [1941](#)
- [local\\_base](#)
  - [L4virtio::Svr::Driver\\_mem\\_region\\_t< DATA >](#), [1942](#)
- [local\\_id\\_received](#)
  - [L4::lpc::Gen\\_fpage](#), [1045](#)
- [log](#)
  - [L4Re::Env](#), [1477](#)
- [Log interface](#), [488](#)
  - [l4re\\_log\\_print](#), [489](#)
  - [l4re\\_log\\_print\\_srv](#), [489](#)
  - [l4re\\_log\\_printn](#), [490](#)
  - [l4re\\_log\\_printn\\_srv](#), [491](#)
- [Logging interface](#), [533](#)
- [loop](#)
  - [L4::Server< LOOP\\_HOOKS >](#), [1258](#)
  - [L4Re::Util::Registry\\_server< LOOP\\_HOOKS >](#), [1620](#)
- [Low-Level Thread Functions](#), [646](#)
- [Low\\_mask](#)
  - [cxx::Bitfield< T, LSB, MSB >](#), [777](#)
- [lower\\_bound\\_node](#)
  - [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >](#), [816](#)
  - [cxx::Bits::Bst< Node, Get\\_key, Compare >](#), [833](#)
- [Lsb](#)
  - [cxx::Bitfield< T, LSB, MSB >](#), [777](#)
- [lseek64](#)
  - [L4Re::Vfs::Regular\\_file](#), [1660](#)
- [Lstr](#)
  - [L4::Factory::Lstr](#), [1005](#)
- [Mag, the GUI Multiplexer](#), [62](#)
- [main\\_thread](#)
  - [L4Re::Env](#), [1477](#)
- [make\\_cap](#)
  - [L4::lpc](#), [680](#)
- [make\\_cap\\_full](#)
  - [L4::lpc](#), [681](#)
- [make\\_cap\\_rw](#)
  - [L4::lpc](#), [681](#)
- [make\\_cap\\_rws](#)
  - [L4::lpc](#), [683](#)
- [make\\_ref\\_cap](#)
  - [L4Re Capability API](#), [532](#)
- [make\\_ref\\_del\\_cap](#)
  - [L4Re Capability API](#), [533](#)
- [make\\_shared\\_cap](#)
  - [L4Re](#), [704](#)

- L4Re::Util, 716
- make\_shared\_del\_cap
  - L4Re, 705
  - L4Re::Util, 716
- make\_unique\_cap
  - L4Re, 706
  - L4Re::Util, 716
- make\_unique\_del\_cap
  - L4Re, 706
  - L4Re::Util, 718
- map
  - L4::Task, 1285
  - L4Re::Dataspace, 1453
  - L4Re::Dma\_space, 1466
  - L4Re::Util::Dataspace\_svr, 1581
- map\_hook
  - L4Re::Util::Dataspace\_svr, 1582
- map\_info
  - L4Re::Dataspace, 1454
  - L4Re::Util::Dataspace\_svr, 1583
- map\_region
  - L4Re::Dataspace, 1455
- Mask
  - cxx::Bitfield< T, LSB, MSB >, 777
- mask
  - L4::lcu, 1016
- Masks
  - cxx::Bitfield< T, LSB, MSB >, 777
- max
  - Small C++ Template Library, 574
- max\_free\_slabs
  - cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >, 765
  - cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >, 771
- max\_queue\_size
  - L4virtio::Driver::Device, 1817
- MB\_ARD\_MEMORY
  - mb\_info.h, 2989
- MB\_ART\_MEMORY
  - mb\_info.h, 2989
- mb\_info.h
  - l4util\_mb\_for\_each\_mmap\_entry, 2989
  - L4UTIL\_MB\_MEMORY, 2989
  - MB\_ARD\_MEMORY, 2989
  - MB\_ART\_MEMORY, 2989
- Mem
  - L4::Type\_info::Demand\_t< CAPS, FLAGS, MEM, PORTS >, 1320
- mem\_alloc
  - L4Re::Env, 1478
- Mem\_alloc\_flags
  - L4Re::Mem\_alloc, 1507
- Mem\_desc
  - L4::Kip::Mem\_desc, 1181
- Mem\_type
  - L4::Kip::Mem\_desc, 1181
- Memory allocator, 491
- l4re\_ma\_alloc, 492
- l4re\_ma\_alloc\_align, 493
- l4re\_ma\_alloc\_align\_svr, 494
- l4re\_ma\_flags, 492
- Memory descriptors (C version), 189
  - l4\_kernel\_info\_get\_mem\_desc\_end, 192
  - l4\_kernel\_info\_get\_mem\_desc\_is\_virtual, 192
  - l4\_kernel\_info\_get\_mem\_desc\_start, 192
  - l4\_kernel\_info\_get\_mem\_desc\_subtype, 192
  - l4\_kernel\_info\_get\_mem\_desc\_type, 192
  - l4\_kernel\_info\_get\_num\_mem\_descs, 193
  - l4\_kernel\_info\_mem\_desc\_t, 190
  - l4\_kernel\_info\_set\_mem\_desc, 193
  - l4\_mem\_archspecific\_acpi\_nvs, 191
  - l4\_mem\_archspecific\_acpi\_tables, 191
  - l4\_mem\_archspecific\_sub\_type\_common\_t, 191
  - l4\_mem\_info\_acpi\_rsd, 191
  - l4\_mem\_info\_sub\_type\_t, 191
  - l4\_mem\_type\_archspecific, 191
  - l4\_mem\_type\_bootloader, 191
  - l4\_mem\_type\_conventional, 191
  - l4\_mem\_type\_dedicated, 191
  - l4\_mem\_type\_info, 191
  - l4\_mem\_type\_reserved, 191
  - l4\_mem\_type\_shared, 191
  - l4\_mem\_type\_t, 191
  - l4\_mem\_type\_undefined, 191
- Memory management - Data Spaces and the Region Map, 22
- Memory operations., 330
  - L4\_mem\_op\_widths, 330
  - l4\_mem\_read, 331
  - L4\_MEM\_WIDTH\_1BYTE, 330
  - L4\_MEM\_WIDTH\_2BYTE, 330
  - L4\_MEM\_WIDTH\_4BYTE, 330
  - l4\_mem\_write, 331
- Memory related, 332
  - l4\_addr\_consts\_t, 335
  - l4\_bytes\_to\_mwords, 336
  - L4\_INVALID\_ADDR, 336
  - L4\_LOG2\_PAGESIZE, 333
  - L4\_LOG2\_SUPERPAGESIZE, 333
  - L4\_PAGEMASK, 334
  - L4\_PAGESHIFT, 334
  - l4\_round\_page, 336
  - l4\_round\_size, 337
  - L4\_SUPERPAGEMASK, 334
  - L4\_SUPERPAGESHIFT, 335
  - L4\_SUPERPAGESIZE, 335
  - l4\_trunc\_page, 338
  - l4\_trunc\_size, 339
- message
  - L4virtio::Svr::Bad\_descriptor, 1847
- Message Items, 364
  - L4\_ITEM\_CONT, 365
  - L4\_ITEM\_MAP, 365
  - l4\_map\_control, 366
  - L4\_MAP\_ITEM\_GRANT, 365

- L4\_MAP\_ITEM\_MAP, [365](#)
- l4\_map\_obj\_control, [366](#)
- l4\_msg\_item\_consts\_t, [365](#)
- L4\_RCV\_ITEM\_LOCAL\_ID, [365](#)
- L4\_RCV\_ITEM\_SINGLE\_CAP, [365](#)
- Message Registers (MRs), [395](#)
- Message Tag, [367](#)
  - l4\_msgtag, [371](#)
  - L4\_MSGTAG\_ERROR, [369](#)
  - L4\_MSGTAG\_FLAGS, [369](#)
  - l4\_msgtag\_flags, [369](#)
  - l4\_msgtag\_flags, [372](#)
  - l4\_msgtag\_has\_error, [373](#)
  - l4\_msgtag\_is\_exception, [373](#)
  - l4\_msgtag\_is\_io\_page\_fault, [375](#)
  - l4\_msgtag\_is\_page\_fault, [376](#)
  - l4\_msgtag\_is\_preemption, [376](#)
  - l4\_msgtag\_is\_sigma0, [377](#)
  - l4\_msgtag\_is\_sys\_exception, [378](#)
  - l4\_msgtag\_items, [378](#)
  - l4\_msgtag\_label, [379](#)
  - L4\_msgtag\_protocol, [370](#)
  - L4\_MSGTAG\_SCHEDULE, [369](#)
  - l4\_msgtag\_t, [369](#)
  - L4\_MSGTAG\_TRANSFER\_FPU, [369](#)
  - l4\_msgtag\_words, [380](#)
  - L4\_platform\_ctl\_proto, [370](#)
  - L4\_PROTO\_ALLOW\_SYSCALL, [370](#)
  - L4\_PROTO\_DEBUGGER, [370](#)
  - L4\_PROTO\_DMA\_SPACE, [370](#)
  - L4\_PROTO\_EXCEPTION, [370](#)
  - L4\_PROTO\_FACTORY, [370](#)
  - L4\_PROTO\_IO\_PAGE\_FAULT, [370](#)
  - L4\_PROTO\_IOMMU, [370](#)
  - L4\_PROTO\_IRQ, [370](#)
  - L4\_PROTO\_IRQ\_MUX, [370](#)
  - L4\_PROTO\_IRQ\_SENDER, [370](#)
  - L4\_PROTO\_KOBJECT, [370](#)
  - L4\_PROTO\_LOG, [370](#)
  - L4\_PROTO\_META, [370](#)
  - L4\_PROTO\_NONE, [370](#)
  - L4\_PROTO\_PAGE\_FAULT, [370](#)
  - L4\_PROTO\_PF\_EXCEPTION, [370](#)
  - L4\_PROTO\_PLATFORM\_CTL, [371](#)
  - L4\_PROTO\_PREEMPTION, [370](#)
  - L4\_PROTO\_SCHEDULER, [370](#)
  - L4\_PROTO\_SEMAPHORE, [370](#)
  - L4\_PROTO\_SIGMA0, [370](#)
  - L4\_PROTO\_SMCCC, [370](#)
  - L4\_PROTO\_SYS\_EXCEPTION, [370](#)
  - L4\_PROTO\_TASK, [370](#)
  - L4\_PROTO\_THREAD, [370](#)
  - L4\_PROTO\_VCPU\_CONTEXT, [370](#)
  - L4\_PROTO\_VM, [370](#)
- min
  - Small C++ Template Library, [575](#)
- mkdir
  - L4Re::Vfs::Directory, [1636](#)
- mmio\_read
  - L4Re::Mmio\_space, [1512](#)
- mmio\_write
  - L4Re::Mmio\_space, [1512](#)
- Mode
  - L4Re::Util::Event\_t< PAYLOAD >, [1596](#)
- mode
  - L4::Uart, [1360](#)
- Mode\_irq
  - L4Re::Util::Event\_t< PAYLOAD >, [1596](#)
- Mode\_polling
  - L4Re::Util::Event\_t< PAYLOAD >, [1596](#)
- modify
  - L4drivers::Register\_tmpl< BITS, BLOCK >, [1434](#)
- modify\_senders
  - L4::Thread, [1295](#)
- Moe, the Root-Task, [45](#)
- mount
  - L4Re::Vfs::File\_system, [1643](#)
  - L4Re::Vfs::Fs, [1647](#)
- move
  - L4::Cap< T >, [952](#)
  - L4::Cap\_base, [960](#)
- Mr\_bytes
  - L4::lpc::Msg, [687](#)
- Mr\_words
  - L4::lpc::Msg, [687](#)
- Msb
  - cxx::Bitfield< T, LSB, MSB >, [777](#)
- msg\_add
  - L4::lpc::Msg, [689](#)
- msg\_get
  - L4::lpc::Msg, [690](#)
- Msg\_ptr
  - L4::lpc::Msg\_ptr< T >, [1086](#)
- msg\_ptr
  - L4::lpc, [684](#)
- msi\_info
  - L4::lcu, [1017](#)
- N
  - cxx::Bits::Direction, [842](#)
- Name-space API, [534](#)
- Name\_max
  - L4Re::Inhibitor, [1497](#)
- Namespace interface, [495](#)
  - l4re\_ns\_query\_srv, [496](#)
  - l4re\_ns\_query\_to\_srv, [497](#)
  - l4re\_ns\_register\_flags, [496](#)
  - l4re\_ns\_register\_obj\_srv, [498](#)
- Ned, the Init Process, [49](#)
- negotiated\_features
  - L4virtio::Svr::Dev\_config, [1910](#)
- next
  - L4Re::Event\_buffer\_t< PAYLOAD >, [1491](#)
  - L4virtio::Svr::Request\_processor, [1946](#)
- next\_avail
  - L4virtio::Svr::Virtqueue, [1974](#)
- next\_bfm\_t

- L4virtio::Virtqueue::Desc::Flags, 2000
- next\_block
  - L4virtio::Svr::Block\_request< Ds\_data >, 1858
- next\_device
  - L4vbus::Device, 1728
- next\_lock\_info
  - L4Re::Inhibitor, 1497
- next\_timeout
  - L4::lpc\_svr::Timeout\_queue, 1148
- no\_demand
  - L4::Type\_info::Demand, 1317
- No\_eager\_map
  - L4Re::Rm::F, 1546
- No\_init
  - L4::Cap\_base, 955
- No\_init\_type
  - L4::Cap\_base, 955
- no\_irq\_bfm\_t
  - L4virtio::Virtqueue::Avail::Flags, 1997
- no\_notify\_bfm\_t
  - L4virtio::Virtqueue::Used::Flags, 2002
- no\_notify\_guest
  - L4virtio::Virtqueue, 1985
- no\_notify\_host
  - L4virtio::Virtqueue, 1986, 1987
- No\_sync
  - L4Re::Dma\_space, 1465
- None
  - L4::Types::Flags< BITS\_ENUM, UNDERLYING >, 1344
  - L4Re::Dma\_space, 1465
- None\_type
  - L4::Types::Flags< BITS\_ENUM, UNDERLYING >, 1344
- Normal
  - L4Re::Dataspace::F, 1458
- notify\_queue
  - L4virtio::Svr::Console::Device, 1868
  - L4virtio::Svr::Console::Virtio\_con, 1890
- NT\_ASRS
  - ELF binary format, 629
- NT\_AUXV
  - ELF binary format, 628
- NT\_FPREGSET
  - ELF binary format, 628
- NT\_GWINDOWS
  - ELF binary format, 628
- NT\_LWPSINFO
  - ELF binary format, 629
- NT\_LWPSTATUS
  - ELF binary format, 629
- NT\_PLATFORM
  - ELF binary format, 628
- NT\_PRCREG
  - ELF binary format, 629
- NT\_PRFPXREG
  - ELF binary format, 629
- NT\_PRPSINFO
  - ELF binary format, 628
- NT\_PRSTATUS
  - ELF binary format, 628
- NT\_PRXREG
  - ELF binary format, 628
- NT\_PSINFO
  - ELF binary format, 629
- NT\_PSTATUS
  - ELF binary format, 629
- NT\_TASKSTRUCT
  - ELF binary format, 628
- NT\_UTSNAME
  - ELF binary format, 629
- NT\_VERSION
  - ELF binary format, 629
- num
  - L4virtio::Virtqueue, 1987
- num\_interfaces
  - L4::Meta, 1210
- NVMe server, 64
- obj\_cap
  - L4::Epiface, 985
  - L4::Epiface\_t0< RPC\_IFACE, BASE >, 992
  - L4::lrqep\_t< Derived, BASE, bool >, 1178
- Object Invocation, 340
  - l4\_ipc, 343
  - l4\_ipc\_call, 345
  - l4\_ipc\_receive, 347
  - l4\_ipc\_reply\_and\_wait, 349
  - l4\_ipc\_send, 351
  - l4\_ipc\_send\_and\_wait, 352
  - l4\_ipc\_sleep, 353
  - l4\_ipc\_sleep\_ms, 354
  - l4\_ipc\_sleep\_us, 355
  - l4\_ipc\_wait, 356
  - l4\_sndfpage\_add, 357
  - l4\_syscall\_flags\_t, 342
  - L4\_SYSF\_CALL, 343
  - L4\_SYSF\_NONE, 342
  - L4\_SYSF\_OPEN\_WAIT, 343
  - L4\_SYSF\_RECV, 343
  - L4\_SYSF\_REPLY, 343
  - L4\_SYSF\_REPLY\_AND\_WAIT, 343
  - L4\_SYSF\_SEND, 343
  - L4\_SYSF\_SEND\_AND\_WAIT, 343
  - L4\_SYSF\_WAIT, 343
- Object\_registry
  - L4Re::Util::Object\_registry, 1610
- object\_size
  - cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >, 765
  - cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >, 771
- objects\_per\_slab
  - cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >, 765
  - cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >, 771

- offset
  - l4\_sched\_cpu\_set\_t, [1398](#)
- operator l4\_msgtag\_t
  - L4::Factory::S, [1009](#)
- operator new
  - Small C++ Template Library, [576](#)
- operator Null\_ptr\_check const \*
  - L4virtio::Svr::Virtqueue::Head\_desc, [1976](#)
- operator value\_type
  - L4drivers::Ro\_register\_tmpl< BITS, BLOCK >, [1439](#)
- operator!
  - cxx::Bits::Direction, [842](#)
- operator!=
  - L4vbus::Device, [1729](#)
- operator<<
  - ipc\_stream, [2226–2228](#)
- operator>>
  - ipc\_stream, [2228–2232](#)
- operator()
  - cxx::Pair\_first\_compare< Cmp, Typ >, [881](#)
- operator->
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >::Node, [822](#)
- operator=
  - L4drivers::Register\_tmpl< BITS, BLOCK >, [1435](#)
  - L4Re::Rm::Unique\_region< T >, [1552](#)
- operator==
  - L4Re::Video::Color\_component, [1666](#)
  - L4Re::Video::Pixel\_info, [1682](#)
  - L4vbus::Device, [1729](#)
- operator[]
  - cxx::Avl\_map< KEY\_TYPE, DATA\_TYPE, COMPARE, ALLOC >, [749](#)
  - cxx::Bitmap\_base, [801](#)
  - cxx::List< D, Alloc >, [863](#)
  - L4drivers::Register\_block< MAX\_BITS, BLOCK >, [1426, 1427](#)
  - L4drivers::Ro\_register\_block< MAX\_BITS, BLOCK >, [1437](#)
- operator\*
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >::Node, [822](#)
- outchar
  - kdebug.h, [2835](#)
- outdec
  - kdebug.h, [2836](#)
- outhex12
  - kdebug.h, [2836](#)
- outhex16
  - kdebug.h, [2837](#)
- outhex20
  - kdebug.h, [2837](#)
- outhex32
  - kdebug.h, [2838](#)
- outhex64
  - kdebug.h, [2839](#)
- outhex8
  - kdebug.h, [2839](#)
- outnstring
  - kdebug.h, [2840](#)
- outstring
  - kdebug.h, [2841](#)
- outumword
  - kdebug.h, [2842](#)
- Overview, [1](#)
- Overwrite
  - L4Re::Namespace, [1517](#)
- p\_flags
  - Elf32\_Phdr, [921](#)
  - Elf64\_Phdr, [931](#)
- p\_type
  - Elf32\_Phdr, [921](#)
  - Elf64\_Phdr, [931](#)
- padding
  - L4Re::Video::Pixel\_info, [1682](#)
- page\_fault
  - L4::Pager, [1216](#)
- page\_shift
  - L4Re::Util::Dataspace\_svr, [1583](#)
- Pager
  - L4Re::Rm::F, [1547](#)
- pager
  - L4::Thread::Attr, [1306](#)
- Pair
  - cxx::Pair< First, Second >, [878](#)
- Pair\_first\_compare
  - cxx::Pair\_first\_compare< Cmp, Typ >, [881](#)
- parent
  - L4Re::Env, [1478, 1479](#)
- Parent API, [535](#)
- Parent interface, [499](#)
- parent.h
  - l4re\_parent\_signal, [2452](#)
- parse\_cmdline
  - Comfortable Command Line Parsing, [612](#)
- Partly\_resolved
  - L4Re::Namespace, [1516](#)
- PF\_ARM\_SB
  - ELF binary format, [621](#)
- PF\_MASKOS
  - ELF binary format, [630](#)
- PF\_MASKPROC
  - ELF binary format, [630](#)
- PF\_R
  - ELF binary format, [630](#)
- PF\_W
  - ELF binary format, [630](#)
- PF\_X
  - ELF binary format, [630](#)
- Phys\_space
  - L4Re::Dma\_space, [1465](#)
- pin
  - L4vbus::Gpio\_module, [1734](#)
  - L4vbus::Gpio\_pin, [1743](#)
- Pinned



- L4Re::Mem\_alloc, 1508
- Pixel\_info
  - L4Re::Video::Pixel\_info, 1678
- pkg/drivers-first/include/ARCH-amd64/asm\_access.h, 2013
- pkg/drivers-first/include/ARCH-arm/asm\_access.h, 2013
- pkg/drivers-first/include/ARCH-arm64/asm\_access.h, 2014
- pkg/drivers-first/include/ARCH-mips/asm\_access.h, 2015
- pkg/drivers-first/include/ARCH-ppc32/asm\_access.h, 2015
- pkg/drivers-first/include/ARCH-riscv/asm\_access.h, 2016
- pkg/drivers-first/include/ARCH-sparc/asm\_access.h, 2017
- pkg/drivers-first/include/ARCH-x86/asm\_access.h, 2017
- pkg/drivers-first/include/asm\_access\_gen.h, 2018
- pkg/drivers-first/include/hw\_mmio\_register\_block, 2018
- pkg/drivers-first/include/hw\_register\_block, 2019
- pkg/drivers-first/include/io\_regblock.h, 2025
- pkg/drivers-first/include/io\_regblock\_port.h, 2027
- pkg/drivers-first/include/Makefile, 2028
- pkg/drivers-first/include/poll\_timeout\_counter.h, 2028
- pkg/drivers-first/uart/include/Makefile, 2028
- pkg/drivers-first/uart/include/uart\_16550.h, 2029
- pkg/drivers-first/uart/include/uart\_16550\_dw.h, 2030
- pkg/drivers-first/uart/include/uart\_apb.h, 2030
- pkg/drivers-first/uart/include/uart\_base.h, 2031
- pkg/drivers-first/uart/include/uart\_cadence.h, 2032
- pkg/drivers-first/uart/include/uart\_dcc-v6.h, 2032
- pkg/drivers-first/uart/include/uart\_dm.h, 2033
- pkg/drivers-first/uart/include/uart\_dummy.h, 2033
- pkg/drivers-first/uart/include/uart\_geni.h, 2034
- pkg/drivers-first/uart/include/uart\_imx.h, 2034
- pkg/drivers-first/uart/include/uart\_leon3.h, 2035
- pkg/drivers-first/uart/include/uart\_linfex.h, 2036
- pkg/drivers-first/uart/include/uart\_lpuart.h, 2036
- pkg/drivers-first/uart/include/uart\_mvebu.h, 2037
- pkg/drivers-first/uart/include/uart\_of.h, 2037
- pkg/drivers-first/uart/include/uart\_omap35x.h, 2038
- pkg/drivers-first/uart/include/uart\_pl011.h, 2038
- pkg/drivers-first/uart/include/uart\_s3c2410.h, 2039
- pkg/drivers-first/uart/include/uart\_sa1000.h, 2040
- pkg/drivers-first/uart/include/uart\_sbi.h, 2041
- pkg/drivers-first/uart/include/uart\_sh.h, 2041
- pkg/l4re-core/ned/lib/include/cmd\_control, 2041
- pkg/l4re-core/ned/lib/include/Makefile, 2028
- Platform Control C API, 241
  - l4\_platform\_ctl\_cpu\_allow\_shutdown, 242
  - l4\_platform\_ctl\_cpu\_disable, 243
  - l4\_platform\_ctl\_cpu\_enable, 244
  - l4\_platform\_ctl\_set\_task\_asid, 244
  - l4\_platform\_ctl\_system\_shutdown, 245
  - l4\_platform\_ctl\_system\_suspend, 246
- pm\_resume
  - L4vbus::Pm< DEC >, 1764
- pm\_suspend
  - L4vbus::Pm< DEC >, 1764
- Poll\_timeout\_counter
  - L4::Poll\_timeout\_counter, 1223
- Poll\_timeout\_kipclock
  - L4::Poll\_timeout\_kipclock, 1226
- pop\_front
  - cxx::Bits::Smart\_ptr\_list< ITEM >, 846
  - cxx::H\_list< T, POLICY >, 854
  - cxx::S\_list< T, POLICY >, 889
- port
  - L4virtio::Svr::Console::Device, 1869
  - L4virtio::Svr::Console::Virtio\_con, 1891
- port\_add
  - L4virtio::Svr::Console::Virtio\_con, 1891
- Port\_added
  - L4virtio::Svr::Console::Port, 1884
- Port\_disabled
  - L4virtio::Svr::Console::Port, 1884
- Port\_failed
  - L4virtio::Svr::Console::Port, 1884
- port\_io.h
  - l4util\_ioport\_map, 2090
- Port\_name
  - L4virtio::Svr::Console::Control\_message, 1860
- Port\_open
  - L4virtio::Svr::Console::Control\_message, 1860
  - L4virtio::Svr::Console::Port, 1884
- port\_open
  - L4virtio::Svr::Console::Virtio\_con, 1892
- port\_read
  - L4virtio::Svr::Console::Device, 1870
- Port\_ready
  - L4virtio::Svr::Console::Control\_message, 1860
  - L4virtio::Svr::Console::Port, 1884
- port\_remove
  - L4virtio::Svr::Console::Virtio\_con, 1894
- Port\_status
  - L4virtio::Svr::Console::Port, 1883
- port\_write
  - L4virtio::Svr::Console::Device, 1871
- Ports
  - L4::Type\_info::Demand\_t< CAPS, FLAGS, MEM, PORTS >, 1320
- print
  - L4Re::Log, 1503
- println
  - L4Re::Log, 1503
- process
  - L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >, 1588
- process\_device\_ready
  - L4virtio::Svr::Console::Device, 1872
  - L4virtio::Svr::Console::Virtio\_con, 1895
- process\_port\_open
  - L4virtio::Svr::Console::Device, 1873
  - L4virtio::Svr::Console::Virtio\_con, 1896
- process\_port\_ready
  - L4virtio::Svr::Console::Device, 1873

- L4virtio::Svr::Console::Virtio\_con, [1896](#)
- process\_request
  - L4virtio::Driver::Block\_device, [1804](#)
- process\_used\_queue
  - L4virtio::Driver::Block\_device, [1805](#)
- Producer, [555](#), [566](#)
  - l4shmc\_chunk\_ready, [556](#)
  - l4shmc\_chunk\_ready\_sig, [556](#)
  - l4shmc\_chunk\_try\_to\_take, [557](#)
  - l4shmc\_chunk\_try\_to\_take\_for\_overwriting, [557](#)
  - l4shmc\_chunk\_try\_to\_take\_for\_writing, [557](#)
  - l4shmc\_is\_chunk\_clear, [558](#)
  - l4shmc\_trigger, [566](#)
- prog.mk - Application Role, [30](#)
- Program Input and Output, [23](#)
- Programming for L4Re, [9](#)
- PROTO\_ANY
  - L4, [671](#)
- proto\_dispatch
  - L4::Kobject\_typeid< T >, [1203](#)
  - L4::Server\_object\_t< IFACE, BASE >, [1267](#)
- PROTO\_EMPTY
  - L4, [671](#)
- protocols.h
  - L4RE\_PROTO\_DATASPACE, [2554](#)
  - L4RE\_PROTO\_DEBUG, [2554](#)
  - L4RE\_PROTO\_DMA\_SPACE, [2554](#)
  - L4RE\_PROTO\_EVENT, [2554](#)
  - L4RE\_PROTO\_GOOS, [2554](#)
  - L4RE\_PROTO\_INHIBITOR, [2554](#)
  - L4RE\_PROTO\_MMIO\_SPACE, [2554](#)
  - L4RE\_PROTO\_NAMESPACE, [2554](#)
  - L4RE\_PROTO\_PARENT, [2554](#)
  - L4RE\_PROTO\_RM, [2554](#)
  - L4RE\_PROTO\_RSVD\_1, [2554](#)
  - L4re\_protocols, [2554](#)
- pSLIM\_BMAP\_START\_LSB
  - bitmap.h, [2423](#)
- PT\_DYNAMIC
  - ELF binary format, [630](#)
- PT\_GNU\_EH\_FRAME
  - ELF binary format, [631](#)
- PT\_GNU\_RELRO
  - ELF binary format, [631](#)
- PT\_GNU\_STACK
  - ELF binary format, [631](#)
- PT\_HIOS
  - ELF binary format, [631](#)
- PT\_HIPROC
  - ELF binary format, [631](#)
- PT\_INTERP
  - ELF binary format, [630](#)
- PT\_L4\_AUX
  - ELF binary format, [631](#)
- PT\_L4\_KIP
  - ELF binary format, [631](#)
- PT\_L4\_STACK
  - ELF binary format, [631](#)
- PT\_LOAD
  - ELF binary format, [630](#)
- PT\_LOOS
  - ELF binary format, [630](#)
- PT\_LOPROC
  - ELF binary format, [631](#)
- PT\_NOTE
  - ELF binary format, [630](#)
- PT\_NULL
  - ELF binary format, [630](#)
- PT\_NUM
  - ELF binary format, [630](#)
- PT\_PHDR
  - ELF binary format, [630](#)
- PT\_SHLIB
  - ELF binary format, [630](#)
- PT\_TLS
  - ELF binary format, [630](#)
- Pthread Support, [24](#)
- ptr
  - cxx::Ref\_ptr< T, CNT >, [886](#)
- push\_back
  - cxx::List\_item, [869](#)
- push\_front
  - cxx::List\_item, [870](#)
- put
  - L4::Factory::S, [1009](#), [1010](#)
  - L4::lpc::loststream, [1053](#)
  - L4::lpc::Ostream, [1091](#), [1092](#)
  - L4Re::Event\_buffer\_t< PAYLOAD >, [1491](#)
- qconfig
  - L4virtio::Svr::Dev\_config, [1910](#)
- query
  - L4Re::Namespace, [1517](#)
- query\_log\_name
  - L4::Debugger, [973](#)
- query\_log\_typeid
  - L4::Debugger, [974](#)
- Query\_result\_flags
  - L4Re::Namespace, [1516](#)
- Query\_timeout
  - L4Re::Namespace, [1516](#)
- R
  - cxx::Bits::Direction, [842](#)
  - L4Re::Dataspace::F, [1458](#)
  - L4Re::Rm::F, [1546](#)
- r
  - L4drivers::Register\_block< MAX\_BITS, BLOCK >, [1427](#), [1428](#)
  - L4drivers::Ro\_register\_block< MAX\_BITS, BLOCK >, [1437](#)
  - L4Re::Video::Pixel\_info, [1683](#)
  - L4vcpu::Vcpu, [1787](#), [1788](#)
- R\_386\_32
  - ELF binary format, [631](#)
- R\_386\_COPY
  - ELF binary format, [631](#)



- R\_386\_GLOB\_DAT
  - ELF binary format, [631](#)
- R\_386\_GOT32
  - ELF binary format, [631](#)
- R\_386\_GOTOFF
  - ELF binary format, [631](#)
- R\_386\_GOTPC
  - ELF binary format, [631](#)
- R\_386\_JMP\_SLOT
  - ELF binary format, [631](#)
- R\_386\_NONE
  - ELF binary format, [631](#)
- R\_386\_NUM
  - ELF binary format, [632](#)
- R\_386\_PC32
  - ELF binary format, [631](#)
- R\_386\_PLT32
  - ELF binary format, [631](#)
- R\_386\_RELATIVE
  - ELF binary format, [631](#)
- R\_386\_TLS\_DTPMOD32
  - ELF binary format, [632](#)
- R\_386\_TLS\_DTPOFF32
  - ELF binary format, [632](#)
- R\_386\_TLS\_GD
  - ELF binary format, [631](#)
- R\_386\_TLS\_GD\_32
  - ELF binary format, [631](#)
- R\_386\_TLS\_GD\_CALL
  - ELF binary format, [631](#)
- R\_386\_TLS\_GD\_POP
  - ELF binary format, [631](#)
- R\_386\_TLS\_GD\_PUSH
  - ELF binary format, [631](#)
- R\_386\_TLS\_GOTIE
  - ELF binary format, [631](#)
- R\_386\_TLS\_IE
  - ELF binary format, [631](#)
- R\_386\_TLS\_IE\_32
  - ELF binary format, [631](#)
- R\_386\_TLS\_LDM
  - ELF binary format, [631](#)
- R\_386\_TLS\_LDM\_32
  - ELF binary format, [631](#)
- R\_386\_TLS\_LDM\_CALL
  - ELF binary format, [631](#)
- R\_386\_TLS\_LDM\_POP
  - ELF binary format, [631](#)
- R\_386\_TLS\_LDM\_PUSH
  - ELF binary format, [631](#)
- R\_386\_TLS\_LDO\_32
  - ELF binary format, [631](#)
- R\_386\_TLS\_LE
  - ELF binary format, [631](#)
- R\_386\_TLS\_LE\_32
  - ELF binary format, [632](#)
- R\_386\_TLS\_TPOFF
  - ELF binary format, [631](#)
- R\_386\_TLS\_TPOFF32
  - ELF binary format, [632](#)
- R\_AARCH64\_NONE
  - ELF binary format, [632](#)
- R\_ARM\_ABS12
  - ELF binary format, [632](#)
- R\_ARM\_ABS16
  - ELF binary format, [632](#)
- R\_ARM\_ABS32
  - ELF binary format, [632](#)
- R\_ARM\_ABS8
  - ELF binary format, [632](#)
- R\_ARM\_COPY
  - ELF binary format, [632](#)
- R\_ARM\_GLOB\_DAT
  - ELF binary format, [632](#)
- R\_ARM\_GOT32
  - ELF binary format, [632](#)
- R\_ARM\_GOTOFF
  - ELF binary format, [632](#)
- R\_ARM\_GOTPC
  - ELF binary format, [632](#)
- R\_ARM\_JUMP\_SLOT
  - ELF binary format, [632](#)
- R\_ARM\_NONE
  - ELF binary format, [632](#)
- R\_ARM\_NUM
  - ELF binary format, [632](#)
- R\_ARM\_PC24
  - ELF binary format, [632](#)
- R\_ARM\_PLT32
  - ELF binary format, [632](#)
- R\_ARM\_REL32
  - ELF binary format, [632](#)
- R\_ARM\_RELATIVE
  - ELF binary format, [632](#)
- R\_ARM\_THM\_PC11
  - ELF binary format, [632](#)
- R\_ARM\_THM\_PC9
  - ELF binary format, [632](#)
- R\_X86\_64\_16
  - ELF binary format, [633](#)
- R\_X86\_64\_32
  - ELF binary format, [633](#)
- R\_X86\_64\_32S
  - ELF binary format, [633](#)
- R\_X86\_64\_64
  - ELF binary format, [633](#)
- R\_X86\_64\_8
  - ELF binary format, [633](#)
- R\_X86\_64\_COPY
  - ELF binary format, [633](#)
- R\_X86\_64\_DTPMOD64
  - ELF binary format, [633](#)
- R\_X86\_64\_DTPOFF32
  - ELF binary format, [633](#)
- R\_X86\_64\_DTPOFF64
  - ELF binary format, [633](#)

- R\_X86\_64\_GLOB\_DAT
  - ELF binary format, [633](#)
- R\_X86\_64\_GOT32
  - ELF binary format, [633](#)
- R\_X86\_64\_GOTPCREL
  - ELF binary format, [633](#)
- R\_X86\_64\_GOTTPOFF
  - ELF binary format, [633](#)
- R\_X86\_64\_JUMP\_SLOT
  - ELF binary format, [633](#)
- R\_X86\_64\_NONE
  - ELF binary format, [633](#)
- R\_X86\_64\_PC16
  - ELF binary format, [633](#)
- R\_X86\_64\_PC32
  - ELF binary format, [633](#)
- R\_X86\_64\_PC8
  - ELF binary format, [633](#)
- R\_X86\_64\_PLT32
  - ELF binary format, [633](#)
- R\_X86\_64\_RELATIVE
  - ELF binary format, [633](#)
- R\_X86\_64\_TLSGD
  - ELF binary format, [633](#)
- R\_X86\_64\_TLSLD
  - ELF binary format, [633](#)
- R\_X86\_64\_TPOFF32
  - ELF binary format, [633](#)
- R\_X86\_64\_TPOFF64
  - ELF binary format, [633](#)
- RAM configuration, [75](#)
- Random number support, [646](#)
  - l4util\_rand, [647](#)
  - l4util\_srand, [647](#)
- rate
  - L4::Uart, [1360](#)
- rbegin
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [817](#)
  - cxx::Bits::Bst< Node, Get\_key, Compare >, [834](#), [835](#)
- rcv\_cap
  - L4::lpc\_svr::Server\_iface, [1140](#), [1141](#)
- rcv\_endpoint.h
  - L4\_RCV\_EP\_BIND\_OP, [2877](#)
  - L4\_rcv\_ep\_ops, [2877](#)
- read
  - L4::lpc, [684](#)
  - L4::Vcon, [1374](#)
  - L4drivers::Ro\_register\_tmpl< BITS, BLOCK >, [1439](#)
- read\_with\_flags
  - L4::Vcon, [1375](#)
- readv
  - L4Re::Vfs::Regular\_file, [1660](#)
- ready
  - L4virtio::Virtqueue, [1988](#)
- realloc\_rcv\_cap
  - L4::lpc\_svr::Server\_iface, [1142](#)
  - L4Re::Util::Br\_manager, [1560](#)
- Realtime API, [381](#)
- receive
  - L4::lpc::lstream, [1062](#)
  - L4::lrc, [1162](#)
- Receiver, [540](#)
- Ref\_ptr
  - cxx::Ref\_ptr< T, CNT >, [885](#)
- refresh
  - L4Re::Util::Video::Goos\_svr, [1626](#)
  - L4Re::Video::View, [1686](#)
- Region
  - L4Re::Rm, [1536](#)
- Region map API, [535](#)
- Region map interface, [499](#)
  - l4re\_rm\_attach, [501](#)
  - l4re\_rm\_attach\_srv, [502](#)
  - L4RE\_RM\_CACHING\_SHIFT, [501](#)
  - l4re\_rm\_detach, [503](#)
  - l4re\_rm\_detach\_ds, [504](#)
  - l4re\_rm\_detach\_ds\_unmap, [505](#)
  - l4re\_rm\_detach\_srv, [506](#)
  - l4re\_rm\_detach\_unmap, [506](#)
  - L4RE\_RM\_F\_ATTACH\_FLAGS, [501](#)
  - L4RE\_RM\_F\_CACHE\_BUFFERED, [501](#)
  - L4RE\_RM\_F\_CACHE\_NORMAL, [501](#)
  - L4RE\_RM\_F\_CACHE\_UNCACHED, [501](#)
  - L4RE\_RM\_F\_CACHING, [501](#)
  - L4RE\_RM\_F\_EAGER\_MAP, [501](#)
  - L4RE\_RM\_F\_IN\_AREA, [501](#)
  - L4RE\_RM\_F\_NO\_ALIAS, [500](#)
  - L4RE\_RM\_F\_NO\_EAGER\_MAP, [501](#)
  - L4RE\_RM\_F\_PAGER, [500](#)
  - L4RE\_RM\_F\_R, [500](#)
  - L4RE\_RM\_F\_RESERVED, [501](#)
  - L4RE\_RM\_F\_SEARCH\_ADDR, [501](#)
  - l4re\_rm\_find, [507](#)
  - l4re\_rm\_find\_srv, [508](#)
  - l4re\_rm\_flags\_values, [500](#)
  - l4re\_rm\_free\_area, [509](#)
  - l4re\_rm\_free\_area\_srv, [510](#)
  - L4RE\_RM\_REGION\_FLAGS, [501](#)
  - l4re\_rm\_reserve\_area, [510](#)
  - l4re\_rm\_reserve\_area\_srv, [511](#)
  - l4re\_rm\_show\_lists, [512](#)
- Region\_flag\_shifts
  - L4Re::Rm, [1537](#)
- Region\_flags
  - L4Re::Rm::F, [1546](#)
- Region\_flags\_mask
  - L4Re::Rm::F, [1547](#)
- register\_del\_irq
  - L4::Thread, [1296](#)
- register\_driver\_irq
  - L4virtio::Svr::Device\_t< DATA >, [1924](#)
- register\_ds
  - L4virtio::Device, [1797](#)

- L4virtio::Driver::Device, [1817](#)
- Register\_flags
  - L4Re::Namespace, [1516](#)
- register\_irq\_obj
  - L4::Registry\_iface, [1235](#)
  - L4Re::Util::Object\_registry, [1611](#)
- register\_obj
  - L4::Registry\_iface, [1235](#), [1236](#)
  - L4Re::Namespace, [1518](#)
  - L4Re::Util::Object\_registry, [1611](#), [1612](#)
- Registry\_server
  - L4Re::Util::Registry\_server< LOOP\_HOOKS >, [1619](#), [1620](#)
- release
  - cxx::Ref\_ptr< T, CNT >, [886](#)
  - L4Re::Inhibitor, [1498](#)
  - L4Re::Rm::Unique\_region< T >, [1553](#)
  - L4Re::Util::Counting\_cap\_alloc< COUNTER-TYPE, Dbg >, [1576](#)
  - L4Re::Util::Dataspace\_svr, [1583](#)
- release\_cap
  - L4::Task, [1286](#)
- release\_ioport
  - L4vbus::Vbus, [1774](#)
- remove
  - cxx::Avl\_tree< Node, Get\_key, Compare >, [760](#)
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [818](#)
  - cxx::H\_list< T, POLICY >, [855](#)
  - cxx::List\_item, [871](#)
  - L4::lpc\_svr::Timeout\_queue, [1149](#)
  - L4virtio::Svr::Driver\_mem\_list\_t< DATA >, [1935](#)
- remove\_all
  - cxx::Bits::Bst< Node, Get\_key, Compare >, [836](#)
- remove\_timeout
  - L4::lpc\_svr::Server\_iface, [1142](#)
  - L4::lpc\_svr::Timeout\_queue\_hooks< HOOKS, BR\_MAN >, [1156](#)
- remove\_tree
  - cxx::Bits::Bst< Node, Get\_key, Compare >, [837](#)
- rename
  - L4Re::Vfs::Directory, [1637](#)
- rend
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [819](#)
  - cxx::Bits::Bst< Node, Get\_key, Compare >, [838](#)
- replace
  - cxx::H\_list< T, POLICY >, [855](#)
- reply\_and\_wait
  - L4::lpc::lostream, [1053](#), [1054](#)
- Reply\_compound
  - Server-Side IPC framework, [542](#)
- Reply\_mode
  - Server-Side IPC framework, [542](#)
- Reply\_separate
  - Server-Side IPC framework, [542](#)
- request\_ioport
  - L4vbus::Vbus, [1774](#)
- reserve\_area
  - L4Re::Rm, [1544](#)
- Reserved
  - L4::Kip::Mem\_desc, [1181](#)
  - L4Re::Rm::F, [1547](#)
- reset
  - L4::lpc::lostream, [1055](#)
  - L4::lpc::lstream, [1063](#)
  - L4Re::Rm::Unique\_region< T >, [1553](#)
- reset\_cmd
  - L4virtio::Svr::Dev\_config, [1911](#)
- reset\_device
  - L4virtio::Svr::Console::Virtio\_con, [1897](#)
- reset\_queue
  - L4virtio::Svr::Dev\_config, [1912](#)
- reset\_queue\_config
  - L4virtio::Svr::Device\_t< DATA >, [1925](#)
- Resize
  - L4virtio::Svr::Console::Control\_message, [1860](#)
- Rights\_mask
  - L4::lpc::Cap< T >, [1041](#)
  - L4Re::Dataspace::F, [1458](#)
  - L4Re::Rm::F, [1546](#)
- ringbuf.h
  - l4shmc\_rb\_attach\_receiver, [2633](#)
  - l4shmc\_rb\_attach\_sender, [2634](#)
  - l4shmc\_rb\_deinit\_buffer, [2634](#)
  - l4shmc\_rb\_init\_buffer, [2634](#)
  - l4shmc\_rb\_init\_receiver, [2635](#)
  - l4shmc\_rb\_receiver\_copy\_out, [2636](#)
  - l4shmc\_rb\_receiver\_notify\_done, [2636](#)
  - l4shmc\_rb\_receiver\_read\_next\_size, [2636](#)
  - l4shmc\_rb\_receiver\_wait\_for\_data, [2636](#)
  - l4shmc\_rb\_sender\_alloc\_packet, [2637](#)
  - l4shmc\_rb\_sender\_commit\_packet, [2637](#)
  - l4shmc\_rb\_sender\_next\_copy\_in, [2638](#)
  - l4shmc\_rb\_sender\_put\_data, [2638](#)
- rm
  - L4Re::Env, [1479](#)
- rmdir
  - L4Re::Vfs::Directory, [1637](#)
- Ro
  - L4Re::Dataspace::F, [1458](#)
  - L4Re::Namespace, [1516](#)
- root
  - L4vbus::Vbus, [1775](#)
- round\_order
  - L4, [675](#)
- Rs
  - L4Re::Namespace, [1517](#)
- run\_thread
  - L4::Scheduler, [1247](#)
- running
  - L4virtio::Svr::Dev\_status, [1919](#)
- Runtime\_error
  - L4::Runtime\_error, [1240](#)
- RW
  - L4Re::Dataspace::F, [1458](#)

- L4Re::Rm::F, [1547](#)
- Rw
  - L4Re::Namespace, [1517](#)
- Rws
  - L4Re::Namespace, [1517](#)
- RWX
  - L4Re::Dataspace::F, [1458](#)
  - L4Re::Rm::F, [1547](#)
- RX
  - L4Re::Dataspace::F, [1458](#)
  - L4Re::Rm::F, [1547](#)
- rx\_pkt
  - L4virtio::Driver::Virtio\_net\_device, [1826](#)
- rx\_queue\_size
  - L4virtio::Driver::Virtio\_net\_device, [1827](#)
- S
  - L4::Factory::S, [1008](#)
- saved\_state
  - L4vcpu::Vcpu, [1788](#)
- scan\_zero
  - cxx::Bitmap< BITS >, [794](#)
  - cxx::Bitmap\_base, [802](#)
- Scheduler, [247](#)
  - I4\_sched\_cpu\_set, [249](#)
  - I4\_sched\_param, [249](#)
  - L4\_SCHEDULER\_CLASS\_FIXED\_PRIO, [248](#)
  - L4\_SCHEDULER\_CLASS\_WFQ, [248](#)
  - L4\_scheduler\_classes, [248](#)
  - I4\_scheduler\_idle\_time, [250](#)
  - L4\_SCHEDULER\_IDLE\_TIME\_OP, [249](#)
  - I4\_scheduler\_info, [251](#)
  - L4\_SCHEDULER\_INFO\_OP, [249](#)
  - I4\_scheduler\_info\_with\_classes, [252](#)
  - I4\_scheduler\_is\_online, [253](#)
  - L4\_scheduler\_ops, [248](#)
  - I4\_scheduler\_run\_thread, [253](#)
  - L4\_SCHEDULER\_RUN\_THREAD\_OP, [249](#)
- scheduler
  - L4Re::Env, [1480](#)
- screen\_info
  - L4Re::Util::Video::Goos\_svr, [1626](#)
- Search\_addr
  - L4Re::Rm::F, [1546](#)
- segment.h
  - fiasco\_amd64\_segment\_info, [2071](#)
  - fiasco\_amd64\_set\_fs, [2072](#), [2076](#)
  - fiasco\_amd64\_set\_segment\_base, [2073](#), [2077](#)
  - L4\_AMD64\_SEGMENT\_FS, [2071](#)
  - L4\_AMD64\_SEGMENT\_GS, [2071](#)
  - L4\_sys\_segment, [2071](#)
  - L4\_task\_ldt\_x86\_consts, [2071](#), [2080](#)
  - L4\_TASK\_LDT\_X86\_ENTRY\_SIZE, [2071](#), [2080](#)
  - L4\_TASK\_LDT\_X86\_MAX\_ENTRIES, [2071](#), [2080](#)
- send
  - L4::lpc::Ostream, [1092](#)
  - L4::Vcon, [1375](#)
  - L4virtio::Driver::Device, [1818](#)
- send\_and\_wait
  - L4virtio::Driver::Device, [1819](#)
- send\_control\_message
  - L4virtio::Svr::Console::Virtio\_con, [1897](#)
- send\_request
  - L4virtio::Driver::Block\_device, [1805](#)
- Sender, [540](#)
- Server
  - L4::Server< LOOP\_HOOKS >, [1257](#)
- Server-Side IPC framework, [540](#)
  - Reply\_compound, [542](#)
  - Reply\_mode, [542](#)
  - Reply\_separate, [542](#)
- server\_iface
  - L4::Epiface, [985](#)
- set
  - cxx::Bitfield< T, LSB, MSB >, [779](#)
  - L4::Kip::Mem\_desc, [1186](#)
  - L4::Poll\_timeout\_counter, [1224](#)
  - L4::Poll\_timeout\_kipclock, [1226](#)
  - I4\_sched\_cpu\_set\_t, [1398](#)
  - L4drivers::Register\_tmpl< BITS, BLOCK >, [1435](#)
  - L4Re::Video::Color\_component, [1666](#)
  - L4vbus::Gpio\_module, [1735](#)
  - L4vbus::Gpio\_pin, [1743](#)
  - L4vcpu::State, [1779](#)
  - L4virtio::Svr::Data\_buffer, [1902](#)
- set\_attr
  - L4::Vcon, [1376](#)
- set\_bit
  - cxx::Bitmap\_base, [802](#)
- set\_blk\_size
  - L4virtio::Svr::Block\_dev\_base< Ds\_data >, [1854](#)
- set\_config\_wce
  - L4virtio::Svr::Block\_dev\_base< Ds\_data >, [1854](#)
- set\_device\_needs\_reset
  - L4virtio::Svr::Dev\_config, [1913](#)
- set\_dirty
  - cxx::Bitfield< T, LSB, MSB >, [779](#)
- set\_discard
  - L4virtio::Svr::Block\_dev\_base< Ds\_data >, [1855](#)
- set\_fd
  - L4Re::Vfs::Fs, [1647](#)
- set\_info
  - L4Re::Video::View, [1687](#)
- set\_lock
  - L4Re::Vfs::Regular\_file, [1661](#)
- set\_mode
  - L4::lcu, [1017](#)
- set\_object\_name
  - L4::Debugger, [974](#)
- set\_raw
  - I4\_vcon\_attr\_t, [1407](#)
- set\_rcv\_cap\_flags
  - L4Re::Util::Br\_manager, [1561](#)
- set\_server
  - L4::Epiface, [986](#)
- set\_size\_max
  - L4virtio::Svr::Block\_dev\_base< Ds\_data >, [1855](#)

- set\_status
  - L4virtio::Device, [1798](#)
  - L4virtio::Svr::Dev\_config, [1913](#)
- set\_status\_flags
  - L4Re::Vfs::Generic\_file, [1650](#)
- set\_topology
  - L4virtio::Svr::Block\_dev\_base< Ds\_data >, [1855](#)
- set\_unshifted
  - cxx::Bitfield< T, LSB, MSB >, [780](#)
- set\_unshifted\_dirty
  - cxx::Bitfield< T, LSB, MSB >, [781](#)
- set\_viewport
  - L4Re::Video::View, [1687](#)
- set\_write\_zeroes
  - L4virtio::Svr::Block\_dev\_base< Ds\_data >, [1856](#)
- setup
  - L4Re::Util::Counting\_cap\_alloc< COUNTER-  
TYPE, Dbg >, [1577](#)
  - L4vbus::Gpio\_module, [1736](#)
  - L4vbus::Gpio\_pin, [1744](#)
  - L4virtio::Virtqueue, [1989](#)
- setup\_device
  - L4virtio::Driver::Block\_device, [1806](#)
  - L4virtio::Driver::Virtio\_net\_device, [1827](#)
- setup\_queue
  - L4virtio::Svr::Device\_t< DATA >, [1925](#)
- setup\_simple
  - L4virtio::Virtqueue, [1990](#)
- sh\_flags
  - Elf32\_Shdr, [924](#)
  - Elf64\_Shdr, [934](#)
- sh\_type
  - Elf32\_Shdr, [924](#)
  - Elf64\_Shdr, [934](#)
- Shared
  - L4::Kip::Mem\_desc, [1181](#)
- Shared Memory Library, [542](#)
  - l4shmc\_area\_overhead, [544](#)
  - l4shmc\_area\_size, [544](#)
  - l4shmc\_area\_size\_free, [544](#)
  - l4shmc\_attach, [544](#)
  - l4shmc\_chunk\_overhead, [545](#)
  - l4shmc\_connect\_chunk\_signal, [545](#)
  - l4shmc\_create, [546](#)
  - l4shmc\_get\_client\_nr, [546](#)
  - l4shmc\_get\_initialized\_clients, [547](#)
  - l4shmc\_mark\_client\_initialized, [547](#)
- Shared\_cap
  - L4Re, [695](#)
  - L4Re::Util, [711](#)
- shared\_cap
  - L4Re, [695](#)
  - L4Re::Util, [711](#)
- Shared\_del\_cap
  - L4Re, [696](#)
  - L4Re::Util, [712](#)
- shared\_del\_cap
  - L4Re, [696](#)
- L4Re::Util, [713](#)
- SHF\_ALLOC
  - ELF binary format, [634](#)
- SHF\_ARM\_COMDEF
  - ELF binary format, [633](#)
- SHF\_ARM\_ENTRYSECT
  - ELF binary format, [633](#)
- SHF\_EXECINSTR
  - ELF binary format, [634](#)
- SHF\_GROUP
  - ELF binary format, [634](#)
- SHF\_INFO\_LINK
  - ELF binary format, [634](#)
- SHF\_LINK\_ORDER
  - ELF binary format, [634](#)
- SHF\_MASKOS
  - ELF binary format, [634](#)
- SHF\_MASKPROC
  - ELF binary format, [634](#)
- SHF\_MERGE
  - ELF binary format, [634](#)
- SHF\_OS\_NONCONFORMING
  - ELF binary format, [634](#)
- SHF\_STRINGS
  - ELF binary format, [634](#)
- SHF\_TLS
  - ELF binary format, [634](#)
- SHF\_WRITE
  - ELF binary format, [634](#)
- shift
  - L4Re::Video::Color\_component, [1666](#)
- Shift\_type
  - cxx::Bitfield< T, LSB, MSB >, [776](#)
- SHN\_ABS
  - ELF binary format, [634](#)
- SHN\_COMMON
  - ELF binary format, [634](#)
- SHN\_HIPROC
  - ELF binary format, [634](#)
- SHN\_HIRESERVE
  - ELF binary format, [634](#)
- SHN\_LOPROC
  - ELF binary format, [634](#)
- SHN\_LORESERVE
  - ELF binary format, [634](#)
- SHN\_UNDEF
  - ELF binary format, [634](#)
- SHT\_DYNAMIC
  - ELF binary format, [635](#)
- SHT\_DYNSYM
  - ELF binary format, [635](#)
- SHT\_FINI\_ARRAY
  - ELF binary format, [635](#)
- SHT\_GROUP
  - ELF binary format, [635](#)
- SHT\_HASH
  - ELF binary format, [635](#)
- SHT\_HIOS

- ELF binary format, [635](#)
- SHT\_HIPROC
  - ELF binary format, [635](#)
- SHT\_HIUSER
  - ELF binary format, [635](#)
- SHT\_INIT\_ARRAY
  - ELF binary format, [635](#)
- SHT\_LOOS
  - ELF binary format, [635](#)
- SHT\_LOPROC
  - ELF binary format, [635](#)
- SHT\_LOUSER
  - ELF binary format, [635](#)
- SHT\_NOBITS
  - ELF binary format, [635](#)
- SHT\_NOTE
  - ELF binary format, [635](#)
- SHT\_NULL
  - ELF binary format, [635](#)
- SHT\_NUM
  - ELF binary format, [635](#)
- SHT\_PREINIT\_ARRAY
  - ELF binary format, [635](#)
- SHT\_PROGBITS
  - ELF binary format, [635](#)
- SHT\_REL
  - ELF binary format, [635](#)
- SHT\_RELA
  - ELF binary format, [635](#)
- SHT\_SHLIB
  - ELF binary format, [635](#)
- SHT\_STRTAB
  - ELF binary format, [635](#)
- SHT\_SYMTAB
  - ELF binary format, [635](#)
- SHT\_SYMTAB\_SHNDX
  - ELF binary format, [635](#)
- shutdown
  - L4::Uart, [1360](#)
  - L4::Uart\_apb, [1366](#)
- si
  - l4\_vcpu\_regs\_t, [1412](#)
- Sigma0 API, [566](#)
  - l4sigma0\_debug\_dump, [568](#)
  - L4SIGMA0\_IPCERROR, [568](#)
  - l4sigma0\_map\_anypage, [568](#)
  - l4sigma0\_map\_errstr, [569](#)
  - l4sigma0\_map\_iomem, [569](#)
  - l4sigma0\_map\_kip, [570](#)
  - l4sigma0\_map\_mem, [570](#)
  - L4SIGMA0\_NOFPAGE, [568](#)
  - L4SIGMA0\_NOTALIGNED, [568](#)
  - L4SIGMA0\_OK, [568](#)
  - l4sigma0\_return\_flags\_t, [568](#)
  - L4SIGMA0\_SMALLERFPAGE, [568](#)
- Sigma0, the Root-Pager, [45](#)
- signal
  - L4Re::Parent, [1525](#)
- Signals, [558](#)
  - l4shmc\_add\_signal, [559](#)
  - l4shmc\_attach\_signal, [559](#)
  - l4shmc\_check\_magic, [561](#)
  - l4shmc\_get\_signal, [561](#)
  - l4shmc\_signal\_cap, [562](#)
- size
  - L4::Kip::Mem\_desc, [1187](#)
  - L4Re::Dataspace, [1456](#)
  - L4Re::Video::Color\_component, [1667](#)
  - L4virtio::Svr::Driver\_mem\_region\_t< DATA >, [1942](#)
- skip
  - L4::lpc::Istream, [1063](#)
  - L4virtio::Svr::Data\_buffer, [1902](#)
- slab\_size
  - cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >, [765](#)
  - cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >, [771](#)
- Small C++ Template Library, [572](#)
  - clamp, [574](#)
  - max, [574](#)
  - min, [575](#)
  - operator new, [576](#)
- Small\_buf
  - L4::lpc::Small\_buf, [1099](#)
- Smart\_cap
  - L4::Smart\_cap< T, SMART >, [1276](#)
- snd\_base
  - L4::Cap\_base, [961](#)
- Space\_attrb
  - L4Re::Dma\_space, [1465](#)
- Spaces and Mappings, [19](#)
- Split\_ds
  - L4Re::Rm, [1537](#)
- Src\_dev\_handle
  - L4vbus::lcu, [1749](#)
- Src\_types
  - L4vbus::lcu, [1749](#)
- ss
  - l4\_exc\_regs\_t, [1386](#)
- stack
  - L4Re::Video::View, [1688](#)
- start
  - L4::Kip::Mem\_desc, [1187](#)
  - L4virtio::Svr::Request\_processor, [1947](#), [1949](#)
- start\_request
  - L4virtio::Driver::Block\_device, [1807](#)
- starts\_with
  - cxx::String, [904](#)
- startup
  - L4::Uart, [1360](#)
  - L4::Uart\_apb, [1366](#)
- State
  - L4vcpu::State, [1778](#)
- state
  - L4vcpu::Vcpu, [1788](#), [1789](#)

- stats\_time
  - L4::Thread, [1297](#)
- status
  - L4virtio::Svr::Dev\_config, [1915](#)
- STB\_GLOBAL
  - ELF binary format, [635](#)
- STB\_HIOS
  - ELF binary format, [635](#)
- STB\_HIPROC
  - ELF binary format, [635](#)
- STB\_LOCAL
  - ELF binary format, [635](#)
- STB\_LOOS
  - ELF binary format, [635](#)
- STB\_LOPROC
  - ELF binary format, [635](#)
- STB\_WEAK
  - ELF binary format, [635](#)
- Str\_cp\_in
  - L4::lpc::Str\_cp\_in< T >, [1103](#)
- str\_cp\_in
  - L4::lpc, [684](#)
- String
  - cxx::String, [902](#)
- Strong
  - L4Re::Namespace, [1517](#)
- STT\_FILE
  - ELF binary format, [636](#)
- STT\_FUNC
  - ELF binary format, [636](#)
- STT\_HIOS
  - ELF binary format, [636](#)
- STT\_HIPROC
  - ELF binary format, [636](#)
- STT\_LOOS
  - ELF binary format, [636](#)
- STT\_LOPROC
  - ELF binary format, [636](#)
- STT\_NOTYPE
  - ELF binary format, [636](#)
- STT\_OBJECT
  - ELF binary format, [636](#)
- STT\_SECTION
  - ELF binary format, [636](#)
- sub\_type
  - L4::Kip::Mem\_desc, [1188](#)
- Super\_pages
  - L4Re::Mem\_alloc, [1508](#)
- supports
  - L4::Meta, [1210](#)
- switch\_log
  - L4::Debugger, [975](#)
- switch\_to
  - L4::Thread, [1298](#)
- symlink
  - L4Re::Vfs::Directory, [1638](#)
- system\_shutdown
  - L4::Platform\_control, [1221](#)
- system\_suspend
  - L4::Platform\_control, [1222](#)
- tag
  - L4::lpc::Istream, [1064](#)
  - L4::lpc::Ostream, [1093](#)
  - L4::lpc::Varg, [1107](#)
- take
  - L4Re::Util::Counting\_cap\_alloc< COUNTER-  
TYPE, Dbg >, [1577](#)
  - L4Re::Util::Dataspace\_svr, [1584](#)
- Task, [255](#)
  - L4\_FP\_ALL\_SPACES, [256](#)
  - L4\_FP\_DELETE\_OBJ, [256](#)
  - L4\_FP\_OTHER\_SPACES, [256](#)
  - l4\_task\_add\_ku\_mem, [256](#)
  - l4\_task\_cap\_equal, [257](#)
  - l4\_task\_cap\_valid, [258](#)
  - l4\_task\_delete\_obj, [259](#)
  - l4\_task\_map, [259](#)
  - l4\_task\_release\_cap, [261](#)
  - l4\_task\_unmap, [262](#)
  - l4\_task\_unmap\_batch, [263](#)
  - l4\_task\_vgicc\_map, [264](#)
  - l4\_unmap\_flags\_t, [256](#)
- task
  - L4Re::Env, [1480](#)
  - L4vcpu::Vcpu, [1789](#)
- test
  - L4::Poll\_timeout\_kipclock, [1227](#)
- test.mk - Test Application Role, [33](#)
- Thread, [265](#)
  - l4\_thread\_arm\_set\_tpidruro, [269](#)
  - L4\_THREAD\_CONTROL\_ALIEN, [268](#)
  - L4\_THREAD\_CONTROL\_BIND\_TASK, [268](#)
  - L4\_thread\_control\_flags, [267](#)
  - L4\_THREAD\_CONTROL\_MR\_IDX\_BIND\_TASK, [268](#)
  - L4\_THREAD\_CONTROL\_MR\_IDX\_BIND\_UTCB, [268](#)
  - L4\_THREAD\_CONTROL\_MR\_IDX\_EXC\_HANDLER, [268](#)
  - L4\_THREAD\_CONTROL\_MR\_IDX\_FLAG\_VALS, [268](#)
  - L4\_THREAD\_CONTROL\_MR\_IDX\_FLAGS, [268](#)
  - L4\_THREAD\_CONTROL\_MR\_IDX\_PAGER, [268](#)
  - L4\_thread\_control\_mr\_indices, [268](#)
  - L4\_THREAD\_CONTROL\_SET\_EXC\_HANDLER, [268](#)
  - L4\_THREAD\_CONTROL\_SET\_PAGER, [268](#)
  - L4\_THREAD\_CONTROL\_UX\_NATIVE, [268](#)
  - l4\_thread\_ex\_regs, [269](#)
  - L4\_THREAD\_EX\_REGS\_ARCH\_MASK, [268](#)
  - L4\_THREAD\_EX\_REGS\_ARM\_SET\_EL\_EL0, [269](#)
  - L4\_THREAD\_EX\_REGS\_ARM\_SET\_EL\_EL1, [269](#)
  - L4\_THREAD\_EX\_REGS\_ARM\_SET\_EL\_KEEP, [269](#)



- L4\_THREAD\_EX\_REGS\_ARM\_SET\_EL\_MASK, 269
- L4\_THREAD\_EX\_REGS\_CANCEL, 268
- L4\_thread\_ex\_regs\_flags, 268
- L4\_thread\_ex\_regs\_flags\_arm, 268
- l4\_thread\_ex\_regs\_ret, 270
- l4\_thread\_ex\_regs\_ret\_u, 271
- L4\_THREAD\_EX\_REGS\_TRIGGER\_EXCEPTION, 268
- l4\_thread\_ex\_regs\_u, 272
- l4\_thread\_modify\_sender\_add, 274
- l4\_thread\_modify\_sender\_commit, 275
- l4\_thread\_modify\_sender\_start, 275
- l4\_thread\_register\_del\_irq, 276
- l4\_thread\_stats\_time, 277
- l4\_thread\_switch, 278
- l4\_thread\_vcpu\_control, 278
- l4\_thread\_vcpu\_control\_ext, 279
- l4\_thread\_vcpu\_control\_ext\_u, 280
- l4\_thread\_vcpu\_control\_u, 281
- l4\_thread\_vcpu\_resume\_commit, 283
- l4\_thread\_vcpu\_resume\_start, 284
- l4\_thread\_yield, 284
- Thread control, 285
  - l4\_thread\_control\_alien, 286
  - l4\_thread\_control\_bind, 287
  - l4\_thread\_control\_commit, 288
  - l4\_thread\_control\_exc\_handler, 289
  - l4\_thread\_control\_pager, 289
  - l4\_thread\_control\_start, 290
  - l4\_thread\_control\_ux\_host\_syscall, 291
- Thread Control Registers (TCRs), 399
- thread.h
  - \_\_L4UTIL\_THREAD\_FUNC, 2915
- throw\_error
  - L4Re, 707
- throw\_ipc\_exception
  - IPC Helpers, 412, 413
- timed\_out
  - L4::Poll\_timeout\_counter, 1224
  - L4::Poll\_timeout\_kipclock, 1228
- timeout
  - L4::lpc\_svr::Timeout, 1145
- timeout\_expired
  - L4::lpc\_svr::Timeout\_queue, 1150
- Timeouts, 381
  - l4\_ipc\_timeout, 383
  - L4\_IPC\_TIMEOUT\_0, 383
  - l4\_rcv\_timeout, 384
  - l4\_snd\_timeout, 384
  - l4\_timeout, 385
  - l4\_timeout\_abs, 385
  - l4\_timeout\_get, 386
  - l4\_timeout\_is\_absolute, 387
  - l4\_timeout\_rel, 388
  - l4\_timeout\_rel\_get, 388
  - l4\_timeout\_s, 383
  - l4\_timeout\_t, 383
  - L4\_TIMEOUT\_US\_MAX, 383
  - l4\_utcb\_mr64\_idx, 389
- Timestamp Counter, 647
  - l4\_busy\_wait\_ns, 648
  - l4\_busy\_wait\_us, 649
  - l4\_calibrate\_tsc, 649
  - l4\_get\_hz, 650
  - l4\_ns\_to\_tsc, 650
  - l4\_rdpmc, 651
  - l4\_rdpmc\_32, 651
  - l4\_rdtsc, 651
  - l4\_rdtsc\_32, 652
  - l4\_tsc\_init, 652
  - l4\_tsc\_to\_ns, 653
  - l4\_tsc\_to\_s\_and\_ns, 653
  - l4\_tsc\_to\_us, 654
- To\_default
  - L4Re::Namespace, 1516
- To\_device
  - L4Re::Dma\_space, 1465
- to\_irq
  - L4vbus::Gpio\_pin, 1745
- To\_non\_blocking
  - L4Re::Namespace, 1516
- total\_objects
  - cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >, 768
  - cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >, 773
- total\_size
  - L4virtio::Virtqueue, 1991, 1992
- trigger
  - L4::Triggerable, 1312
- trunc\_order
  - L4, 676
- Trusted
  - L4Re::Namespace, 1517
- Tutorial, 7
- tx
  - L4virtio::Driver::Virtio\_net\_device, 1828
- tx\_queue\_size
  - L4virtio::Driver::Virtio\_net\_device, 1829
- type
  - L4::lpc::Varg, 1108
  - L4::Kip::Mem\_desc, 1188
  - L4Re::Vfs::Be\_file\_system, 1633
  - L4Re::Vfs::File\_system, 1643
- types
  - L4\_TYPES\_FLAGS\_OPS\_DEF, 2764
- types.h
  - l4\_capability\_next, 2146
- unbind
  - L4::lcu, 1018
  - L4::lommu, 1028
- Uncacheable
  - L4Re::Dataspace::F, 1458
- Undefined
  - L4::Kip::Mem\_desc, 1181



- Unique\_cap
  - L4Re, [697](#)
  - L4Re::Util, [713](#)
- unique\_cap
  - L4Re, [697](#)
  - L4Re::Util, [714](#)
- Unique\_del\_cap
  - L4Re, [697](#)
  - L4Re::Util, [714](#)
- unique\_del\_cap
  - L4Re, [698](#)
  - L4Re::Util, [715](#)
- Unique\_region
  - L4Re::Rm::Unique\_region< T >, [1550](#), [1551](#)
- unlink
  - L4Re::Namespace, [1519](#)
  - L4Re::Vfs::Directory, [1638](#)
- unlock\_all\_locks
  - L4Re::Vfs::Be\_file, [1630](#)
  - L4Re::Vfs::Generic\_file, [1651](#)
- unmap
  - L4::Task, [1287](#)
  - L4Re::Dma\_space, [1468](#)
- unmap\_batch
  - L4::Task, [1288](#)
- unmask
  - L4::Irq, [1163](#)
  - L4::Irq\_eoi, [1166](#)
- unregister\_obj
  - L4::Registry\_iface, [1237](#)
  - L4Re::Util::Object\_registry, [1613](#)
- up
  - L4::Semaphore, [1253](#)
- used\_align
  - L4virtio::Virtqueue, [1993](#)
- Used\_elem
  - L4virtio::Virtqueue::Used\_elem, [2003](#)
- used\_size
  - L4virtio::Virtqueue, [1993](#)
- utcb\_area
  - L4Re::Env, [1480](#), [1481](#)
- Utility Functions, [576](#)
  - l4\_sleep, [578](#)
  - l4\_touch\_ro, [579](#)
  - l4\_touch\_rw, [580](#)
  - l4\_usleep, [580](#)
  - l4util\_micros2l4to, [580](#)
  - l4util\_splitlog2\_hdl, [581](#)
  - l4util\_splitlog2\_size, [582](#)
- Uvmm, the virtual machine monitor, [67](#)
- ux\_host\_syscall
  - L4::Thread::Attr, [1307](#)
- V\_flags
  - L4Re::Video::View, [1686](#)
- val
  - cxx::Bitfield< T, LSB, MSB >, [782](#)
- val\_dirty
  - cxx::Bitfield< T, LSB, MSB >, [783](#)
- val\_unshifted
  - cxx::Bitfield< T, LSB, MSB >, [785](#)
- valid
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >::Node, [822](#)
  - L4virtio::Svr::Virtqueue::Head\_desc, [1976](#)
- validate
  - L4::Cap\_base, [961](#), [962](#)
- value
  - L4::lpc::Varg, [1108](#)
- Varg\_list\_ref
  - L4::lpc::Varg\_list\_ref, [1113](#)
- vbe\_ctrl\_info
  - l4util\_l4mod\_info, [1709](#)
- Vbus API, [536](#)
- vbus.h
  - L4VBUS\_ICU\_SRC\_DEV\_HANDLE, [3004](#)
  - l4vbus\_icu\_src\_types, [3003](#)
  - L4VBUS\_NULL, [3003](#)
  - L4VBUS\_ROOT\_BUS, [3003](#)
- vbus\_interfaces.h
  - L4VBUS\_IFACE\_SHIFT, [3010](#)
  - l4vbus\_iface\_type\_t, [3010](#)
  - L4VBUS\_INTERFACE\_BUS, [3011](#)
  - L4VBUS\_INTERFACE\_GENERIC, [3011](#)
  - L4VBUS\_INTERFACE\_GPIO, [3011](#)
  - L4VBUS\_INTERFACE\_ICU, [3011](#)
  - L4VBUS\_INTERFACE\_PCI, [3011](#)
  - L4VBUS\_INTERFACE\_PCIDEV, [3011](#)
  - L4VBUS\_INTERFACE\_PM, [3011](#)
  - l4vbus\_subinterface\_supported, [3011](#)
- vbus\_types.h
  - L4VBUS\_DEVICE\_F\_CHILDREN, [3017](#)
  - l4vbus\_device\_flags\_t, [3016](#)
  - L4VBUS\_RESOURCE\_BUS, [3017](#)
  - L4VBUS\_RESOURCE\_DMA\_DOMAIN, [3017](#)
  - L4VBUS\_RESOURCE\_F\_MEM\_MMIO\_READ, [3017](#)
  - L4VBUS\_RESOURCE\_F\_MEM\_MMIO\_WRITE, [3017](#)
  - L4VBUS\_RESOURCE\_F\_MEM\_R, [3017](#)
  - L4VBUS\_RESOURCE\_F\_MEM\_W, [3017](#)
  - l4vbus\_resource\_flags\_t, [3017](#)
  - L4VBUS\_RESOURCE\_GPIO, [3017](#)
  - L4VBUS\_RESOURCE\_INVALID, [3017](#)
  - L4VBUS\_RESOURCE\_IRQ, [3017](#)
  - L4VBUS\_RESOURCE\_MAX, [3017](#)
  - L4VBUS\_RESOURCE\_MEM, [3017](#)
  - L4VBUS\_RESOURCE\_PORT, [3017](#)
  - l4vbus\_resource\_type\_t, [3017](#)
- vcon.h
  - L4\_vcon\_read\_flags, [2936](#)
  - L4\_VCON\_READ\_SIZE\_MASK, [2936](#)
  - L4\_VCON\_READ\_STAT\_BREAK, [2936](#)
  - L4\_VCON\_READ\_STAT\_DONE, [2936](#)
- vCPU API, [292](#)
  - L4\_VCPU\_F\_EXCEPTIONS, [295](#)
  - L4\_VCPU\_F\_FPU\_ENABLED, [295](#)

- L4\_VCPU\_F\_IRQ, [294](#)
- L4\_VCPU\_F\_PAGE\_FAULTS, [294](#)
- L4\_VCPU\_F\_USER\_MODE, [295](#)
- L4\_VCPU\_OFFSET\_EXT\_INFOS, [295](#), [296](#)
- L4\_VCPU\_OFFSET\_EXT\_STATE, [295](#), [296](#)
- L4\_VCPU\_SF\_IRQ\_PENDING, [296](#)
- L4\_vcpu\_state\_flags, [294](#)
- L4\_vcpu\_state\_offset, [295](#), [296](#)
- L4\_vcpu\_sticky\_flags, [296](#)
- vCPU Support Library, [654](#)
  - l4vcpu\_irq\_disable, [655](#)
  - l4vcpu\_irq\_disable\_save, [656](#)
  - l4vcpu\_irq\_enable, [657](#)
  - l4vcpu\_irq\_restore, [658](#)
  - l4vcpu\_is\_irq\_entry, [659](#)
  - l4vcpu\_is\_page\_fault\_entry, [659](#)
  - l4vcpu\_print\_state, [660](#)
  - l4vcpu\_wait\_for\_event, [661](#)
- vcpu.h
  - l4\_vcpu\_check\_version, [2941](#)
- vcpu\_control
  - L4::Thread, [1298](#)
- vcpu\_control\_ext
  - L4::Thread, [1299](#)
- vcpu\_resume\_commit
  - L4::Thread, [1300](#)
- vcpu\_resume\_start
  - L4::Thread, [1301](#)
- version
  - l4\_vcpu\_state\_t, [1415](#)
- vgicc\_map
  - L4::Vm, [1382](#)
- vicu
  - L4vbus::lcu, [1749](#)
- Video API, [513](#)
  - F\_l4re\_video\_goos\_auto\_refresh, [515](#)
  - F\_l4re\_video\_goos\_dynamic\_buffers, [515](#)
  - F\_l4re\_video\_goos\_dynamic\_views, [515](#)
  - F\_l4re\_video\_goos\_pointer, [515](#)
  - F\_l4re\_video\_view\_above, [515](#)
  - F\_l4re\_video\_view\_dyn\_allocated, [515](#)
  - F\_l4re\_video\_view\_flags\_mask, [515](#)
  - F\_l4re\_video\_view\_none, [515](#)
  - F\_l4re\_video\_view\_set\_background, [515](#)
  - F\_l4re\_video\_view\_set\_buffer, [515](#)
  - F\_l4re\_video\_view\_set\_buffer\_offset, [515](#)
  - F\_l4re\_video\_view\_set\_bytes\_per\_line, [515](#)
  - F\_l4re\_video\_view\_set\_flags, [515](#)
  - F\_l4re\_video\_view\_set\_pixel, [515](#)
  - F\_l4re\_video\_view\_set\_position, [515](#)
  - l4re\_video\_goos\_create\_buffer, [515](#)
  - l4re\_video\_goos\_create\_view, [516](#)
  - l4re\_video\_goos\_delete\_buffer, [516](#)
  - l4re\_video\_goos\_delete\_view, [516](#)
  - l4re\_video\_goos\_get\_static\_buffer, [516](#)
  - l4re\_video\_goos\_get\_view, [517](#)
  - l4re\_video\_goos\_info, [517](#)
  - l4re\_video\_goos\_info\_flags\_t, [514](#)
  - l4re\_video\_goos\_refresh, [518](#)
  - l4re\_video\_view\_get\_info, [518](#)
  - l4re\_video\_view\_info\_flags\_t, [515](#)
  - l4re\_video\_view\_refresh, [518](#)
  - l4re\_video\_view\_set\_info, [519](#)
  - l4re\_video\_view\_set\_viewport, [519](#)
  - l4re\_video\_view\_stack, [519](#)
  - l4re\_video\_view\_t, [514](#)
- view
  - L4Re::Video::Goos, [1674](#)
- view\_info
  - L4Re::Util::Video::Goos\_svr, [1626](#)
- Virtio\_con
  - L4virtio::Svr::Console::Virtio\_con, [1888](#)
- Virtual Console, [296](#)
  - l4\_vcon\_attr\_t, [298](#)
  - L4\_VCON\_ECHO, [299](#)
  - l4\_vcon\_get\_attr, [299](#)
  - l4\_vcon\_get\_attr\_u, [300](#)
  - L4\_vcon\_i\_flags, [298](#)
  - L4\_VCON\_ICANON, [299](#)
  - L4\_VCON\_ICRNL, [298](#)
  - L4\_VCON\_IGNCR, [298](#)
  - L4\_VCON\_INLCR, [298](#)
  - L4\_vcon\_l\_flags, [299](#)
  - L4\_vcon\_o\_flags, [299](#)
  - L4\_VCON\_OCRNL, [299](#)
  - L4\_VCON\_ONLCR, [299](#)
  - L4\_VCON\_ONLRET, [299](#)
  - l4\_vcon\_read, [301](#)
  - L4\_VCON\_READ\_SIZE, [299](#)
  - l4\_vcon\_read\_u, [302](#)
  - l4\_vcon\_read\_with\_flags, [303](#)
  - l4\_vcon\_send, [304](#)
  - l4\_vcon\_send\_u, [305](#)
  - l4\_vcon\_set\_attr, [306](#)
  - l4\_vcon\_set\_attr\_raw, [307](#)
  - l4\_vcon\_set\_attr\_u, [308](#)
  - L4\_vcon\_size\_consts, [299](#)
  - l4\_vcon\_write, [309](#)
  - L4\_VCON\_WRITE\_SIZE, [299](#)
  - l4\_vcon\_write\_u, [309](#)
- Virtual Machines, [311](#)
- Virtual Registers (UTCBs), [389](#)
  - l4\_utcb\_br, [391](#)
  - l4\_utcb\_mr, [392](#)
  - l4\_utcb\_t, [391](#)
  - l4\_utcb\_tcr, [393](#)
- VM API for SVM, [311](#)
- VM API for TZ, [312](#)
- VM API for VMX, [313](#)
  - l4\_ext\_vcpu\_state\_vmx\_t, [315](#)
  - L4\_VM\_VMX\_BASIC\_REG, [316](#)
  - L4\_vm\_vmx\_caps\_regs, [316](#)
  - l4\_vm\_vmx\_clear, [317](#)
  - L4\_VM\_VMX\_CR0\_FIXED0\_REG, [316](#)
  - L4\_VM\_VMX\_CR0\_FIXED1\_REG, [316](#)
  - L4\_VM\_VMX\_CR4\_FIXED0\_REG, [316](#)

- L4\_VM\_VMX\_CR4\_FIXED1\_REG, [316](#)
  - L4\_vm\_vmx\_dfl1\_regs, [316](#)
  - L4\_VM\_VMX\_ENTRY\_CTL5\_DFL1\_REG, [316](#)
  - L4\_VM\_VMX\_EPT\_VPID\_CAP\_REG, [316](#)
  - L4\_VM\_VMX\_EXIT\_CTL5\_DFL1\_REG, [316](#)
  - l4\_vm\_vmx\_field\_len, [318](#)
  - l4\_vm\_vmx\_field\_order, [319](#)
  - l4\_vm\_vmx\_get\_caps, [319](#)
  - l4\_vm\_vmx\_get\_caps\_default1, [320](#)
  - l4\_vm\_vmx\_get\_cr2\_index, [320](#)
  - l4\_vm\_vmx\_get\_hw\_vmcs, [321](#)
  - L4\_VM\_VMX\_MISC\_REG, [316](#)
  - L4\_VM\_VMX\_NESTED\_REVISION, [316](#)
  - L4\_VM\_VMX\_NUM\_CAPS\_REGS, [316](#)
  - L4\_VM\_VMX\_NUM\_DFL1\_REGS, [316](#)
  - L4\_VM\_VMX\_PINBASED\_CTL5\_DFL1\_REG, [316](#)
  - L4\_VM\_VMX\_PROCBASED\_CTL52\_REG, [316](#)
  - L4\_VM\_VMX\_PROCBASED\_CTL5\_DFL1\_REG, [316](#)
  - l4\_vm\_vmx\_ptr\_load, [321](#)
  - l4\_vm\_vmx\_read, [322](#)
  - l4\_vm\_vmx\_read\_16, [323](#)
  - l4\_vm\_vmx\_read\_32, [323](#)
  - l4\_vm\_vmx\_read\_64, [324](#)
  - l4\_vm\_vmx\_read\_nat, [325](#)
  - l4\_vm\_vmx\_set\_hw\_vmcs, [325](#)
  - L4\_vm\_vmx\_sw\_fields, [316](#)
  - L4\_VM\_VMX\_TRUE\_ENTRY\_CTL5\_REG, [316](#)
  - L4\_VM\_VMX\_TRUE\_EXIT\_CTL5\_REG, [316](#)
  - L4\_VM\_VMX\_TRUE\_PINBASED\_CTL5\_REG, [316](#)
  - L4\_VM\_VMX\_TRUE\_PROCBASED\_CTL5\_REG, [316](#)
  - L4\_VM\_VMX\_VMCS\_CR2, [317](#)
  - L4\_VM\_VMX\_VMCS\_ENUM\_REG, [316](#)
  - L4\_VM\_VMX\_VMCS\_MSR\_CSTAR, [317](#)
  - L4\_VM\_VMX\_VMCS\_MSR\_KERNEL\_GS\_BASE, [317](#)
  - L4\_VM\_VMX\_VMCS\_MSR\_LSTAR, [317](#)
  - L4\_VM\_VMX\_VMCS\_MSR\_STAR, [317](#)
  - L4\_VM\_VMX\_VMCS\_MSR\_SYSCALL\_MASK, [317](#)
  - L4\_VM\_VMX\_VMCS\_MSR\_TSC\_AUX, [317](#)
  - L4\_VM\_VMX\_VMCS\_XCR0, [317](#)
  - l4\_vm\_vmx\_write, [326](#)
  - l4\_vm\_vmx\_write\_16, [327](#)
  - l4\_vm\_vmx\_write\_32, [328](#)
  - l4\_vm\_vmx\_write\_64, [328](#)
  - l4\_vm\_vmx\_write\_nat, [329](#)
  - l4\_vmx\_offset\_table\_t, [315](#)
- W
- L4Re::Dataspace::F, [1458](#)
  - L4Re::Rm::F, [1547](#)
- W\_bits
- cxx::Bitmap\_base, [798](#)
- wait
- L4::lpc::Istream, [1065](#), [1066](#)
  - L4::Irq, [1164](#)
  - L4virtio::Driver::Device, [1820](#)
  - wait\_for\_event
    - L4vcpu::Vcpu, [1790](#)
  - wait\_for\_next\_used
    - L4virtio::Driver::Device, [1821](#)
  - wait\_rx
    - L4virtio::Driver::Virtio\_net\_device, [1830](#)
  - Wd\_16bit
    - L4Re::Mmio\_space, [1512](#)
  - Wd\_32bit
    - L4Re::Mmio\_space, [1512](#)
  - Wd\_64bit
    - L4Re::Mmio\_space, [1512](#)
  - Wd\_8bit
    - L4Re::Mmio\_space, [1512](#)
  - Word\_bytes
    - L4::lpc::Msg, [687](#)
  - word\_index
    - cxx::Bitmap\_base, [803](#)
  - write
    - L4::Uart, [1361](#)
    - L4::Uart\_apb, [1366](#)
    - L4::Vcon, [1377](#)
    - L4drivers::Register\_tmpl< BITS, BLOCK >, [1435](#)
  - write\_bfm\_t
    - L4virtio::Virtqueue::Desc::Flags, [2000](#)
  - write\_now
    - cxx, [666](#)
  - writew
    - L4Re::Vfs::Regular\_file, [1661](#)
- X
- L4Re::Dataspace::F, [1458](#)
  - L4Re::Rm::F, [1547](#)
- x86 Virtual Registers (UTCB), [400](#)
- L4\_UTCB\_BUF\_REGS\_OFFSET, [401](#)
  - L4\_utcb\_consts\_x86, [401](#)
  - L4\_UTCB\_EXCEPTION\_REGS\_SIZE, [401](#)
  - L4\_UTCB\_GENERIC\_BUFFERS\_SIZE, [401](#)
  - L4\_UTCB\_GENERIC\_DATA\_SIZE, [401](#)
  - L4\_UTCB\_INHERIT\_FPU, [401](#)
  - L4\_UTCB\_MSG\_REGS\_OFFSET, [401](#)
  - L4\_UTCB\_OFFSET, [401](#)
  - L4\_UTCB\_THREAD\_REGS\_OFFSET, [401](#)
- x86/l4/sys/\_\_kip-arch.h, [2092](#)
- x86/l4/sys/\_\_vcpu-arch.h, [2099](#), [2100](#)
- x86/l4/sys/cache.h, [2703](#), [2704](#)
- x86/l4/sys/consts.h, [2488](#)
- x86/l4/sys/ipc-invoke.h, [3021](#)
- x86/l4/sys/ktrace\_events.h, [2107](#)
- x86/l4/sys/l4int.h, [2861](#)
- x86/l4/sys/linkage.h, [2112](#), [2113](#)
- x86/l4/sys/segment.h, [2079](#), [2080](#)
- x86/l4/sys/utcb.h, [2928](#), [2930](#)
- x86/l4/sys/vm.h, [2118](#)
- x86/l4/util/bitops\_arch.h, [2122](#)
- x86/l4/util/cpu.h, [2128](#), [2129](#)
- x86/l4/util/idt.h, [2044](#), [2045](#)
- x86/l4/util/irq.h, [2825](#), [2826](#)

x86/l4/util/l4\_macros.h, [2132](#), [2133](#)  
x86/l4/util/mbi\_argv.h, [2136](#), [2137](#)  
x86/l4/util/perform.h, [2052](#), [2053](#)  
x86/l4/util/port\_io.h, [2085](#), [2086](#)  
x86/l4/util/rdtsc.h, [2062](#), [2064](#)  
x86/l4/util/spin.h, [2068](#), [2069](#)  
x86/l4f/l4/sys/ipc-l42-gcc3-nopic.h, [3022](#)  
x86/l4f/l4/sys/ipc.h, [2803](#), [2804](#)  
x86/l4f/l4/sys/segment.h, [2081](#), [2082](#)  
x86/l4f/l4/util/port\_io.h, [2088](#), [2091](#)