

L4Re Operating System Framework – Interface and Usage Documentation

Generated on Tue Oct 17 2023 12:44:38 for L4Re Operating System Framework – Interface
and Usage Documentation by Doxygen 1.9.8

Tue Oct 17 2023 12:44:38

1 Overview	1
2 Introduction	3
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
3 Tutorial	7
3.1 Customizations	8
4 Programming for L4Re	9
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
5 L4Re Servers	43
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	49
5.3.1.2 Access to L4Re::Env Capabilities	49
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	52
5.3.2 Command Line Options	52
5.4 Io, the Io Server	53
5.5 l4vio_net_p2p, a virtual network point-to-point link	58
5.6 Uvmm, the virtual machine monitor	60
5.6.1 RAM configuration	68
5.7 NVMe server	69
5.8 Mag, the GUI Multiplexer	71
5.9 Cons, the Console Multiplexer	73

5.10 AHCI driver	74
6 Bootstrap, the L4 kernel bootstrapper	77
7 Deprecated List	81
8 Topic Index	83
8.1 Topics	83
9 Namespace Index	87
9.1 Namespace List	87
10 Hierarchical Index	89
10.1 Class Hierarchy	89
11 Data Structure Index	101
11.1 Data Structures	101
12 File Index	117
12.1 File List	117
13 Topic Documentation	129
13.1 Base API	129
13.1.1 Detailed Description	132
13.1.2 Basic Macros	132
13.1.2.1 Detailed Description	133
13.1.2.2 Macro Definition Documentation	134
13.1.2.3 Function Documentation	135
13.1.3 C++ IPC Interface Definition.	136
13.1.3.1 Detailed Description	136
13.1.3.2 Internal Helpers	136
13.1.4 Cache Consistency	137
13.1.4.1 Detailed Description	137
13.1.4.2 Function Documentation	137
13.1.5 Capabilities	140
13.1.5.1 Detailed Description	141
13.1.5.2 Typedef Documentation	141
13.1.5.3 Enumeration Type Documentation	141
13.1.5.4 Function Documentation	142
13.1.6 Error codes	144
13.1.6.1 Detailed Description	145
13.1.6.2 Enumeration Type Documentation	145
13.1.7 Fiasco extensions	146
13.1.7.1 Detailed Description	147
13.1.7.2 Function Documentation	147

13.1.7.3 Fiasco real time scheduling extensions	149
13.1.7.4 Kernel Debugger	150
13.1.7.5 Kernel Tracing	156
13.1.8 Flex pages	159
13.1.8.1 Detailed Description	161
13.1.8.2 Enumeration Type Documentation	161
13.1.8.3 Function Documentation	166
13.1.9 Integer Types	176
13.1.9.1 Detailed Description	177
13.1.10 Kernel Interface Page	177
13.1.10.1 Detailed Description	179
13.1.10.2 Enumeration Type Documentation	179
13.1.10.3 Function Documentation	179
13.1.10.4 Fiasco-UX Virtual devices	184
13.1.10.5 Memory descriptors (C version)	185
13.1.11 Kernel Objects	190
13.1.11.1 Detailed Description	191
13.1.11.2 DMA space	191
13.1.11.3 Factory	192
13.1.11.4 IPC-Gate API	200
13.1.11.5 IRQs	203
13.1.11.6 Interrupt controller	217
13.1.11.7 Kernel-provided semaphore	233
13.1.11.8 L4 kernel object type information	235
13.1.11.9 Platform Control C API	236
13.1.11.10 Scheduler	241
13.1.11.11 Task	249
13.1.11.12 Thread	259
13.1.11.13 Virtual Console	289
13.1.11.14 Virtual Machines	303
13.1.12 Memory operations.	319
13.1.12.1 Detailed Description	319
13.1.12.2 Enumeration Type Documentation	319
13.1.12.3 Function Documentation	320
13.1.13 Memory related	321
13.1.13.1 Detailed Description	322
13.1.13.2 Macro Definition Documentation	322
13.1.13.3 Enumeration Type Documentation	324
13.1.13.4 Function Documentation	325
13.1.14 Object Invocation	328
13.1.14.1 Detailed Description	330
13.1.14.2 Enumeration Type Documentation	331

13.1.14.3 Function Documentation	332
13.1.14.4 Error Handling	347
13.1.14.5 Message Items	352
13.1.14.6 Message Tag	356
13.1.14.7 Realtime API	370
13.1.14.8 Timeouts	370
13.1.14.9 Virtual Registers (UTCBS)	378
13.2 EDID parsing functionality	390
13.2.1 Detailed Description	391
13.2.2 Enumeration Type Documentation	391
13.2.2.1 Libedid_consts	391
13.2.3 Function Documentation	391
13.2.3.1 libedid_check_header()	391
13.2.3.2 libedid_checksum()	392
13.2.3.3 libedid_dump()	392
13.2.3.4 libedid_dump_standard_timings()	392
13.2.3.5 libedid_num_ext_blocks()	393
13.2.3.6 libedid_pnp_id()	393
13.2.3.7 libedid_prefered_resolution()	393
13.2.3.8 libedid_revision()	394
13.2.3.9 libedid_version()	394
13.3 IO interface	394
13.3.1 Detailed Description	395
13.3.2 Typedef Documentation	395
13.3.2.1 l4io_resource_t	395
13.3.3 Enumeration Type Documentation	395
13.3.3.1 l4io_device_types_t	395
13.3.3.2 l4io_iomem_flags_t	396
13.3.3.3 l4io_resource_types_t	396
13.3.4 Function Documentation	396
13.3.4.1 l4io_has_resource()	396
13.3.4.2 l4io_lookup_device()	397
13.3.4.3 l4io_lookup_resource()	397
13.3.4.4 l4io_release_iomem()	398
13.3.4.5 l4io_release_ioport()	398
13.3.4.6 l4io_request_iomem()	398
13.3.4.7 l4io_request_iomem_region()	399
13.3.4.8 l4io_request_ioport()	400
13.3.4.9 l4io_request_resource_iomem()	400
13.4 IRQ handling library	401
13.4.1 Detailed Description	401
13.4.2 Interface for asynchronous ISR handlers.	401

13.4.2.1 Detailed Description	402
13.4.2.2 Function Documentation	402
13.4.2.3 Interface for asynchronous ISR handlers with a given IRQ capability.	403
13.4.3 Interface using direct functionality.	404
13.4.3.1 Detailed Description	405
13.4.3.2 Function Documentation	405
13.4.3.3 Interface using direct functionality.	408
13.5 L4 IPC Opcodes	410
13.5.1 Detailed Description	411
13.5.2 Enumeration Type Documentation	411
13.5.2.1 L4_icu_opcode	411
13.5.2.2 L4_ipc_gate_ops	412
13.5.2.3 L4_platform_ctl_ops	412
13.5.2.4 L4_task_ops	413
13.5.2.5 L4_thread_ops	413
13.5.2.6 L4_vcon_ops	414
13.6 L4 VIRTIO Interface	414
13.6.1 Detailed Description	415
13.6.2 L4 VIRTIO Block Device	415
13.6.2.1 Detailed Description	415
13.6.2.2 Enumeration Type Documentation	415
13.6.3 L4 VIRTIO Input Device	416
13.6.3.1 Detailed Description	417
13.6.4 L4 VIRTIO Network Device	417
13.6.4.1 Detailed Description	418
13.6.5 L4 VIRTIO Transport Layer	418
13.6.5.1 Detailed Description	419
13.6.5.2 Typedef Documentation	420
13.6.5.3 Enumeration Type Documentation	420
13.6.5.4 Function Documentation	422
13.7 L4 Vbus functions	426
13.7.1 Detailed Description	427
13.7.2 Enumeration Type Documentation	427
13.7.2.1 L4vbus_dma_domain_assign_flags	427
13.7.3 Function Documentation	428
13.7.3.1 l4vbus_assign_dma_domain()	428
13.7.3.2 l4vbus_get_adr()	429
13.7.3.3 l4vbus_get_device()	429
13.7.3.4 l4vbus_get_device_by_hid()	430
13.7.3.5 l4vbus_get_hid()	431
13.7.3.6 l4vbus_get_next_device()	431
13.7.3.7 l4vbus_get_resource()	433

13.7.3.8 l4vbus_is_compatible()	434
13.7.3.9 l4vbus_release_ioport()	435
13.7.3.10 l4vbus_request_ioport()	435
13.7.3.11 l4vbus_vicu_get_cap()	436
13.7.4 L4vbus GPIO functions	437
13.7.4.1 Detailed Description	438
13.7.4.2 Enumeration Type Documentation	438
13.7.4.3 Function Documentation	438
13.7.5 L4vbus PCI functions	446
13.7.5.1 Detailed Description	447
13.7.5.2 Function Documentation	447
13.7.6 L4vbus power management functions	452
13.7.6.1 Detailed Description	452
13.7.6.2 Function Documentation	452
13.8 L4Re C Interface	454
13.8.1 Detailed Description	456
13.8.2 Capability allocator	456
13.8.2.1 Detailed Description	457
13.8.2.2 Function Documentation	457
13.8.3 DMA Space Interface	457
13.8.3.1 Detailed Description	458
13.8.3.2 Typedef Documentation	458
13.8.3.3 Function Documentation	458
13.8.4 Dataspace interface	461
13.8.4.1 Detailed Description	462
13.8.4.2 Enumeration Type Documentation	462
13.8.4.3 Function Documentation	462
13.8.5 Debug interface	465
13.8.5.1 Detailed Description	466
13.8.5.2 Function Documentation	466
13.8.6 Event interface	466
13.8.6.1 Detailed Description	467
13.8.6.2 Function Documentation	467
13.8.7 Initial Environment	469
13.8.7.1 Detailed Description	470
13.8.7.2 Function Documentation	470
13.8.8 Kumem allocator utility	474
13.8.9 L4Re Util C Interface	474
13.8.10 Log interface	474
13.8.10.1 Detailed Description	475
13.8.10.2 Function Documentation	475
13.8.11 Memory allocator	477

13.8.11.1 Detailed Description	478
13.8.11.2 Enumeration Type Documentation	478
13.8.11.3 Function Documentation	478
13.8.12 Namespace interface	481
13.8.12.1 Detailed Description	482
13.8.12.2 Enumeration Type Documentation	482
13.8.12.3 Function Documentation	482
13.8.13 Region map interface	485
13.8.13.1 Detailed Description	486
13.8.13.2 Enumeration Type Documentation	486
13.8.13.3 Function Documentation	487
13.8.14 Video API	499
13.8.14.1 Detailed Description	500
13.8.14.2 Typedef Documentation	500
13.8.14.3 Enumeration Type Documentation	500
13.8.14.4 Function Documentation	501
13.9 L4Re C++ Interface	506
13.9.1 Detailed Description	508
13.9.2 Auxiliary data	508
13.9.2.1 Detailed Description	509
13.9.3 C++ Exceptions	509
13.9.3.1 Detailed Description	510
13.9.4 Console API	510
13.9.4.1 Detailed Description	510
13.9.5 Debugging API	510
13.9.5.1 Detailed Description	511
13.9.6 Event API	511
13.9.6.1 Detailed Description	511
13.9.7 L4Re ELF Auxiliary Information	512
13.9.7.1 Detailed Description	513
13.9.7.2 Macro Definition Documentation	513
13.9.7.3 Enumeration Type Documentation	513
13.9.8 L4Re Protocol identifiers	514
13.9.8.1 Detailed Description	515
13.9.9 L4Re Util C++ Interface	515
13.9.9.1 Detailed Description	515
13.9.9.2 Kumem utilities	516
13.9.9.3 L4Re Capability API	517
13.9.10 Logging interface	520
13.9.10.1 Detailed Description	520
13.9.11 Name-space API	520
13.9.11.1 Detailed Description	521

13.9.12 Parent API	521
13.9.12.1 Detailed Description	521
13.9.13 Region map API	522
13.9.13.1 Detailed Description	522
13.9.14 Vbus API	523
13.9.14.1 Detailed Description	523
13.10 L4SHM-based ring buffer implementation	524
13.10.1 Detailed Description	524
13.10.2 Internal	524
13.10.3 Receiver	525
13.10.4 Sender	525
13.11 Server-Side IPC framework	525
13.11.1 Detailed Description	526
13.11.2 Enumeration Type Documentation	526
13.11.2.1 Reply_mode	526
13.12 Shared Memory Library	527
13.12.1 Detailed Description	528
13.12.2 Function Documentation	528
13.12.2.1 l4shmc_area_overhead()	528
13.12.2.2 l4shmc_area_size()	528
13.12.2.3 l4shmc_area_size_free()	529
13.12.2.4 l4shmc_attach()	529
13.12.2.5 l4shmc_chunk_overhead()	530
13.12.2.6 l4shmc_connect_chunk_signal()	530
13.12.2.7 l4shmc_create()	530
13.12.2.8 l4shmc_get_client_nr()	531
13.12.2.9 l4shmc_get_initialized_clients()	531
13.12.2.10 l4shmc_mark_client_initialized()	532
13.12.3 Chunks	532
13.12.3.1 Detailed Description	533
13.12.3.2 Function Documentation	533
13.12.3.3 Consumer	536
13.12.3.4 Producer	540
13.12.4 Signals	544
13.12.4.1 Detailed Description	545
13.12.4.2 Function Documentation	545
13.12.4.3 Consumer	547
13.12.4.4 Producer	551
13.13 Sigma0 API	552
13.13.1 Detailed Description	553
13.13.2 Enumeration Type Documentation	553
13.13.2.1 l4sigma0_return_flags_t	553

13.13.3 Function Documentation	554
13.13.3.1 l4sigma0_debug_dump()	554
13.13.3.2 l4sigma0_map_anypage()	554
13.13.3.3 l4sigma0_map_errstr()	555
13.13.3.4 l4sigma0_map_iomem()	555
13.13.3.5 l4sigma0_map_kip()	556
13.13.3.6 l4sigma0_map_mem()	556
13.13.4 Internal constants	557
13.13.4.1 Detailed Description	558
13.14 Small C++ Template Library	558
13.14.1 Detailed Description	559
13.14.2 Function Documentation	559
13.14.2.1 clamp()	559
13.14.2.2 max()	560
13.14.2.3 min()	560
13.14.2.4 operator new()	562
13.15 Utility Functions	562
13.15.1 Detailed Description	564
13.15.2 Function Documentation	564
13.15.2.1 l4_sleep()	564
13.15.2.2 l4_touch_ro()	565
13.15.2.3 l4_touch_rw()	566
13.15.2.4 l4_usleep()	566
13.15.2.5 l4util_micros2l4to()	567
13.15.2.6 l4util_splitlog2_hdl()	567
13.15.2.7 l4util_splitlog2_size()	568
13.15.3 Atomic Instructions	569
13.15.3.1 Detailed Description	571
13.15.3.2 Function Documentation	571
13.15.4 Bit Manipulation	586
13.15.4.1 Detailed Description	587
13.15.4.2 Function Documentation	587
13.15.5 Bitmap graphics and fonts	593
13.15.5.1 Detailed Description	593
13.15.5.2 Functions for rendering bitmap data in frame buffers	594
13.15.5.3 Functions for rendering bitmap fonts to frame buffers	594
13.15.6 CPU related functions	594
13.15.6.1 Detailed Description	595
13.15.6.2 Function Documentation	595
13.15.7 Comfortable Command Line Parsing	597
13.15.7.1 Detailed Description	597
13.15.7.2 Function Documentation	597

13.15.8 ELF binary format	599
13.15.8.1 Detailed Description	605
13.15.8.2 Macro Definition Documentation	605
13.15.8.3 Enumeration Type Documentation	606
13.15.9 Functions to manipulate the local IDT	623
13.15.9.1 Detailed Description	624
13.15.10 IA32 Port I/O API	624
13.15.10.1 Detailed Description	624
13.15.10.2 Function Documentation	624
13.15.11 Internal functions	630
13.15.11.1 Detailed Description	631
13.15.12 Kernel Interface Page API	631
13.15.12.1 Detailed Description	631
13.15.12.2 Macro Definition Documentation	631
13.15.12.3 Function Documentation	632
13.15.13 Low-Level Thread Functions	633
13.15.14 Random number support	633
13.15.14.1 Detailed Description	634
13.15.14.2 Function Documentation	634
13.15.15 Timestamp Counter	634
13.15.15.1 Detailed Description	635
13.15.15.2 Function Documentation	635
13.16 vCPU Support Library	641
13.16.1 Detailed Description	642
13.16.2 Function Documentation	642
13.16.2.1 l4vcpu_irq_disable()	642
13.16.2.2 l4vcpu_irq_disable_save()	643
13.16.2.3 l4vcpu_irq_enable()	644
13.16.2.4 l4vcpu_irq_restore()	645
13.16.2.5 l4vcpu_is_irq_entry()	646
13.16.2.6 l4vcpu_is_page_fault_entry()	647
13.16.2.7 l4vcpu_print_state()	647
13.16.2.8 l4vcpu_wait_for_event()	648
13.16.3 Extended vCPU support	649
13.16.3.1 Detailed Description	649
13.16.3.2 Function Documentation	649
14 Namespace Documentation	651
14.1 cxx Namespace Reference	651
14.1.1 Detailed Description	653
14.1.2 Function Documentation	653
14.1.2.1 access_once()	653

14.1.2.2 write_now()	654
14.2 cxx::Bits Namespace Reference	655
14.2.1 Detailed Description	655
14.3 L4 Namespace Reference	655
14.3.1 Detailed Description	659
14.3.2 Enumeration Type Documentation	659
14.3.2.1 anonymous enum	659
14.3.3 Function Documentation	660
14.3.3.1 cap_cast() [1/2]	660
14.3.3.2 cap_cast() [2/2]	660
14.3.3.3 cap_dynamic_cast()	661
14.3.3.4 cap_reinterpret_cast() [1/2]	662
14.3.3.5 cap_reinterpret_cast() [2/2]	662
14.3.3.6 round_order()	663
14.3.3.7 throw_ipc_exception() [1/2]	664
14.3.3.8 throw_ipc_exception() [2/2]	665
14.3.3.9 trunc_order()	665
14.4 L4::lpc Namespace Reference	666
14.4.1 Detailed Description	668
14.4.2 Function Documentation	668
14.4.2.1 buf_cp_in()	668
14.4.2.2 buf_cp_out()	669
14.4.2.3 buf_in()	669
14.4.2.4 make_cap()	670
14.4.2.5 make_cap_full()	670
14.4.2.6 make_cap_rw()	671
14.4.2.7 make_cap_rws()	672
14.4.2.8 msg_ptr()	672
14.4.2.9 read()	673
14.4.2.10 str_cp_in()	673
14.5 L4::lpc::Msg Namespace Reference	674
14.5.1 Detailed Description	675
14.5.2 Enumeration Type Documentation	675
14.5.2.1 anonymous enum	675
14.5.3 Function Documentation	675
14.5.3.1 align_to() [1/2]	675
14.5.3.2 align_to() [2/2]	677
14.5.3.3 check_size() [1/2]	678
14.5.3.4 check_size() [2/2]	678
14.5.3.5 msg_add()	679
14.5.3.6 msg_get()	679
14.6 L4::lpc_svr Namespace Reference	680

14.6.1 Detailed Description	681
14.7 L4::Typeid Namespace Reference	681
14.7.1 Detailed Description	681
14.8 L4::Types Namespace Reference	682
14.8.1 Detailed Description	682
14.9 L4Re Namespace Reference	682
14.9.1 Detailed Description	684
14.9.2 Typedef Documentation	685
14.9.2.1 Shared_cap	685
14.9.2.2 shared_cap	685
14.9.2.3 Shared_del_cap	685
14.9.2.4 shared_del_cap	686
14.9.2.5 Unique_cap	686
14.9.2.6 unique_cap	687
14.9.2.7 Unique_del_cap	687
14.9.2.8 unique_del_cap	688
14.9.3 Function Documentation	688
14.9.3.1 chkcap()	688
14.9.3.2 chkipc()	689
14.9.3.3 chksys() [1/3]	690
14.9.3.4 chksys() [2/3]	691
14.9.3.5 chksys() [3/3]	692
14.9.3.6 make_shared_cap()	693
14.9.3.7 make_shared_del_cap()	694
14.9.3.8 make_unique_cap()	695
14.9.3.9 make_unique_del_cap()	696
14.9.3.10 throw_error()	696
14.10 L4Re::Util Namespace Reference	697
14.10.1 Detailed Description	699
14.10.2 Typedef Documentation	700
14.10.2.1 Shared_cap	700
14.10.2.2 shared_cap	700
14.10.2.3 Shared_del_cap	701
14.10.2.4 shared_del_cap	702
14.10.2.5 Unique_cap	702
14.10.2.6 unique_cap	703
14.10.2.7 Unique_del_cap	703
14.10.2.8 unique_del_cap	704
14.10.3 Function Documentation	705
14.10.3.1 make_shared_cap()	705
14.10.3.2 make_shared_del_cap()	705
14.10.3.3 make_unique_cap()	705

14.10.3.4 make_unique_del_cap()	706
14.11 L4Re::Vfs Namespace Reference	706
14.11.1 Detailed Description	707
14.12 L4vbus Namespace Reference	707
14.12.1 Detailed Description	707
14.13 L4virtio Namespace Reference	708
14.13.1 Detailed Description	708
15 Data Structure Documentation	709
15.1 Block_device::Device_discard_feature Struct Reference	709
15.1.1 Detailed Description	709
15.2 Block_device::Device_mgr< DEV, FACTORY > Class Template Reference	710
15.2.1 Detailed Description	710
15.3 Block_device::Dma_region_info Struct Reference	711
15.3.1 Detailed Description	711
15.4 Block_device::Errand::Errand Class Reference	711
15.4.1 Detailed Description	714
15.4.2 Member Function Documentation	714
15.4.2.1 expired()	714
15.5 Block_device::Errand::Poll_errand Class Reference	715
15.5.1 Detailed Description	717
15.5.2 Member Function Documentation	717
15.5.2.1 expired()	717
15.6 Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, bool > Class Template Reference	718
15.6.1 Detailed Description	719
15.7 Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, true > Class Template Reference	719
15.7.1 Detailed Description	719
15.8 Block_device::Inout_block Struct Reference	720
15.8.1 Detailed Description	720
15.9 Block_device::Inout_memory< DEV > Class Template Reference	721
15.9.1 Detailed Description	721
15.10 Block_device::Mem_region_info Struct Reference	722
15.10.1 Detailed Description	722
15.11 Block_device::Partition_info Struct Reference	722
15.11.1 Detailed Description	723
15.12 Block_device::Partition_reader< DEV > Class Template Reference	723
15.12.1 Detailed Description	724
15.13 Block_device::Partitioned_device< BASE_DEV > Class Template Reference	724
15.13.1 Detailed Description	725
15.14 Block_device::Pending_request Struct Reference	725
15.14.1 Detailed Description	726

15.14.2 Member Function Documentation	726
15.14.2.1 fail_request()	726
15.14.2.2 handle_request()	726
15.14.2.3 is_owner()	727
15.15 Block_device::Pending_request::Owner Struct Reference	727
15.15.1 Detailed Description	728
15.16 Block_device::Simple_request_queue Class Reference	728
15.16.1 Detailed Description	728
15.17 cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC > Class Template Reference	729
15.17.1 Detailed Description	732
15.17.2 Constructor & Destructor Documentation	732
15.17.2.1 Avl_map()	732
15.17.3 Member Function Documentation	733
15.17.3.1 insert()	733
15.17.3.2 operator[]() [1/2]	734
15.17.3.3 operator[]() [2/2]	734
15.18 cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC > Class Template Reference	735
15.18.1 Detailed Description	739
15.19 cxx::Avl_tree< Node, Get_key, Compare > Class Template Reference	739
15.19.1 Detailed Description	743
15.19.2 Member Typedef Documentation	744
15.19.2.1 Iterator	744
15.19.3 Member Function Documentation	744
15.19.3.1 insert()	744
15.19.3.2 remove()	745
15.20 cxx::Avl_tree_node Class Reference	746
15.20.1 Detailed Description	748
15.21 cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference	748
15.21.1 Detailed Description	749
15.21.2 Member Enumeration Documentation	750
15.21.2.1 anonymous enum	750
15.21.3 Member Function Documentation	750
15.21.3.1 alloc()	750
15.21.3.2 free()	751
15.21.3.3 free_objects()	752
15.21.3.4 total_objects()	753
15.22 cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i Struct Reference	754
15.22.1 Detailed Description	754
15.23 cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference	754
15.23.1 Detailed Description	756
15.23.2 Member Enumeration Documentation	756
15.23.2.1 anonymous enum	756

15.23.3 Member Function Documentation	757
15.23.3.1 alloc()	757
15.23.3.2 free()	757
15.23.3.3 free_objects()	758
15.23.3.4 total_objects()	759
15.24 cxx::Bitfield< T, LSB, MSB > Class Template Reference	759
15.24.1 Detailed Description	761
15.24.2 Member Typedef Documentation	761
15.24.2.1 Bits_type	761
15.24.2.2 Shift_type	762
15.24.3 Member Enumeration Documentation	762
15.24.3.1 anonymous enum	762
15.24.3.2 Masks	762
15.24.4 Member Function Documentation	762
15.24.4.1 get()	762
15.24.4.2 get_unshifted()	763
15.24.4.3 set()	764
15.24.4.4 set_dirty()	764
15.24.4.5 set_unshifted()	765
15.24.4.6 set_unshifted_dirty()	766
15.24.4.7 val()	767
15.24.4.8 val_dirty()	768
15.24.4.9 val_unshifted()	771
15.25 cxx::Bitfield< T, LSB, MSB >::Value< TT > Class Template Reference	771
15.25.1 Detailed Description	772
15.26 cxx::Bitfield< T, LSB, MSB >::Value_base< TT > Class Template Reference	773
15.26.1 Detailed Description	773
15.27 cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT > Class Template Reference	774
15.27.1 Detailed Description	775
15.28 cxx::Bitmap< BITS > Class Template Reference	775
15.28.1 Detailed Description	778
15.28.2 Member Function Documentation	779
15.28.2.1 scan_zero()	779
15.29 cxx::Bitmap_base Class Reference	779
15.29.1 Detailed Description	782
15.29.2 Member Enumeration Documentation	782
15.29.2.1 anonymous enum	782
15.29.3 Member Function Documentation	783
15.29.3.1 bit() [1/2]	783
15.29.3.2 bit() [2/2]	783
15.29.3.3 bit_index()	784
15.29.3.4 clear_bit()	784

15.29.3.5 operator[]() [1/2]	784
15.29.3.6 operator[]() [2/2]	785
15.29.3.7 scan_zero()	785
15.29.3.8 set_bit()	786
15.29.3.9 word_index()	786
15.30 cxx::Bitmap_base::Bit Class Reference	787
15.30.1 Detailed Description	787
15.31 cxx::Bitmap_base::Char< BITS > Class Template Reference	788
15.31.1 Detailed Description	788
15.32 cxx::Bitmap_base::Word< BITS > Class Template Reference	788
15.32.1 Detailed Description	789
15.33 cxx::Bits::Avl_map_get_key< KEY_TYPE > Struct Template Reference	790
15.33.1 Detailed Description	790
15.34 cxx::Bits::Avl_set_get_key< KEY_TYPE > Struct Template Reference	790
15.34.1 Detailed Description	791
15.35 cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > Class Template Reference	791
15.35.1 Detailed Description	793
15.35.2 Member Enumeration Documentation	794
15.35.2.1 anonymous enum	794
15.35.3 Constructor & Destructor Documentation	794
15.35.3.1 Base_avl_set() [1/2]	794
15.35.3.2 Base_avl_set() [2/2]	795
15.35.4 Member Function Documentation	795
15.35.4.1 begin() [1/2]	795
15.35.4.2 begin() [2/2]	796
15.35.4.3 end() [1/2]	796
15.35.4.4 end() [2/2]	797
15.35.4.5 erase()	798
15.35.4.6 find_node()	798
15.35.4.7 insert()	799
15.35.4.8 lower_bound_node()	800
15.35.4.9 rbegin() [1/2]	801
15.35.4.10 rbegin() [2/2]	802
15.35.4.11 remove()	802
15.35.4.12 rend() [1/2]	803
15.35.4.13 rend() [2/2]	804
15.36 cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node Class Reference	804
15.36.1 Detailed Description	805
15.36.2 Member Function Documentation	806
15.36.2.1 operator*()	806
15.36.2.2 operator->()	806
15.36.2.3 valid()	806

15.37 cxx::Bits::Basic_list< POLICY > Class Template Reference	807
15.37.1 Detailed Description	808
15.37.2 Member Function Documentation	808
15.37.2.1 clear()	808
15.37.2.2 iter()	808
15.38 cxx::Bits::Bst< Node, Get_key, Compare > Class Template Reference	809
15.38.1 Detailed Description	812
15.38.2 Member Function Documentation	812
15.38.2.1 begin() [1/2]	812
15.38.2.2 begin() [2/2]	813
15.38.2.3 dir() [1/2]	814
15.38.2.4 dir() [2/2]	814
15.38.2.5 end() [1/2]	815
15.38.2.6 end() [2/2]	815
15.38.2.7 find()	816
15.38.2.8 find_node()	817
15.38.2.9 lower_bound_node()	817
15.38.2.10 rbegin() [1/2]	818
15.38.2.11 rbegin() [2/2]	819
15.38.2.12 remove_all()	820
15.38.2.13 remove_tree()	821
15.38.2.14 rend() [1/2]	822
15.38.2.15 rend() [2/2]	822
15.39 cxx::Bits::Bst_node Class Reference	823
15.39.1 Detailed Description	825
15.40 cxx::Bits::Direction Struct Reference	825
15.40.1 Detailed Description	826
15.40.2 Member Enumeration Documentation	826
15.40.2.1 Direction_e	826
15.40.3 Member Function Documentation	826
15.40.3.1 operator"()()	826
15.41 cxx::Bits::Smart_ptr_list< ITEM > Class Template Reference	827
15.41.1 Detailed Description	829
15.41.2 Member Function Documentation	830
15.41.2.1 pop_front()	830
15.42 cxx::Bits::Smart_ptr_list_item< T, STORE_T > Class Template Reference	830
15.42.1 Detailed Description	831
15.43 cxx::H_list< T, POLICY > Class Template Reference	832
15.43.1 Detailed Description	835
15.43.2 Member Function Documentation	835
15.43.2.1 erase()	835
15.43.2.2 insert()	836

15.43.2.3 insert_after()	836
15.43.2.4 insert_before()	837
15.43.2.5 iter()	838
15.43.2.6 pop_front()	838
15.43.2.7 remove()	839
15.43.2.8 replace()	840
15.44 cxx::H_list_item_t< ELEM_TYPE > Class Template Reference	840
15.44.1 Detailed Description	842
15.44.2 Constructor & Destructor Documentation	842
15.44.2.1 H_list_item_t()	842
15.44.2.2 ~H_list_item_t()	842
15.45 cxx::H_list_t< T > Struct Template Reference	843
15.45.1 Detailed Description	846
15.46 cxx::List< D, Alloc > Class Template Reference	846
15.46.1 Detailed Description	847
15.46.2 Member Function Documentation	847
15.46.2.1 operator[]() [1/2]	847
15.46.2.2 operator[]() [2/2]	848
15.47 cxx::List< D, Alloc >::Iter Class Reference	848
15.47.1 Detailed Description	848
15.48 cxx::List_alloc Class Reference	849
15.48.1 Detailed Description	849
15.48.2 Constructor & Destructor Documentation	849
15.48.2.1 List_alloc()	849
15.48.3 Member Function Documentation	850
15.48.3.1 alloc()	850
15.48.3.2 alloc_max()	850
15.48.3.3 avail()	851
15.48.3.4 free()	851
15.49 cxx::List_item Class Reference	852
15.49.1 Detailed Description	853
15.49.2 Member Function Documentation	853
15.49.2.1 push_back()	853
15.49.2.2 push_front()	854
15.49.2.3 remove()	855
15.50 cxx::List_item::Iter Class Reference	856
15.50.1 Detailed Description	857
15.51 cxx::List_item::T_iter< T, Poly > Class Template Reference	857
15.51.1 Detailed Description	858
15.52 cxx::Lt_functor< Obj > Struct Template Reference	859
15.52.1 Detailed Description	859
15.53 cxx::New_allocator< _Type > Class Template Reference	859

15.53.1 Detailed Description	860
15.54 <code>cx::Nothrow</code> Class Reference	860
15.54.1 Detailed Description	861
15.55 <code>cx::Pair< First, Second ></code> Struct Template Reference	861
15.55.1 Detailed Description	862
15.55.2 Constructor & Destructor Documentation	862
15.55.2.1 <code>Pair()</code>	862
15.56 <code>cx::Pair_first_compare< Cmp, Typ ></code> Class Template Reference	863
15.56.1 Detailed Description	863
15.56.2 Constructor & Destructor Documentation	863
15.56.2.1 <code>Pair_first_compare()</code>	863
15.56.3 Member Function Documentation	864
15.56.3.1 <code>operator()</code>	864
15.57 <code>cx::Ref_obj_list_item< T ></code> Struct Template Reference	864
15.57.1 Detailed Description	866
15.58 <code>cx::Ref_ptr< T, CNT ></code> Class Template Reference	866
15.58.1 Detailed Description	867
15.58.2 Constructor & Destructor Documentation	868
15.58.2.1 <code>Ref_ptr()</code> [1/3]	868
15.58.2.2 <code>Ref_ptr()</code> [2/3]	868
15.58.2.3 <code>Ref_ptr()</code> [3/3]	868
15.58.3 Member Function Documentation	869
15.58.3.1 <code>get()</code>	869
15.58.3.2 <code>ptr()</code>	869
15.58.3.3 <code>release()</code>	869
15.59 <code>cx::S_list< T, POLICY ></code> Class Template Reference	870
15.59.1 Detailed Description	872
15.59.2 Member Function Documentation	872
15.59.2.1 <code>pop_front()</code>	872
15.60 <code>cx::Slab< Type, Slab_size, Max_free, Alloc ></code> Class Template Reference	873
15.60.1 Detailed Description	875
15.60.2 Member Function Documentation	876
15.60.2.1 <code>alloc()</code>	876
15.60.2.2 <code>free()</code>	876
15.61 <code>cx::Slab_static< Type, Slab_size, Max_free, Alloc ></code> Class Template Reference	877
15.61.1 Detailed Description	879
15.61.2 Member Function Documentation	880
15.61.2.1 <code>alloc()</code>	880
15.62 <code>cx::static_vector< T, IDX ></code> Class Template Reference	881
15.62.1 Detailed Description	881
15.63 <code>cx::String</code> Class Reference	882
15.63.1 Detailed Description	884

15.63.2 Constructor & Destructor Documentation	885
15.63.2.1 String()	885
15.63.3 Member Function Documentation	885
15.63.3.1 find()	885
15.63.3.2 from_dec()	886
15.63.3.3 from_hex()	887
15.63.3.4 starts_with()	887
15.64 cxx::Weak_ref< T > Class Template Reference	888
15.64.1 Detailed Description	890
15.65 cxx::Weak_ref_base Class Reference	891
15.65.1 Detailed Description	893
15.66 Elf32_Auxv Struct Reference	894
15.66.1 Detailed Description	894
15.66.2 Field Documentation	894
15.66.2.1 atype	894
15.67 Elf32_Dyn Struct Reference	895
15.67.1 Detailed Description	895
15.67.2 Field Documentation	895
15.67.2.1 d_tag	895
15.68 Elf32_Ehdr Struct Reference	896
15.68.1 Detailed Description	897
15.68.2 Field Documentation	897
15.68.2.1 e_flags	897
15.68.2.2 e_machine	897
15.68.2.3 e_type	898
15.68.2.4 e_version	898
15.69 Elf32_Phdr Struct Reference	898
15.69.1 Detailed Description	899
15.69.2 Field Documentation	899
15.69.2.1 p_flags	899
15.69.2.2 p_type	899
15.70 Elf32_Rel Struct Reference	900
15.70.1 Detailed Description	900
15.71 Elf32_Rela Struct Reference	900
15.71.1 Detailed Description	901
15.72 Elf32_Shdr Struct Reference	901
15.72.1 Detailed Description	902
15.72.2 Field Documentation	902
15.72.2.1 sh_flags	902
15.72.2.2 sh_type	902
15.73 Elf32_Sym Struct Reference	903
15.73.1 Detailed Description	903

15.74 Elf64_Auxv Struct Reference	904
15.74.1 Detailed Description	904
15.74.2 Field Documentation	904
15.74.2.1 atype	904
15.75 Elf64_Dyn Struct Reference	905
15.75.1 Detailed Description	905
15.75.2 Field Documentation	905
15.75.2.1 d_tag	905
15.76 Elf64_Ehdr Struct Reference	906
15.76.1 Detailed Description	907
15.76.2 Field Documentation	907
15.76.2.1 e_flags	907
15.76.2.2 e_machine	907
15.76.2.3 e_type	908
15.76.2.4 e_version	908
15.77 Elf64_Phdr Struct Reference	908
15.77.1 Detailed Description	909
15.77.2 Field Documentation	909
15.77.2.1 p_flags	909
15.77.2.2 p_type	909
15.78 Elf64_Rel Struct Reference	910
15.78.1 Detailed Description	910
15.79 Elf64_Rela Struct Reference	910
15.79.1 Detailed Description	911
15.80 Elf64_Shdr Struct Reference	911
15.80.1 Detailed Description	912
15.80.2 Field Documentation	912
15.80.2.1 sh_flags	912
15.80.2.2 sh_type	912
15.81 Elf64_Sym Struct Reference	913
15.81.1 Detailed Description	913
15.82 gfxbitmap_offset Struct Reference	914
15.82.1 Detailed Description	914
15.83 L4::Alloc_list Class Reference	915
15.83.1 Detailed Description	915
15.84 L4::Arm_smccc Class Reference	915
15.84.1 Detailed Description	916
15.84.2 Member Function Documentation	916
15.84.2.1 call()	916
15.85 L4::Base_exception Class Reference	917
15.85.1 Detailed Description	919
15.86 L4::Basic_registry Class Reference	919

15.86.1 Detailed Description	920
15.86.2 Member Function Documentation	920
15.86.2.1 dispatch()	920
15.86.2.2 find()	921
15.87 L4::Bounds_error Class Reference	922
15.87.1 Detailed Description	924
15.88 L4::Cap< T > Class Template Reference	924
15.88.1 Detailed Description	927
15.88.2 Constructor & Destructor Documentation	928
15.88.2.1 Cap() [1/4]	928
15.88.2.2 Cap() [2/4]	928
15.88.2.3 Cap() [3/4]	928
15.88.2.4 Cap() [4/4]	928
15.88.3 Member Function Documentation	929
15.88.3.1 copy()	929
15.88.3.2 move()	929
15.89 L4::Cap_base Class Reference	930
15.89.1 Detailed Description	931
15.89.2 Member Enumeration Documentation	932
15.89.2.1 Cap_type	932
15.89.2.2 No_init_type	932
15.89.3 Constructor & Destructor Documentation	932
15.89.3.1 Cap_base() [1/2]	932
15.89.3.2 Cap_base() [2/2]	932
15.89.4 Member Function Documentation	933
15.89.4.1 cap()	933
15.89.4.2 copy()	934
15.89.4.3 fpage()	935
15.89.4.4 is_valid()	936
15.89.4.5 move()	937
15.89.4.6 snd_base()	938
15.89.4.7 validate() [1/2]	938
15.89.4.8 validate() [2/2]	939
15.90 L4::Com_error Class Reference	940
15.90.1 Detailed Description	943
15.90.2 Constructor & Destructor Documentation	943
15.90.2.1 Com_error()	943
15.91 L4::Debugger Class Reference	944
15.91.1 Detailed Description	947
15.91.2 Member Function Documentation	947
15.91.2.1 get_object_name()	947
15.91.2.2 global_id()	948

15.91.2.3 kobj_to_id()	948
15.91.2.4 query_log_name()	949
15.91.2.5 query_log_typeid()	950
15.91.2.6 set_object_name()	951
15.91.2.7 switch_log()	951
15.92 L4::Element_already_exists Class Reference	952
15.92.1 Detailed Description	955
15.93 L4::Element_not_found Class Reference	955
15.93.1 Detailed Description	958
15.94 L4::Epiface Struct Reference	958
15.94.1 Detailed Description	960
15.94.2 Member Function Documentation	960
15.94.2.1 dispatch()	960
15.94.2.2 get_buffer_demand()	961
15.94.2.3 obj_cap()	961
15.94.2.4 server_iface()	962
15.94.2.5 set_server()	962
15.95 L4::Epiface_t< Derived, IFACE, BASE, bool > Struct Template Reference	963
15.95.1 Detailed Description	965
15.95.2 Member Function Documentation	966
15.95.2.1 dispatch()	966
15.96 L4::Epiface_t0< RPC_IFACE, BASE > Struct Template Reference	966
15.96.1 Detailed Description	968
15.96.2 Member Function Documentation	968
15.96.2.1 obj_cap()	968
15.97 L4::Exception Class Reference	969
15.97.1 Detailed Description	969
15.97.2 Member Function Documentation	969
15.97.2.1 exception()	969
15.98 L4::Exception_tracer Class Reference	970
15.98.1 Detailed Description	971
15.99 L4::Factory Class Reference	972
15.99.1 Detailed Description	975
15.99.2 Member Function Documentation	975
15.99.2.1 create() [1/2]	975
15.99.2.2 create() [2/2]	976
15.99.2.3 create_factory()	977
15.99.2.4 create_gate()	978
15.99.2.5 create_task()	979
15.100 L4::Factory::Lstr Struct Reference	980
15.100.1 Detailed Description	981
15.100.2 Constructor & Destructor Documentation	981

15.100.2.1 Lstr()	981
15.101 L4::Factory::Nil Struct Reference	982
15.101.1 Detailed Description	982
15.102 L4::Factory::S Class Reference	982
15.102.1 Detailed Description	983
15.102.2 Constructor & Destructor Documentation	984
15.102.2.1 S() [1/2]	984
15.102.2.2 S() [2/2]	984
15.102.3 Member Function Documentation	984
15.102.3.1 operator l4_msgtag_t()	984
15.102.3.2 put() [1/5]	985
15.102.3.3 put() [2/5]	985
15.102.3.4 put() [3/5]	985
15.102.3.5 put() [4/5]	986
15.102.3.6 put() [5/5]	986
15.103 L4::lcu Class Reference	986
15.103.1 Detailed Description	990
15.103.2 Member Function Documentation	990
15.103.2.1 bind()	990
15.103.2.2 info()	991
15.103.2.3 mask()	992
15.103.2.4 msi_info()	993
15.103.2.5 set_mode()	994
15.103.2.6 unbind()	994
15.104 L4::lcu::Info Class Reference	995
15.104.1 Detailed Description	996
15.105 L4::Invalid_capability Class Reference	997
15.105.1 Detailed Description	999
15.105.2 Constructor & Destructor Documentation	999
15.105.2.1 Invalid_capability()	999
15.105.3 Member Function Documentation	999
15.105.3.1 cap()	999
15.106 L4::io_pager Class Reference	1000
15.106.1 Detailed Description	1001
15.106.2 Member Function Documentation	1001
15.106.2.1 io_page_fault()	1001
15.107 L4::lomu Class Reference	1002
15.107.1 Detailed Description	1003
15.107.2 Member Function Documentation	1004
15.107.2.1 bind()	1004
15.107.2.2 unbind()	1004
15.108 L4::IOModifier Class Reference	1004

15.108.1 Detailed Description	1005
15.109 L4::lpc::Array< ELEM_TYPE, LEN_TYPE > Struct Template Reference	1005
15.109.1 Detailed Description	1007
15.110 L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX > Struct Template Reference	1008
15.110.1 Detailed Description	1009
15.111 L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE > Struct Template Reference	1009
15.111.1 Detailed Description	1010
15.112 L4::lpc::As_value< T > Struct Template Reference	1010
15.112.1 Detailed Description	1011
15.113 L4::lpc::Buf_item Class Reference	1011
15.113.1 Detailed Description	1012
15.114 L4::lpc::Call Struct Reference	1012
15.114.1 Detailed Description	1013
15.115 L4::lpc::Call_t< RIGHTS > Struct Template Reference	1013
15.115.1 Detailed Description	1014
15.116 L4::lpc::Call_zero_send_timeout Struct Reference	1014
15.116.1 Detailed Description	1015
15.117 L4::lpc::Cap< T > Class Template Reference	1016
15.117.1 Detailed Description	1017
15.117.2 Member Enumeration Documentation	1017
15.117.2.1 anonymous enum	1017
15.117.3 Constructor & Destructor Documentation	1017
15.117.3.1 Cap()	1017
15.117.4 Member Function Documentation	1018
15.117.4.1 from_ci()	1018
15.118 L4::lpc::Gen_fpage< T > Class Template Reference	1018
15.118.1 Detailed Description	1019
15.118.2 Member Function Documentation	1020
15.118.2.1 cap_received()	1020
15.118.2.2 id_received()	1020
15.118.2.3 is_compound()	1020
15.118.2.4 local_id_received()	1021
15.119 L4::lpc::In_out< T > Struct Template Reference	1021
15.119.1 Detailed Description	1021
15.120 L4::lpc::Iostream Class Reference	1022
15.120.1 Detailed Description	1026
15.120.2 Constructor & Destructor Documentation	1026
15.120.2.1 Iostream()	1026
15.120.3 Member Function Documentation	1027
15.120.3.1 call()	1027
15.120.3.2 get() [1/3]	1028
15.120.3.3 get() [2/3]	1028

15.120.3.4 get() [3/3]	1028
15.120.3.5 put() [1/2]	1029
15.120.3.6 put() [2/2]	1029
15.120.3.7 reply_and_wait() [1/2]	1030
15.120.3.8 reply_and_wait() [2/2]	1030
15.120.3.9 reset()	1032
15.121 L4::Ipc::Istream Class Reference	1032
15.121.1 Detailed Description	1035
15.121.2 Constructor & Destructor Documentation	1035
15.121.2.1 Istream()	1035
15.121.3 Member Function Documentation	1036
15.121.3.1 get() [1/3]	1036
15.121.3.2 get() [2/3]	1036
15.121.3.3 get() [3/3]	1037
15.121.3.4 receive()	1038
15.121.3.5 reset()	1039
15.121.3.6 skip()	1040
15.121.3.7 tag() [1/2]	1040
15.121.3.8 tag() [2/2]	1040
15.121.3.9 wait() [1/2]	1041
15.121.3.10 wait() [2/2]	1042
15.122 L4::Ipc::Msg::Cnt_val_ops< MTYPE, DIR, CLASS > Struct Template Reference	1043
15.122.1 Detailed Description	1044
15.123 L4::Ipc::Msg::Cls_buffer Struct Reference	1044
15.123.1 Detailed Description	1045
15.124 L4::Ipc::Msg::Cls_data Struct Reference	1045
15.124.1 Detailed Description	1046
15.125 L4::Ipc::Msg::Cls_item Struct Reference	1046
15.125.1 Detailed Description	1047
15.126 L4::Ipc::Msg::Dir_in Struct Reference	1047
15.126.1 Detailed Description	1048
15.127 L4::Ipc::Msg::Dir_out Struct Reference	1048
15.127.1 Detailed Description	1049
15.128 L4::Ipc::Msg::Do_in_data Struct Reference	1049
15.128.1 Detailed Description	1050
15.129 L4::Ipc::Msg::Do_in_items Struct Reference	1050
15.129.1 Detailed Description	1051
15.130 L4::Ipc::Msg::Do_out_data Struct Reference	1051
15.130.1 Detailed Description	1052
15.131 L4::Ipc::Msg::Do_out_items Struct Reference	1052
15.131.1 Detailed Description	1053
15.132 L4::Ipc::Msg::Do_rcv_buffers Struct Reference	1053

15.132.1 Detailed Description	1054
15.133 L4::lpc::Msg::Elem< Array< A, LEN > & > Struct Template Reference	1055
15.133.1 Detailed Description	1055
15.134 L4::lpc::Msg::Elem< Array< A, LEN > > Struct Template Reference	1056
15.134.1 Detailed Description	1056
15.135 L4::lpc::Msg::Elem< Array_ref< A, LEN > & > Struct Template Reference	1057
15.135.1 Detailed Description	1057
15.136 L4::lpc::Msg::Is_valid_rpc_type< T > Struct Template Reference	1058
15.136.1 Detailed Description	1059
15.137 L4::lpc::Msg::Svr_arg_pack< IPC_TYPE > Struct Template Reference	1060
15.137.1 Detailed Description	1060
15.138 L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS > Struct Template Reference	1060
15.138.1 Detailed Description	1061
15.139 L4::lpc::Msg_ptr< T > Class Template Reference	1061
15.139.1 Detailed Description	1062
15.139.2 Constructor & Destructor Documentation	1062
15.139.2.1 Msg_ptr()	1062
15.140 L4::lpc::Opt< T > Struct Template Reference	1062
15.140.1 Detailed Description	1064
15.141 L4::lpc::Ostream Class Reference	1064
15.141.1 Detailed Description	1067
15.141.2 Member Function Documentation	1067
15.141.2.1 put() [1/2]	1067
15.141.2.2 put() [2/2]	1068
15.141.2.3 send()	1068
15.141.2.4 tag() [1/2]	1069
15.141.2.5 tag() [2/2]	1069
15.142 L4::lpc::Out< T > Struct Template Reference	1070
15.142.1 Detailed Description	1070
15.143 L4::lpc::Ret_array< T > Struct Template Reference	1071
15.143.1 Detailed Description	1071
15.144 L4::lpc::Send_only Struct Reference	1072
15.144.1 Detailed Description	1072
15.145 L4::lpc::Small_buf Class Reference	1073
15.145.1 Detailed Description	1073
15.145.2 Constructor & Destructor Documentation	1073
15.145.2.1 Small_buf() [1/2]	1073
15.145.2.2 Small_buf() [2/2]	1074
15.146 L4::lpc::Snd_item Class Reference	1074
15.146.1 Detailed Description	1075
15.147 L4::lpc::Str_cp_in< T > Class Template Reference	1075
15.147.1 Detailed Description	1075

15.147.2 Constructor & Destructor Documentation	1076
15.147.2.1 Str_cp_in()	1076
15.148 L4::lpc::Varg Class Reference	1076
15.148.1 Detailed Description	1077
15.148.2 Member Function Documentation	1078
15.148.2.1 data()	1078
15.148.2.2 get_value()	1078
15.148.2.3 is_nil()	1079
15.148.2.4 is_of()	1079
15.148.2.5 is_of_int()	1080
15.148.2.6 length()	1080
15.148.2.7 tag()	1081
15.148.2.8 type()	1081
15.148.2.9 value()	1081
15.149 L4::lpc::Varg_list< MAX > Class Template Reference	1082
15.149.1 Detailed Description	1084
15.150 L4::lpc::Varg_list_ref Class Reference	1084
15.150.1 Detailed Description	1085
15.150.2 Constructor & Destructor Documentation	1086
15.150.2.1 Varg_list_ref()	1086
15.151 L4::lpc::Varg_list_ref::Iterator Class Reference	1086
15.151.1 Detailed Description	1087
15.152 L4::lpc_gate Class Reference	1087
15.152.1 Detailed Description	1091
15.152.2 Member Function Documentation	1092
15.152.2.1 get_infos()	1092
15.153 L4::lpc_svr::Br_manager_no_buffers Class Reference	1092
15.153.1 Detailed Description	1095
15.153.2 Member Function Documentation	1095
15.153.2.1 alloc_buffer_demand()	1095
15.154 L4::lpc_svr::Compound_reply Struct Reference	1096
15.154.1 Detailed Description	1098
15.155 L4::lpc_svr::Default_loop_hooks Struct Reference	1098
15.155.1 Detailed Description	1101
15.156 L4::lpc_svr::Default_setup_wait Struct Reference	1101
15.156.1 Detailed Description	1102
15.157 L4::lpc_svr::Default_timeout Struct Reference	1102
15.157.1 Detailed Description	1104
15.158 L4::lpc_svr::Direct_dispatch< R > Struct Template Reference	1104
15.158.1 Detailed Description	1105
15.159 L4::lpc_svr::Direct_dispatch< R * > Struct Template Reference	1106
15.159.1 Detailed Description	1106

15.160 L4::lpc_svr::Exc_dispatch< R, Exc > Struct Template Reference	1107
15.160.1 Detailed Description	1108
15.161 L4::lpc_svr::Ignore_errors Struct Reference	1109
15.161.1 Detailed Description	1110
15.162 L4::lpc_svr::Server_iface Class Reference	1110
15.162.1 Detailed Description	1112
15.162.2 Member Function Documentation	1112
15.162.2.1 add_timeout()	1112
15.162.2.2 alloc_buffer_demand()	1112
15.162.2.3 get_rcv_cap()	1113
15.162.2.4 rcv_cap() [1/2]	1113
15.162.2.5 rcv_cap() [2/2]	1114
15.162.2.6 realloc_rcv_cap()	1115
15.162.2.7 remove_timeout()	1115
15.163 L4::lpc_svr::Timeout Class Reference	1116
15.163.1 Detailed Description	1117
15.163.2 Member Function Documentation	1118
15.163.2.1 expired()	1118
15.163.2.2 timeout()	1118
15.164 L4::lpc_svr::Timeout_queue Class Reference	1119
15.164.1 Detailed Description	1119
15.164.2 Member Function Documentation	1120
15.164.2.1 add()	1120
15.164.2.2 handle_expired_timeouts()	1121
15.164.2.3 next_timeout()	1122
15.164.2.4 remove()	1122
15.164.2.5 timeout_expired()	1123
15.165 L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN > Class Template Reference	1124
15.165.1 Detailed Description	1127
15.165.2 Member Function Documentation	1128
15.165.2.1 add_timeout()	1128
15.165.2.2 remove_timeout()	1129
15.166 L4::lpc_svr::Irq Class Reference	1130
15.166.1 Detailed Description	1134
15.166.2 Member Function Documentation	1134
15.166.2.1 detach()	1134
15.166.2.2 receive()	1135
15.166.2.3 unmask()	1136
15.166.2.4 wait()	1137
15.167 L4::lpc_svr::Irq_eoi Class Reference	1138
15.167.1 Detailed Description	1139
15.167.2 Member Function Documentation	1139

15.167.2.1 unmask()	1139
15.168 L4::Irq_handler_object Struct Reference	1140
15.168.1 Detailed Description	1143
15.169 L4::Irq_mux Struct Reference	1144
15.169.1 Detailed Description	1147
15.169.2 Member Function Documentation	1147
15.169.2.1 chain()	1147
15.170 L4::Irqp_t< Derived, BASE, bool > Struct Template Reference	1148
15.170.1 Detailed Description	1151
15.170.2 Member Function Documentation	1152
15.170.2.1 dispatch()	1152
15.170.2.2 obj_cap()	1152
15.171 L4::Kip::Mem_desc Class Reference	1153
15.171.1 Detailed Description	1154
15.171.2 Member Enumeration Documentation	1154
15.171.2.1 Arch_sub_type_common	1154
15.171.2.2 Info_sub_type	1155
15.171.2.3 Mem_type	1155
15.171.3 Constructor & Destructor Documentation	1155
15.171.3.1 Mem_desc()	1155
15.171.4 Member Function Documentation	1156
15.171.4.1 all() [1/2]	1156
15.171.4.2 all() [2/2]	1157
15.171.4.3 count() [1/2]	1157
15.171.4.4 count() [2/2]	1158
15.171.4.5 end()	1159
15.171.4.6 first()	1159
15.171.4.7 is_virtual()	1160
15.171.4.8 set()	1160
15.171.4.9 size()	1161
15.171.4.10 start()	1161
15.171.4.11 sub_type()	1162
15.171.4.12 type()	1162
15.172 L4::Kobject Class Reference	1163
15.172.1 Detailed Description	1163
15.172.2 Member Function Documentation	1164
15.172.2.1 cap()	1164
15.172.2.2 dec_refcnt()	1165
15.173 L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND > Class Template Reference	1166
15.173.1 Detailed Description	1167
15.173.2 Member Typedef Documentation	1168
15.173.2.1 __lface	1168

15.173.2.2	__iface_list	1168
15.173.2.3	Class	1168
15.173.3	Member Function Documentation	1168
15.173.3.1	__check_protocols__()	1168
15.173.3.2	c()	1169
15.174 L4::	Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND > Struct Template Reference	1169
15.174.1	Detailed Description	1171
15.174.2	Member Typedef Documentation	1172
15.174.2.1	__iface	1172
15.174.2.2	__iface_list	1172
15.174.2.3	Class	1172
15.174.3	Member Function Documentation	1172
15.174.3.1	__check_protocols__()	1172
15.174.3.2	c()	1173
15.175 L4::	Kobject_demand< T > Struct Template Reference	1173
15.175.1	Detailed Description	1173
15.176 L4::	Kobject_t< Derived, Base, PROTO, S_DEMAND > Class Template Reference	1174
15.176.1	Detailed Description	1175
15.177 L4::	Kobject_typeid< T > Struct Template Reference	1175
15.177.1	Detailed Description	1176
15.177.2	Member Typedef Documentation	1176
15.177.2.1	Demand	1176
15.177.3	Member Function Documentation	1177
15.177.3.1	demand()	1177
15.177.3.2	id()	1177
15.177.3.3	proto_dispatch()	1177
15.178 L4::	Kobject_typeid< void > Struct Reference	1178
15.178.1	Detailed Description	1179
15.179 L4::	Kobject_x< Derived, ARGS > Struct Template Reference	1179
15.179.1	Detailed Description	1180
15.180 L4::	Meta Class Reference	1180
15.180.1	Detailed Description	1183
15.180.2	Member Function Documentation	1183
15.180.2.1	interface()	1183
15.180.2.2	num_interfaces()	1184
15.180.2.3	supports()	1184
15.181 L4::	Out_of_memory Class Reference	1185
15.181.1	Detailed Description	1187
15.182 L4::	Pager Class Reference	1188
15.182.1	Detailed Description	1190
15.182.2	Member Function Documentation	1190
15.182.2.1	page_fault()	1190

15.183 L4::Platform_control Class Reference	1192
15.183.1 Detailed Description	1194
15.183.2 Member Function Documentation	1194
15.183.2.1 cpu_allow_shutdown()	1194
15.183.2.2 cpu_disable()	1195
15.183.2.3 cpu_enable()	1195
15.183.2.4 system_shutdown()	1195
15.183.2.5 system_suspend()	1196
15.184 L4::Poll_timeout_counter Class Reference	1196
15.184.1 Detailed Description	1197
15.184.2 Constructor & Destructor Documentation	1197
15.184.2.1 Poll_timeout_counter()	1197
15.184.3 Member Function Documentation	1198
15.184.3.1 set()	1198
15.184.3.2 timed_out()	1198
15.185 L4::Poll_timeout_kipclock Class Reference	1199
15.185.1 Detailed Description	1199
15.185.2 Constructor & Destructor Documentation	1200
15.185.2.1 Poll_timeout_kipclock()	1200
15.185.3 Member Function Documentation	1200
15.185.3.1 set()	1200
15.185.3.2 test()	1201
15.185.3.3 timed_out()	1202
15.186 L4::Proto_t< P > Struct Template Reference	1202
15.186.1 Detailed Description	1202
15.187 L4::Rcv_endpoint Class Reference	1203
15.187.1 Detailed Description	1206
15.187.2 Member Function Documentation	1206
15.187.2.1 bind_thread()	1206
15.188 L4::Registry_iface Class Reference	1207
15.188.1 Detailed Description	1209
15.188.2 Member Function Documentation	1209
15.188.2.1 register_irq_obj()	1209
15.188.2.2 register_obj() [1/3]	1209
15.188.2.3 register_obj() [2/3]	1210
15.188.2.4 register_obj() [3/3]	1210
15.188.2.5 unregister_obj()	1211
15.189 L4::Runtime_error Class Reference	1211
15.189.1 Detailed Description	1214
15.189.2 Constructor & Destructor Documentation	1214
15.189.2.1 Runtime_error()	1214
15.189.3 Member Function Documentation	1215

15.189.3.1 err_no()	1215
15.189.3.2 extra_str()	1215
15.190 L4::Scheduler Class Reference	1215
15.190.1 Detailed Description	1219
15.190.2 Member Function Documentation	1219
15.190.2.1 idle_time()	1219
15.190.2.2 info()	1220
15.190.2.3 is_online()	1221
15.190.2.4 run_thread()	1221
15.191 L4::Semaphore Struct Reference	1222
15.191.1 Detailed Description	1226
15.191.2 Member Function Documentation	1226
15.191.2.1 down()	1226
15.191.2.2 up()	1227
15.192 L4::Server< LOOP_HOOKS > Class Template Reference	1228
15.192.1 Detailed Description	1230
15.192.2 Constructor & Destructor Documentation	1231
15.192.2.1 Server()	1231
15.192.3 Member Function Documentation	1231
15.192.3.1 internal_loop()	1231
15.192.3.2 loop()	1232
15.193 L4::Server_object Class Reference	1233
15.193.1 Detailed Description	1235
15.193.2 Member Function Documentation	1235
15.193.2.1 dispatch() [1/2]	1235
15.193.2.2 dispatch() [2/2]	1236
15.194 L4::Server_object_t< IFACE, BASE > Struct Template Reference	1237
15.194.1 Detailed Description	1240
15.194.2 Member Function Documentation	1241
15.194.2.1 dispatch_meta_request()	1241
15.194.2.2 get_buffer_demand()	1241
15.194.2.3 proto_dispatch()	1241
15.195 L4::Server_object_x< Derived, IFACE, BASE > Struct Template Reference	1243
15.195.1 Detailed Description	1245
15.196 L4::Smart_cap< T, SMART > Class Template Reference	1246
15.196.1 Detailed Description	1250
15.196.2 Constructor & Destructor Documentation	1250
15.196.2.1 Smart_cap()	1250
15.197 L4::String Class Reference	1250
15.197.1 Detailed Description	1251
15.198 L4::Task Class Reference	1251
15.198.1 Detailed Description	1255

15.198.2 Member Function Documentation	1255
15.198.2.1 add_ku_mem()	1255
15.198.2.2 cap_equal()	1256
15.198.2.3 cap_valid()	1257
15.198.2.4 delete_obj()	1257
15.198.2.5 map()	1258
15.198.2.6 release_cap()	1259
15.198.2.7 unmap()	1260
15.198.2.8 unmap_batch()	1261
15.199 L4::Thread Class Reference	1262
15.199.1 Detailed Description	1266
15.199.2 Member Function Documentation	1266
15.199.2.1 control()	1266
15.199.2.2 ex_regs() [1/2]	1267
15.199.2.3 ex_regs() [2/2]	1268
15.199.2.4 modify_senders()	1269
15.199.2.5 register_del_irq()	1270
15.199.2.6 stats_time()	1271
15.199.2.7 switch_to()	1272
15.199.2.8 vcpu_control()	1272
15.199.2.9 vcpu_control_ext()	1273
15.199.2.10 vcpu_resume_commit()	1274
15.199.2.11 vcpu_resume_start()	1275
15.200 L4::Thread::Attr Class Reference	1276
15.200.1 Detailed Description	1277
15.200.2 Constructor & Destructor Documentation	1277
15.200.2.1 Attr()	1277
15.200.3 Member Function Documentation	1277
15.200.3.1 alien()	1277
15.200.3.2 bind()	1278
15.200.3.3 exc_handler() [1/2]	1279
15.200.3.4 exc_handler() [2/2]	1279
15.200.3.5 pager() [1/2]	1280
15.200.3.6 pager() [2/2]	1280
15.200.3.7 ux_host_syscall()	1281
15.201 L4::Thread::Modify_senders Class Reference	1281
15.201.1 Detailed Description	1282
15.201.2 Member Function Documentation	1282
15.201.2.1 add()	1282
15.202 L4::Triggerable Struct Reference	1283
15.202.1 Detailed Description	1286
15.202.2 Member Function Documentation	1286

15.202.2.1 trigger()	1286
15.203 L4::Type_info Struct Reference	1287
15.203.1 Detailed Description	1288
15.204 L4::Type_info::Demand Class Reference	1288
15.204.1 Detailed Description	1290
15.204.2 Constructor & Destructor Documentation	1291
15.204.2.1 Demand()	1291
15.204.3 Member Function Documentation	1291
15.204.3.1 no_demand()	1291
15.205 L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS > Struct Template Reference	1292
15.205.1 Detailed Description	1294
15.205.2 Member Enumeration Documentation	1294
15.205.2.1 anonymous enum	1294
15.206 L4::Type_info::Demand_union_t< D1, D2 > Struct Template Reference	1294
15.206.1 Detailed Description	1297
15.207 L4::Typeid::Detail::_Rpc< OPCODE, O, X > Struct Template Reference	1297
15.207.1 Detailed Description	1298
15.208 L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y > Struct Template Reference	1299
15.208.1 Detailed Description	1299
15.209 L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... > Struct Template Reference	1299
15.209.1 Detailed Description	1300
15.210 L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y > Struct Template Reference	1301
15.210.1 Detailed Description	1301
15.211 L4::Typeid::Detail::Rpc_end Struct Reference	1301
15.211.1 Detailed Description	1302
15.212 L4::Typeid::P_dispatch< LIST > Struct Template Reference	1302
15.212.1 Detailed Description	1303
15.213 L4::Typeid::Raw_ipc< CLASS > Struct Template Reference	1303
15.213.1 Detailed Description	1303
15.214 L4::Typeid::Rpc_nocode< OPERATION > Struct Template Reference	1304
15.214.1 Detailed Description	1305
15.215 L4::Typeid::Rpc< RPCS > Struct Template Reference	1306
15.215.1 Detailed Description	1307
15.216 L4::Typeid::Rpc_code< OPCODE_TYPE > Struct Template Reference	1308
15.216.1 Detailed Description	1308
15.217 L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS > Struct Template Reference	1309
15.217.1 Detailed Description	1310
15.218 L4::Typeid::Rpc_sys< ARG > Struct Template Reference	1311
15.218.1 Detailed Description	1312
15.219 L4::Types::Bool< V > Struct Template Reference	1313
15.219.1 Detailed Description	1313
15.220 L4::Types::False Struct Reference	1314

15.220.1 Detailed Description	1315
15.221 L4::Types::Flags< BITS_ENUM, UNDERLYING > Class Template Reference	1316
15.221.1 Detailed Description	1317
15.221.2 Member Enumeration Documentation	1318
15.221.2.1 None_type	1318
15.221.3 Constructor & Destructor Documentation	1318
15.221.3.1 Flags() [1/2]	1318
15.221.3.2 Flags() [2/2]	1319
15.221.4 Member Function Documentation	1319
15.221.4.1 clear()	1319
15.221.4.2 from_raw()	1319
15.222 L4::Types::Flags_ops_t< DT > Struct Template Reference	1320
15.222.1 Detailed Description	1321
15.223 L4::Types::Flags_t< DT, T > Struct Template Reference	1322
15.223.1 Detailed Description	1324
15.224 L4::Types::Int_for_size< SIZE, bool > Struct Template Reference	1324
15.224.1 Detailed Description	1325
15.225 L4::Types::Int_for_type< T > Struct Template Reference	1325
15.225.1 Detailed Description	1325
15.226 L4::Types::Same< A, B > Struct Template Reference	1326
15.226.1 Detailed Description	1327
15.227 L4::Types::True Struct Reference	1328
15.227.1 Detailed Description	1329
15.228 L4::Uart Class Reference	1330
15.228.1 Detailed Description	1331
15.228.2 Member Function Documentation	1331
15.228.2.1 change_mode()	1331
15.228.2.2 char_avail()	1331
15.228.2.3 enable_rx_irq()	1332
15.228.2.4 generic_write()	1332
15.228.2.5 get_char()	1333
15.228.2.6 mode()	1333
15.228.2.7 rate()	1333
15.228.2.8 shutdown()	1334
15.228.2.9 startup()	1334
15.228.2.10 write()	1334
15.229 L4::Unknown_error Class Reference	1335
15.229.1 Detailed Description	1337
15.230 L4::Vcon Class Reference	1337
15.230.1 Detailed Description	1341
15.230.2 Member Function Documentation	1341
15.230.2.1 get_attr()	1341

15.230.2.2 read()	1342
15.230.2.3 read_with_flags()	1343
15.230.2.4 send()	1344
15.230.2.5 set_attr()	1344
15.230.2.6 write()	1345
15.231 L4::Vm Class Reference	1346
15.231.1 Detailed Description	1350
15.231.2 Member Function Documentation	1350
15.231.2.1 vgicc_map()	1350
15.232 l4_buf_regs_t Struct Reference	1351
15.232.1 Detailed Description	1352
15.233 l4_exc_regs_t Struct Reference	1352
15.233.1 Detailed Description	1354
15.233.2 Field Documentation	1354
15.233.2.1 flags	1354
15.233.2.2 ss	1355
15.234 l4_fpage_t Union Reference	1355
15.234.1 Detailed Description	1355
15.235 l4_icu_info_t Struct Reference	1356
15.235.1 Detailed Description	1357
15.235.2 Field Documentation	1357
15.235.2.1 features	1357
15.236 l4_icu_msi_info_t Struct Reference	1357
15.236.1 Detailed Description	1358
15.237 l4_kernel_info_mem_desc_t Struct Reference	1358
15.237.1 Detailed Description	1358
15.238 l4_kernel_info_t Struct Reference	1359
15.238.1 Detailed Description	1361
15.239 l4_msg_regs_t Union Reference	1361
15.239.1 Detailed Description	1361
15.240 l4_msgtag_t Struct Reference	1362
15.240.1 Detailed Description	1363
15.240.2 Member Function Documentation	1363
15.240.2.1 flags()	1363
15.241 l4_sched_cpu_set_t Struct Reference	1364
15.241.1 Detailed Description	1364
15.241.2 Member Function Documentation	1365
15.241.2.1 granularity()	1365
15.241.2.2 offset()	1365
15.241.2.3 set()	1366
15.241.3 Field Documentation	1367
15.241.3.1 gran_offset	1367

15.242 l4_sched_param_t Struct Reference	1368
15.242.1 Detailed Description	1368
15.243 l4_snd_fpage_t Struct Reference	1369
15.243.1 Detailed Description	1369
15.244 l4_thread_regs_t Struct Reference	1370
15.244.1 Detailed Description	1370
15.245 l4_timeout_s Struct Reference	1371
15.245.1 Detailed Description	1371
15.246 l4_timeout_t Union Reference	1372
15.246.1 Detailed Description	1372
15.247 l4_vcon_attr_t Struct Reference	1373
15.247.1 Detailed Description	1373
15.247.2 Member Function Documentation	1374
15.247.2.1 set_raw()	1374
15.248 l4_vcpu_arch_state_t Struct Reference	1374
15.248.1 Detailed Description	1375
15.249 l4_vcpu_ipc_regs_t Struct Reference	1375
15.249.1 Detailed Description	1376
15.250 l4_vcpu_regs_t Struct Reference	1376
15.250.1 Detailed Description	1378
15.250.2 Field Documentation	1378
15.250.2.1 ax	1378
15.250.2.2 bp	1378
15.250.2.3 bx	1378
15.250.2.4 cx	1378
15.250.2.5 di	1379
15.250.2.6 dx	1379
15.250.2.7 si	1379
15.251 l4_vcpu_state_t Struct Reference	1380
15.251.1 Detailed Description	1382
15.251.2 Field Documentation	1382
15.251.2.1 version	1382
15.252 l4_vhw_descriptor Struct Reference	1383
15.252.1 Detailed Description	1384
15.253 l4_vhw_entry Struct Reference	1384
15.253.1 Detailed Description	1385
15.254 l4_vm_svm_vmcb_control_area Struct Reference	1385
15.254.1 Detailed Description	1385
15.255 l4_vm_svm_vmcb_state_save_area Struct Reference	1386
15.255.1 Detailed Description	1386
15.256 l4_vm_svm_vmcb_state_save_area_seg Struct Reference	1387
15.256.1 Detailed Description	1387

15.257 l4_vm_svm_vmcb_t Struct Reference	1387
15.257.1 Detailed Description	1388
15.258 l4_vm_tz_state Struct Reference	1388
15.258.1 Detailed Description	1389
15.259 L4drivers::Mmio_register_block< MAX_BITS > Struct Template Reference	1389
15.259.1 Detailed Description	1390
15.260 L4drivers::Register_block< MAX_BITS, BLOCK > Class Template Reference	1390
15.260.1 Detailed Description	1391
15.260.2 Member Function Documentation	1392
15.260.2.1 operator[]() [1/2]	1392
15.260.2.2 operator[]() [2/2]	1392
15.260.2.3 r() [1/2]	1394
15.260.2.4 r() [2/2]	1394
15.261 L4drivers::Register_block_base< MAX_BITS > Struct Template Reference	1395
15.261.1 Detailed Description	1395
15.262 L4drivers::Register_block_impl< BASE, MAX_BITS > Struct Template Reference	1396
15.262.1 Detailed Description	1397
15.263 L4drivers::Register_block_tmpl< BLOCK > Class Template Reference	1397
15.263.1 Detailed Description	1398
15.264 L4drivers::Register_tmpl< BITS, BLOCK > Class Template Reference	1399
15.264.1 Detailed Description	1400
15.264.2 Member Function Documentation	1401
15.264.2.1 clear()	1401
15.264.2.2 modify()	1401
15.264.2.3 operator=()	1402
15.264.2.4 set()	1402
15.264.2.5 write()	1402
15.265 L4drivers::Ro_register_block< MAX_BITS, BLOCK > Class Template Reference	1403
15.265.1 Detailed Description	1403
15.265.2 Member Function Documentation	1404
15.265.2.1 operator[]()	1404
15.265.2.2 r()	1404
15.266 L4drivers::Ro_register_tmpl< BITS, BLOCK > Class Template Reference	1405
15.266.1 Detailed Description	1406
15.266.2 Member Function Documentation	1406
15.266.2.1 operator value_type()	1406
15.266.2.2 read()	1407
15.267 L4Re::Cap_alloc Class Reference	1407
15.267.1 Detailed Description	1408
15.267.2 Member Function Documentation	1408
15.267.2.1 alloc() [1/2]	1408
15.267.2.2 alloc() [2/2]	1409

15.267.2.3 free()	1409
15.267.2.4 get_cap_alloc()	1410
15.268 L4Re::Console Class Reference	1410
15.268.1 Detailed Description	1413
15.269 L4Re::Dataspace Class Reference	1413
15.269.1 Detailed Description	1417
15.269.2 Member Function Documentation	1417
15.269.2.1 allocate()	1417
15.269.2.2 clear()	1418
15.269.2.3 copy_in()	1418
15.269.2.4 flags()	1419
15.269.2.5 info()	1420
15.269.2.6 map()	1421
15.269.2.7 map_region()	1422
15.269.2.8 size()	1423
15.270 L4Re::Dataspace::F Struct Reference	1424
15.270.1 Detailed Description	1424
15.270.2 Member Enumeration Documentation	1424
15.270.2.1 anonymous enum	1424
15.270.2.2 Flags	1425
15.271 L4Re::Dataspace::Stats Struct Reference	1425
15.271.1 Detailed Description	1426
15.272 L4Re::Debug_obj Class Reference	1426
15.272.1 Detailed Description	1428
15.272.2 Member Function Documentation	1428
15.272.2.1 debug()	1428
15.273 L4Re::Default_event_payload Struct Reference	1429
15.273.1 Detailed Description	1429
15.274 L4Re::Dma_space Class Reference	1430
15.274.1 Detailed Description	1431
15.274.2 Member Typedef Documentation	1431
15.274.2.1 Attributes	1431
15.274.3 Member Enumeration Documentation	1431
15.274.3.1 Attribute	1431
15.274.3.2 Direction	1432
15.274.3.3 Space_attrib	1432
15.274.4 Member Function Documentation	1432
15.274.4.1 associate()	1432
15.274.4.2 disassociate()	1433
15.274.4.3 map()	1433
15.274.4.4 unmap()	1435
15.275 L4Re::Env Class Reference	1436

15.275.1 Detailed Description	1438
15.275.2 Member Function Documentation	1439
15.275.2.1 env()	1439
15.275.2.2 factory() [1/2]	1439
15.275.2.3 factory() [2/2]	1439
15.275.2.4 first_free_cap() [1/2]	1440
15.275.2.5 first_free_cap() [2/2]	1440
15.275.2.6 first_free_utcb() [1/2]	1440
15.275.2.7 first_free_utcb() [2/2]	1441
15.275.2.8 get()	1441
15.275.2.9 get_cap() [1/2]	1442
15.275.2.10 get_cap() [2/2]	1443
15.275.2.11 initial_caps() [1/2]	1443
15.275.2.12 initial_caps() [2/2]	1443
15.275.2.13 log() [1/2]	1444
15.275.2.14 log() [2/2]	1444
15.275.2.15 main_thread() [1/2]	1444
15.275.2.16 main_thread() [2/2]	1445
15.275.2.17 mem_alloc() [1/2]	1445
15.275.2.18 mem_alloc() [2/2]	1445
15.275.2.19 parent() [1/2]	1446
15.275.2.20 parent() [2/2]	1446
15.275.2.21 rm() [1/2]	1446
15.275.2.22 rm() [2/2]	1446
15.275.2.23 scheduler() [1/2]	1447
15.275.2.24 scheduler() [2/2]	1447
15.275.2.25 task()	1447
15.275.2.26 utcb_area() [1/2]	1448
15.275.2.27 utcb_area() [2/2]	1448
15.276 L4Re::Event Class Reference	1448
15.276.1 Detailed Description	1452
15.276.2 Member Function Documentation	1452
15.276.2.1 get_axis_info()	1452
15.276.2.2 get_buffer()	1453
15.276.2.3 get_num_streams()	1453
15.276.2.4 get_stream_info()	1453
15.276.2.5 get_stream_info_for_id()	1454
15.276.2.6 get_stream_state_for_id()	1454
15.277 L4Re::Event_buffer_t< PAYLOAD > Class Template Reference	1455
15.277.1 Detailed Description	1457
15.277.2 Constructor & Destructor Documentation	1457
15.277.2.1 Event_buffer_t()	1457

15.277.3 Member Function Documentation	1458
15.277.3.1 next()	1458
15.277.3.2 put()	1458
15.278 L4Re::Event_buffer_t< PAYLOAD >::Event Struct Reference	1459
15.278.1 Detailed Description	1460
15.279 L4Re::Inhibitor Class Reference	1460
15.279.1 Detailed Description	1463
15.279.2 Member Enumeration Documentation	1463
15.279.2.1 anonymous enum	1463
15.279.3 Member Function Documentation	1464
15.279.3.1 acquire()	1464
15.279.3.2 next_lock_info()	1464
15.279.3.3 release()	1465
15.280 L4Re::Log Class Reference	1465
15.280.1 Detailed Description	1470
15.280.2 Member Function Documentation	1470
15.280.2.1 print()	1470
15.280.2.2 printf()	1470
15.281 L4Re::Mem_alloc Class Reference	1470
15.281.1 Detailed Description	1474
15.281.2 Member Enumeration Documentation	1474
15.281.2.1 Mem_alloc_flags	1474
15.281.3 Member Function Documentation	1475
15.281.3.1 alloc()	1475
15.282 L4Re::Mmio_space Struct Reference	1476
15.282.1 Detailed Description	1478
15.282.2 Member Enumeration Documentation	1478
15.282.2.1 Access_width	1478
15.282.3 Member Function Documentation	1479
15.282.3.1 mmio_read()	1479
15.282.3.2 mmio_write()	1479
15.283 L4Re::Namespace Class Reference	1480
15.283.1 Detailed Description	1483
15.283.2 Member Enumeration Documentation	1483
15.283.2.1 Query_result_flags	1483
15.283.2.2 Query_timeout	1483
15.283.2.3 Register_flags	1483
15.283.3 Member Function Documentation	1484
15.283.3.1 query() [1/2]	1484
15.283.3.2 query() [2/2]	1485
15.283.3.3 register_obj()	1486
15.283.3.4 unlink()	1486

15.284 L4Re::Ned::Cmd_control Class Reference	1487
15.284.1 Detailed Description	1487
15.284.2 Member Function Documentation	1488
15.284.2.1 execute() [1/2]	1488
15.284.2.2 execute() [2/2]	1489
15.285 L4Re::Parent Class Reference	1489
15.285.1 Detailed Description	1492
15.285.2 Member Function Documentation	1492
15.285.2.1 signal()	1492
15.286 L4Re::Random Struct Reference	1493
15.286.1 Detailed Description	1497
15.286.2 Member Function Documentation	1497
15.286.2.1 get_random()	1497
15.287 L4Re::Rm Class Reference	1498
15.287.1 Detailed Description	1503
15.287.2 Member Typedef Documentation	1503
15.287.2.1 Area	1503
15.287.2.2 Region	1503
15.287.3 Member Enumeration Documentation	1503
15.287.3.1 Detach_flags	1503
15.287.3.2 Detach_result	1504
15.287.3.3 Region_flag_shifts	1504
15.287.4 Member Function Documentation	1504
15.287.4.1 attach() [1/2]	1504
15.287.4.2 attach() [2/2]	1505
15.287.4.3 detach() [1/3]	1506
15.287.4.4 detach() [2/3]	1507
15.287.4.5 detach() [3/3]	1508
15.287.4.6 find()	1508
15.287.4.7 free_area()	1509
15.287.4.8 get_areas()	1510
15.287.4.9 get_regions()	1510
15.287.4.10 reserve_area() [1/2]	1511
15.287.4.11 reserve_area() [2/2]	1511
15.288 L4Re::Rm::F Struct Reference	1512
15.288.1 Detailed Description	1513
15.288.2 Member Enumeration Documentation	1513
15.288.2.1 Attach_flags	1513
15.288.2.2 Region_flags	1513
15.289 L4Re::Rm::Range Struct Reference	1514
15.289.1 Detailed Description	1515
15.290 L4Re::Rm::Unique_region< T > Class Template Reference	1515

15.290.1 Detailed Description	1517
15.290.2 Constructor & Destructor Documentation	1517
15.290.2.1 Unique_region() [1/3]	1517
15.290.2.2 Unique_region() [2/3]	1517
15.290.2.3 Unique_region() [3/3]	1518
15.290.2.4 ~Unique_region()	1518
15.290.3 Member Function Documentation	1518
15.290.3.1 get()	1518
15.290.3.2 is_valid()	1519
15.290.3.3 operator=()	1520
15.290.3.4 release()	1520
15.290.3.5 reset()	1520
15.291 L4Re::Smart_cap_auto< Unmap_flags > Class Template Reference	1521
15.291.1 Detailed Description	1522
15.292 L4Re::Smart_count_cap< Unmap_flags > Class Template Reference	1522
15.292.1 Detailed Description	1523
15.293 L4Re::Util::Br_manager Class Reference	1523
15.293.1 Detailed Description	1526
15.293.2 Member Function Documentation	1526
15.293.2.1 alloc_buffer_demand()	1526
15.293.2.2 get_rcv_cap()	1527
15.293.2.3 realloc_rcv_cap()	1528
15.293.2.4 set_rcv_cap_flags()	1528
15.294 L4Re::Util::Br_manager_hooks Struct Reference	1529
15.294.1 Detailed Description	1531
15.295 L4Re::Util::Br_manager_timeout_hooks Struct Reference	1531
15.295.1 Detailed Description	1534
15.296 L4Re::Util::Cap_alloc_base Class Reference	1535
15.296.1 Detailed Description	1535
15.297 L4Re::Util::Counter< COUNTER > Struct Template Reference	1536
15.297.1 Detailed Description	1536
15.298 L4Re::Util::Counter_atomic< COUNTER > Struct Template Reference	1536
15.298.1 Detailed Description	1537
15.299 L4Re::Util::Counting_cap_alloc< COUNTERTYPE > Class Template Reference	1537
15.299.1 Detailed Description	1538
15.299.2 Constructor & Destructor Documentation	1539
15.299.2.1 Counting_cap_alloc()	1539
15.299.3 Member Function Documentation	1539
15.299.3.1 alloc() [1/2]	1539
15.299.3.2 alloc() [2/2]	1540
15.299.3.3 free()	1540
15.299.3.4 release()	1541

15.299.3.5 setup()	1542
15.299.3.6 take()	1542
15.300 L4Re::Util::Dataspace_svr Class Reference	1543
15.300.1 Detailed Description	1544
15.300.2 Member Function Documentation	1544
15.300.2.1 allocate()	1544
15.300.2.2 clear()	1544
15.300.2.3 copy()	1545
15.300.2.4 is_static()	1545
15.300.2.5 map()	1545
15.300.2.6 map_hook()	1546
15.300.2.7 page_shift()	1546
15.300.2.8 release()	1547
15.300.2.9 take()	1547
15.301 L4Re::Util::Event_buffer_consumer_t< PAYLOAD > Class Template Reference	1548
15.301.1 Detailed Description	1550
15.301.2 Member Function Documentation	1550
15.301.2.1 foreach_available_event()	1550
15.301.2.2 process()	1551
15.302 L4Re::Util::Event_buffer_t< PAYLOAD > Class Template Reference	1552
15.302.1 Detailed Description	1555
15.302.2 Member Function Documentation	1555
15.302.2.1 attach()	1555
15.302.2.2 buf()	1555
15.302.2.3 detach()	1556
15.303 L4Re::Util::Event_svr< SVR > Class Template Reference	1556
15.303.1 Detailed Description	1558
15.304 L4Re::Util::Event_t< PAYLOAD > Class Template Reference	1558
15.304.1 Detailed Description	1559
15.304.2 Member Enumeration Documentation	1559
15.304.2.1 Mode	1559
15.304.3 Member Function Documentation	1559
15.304.3.1 buffer()	1559
15.304.3.2 init()	1560
15.304.3.3 init_poll()	1561
15.304.3.4 irq()	1561
15.305 L4Re::Util::Item_alloc_base Class Reference	1562
15.305.1 Detailed Description	1562
15.306 L4Re::Util::Names::Name Class Reference	1562
15.306.1 Detailed Description	1566
15.307 L4Re::Util::Names::Name_space Class Reference	1566
15.307.1 Detailed Description	1567

15.307.2 Member Function Documentation	1567
15.307.2.1 alloc_dynamic_entry()	1567
15.307.2.2 copy_receive_cap()	1567
15.307.2.3 free_capability()	1568
15.307.2.4 free_dynamic_entry()	1568
15.307.2.5 free_epiface()	1568
15.307.2.6 get_epiface()	1568
15.308 L4Re::Util::Object_registry Class Reference	1569
15.308.1 Detailed Description	1572
15.308.2 Constructor & Destructor Documentation	1572
15.308.2.1 Object_registry() [1/2]	1572
15.308.2.2 Object_registry() [2/2]	1572
15.308.3 Member Function Documentation	1573
15.308.3.1 register_irq_obj()	1573
15.308.3.2 register_obj() [1/3]	1573
15.308.3.3 register_obj() [2/3]	1574
15.308.3.4 register_obj() [3/3]	1574
15.308.3.5 unregister_obj()	1575
15.309 L4Re::Util::Ref_cap< T > Struct Template Reference	1576
15.309.1 Detailed Description	1576
15.310 L4Re::Util::Ref_del_cap< T > Struct Template Reference	1577
15.310.1 Detailed Description	1577
15.311 L4Re::Util::Registry_server< LOOP_HOOKS > Class Template Reference	1578
15.311.1 Detailed Description	1582
15.311.2 Constructor & Destructor Documentation	1582
15.311.2.1 Registry_server() [1/3]	1582
15.311.2.2 Registry_server() [2/3]	1582
15.311.2.3 Registry_server() [3/3]	1583
15.311.3 Member Function Documentation	1583
15.311.3.1 loop()	1583
15.312 L4Re::Util::Smart_cap_auto< Unmap_flags > Class Template Reference	1584
15.312.1 Detailed Description	1584
15.313 L4Re::Util::Smart_count_cap< Unmap_flags > Class Template Reference	1585
15.313.1 Detailed Description	1585
15.314 L4Re::Util::Vcon_svr< SVR > Class Template Reference	1586
15.314.1 Detailed Description	1586
15.315 L4Re::Util::Video::Goos_svr Class Reference	1587
15.315.1 Detailed Description	1588
15.315.2 Member Function Documentation	1588
15.315.2.1 get_fb()	1588
15.315.2.2 init_infos()	1589
15.315.2.3 refresh()	1589

15.315.2.4 screen_info()	1589
15.315.2.5 view_info()	1590
15.316 L4Re::Vfs::Be_file Class Reference	1590
15.316.1 Detailed Description	1592
15.316.2 Member Function Documentation	1593
15.316.2.1 data_space()	1593
15.316.2.2 fstat64()	1593
15.316.2.3 unlock_all_locks()	1593
15.317 L4Re::Vfs::Be_file_system Class Reference	1594
15.317.1 Detailed Description	1595
15.317.2 Constructor & Destructor Documentation	1596
15.317.2.1 Be_file_system()	1596
15.317.2.2 ~Be_file_system()	1596
15.317.3 Member Function Documentation	1596
15.317.3.1 type()	1596
15.318 L4Re::Vfs::Directory Class Reference	1597
15.318.1 Detailed Description	1598
15.318.2 Member Function Documentation	1599
15.318.2.1 faccessat()	1599
15.318.2.2 link()	1599
15.318.2.3 mkdir()	1599
15.318.2.4 rename()	1600
15.318.2.5 rmdir()	1600
15.318.2.6 symlink()	1601
15.318.2.7 unlink()	1601
15.319 L4Re::Vfs::File Class Reference	1602
15.319.1 Detailed Description	1604
15.320 L4Re::Vfs::File_system Class Reference	1605
15.320.1 Detailed Description	1606
15.320.2 Member Function Documentation	1606
15.320.2.1 mount()	1606
15.320.2.2 type()	1607
15.321 L4Re::Vfs::Fs Class Reference	1607
15.321.1 Detailed Description	1608
15.321.2 Member Function Documentation	1609
15.321.2.1 alloc_fd()	1609
15.321.2.2 free_fd()	1609
15.321.2.3 get_file()	1609
15.321.2.4 mount()	1610
15.321.2.5 set_fd()	1610
15.322 L4Re::Vfs::Generic_file Class Reference	1611
15.322.1 Detailed Description	1612

15.322.2 Member Function Documentation	1612
15.322.2.1 fchmod()	1612
15.322.2.2 fstat64()	1613
15.322.2.3 get_status_flags()	1613
15.322.2.4 set_status_flags()	1613
15.322.2.5 unlock_all_locks()	1614
15.323 L4Re::Vfs::Mman Class Reference	1614
15.323.1 Detailed Description	1616
15.324 L4Re::Vfs::Ops Class Reference	1616
15.324.1 Detailed Description	1619
15.325 L4Re::Vfs::Regular_file Class Reference	1619
15.325.1 Detailed Description	1621
15.325.2 Member Function Documentation	1622
15.325.2.1 data_space()	1622
15.325.2.2 fdatasync()	1622
15.325.2.3 fsync()	1622
15.325.2.4 ftruncate64()	1622
15.325.2.5 get_lock()	1623
15.325.2.6 lseek64()	1623
15.325.2.7 readv()	1624
15.325.2.8 set_lock()	1624
15.325.2.9 writev()	1624
15.326 L4Re::Vfs::Special_file Class Reference	1625
15.326.1 Detailed Description	1626
15.326.2 Member Function Documentation	1626
15.326.2.1 ioctl()	1626
15.327 L4Re::Video::Color_component Class Reference	1627
15.327.1 Detailed Description	1628
15.327.2 Constructor & Destructor Documentation	1628
15.327.2.1 Color_component()	1628
15.327.3 Member Function Documentation	1628
15.327.3.1 dump()	1628
15.327.3.2 get()	1628
15.327.3.3 operator==()	1629
15.327.3.4 set()	1629
15.327.3.5 shift()	1629
15.327.3.6 size()	1630
15.328 L4Re::Video::Goos Class Reference	1631
15.328.1 Detailed Description	1634
15.328.2 Member Enumeration Documentation	1634
15.328.2.1 Flags	1634
15.328.3 Member Function Documentation	1634

15.328.3.1 create_buffer()	1634
15.328.3.2 create_view()	1635
15.328.3.3 delete_buffer()	1635
15.328.3.4 delete_view()	1636
15.328.3.5 get_static_buffer()	1636
15.328.3.6 info()	1636
15.328.3.7 view()	1637
15.329 L4Re::Video::Goos::Info Struct Reference	1637
15.329.1 Detailed Description	1639
15.330 L4Re::Video::Pixel_info Class Reference	1639
15.330.1 Detailed Description	1640
15.330.2 Constructor & Destructor Documentation	1641
15.330.2.1 Pixel_info() [1/2]	1641
15.330.2.2 Pixel_info() [2/2]	1641
15.330.3 Member Function Documentation	1641
15.330.3.1 a() [1/2]	1641
15.330.3.2 a() [2/2]	1642
15.330.3.3 b() [1/2]	1642
15.330.3.4 b() [2/2]	1642
15.330.3.5 bits_per_pixel()	1643
15.330.3.6 bytes_per_pixel() [1/2]	1643
15.330.3.7 bytes_per_pixel() [2/2]	1643
15.330.3.8 dump()	1644
15.330.3.9 g() [1/2]	1644
15.330.3.10 g() [2/2]	1644
15.330.3.11 has_alpha()	1645
15.330.3.12 operator==()	1645
15.330.3.13 padding()	1646
15.330.3.14 r() [1/2]	1646
15.330.3.15 r() [2/2]	1646
15.331 L4Re::Video::View Class Reference	1647
15.331.1 Detailed Description	1648
15.331.2 Member Enumeration Documentation	1648
15.331.2.1 Flags	1648
15.331.2.2 V_flags	1649
15.331.3 Member Function Documentation	1649
15.331.3.1 info()	1649
15.331.3.2 refresh()	1649
15.331.3.3 set_info()	1650
15.331.3.4 set_viewport()	1650
15.331.3.5 stack()	1651
15.332 L4Re::Video::View::Info Struct Reference	1651

15.332.1 Detailed Description	1653
15.333 l4re_aux_t Struct Reference	1654
15.333.1 Detailed Description	1654
15.334 l4re_ds_stats_t Struct Reference	1655
15.334.1 Detailed Description	1655
15.335 l4re_elf_aux_mword_t Struct Reference	1655
15.335.1 Detailed Description	1656
15.336 l4re_elf_aux_t Struct Reference	1656
15.336.1 Detailed Description	1656
15.337 l4re_elf_aux_vma_t Struct Reference	1656
15.337.1 Detailed Description	1657
15.338 l4re_env_cap_entry_t Struct Reference	1657
15.338.1 Detailed Description	1658
15.338.2 Constructor & Destructor Documentation	1658
15.338.2.1 l4re_env_cap_entry_t()	1658
15.338.3 Field Documentation	1658
15.338.3.1 flags	1658
15.339 l4re_env_t Struct Reference	1659
15.339.1 Detailed Description	1660
15.339.2 Field Documentation	1660
15.339.2.1 caps	1660
15.340 l4re_event_t Struct Reference	1661
15.340.1 Detailed Description	1661
15.341 l4re_video_color_component_t Struct Reference	1662
15.341.1 Detailed Description	1662
15.342 l4re_video_goos_info_t Struct Reference	1662
15.342.1 Detailed Description	1664
15.343 l4re_video_pixel_info_t Struct Reference	1664
15.343.1 Detailed Description	1665
15.344 l4re_video_view_info_t Struct Reference	1665
15.344.1 Detailed Description	1666
15.345 l4re_video_view_t Struct Reference	1666
15.345.1 Detailed Description	1667
15.346 l4shmc_ringbuf_head_t Struct Reference	1667
15.346.1 Detailed Description	1668
15.347 l4shmc_ringbuf_t Struct Reference	1668
15.347.1 Detailed Description	1669
15.348 l4util_idt_desc_t Struct Reference	1669
15.348.1 Detailed Description	1670
15.349 l4util_idt_header_t Struct Reference	1670
15.349.1 Detailed Description	1671
15.350 l4util_l4mod_info Struct Reference	1671

15.350.1 Detailed Description	1672
15.350.2 Field Documentation	1672
15.350.2.1 vbe_ctrl_info	1672
15.351 l4util_l4mod_mod Struct Reference	1672
15.351.1 Detailed Description	1673
15.352 l4util_mb_addr_range_t Struct Reference	1673
15.352.1 Detailed Description	1674
15.353 l4util_mb_apm_t Struct Reference	1674
15.353.1 Detailed Description	1674
15.354 l4util_mb_drive_t Struct Reference	1675
15.354.1 Detailed Description	1675
15.354.2 Field Documentation	1676
15.354.2.1 drive_cylinders	1676
15.354.2.2 drive_mode	1676
15.354.2.3 drive_number	1676
15.355 l4util_mb_info_t Struct Reference	1677
15.355.1 Detailed Description	1678
15.356 l4util_mb_mod_t Struct Reference	1678
15.356.1 Detailed Description	1679
15.357 l4util_mb_vbe_ctrl_t Struct Reference	1679
15.357.1 Detailed Description	1680
15.358 l4util_mb_vbe_mode_t Struct Reference	1680
15.358.1 Detailed Description	1683
15.359 L4vbus::Device Class Reference	1684
15.359.1 Detailed Description	1686
15.359.2 Constructor & Destructor Documentation	1686
15.359.2.1 Device()	1686
15.359.3 Member Function Documentation	1687
15.359.3.1 bus_cap()	1687
15.359.3.2 dev_handle()	1688
15.359.3.3 device()	1688
15.359.3.4 device_by_hid()	1689
15.359.3.5 get_resource()	1690
15.359.3.6 is_compatible()	1690
15.359.3.7 next_device()	1691
15.359.3.8 operator"!=()	1692
15.359.3.9 operator==()	1692
15.360 L4vbus::Gpio_module Class Reference	1693
15.360.1 Detailed Description	1696
15.360.2 Member Function Documentation	1696
15.360.2.1 config_pad()	1696
15.360.2.2 get()	1696

15.360.2.3 pin()	1697
15.360.2.4 set()	1698
15.360.2.5 setup()	1699
15.361 L4vbus::Gpio_module::Pin_slice Struct Reference	1700
15.361.1 Detailed Description	1700
15.362 L4vbus::Gpio_pin Class Reference	1700
15.362.1 Detailed Description	1704
15.362.2 Member Function Documentation	1704
15.362.2.1 config_get()	1704
15.362.2.2 config_pad()	1705
15.362.2.3 config_pull()	1705
15.362.2.4 get()	1706
15.362.2.5 pin()	1706
15.362.2.6 set()	1706
15.362.2.7 setup()	1707
15.362.2.8 to_irq()	1708
15.363 L4vbus::Icu Class Reference	1709
15.363.1 Detailed Description	1712
15.363.2 Member Enumeration Documentation	1712
15.363.2.1 Src_types	1712
15.363.3 Member Function Documentation	1712
15.363.3.1 vicu()	1712
15.364 L4vbus::Pci_dev Class Reference	1713
15.364.1 Detailed Description	1716
15.364.2 Member Function Documentation	1717
15.364.2.1 cfg_read()	1717
15.364.2.2 cfg_write()	1717
15.364.2.3 irq_enable()	1718
15.365 L4vbus::Pci_host_bridge Class Reference	1719
15.365.1 Detailed Description	1723
15.365.2 Member Function Documentation	1723
15.365.2.1 cfg_read()	1723
15.365.2.2 cfg_write()	1724
15.365.2.3 irq_enable()	1724
15.366 L4vbus::Pm< DEC > Class Template Reference	1725
15.366.1 Detailed Description	1727
15.366.2 Member Function Documentation	1727
15.366.2.1 pm_resume()	1727
15.366.2.2 pm_suspend()	1727
15.367 L4vbus::Vbus Class Reference	1728
15.367.1 Detailed Description	1735
15.367.2 Member Function Documentation	1735

15.367.2.1 assign_dma_domain() [1/2]	1735
15.367.2.2 assign_dma_domain() [2/2]	1736
15.367.2.3 release_ioport()	1737
15.367.2.4 request_ioport()	1737
15.367.2.5 root()	1738
15.368 l4vbus_device_t Struct Reference	1739
15.368.1 Detailed Description	1739
15.369 l4vbus_resource_t Struct Reference	1740
15.369.1 Detailed Description	1740
15.370 L4vcpu::State Class Reference	1741
15.370.1 Detailed Description	1741
15.370.2 Constructor & Destructor Documentation	1741
15.370.2.1 State()	1741
15.370.3 Member Function Documentation	1742
15.370.3.1 add()	1742
15.370.3.2 clear()	1742
15.370.3.3 set()	1742
15.371 L4vcpu::Vcpu Class Reference	1743
15.371.1 Detailed Description	1746
15.371.2 Member Function Documentation	1746
15.371.2.1 cast() [1/2]	1746
15.371.2.2 cast() [2/2]	1746
15.371.2.3 entry_ip()	1746
15.371.2.4 entry_sp()	1747
15.371.2.5 ext_alloc()	1747
15.371.2.6 i() [1/2]	1748
15.371.2.7 i() [2/2]	1748
15.371.2.8 irq_disable_save()	1748
15.371.2.9 irq_enable()	1749
15.371.2.10 irq_restore()	1749
15.371.2.11 is_irq_entry()	1750
15.371.2.12 is_page_fault_entry()	1750
15.371.2.13 r() [1/2]	1751
15.371.2.14 r() [2/2]	1751
15.371.2.15 saved_state() [1/2]	1751
15.371.2.16 saved_state() [2/2]	1751
15.371.2.17 state() [1/2]	1752
15.371.2.18 state() [2/2]	1752
15.371.2.19 task()	1752
15.371.2.20 wait_for_event()	1753
15.372 L4virtio::Device Class Reference	1754
15.372.1 Detailed Description	1758

15.372.2 Member Function Documentation	1758
15.372.2.1 config_queue()	1758
15.372.2.2 device_config()	1759
15.372.2.3 device_notification_irq()	1760
15.372.2.4 register_ds()	1760
15.372.2.5 set_status()	1761
15.373 L4virtio::Driver::Block_device Class Reference	1762
15.373.1 Detailed Description	1766
15.373.2 Member Function Documentation	1766
15.373.2.1 add_block()	1766
15.373.2.2 process_request()	1767
15.373.2.3 process_used_queue()	1768
15.373.2.4 send_request()	1768
15.373.2.5 setup_device()	1769
15.373.2.6 start_request()	1770
15.374 L4virtio::Driver::Block_device::Handle Class Reference	1771
15.374.1 Detailed Description	1772
15.375 L4virtio::Driver::Device Class Reference	1772
15.375.1 Detailed Description	1775
15.375.2 Member Function Documentation	1775
15.375.2.1 bind_notification_irq()	1775
15.375.2.2 config_queue()	1776
15.375.2.3 driver_acknowledge()	1777
15.375.2.4 driver_connect()	1778
15.375.2.5 feature_negotiated()	1779
15.375.2.6 max_queue_size()	1780
15.375.2.7 register_ds()	1781
15.375.2.8 send()	1782
15.375.2.9 send_and_wait()	1782
15.375.2.10 wait()	1783
15.375.2.11 wait_for_next_used()	1784
15.376 L4virtio::Driver::Virtio_net_device Class Reference	1785
15.376.1 Detailed Description	1789
15.376.2 Member Function Documentation	1789
15.376.2.1 finish_rx()	1789
15.376.2.2 rx_pkt()	1789
15.376.2.3 rx_queue_size()	1790
15.376.2.4 setup_device()	1790
15.376.2.5 tx()	1791
15.376.2.6 tx_queue_size()	1792
15.376.2.7 wait_rx()	1793
15.377 L4virtio::Driver::Virtio_net_device::Packet Struct Reference	1794

15.377.1 Detailed Description	1794
15.378 L4virtio::Driver::Virtqueue Class Reference	1794
15.378.1 Detailed Description	1798
15.378.2 Member Function Documentation	1799
15.378.2.1 alloc_descriptor()	1799
15.378.2.2 desc()	1799
15.378.2.3 enqueue_descriptor()	1800
15.378.2.4 find_next_used()	1801
15.378.2.5 free_descriptor()	1801
15.378.2.6 init_queue() [1/2]	1802
15.378.2.7 init_queue() [2/2]	1803
15.378.2.8 initialize_rings()	1804
15.379 L4virtio::Ptr< T > Class Template Reference	1805
15.379.1 Detailed Description	1807
15.379.2 Member Enumeration Documentation	1807
15.379.2.1 Invalid_type	1807
15.379.3 Member Function Documentation	1807
15.379.3.1 get()	1807
15.379.3.2 is_valid()	1808
15.380 L4virtio::Svr::Bad_descriptor Struct Reference	1809
15.380.1 Detailed Description	1810
15.380.2 Member Enumeration Documentation	1810
15.380.2.1 Error	1810
15.380.3 Constructor & Destructor Documentation	1810
15.380.3.1 Bad_descriptor()	1810
15.380.4 Member Function Documentation	1810
15.380.4.1 message()	1810
15.381 L4virtio::Svr::Block_dev_base< Ds_data > Class Template Reference	1811
15.381.1 Detailed Description	1815
15.381.2 Constructor & Destructor Documentation	1815
15.381.2.1 Block_dev_base()	1815
15.381.3 Member Function Documentation	1816
15.381.3.1 finalize_request()	1816
15.381.3.2 get_writeback()	1817
15.381.3.3 process_request()	1817
15.381.3.4 set_blk_size()	1818
15.381.3.5 set_config_wce()	1818
15.381.3.6 set_discard()	1818
15.381.3.7 set_size_max()	1818
15.381.3.8 set_topology()	1819
15.381.3.9 set_write_zeroes()	1819
15.382 L4virtio::Svr::Block_request< Ds_data > Class Template Reference	1820

15.382.1 Detailed Description	1820
15.382.2 Member Function Documentation	1820
15.382.2.1 data_size()	1820
15.382.2.2 next_block()	1821
15.383 L4virtio::Svr::Data_buffer Struct Reference	1822
15.383.1 Detailed Description	1823
15.383.2 Constructor & Destructor Documentation	1823
15.383.2.1 Data_buffer()	1823
15.383.3 Member Function Documentation	1824
15.383.3.1 copy_to()	1824
15.383.3.2 done()	1824
15.383.3.3 set()	1825
15.383.3.4 skip()	1825
15.384 L4virtio::Svr::Dev_config Class Reference	1826
15.384.1 Detailed Description	1827
15.384.2 Constructor & Destructor Documentation	1828
15.384.2.1 Dev_config() [1/2]	1828
15.384.2.2 Dev_config() [2/2]	1829
15.384.3 Member Function Documentation	1830
15.384.3.1 change_queue_config()	1830
15.384.3.2 ds()	1830
15.384.3.3 get_cmd()	1831
15.384.3.4 guest_features()	1831
15.384.3.5 hdr()	1832
15.384.3.6 negotiated_features()	1832
15.384.3.7 qconfig()	1833
15.384.3.8 reset_cmd()	1834
15.384.3.9 reset_queue()	1834
15.384.3.10 set_device_needs_reset()	1835
15.384.3.11 set_status()	1836
15.384.3.12 status()	1837
15.385 L4virtio::Svr::Dev_features Struct Reference	1838
15.385.1 Detailed Description	1838
15.386 L4virtio::Svr::Dev_status Struct Reference	1839
15.386.1 Detailed Description	1840
15.386.2 Member Function Documentation	1840
15.386.2.1 running()	1840
15.387 L4virtio::Svr::Device_t< DATA > Class Template Reference	1840
15.387.1 Detailed Description	1843
15.387.2 Member Function Documentation	1843
15.387.2.1 add_trusted_dataspaces()	1843
15.387.2.2 device_error()	1844

15.387.2.3 device_notify_irq()	1844
15.387.2.4 handle_mem_cmd_write()	1845
15.387.2.5 init_mem_info()	1845
15.387.2.6 register_driver_irq()	1845
15.387.2.7 reset_queue_config()	1846
15.387.2.8 setup_queue()	1847
15.388 L4virtio::Svr::Driver_mem_list_t< DATA > Class Template Reference	1848
15.388.1 Detailed Description	1850
15.388.2 Member Function Documentation	1850
15.388.2.1 add()	1850
15.388.2.2 find()	1851
15.388.2.3 full()	1851
15.388.2.4 init()	1852
15.388.2.5 load_desc() [1/3]	1852
15.388.2.6 load_desc() [2/3]	1853
15.388.2.7 load_desc() [3/3]	1854
15.388.2.8 remove()	1855
15.389 L4virtio::Svr::Driver_mem_region_t< DATA > Class Template Reference	1856
15.389.1 Detailed Description	1857
15.389.2 Constructor & Destructor Documentation	1858
15.389.2.1 Driver_mem_region_t()	1858
15.389.3 Member Function Documentation	1859
15.389.3.1 contains()	1859
15.389.3.2 drv_base()	1860
15.389.3.3 ds()	1860
15.389.3.4 ds_offset()	1861
15.389.3.5 empty()	1861
15.389.3.6 flags()	1861
15.389.3.7 is_writable()	1861
15.389.3.8 local()	1861
15.389.3.9 local_base()	1862
15.389.3.10 size()	1863
15.390 L4virtio::Svr::Request_processor Class Reference	1863
15.390.1 Detailed Description	1864
15.390.2 Member Function Documentation	1864
15.390.2.1 current_flags()	1864
15.390.2.2 has_more()	1865
15.390.2.3 next()	1865
15.390.2.4 start() [1/2]	1867
15.390.2.5 start() [2/2]	1868
15.391 L4virtio::Svr::Virtqueue Class Reference	1870
15.391.1 Detailed Description	1873

15.391.2 Member Function Documentation	1873
15.391.2.1 consumed()	1873
15.391.2.2 desc()	1874
15.391.2.3 desc_avail()	1875
15.391.2.4 disable_notify()	1875
15.391.2.5 enable_notify()	1876
15.391.2.6 next_avail()	1876
15.392 L4virtio::Svr::Virtqueue::Head_desc Class Reference	1877
15.392.1 Detailed Description	1877
15.392.2 Member Function Documentation	1877
15.392.2.1 desc()	1877
15.392.2.2 operator Null_ptr_check const *()	1878
15.392.2.3 valid()	1878
15.393 L4virtio::Virtqueue Class Reference	1879
15.393.1 Detailed Description	1882
15.393.2 Member Function Documentation	1882
15.393.2.1 avail_align()	1882
15.393.2.2 avail_size()	1882
15.393.2.3 desc_align()	1883
15.393.2.4 desc_size()	1884
15.393.2.5 disable()	1885
15.393.2.6 dump()	1885
15.393.2.7 get_avail_idx()	1886
15.393.2.8 get_tail_avail_idx()	1886
15.393.2.9 no_notify_guest()	1886
15.393.2.10 no_notify_host() [1/2]	1887
15.393.2.11 no_notify_host() [2/2]	1887
15.393.2.12 num()	1888
15.393.2.13 ready()	1888
15.393.2.14 setup()	1889
15.393.2.15 setup_simple()	1890
15.393.2.16 total_size() [1/2]	1891
15.393.2.17 total_size() [2/2]	1892
15.393.2.18 used_align()	1893
15.393.2.19 used_size()	1893
15.394 L4virtio::Virtqueue::Avail Class Reference	1894
15.394.1 Detailed Description	1895
15.395 L4virtio::Virtqueue::Avail::Flags Struct Reference	1896
15.395.1 Detailed Description	1896
15.395.2 Member Typedef Documentation	1896
15.395.2.1 no_irq_bfm_t	1896
15.396 L4virtio::Virtqueue::Desc Class Reference	1897

15.396.1 Detailed Description	1898
15.397 L4virtio::Virtqueue::Desc::Flags Struct Reference	1898
15.397.1 Detailed Description	1899
15.397.2 Member Typedef Documentation	1899
15.397.2.1 indirect_bfm_t	1899
15.397.2.2 next_bfm_t	1899
15.397.2.3 write_bfm_t	1899
15.398 L4virtio::Virtqueue::Used Class Reference	1900
15.398.1 Detailed Description	1900
15.399 L4virtio::Virtqueue::Used::Flags Struct Reference	1901
15.399.1 Detailed Description	1901
15.399.2 Member Typedef Documentation	1901
15.399.2.1 no_notify_bfm_t	1901
15.400 L4virtio::Virtqueue::Used_elem Struct Reference	1902
15.400.1 Detailed Description	1902
15.400.2 Constructor & Destructor Documentation	1902
15.400.2.1 Used_elem()	1902
15.401 l4virtio_block_config_t Struct Reference	1903
15.401.1 Detailed Description	1904
15.402 l4virtio_block_discard_t Struct Reference	1904
15.402.1 Detailed Description	1904
15.403 l4virtio_block_header_t Struct Reference	1905
15.403.1 Detailed Description	1905
15.404 l4virtio_config_hdr_t Struct Reference	1906
15.404.1 Detailed Description	1907
15.404.2 Field Documentation	1907
15.404.2.1 status	1907
15.405 l4virtio_config_queue_t Struct Reference	1907
15.405.1 Detailed Description	1908
15.406 l4virtio_input_absinfo_t Struct Reference	1909
15.406.1 Detailed Description	1909
15.407 l4virtio_input_config_t Struct Reference	1909
15.407.1 Detailed Description	1910
15.408 l4virtio_input_devids_t Struct Reference	1910
15.408.1 Detailed Description	1910
15.409 l4virtio_input_event_t Struct Reference	1910
15.409.1 Detailed Description	1911
15.410 l4virtio_net_config_t Struct Reference	1911
15.410.1 Detailed Description	1911
15.411 l4virtio_net_header_t Struct Reference	1911
15.411.1 Detailed Description	1912

16 File Documentation	1913
16.1 asm_access.h	1913
16.2 asm_access.h	1913
16.3 asm_access.h	1914
16.4 asm_access.h	1915
16.5 asm_access.h	1915
16.6 asm_access.h	1916
16.7 asm_access.h	1916
16.8 asm_access_gen.h	1917
16.9 hw_mmio_register_block	1917
16.10 hw_register_block	1918
16.11 io_regblock.h	1923
16.12 io_regblock_port.h	1926
16.13 poll_timeout_counter.h	1926
16.14 uart_16550.h	1927
16.15 uart_16550_dw.h	1928
16.16 uart_base.h	1929
16.17 uart_cadence.h	1930
16.18 uart_dcc-v6.h	1930
16.19 uart_dm.h	1931
16.20 uart_dummy.h	1931
16.21 uart_geni.h	1932
16.22 uart_imx.h	1932
16.23 uart_leon3.h	1933
16.24 uart_linflex.h	1934
16.25 uart_lpuart.h	1934
16.26 uart_mvebu.h	1935
16.27 uart_of.h	1935
16.28 uart_omap35x.h	1936
16.29 uart_pl011.h	1936
16.30 uart_s3c2410.h	1937
16.31 uart_sa1000.h	1938
16.32 uart_sh.h	1938
16.33 cmd_control	1939
16.34 Makefile	1940
16.35 Makefile	1940
16.36 Makefile	1940
16.37 amd64/l4/util/idt.h File Reference	1940
16.37.1 Detailed Description	1941
16.38 idt.h	1942
16.39 x86/l4/util/idt.h File Reference	1942
16.39.1 Detailed Description	1943

16.40	idt.h	1943
16.41	amd64/l4/util/perform.h File Reference	1944
16.41.1	Detailed Description	1945
16.42	perform.h	1945
16.43	x86/l4/util/perform.h File Reference	1950
16.43.1	Detailed Description	1951
16.44	perform.h	1951
16.45	amd64/l4/util/rdtsc.h File Reference	1956
16.45.1	Detailed Description	1958
16.46	rdtsc.h	1958
16.47	x86/l4/util/rdtsc.h File Reference	1960
16.47.1	Detailed Description	1962
16.48	rdtsc.h	1962
16.49	amd64/l4/util/spin.h File Reference	1965
16.49.1	Detailed Description	1965
16.50	spin.h	1965
16.51	x86/l4/util/spin.h File Reference	1966
16.51.1	Detailed Description	1966
16.52	spin.h	1967
16.53	amd64/l4/util/util.h File Reference	1967
16.53.1	Detailed Description	1968
16.53.2	Function Documentation	1968
16.53.2.1	l4_sleep()	1968
16.53.2.2	l4util_micros2l4to()	1969
16.54	util.h	1969
16.55	util.h	1970
16.56	x86/l4/util/util.h File Reference	1971
16.56.1	Detailed Description	1972
16.56.2	Function Documentation	1972
16.56.2.1	l4_sleep()	1972
16.56.2.2	l4util_micros2l4to()	1973
16.57	util.h	1973
16.58	amd64/l4/sys/segment.h File Reference	1974
16.58.1	Detailed Description	1975
16.58.2	Enumeration Type Documentation	1975
16.58.2.1	L4_sys_segment	1975
16.58.2.2	L4_task_ldt_x86_consts	1976
16.58.3	Function Documentation	1976
16.58.3.1	fiasco_amd64_segment_info()	1976
16.58.3.2	fiasco_amd64_set_fs()	1977
16.58.3.3	fiasco_amd64_set_segment_base()	1978
16.59	segment.h	1979

16.60 amd64/l4/l4/sys/segment.h File Reference	1880
16.60.1 Detailed Description	1881
16.60.2 Function Documentation	1881
16.60.2.1 fiasco_amd64_set_fs()	1881
16.60.2.2 fiasco_amd64_set_segment_base()	1882
16.61 segment.h	1883
16.62 x86/l4/l4/sys/segment.h File Reference	1883
16.62.1 Detailed Description	1884
16.62.2 Enumeration Type Documentation	1885
16.62.2.1 L4_task_ldt_x86_consts	1885
16.63 segment.h	1886
16.64 x86/l4/l4/sys/segment.h File Reference	1887
16.64.1 Detailed Description	1887
16.65 segment.h	1888
16.66 amd64/l4/util/port_io.h File Reference	1888
16.66.1 Detailed Description	1889
16.67 port_io.h	1889
16.68 amd64/l4/l4/util/port_io.h File Reference	1889
16.68.1 Detailed Description	1900
16.69 port_io.h	1900
16.70 x86/l4/util/port_io.h File Reference	1900
16.70.1 Detailed Description	1902
16.71 port_io.h	1902
16.72 x86/l4/l4/util/port_io.h File Reference	1904
16.72.1 Detailed Description	1905
16.72.2 Function Documentation	1906
16.72.2.1 l4util_ioport_map()	1906
16.73 port_io.h	1907
16.74 __kip-arch.h	1907
16.75 __kip-arch.h	1908
16.76 __kip-arch.h	1908
16.77 amd64/l4/sys/__vcpu-arch.h File Reference	1908
16.77.1 Detailed Description	1909
16.77.2 Enumeration Type Documentation	2000
16.77.2.1 anonymous enum	2000
16.78 __vcpu-arch.h	2000
16.79 arm/l4/sys/__vcpu-arch.h File Reference	2001
16.79.1 Detailed Description	2002
16.79.2 Enumeration Type Documentation	2002
16.79.2.1 anonymous enum	2002
16.79.2.2 L4_vcpu_e_field_ids	2002
16.80 __vcpu-arch.h	2003

16.81 x86/l4/sys/__vcpu-arch.h File Reference	2004
16.81.1 Detailed Description	2005
16.81.2 Enumeration Type Documentation	2005
16.81.2.1 anonymous enum	2005
16.82 __vcpu-arch.h	2005
16.83 ktrace_events.h	2006
16.84 ktrace_events.h	2009
16.85 ktrace_events.h	2012
16.86 amd64/l4/sys/linkage.h File Reference	2016
16.86.1 Detailed Description	2016
16.87 linkage.h	2016
16.88 arm/l4/sys/linkage.h File Reference	2017
16.88.1 Detailed Description	2017
16.89 linkage.h	2017
16.90 x86/l4/sys/linkage.h File Reference	2017
16.90.1 Detailed Description	2018
16.91 linkage.h	2018
16.92 arm/l4/sys/mem_op.h File Reference	2018
16.92.1 Detailed Description	2019
16.93 mem_op.h	2020
16.94 vm.h	2021
16.95 arm/l4/sys/vm.h File Reference	2021
16.95.1 Detailed Description	2021
16.96 vm.h	2022
16.97 vm.h	2023
16.98 amd64/l4/util/bitops_arch.h File Reference	2023
16.98.1 Detailed Description	2023
16.99 bitops_arch.h	2023
16.100 arm/l4/util/bitops_arch.h File Reference	2026
16.100.1 Detailed Description	2027
16.101 bitops_arch.h	2027
16.102 x86/l4/util/bitops_arch.h File Reference	2027
16.102.1 Detailed Description	2027
16.103 bitops_arch.h	2027
16.104 amd64/l4/util/cpu.h File Reference	2030
16.104.1 Detailed Description	2031
16.105 cpu.h	2032
16.106 arm/l4/util/cpu.h File Reference	2033
16.106.1 Detailed Description	2033
16.107 cpu.h	2033
16.108 x86/l4/util/cpu.h File Reference	2033
16.108.1 Detailed Description	2034

16.109 <code>cpu.h</code>	2034
16.110 <code>amd64/l4/util/l4_macros.h</code> File Reference	2035
16.110.1 Detailed Description	2035
16.111 <code>l4_macros.h</code>	2036
16.112 <code>arm/l4/util/l4_macros.h</code> File Reference	2036
16.112.1 Detailed Description	2036
16.113 <code>l4_macros.h</code>	2036
16.114 <code>l4/util/l4_macros.h</code> File Reference	2037
16.114.1 Detailed Description	2037
16.115 <code>l4_macros.h</code>	2037
16.116 <code>x86/l4/util/l4_macros.h</code> File Reference	2037
16.116.1 Detailed Description	2038
16.117 <code>l4_macros.h</code>	2038
16.118 <code>amd64/l4/util/mbi_argv.h</code> File Reference	2038
16.118.1 Detailed Description	2039
16.119 <code>mbi_argv.h</code>	2039
16.120 <code>arm/l4/util/mbi_argv.h</code> File Reference	2040
16.120.1 Detailed Description	2040
16.121 <code>mbi_argv.h</code>	2041
16.122 <code>x86/l4/util/mbi_argv.h</code> File Reference	2041
16.122.1 Detailed Description	2042
16.123 <code>mbi_argv.h</code>	2042
16.124 <code>arm/l4/l4/sys/syscall_defs.h</code> File Reference	2043
16.124.1 Detailed Description	2043
16.125 <code>syscall_defs.h</code>	2043
16.126 <code>contrib/libio-io/l4/io/io.h</code> File Reference	2044
16.126.1 Function Documentation	2045
16.126.1.1 <code>l4io_get_root_device()</code>	2045
16.126.1.2 <code>l4io_iterate_devices()</code>	2045
16.126.1.3 <code>l4io_request_all_ioports()</code>	2045
16.126.1.4 <code>l4io_request_icu()</code>	2046
16.127 <code>io.h</code>	2046
16.128 <code>l4/cxx/alloc.h</code> File Reference	2047
16.128.1 Detailed Description	2047
16.129 <code>alloc.h</code>	2048
16.130 <code>arith</code>	2048
16.131 <code>l4/cxx/avl_map</code> File Reference	2049
16.131.1 Detailed Description	2050
16.132 <code>avl_map</code>	2050
16.133 <code>l4/cxx/avl_set</code> File Reference	2052
16.133.1 Detailed Description	2053
16.134 <code>avl_set</code>	2053

16.135 I4/cxx/avl_tree File Reference	2056
16.135.1 Detailed Description	2058
16.136 avl_tree	2058
16.137 I4/cxx/basic_ostream File Reference	2062
16.137.1 Detailed Description	2063
16.138 basic_ostream	2063
16.139 I4/cxx/basic_vector.h File Reference	2066
16.139.1 Detailed Description	2067
16.140 basic_vector.h	2067
16.141 bitfield	2068
16.142 bitmap	2070
16.143 I4/cxx/bits/bst.h File Reference	2072
16.143.1 Detailed Description	2074
16.144 bst.h	2075
16.145 I4/cxx/bits/bst_base.h File Reference	2077
16.145.1 Detailed Description	2079
16.146 bst_base.h	2079
16.147 I4/cxx/bits/bst_iter.h File Reference	2080
16.147.1 Detailed Description	2081
16.148 bst_iter.h	2082
16.149 list_basics.h	2083
16.150 I4/cxx/bits/smart_ptr_list.h File Reference	2085
16.150.1 Detailed Description	2086
16.151 smart_ptr_list.h	2087
16.152 type_traits.h	2088
16.153 dlist	2091
16.154 I4/cxx/exceptions File Reference	2094
16.154.1 Detailed Description	2096
16.155 exceptions	2096
16.156 hlist	2098
16.157 I4/cxx/iostream File Reference	2101
16.157.1 Detailed Description	2102
16.158 iostream	2102
16.159 I4/cxx/ipc_helper File Reference	2102
16.159.1 Detailed Description	2103
16.160 ipc_helper	2103
16.161 I4/cxx/ipc_stream File Reference	2104
16.161.1 Detailed Description	2107
16.161.2 Function Documentation	2107
16.161.2.1 operator<<() [1/4]	2107
16.161.2.2 operator<<() [2/4]	2107
16.161.2.3 operator<<() [3/4]	2108

16.161.2.4 operator<<() [4/4]	2109
16.161.2.5 operator>>() [1/6]	2110
16.161.2.6 operator>>() [2/6]	2111
16.161.2.7 operator>>() [3/6]	2112
16.161.2.8 operator>>() [4/6]	2113
16.161.2.9 operator>>() [5/6]	2113
16.161.2.10 operator>>() [6/6]	2114
16.162 ipc_stream	2115
16.163 ipc_timeout_queue	2123
16.164 l4/cxx/l4iostream File Reference	2125
16.164.1 Detailed Description	2125
16.165 l4iostream	2126
16.166 l4/cxx/l4types.h File Reference	2126
16.166.1 Detailed Description	2127
16.167 l4types.h	2127
16.168 list	2128
16.169 list_alloc	2132
16.170 l4/cxx/main_thread File Reference	2137
16.170.1 Detailed Description	2138
16.171 main_thread	2138
16.172 minmax	2139
16.173 observer	2139
16.174 l4/cxx/pair File Reference	2140
16.174.1 Detailed Description	2141
16.175 pair	2141
16.176 ref_ptr	2142
16.177 l4/cxx/ref_ptr_list File Reference	2145
16.177.1 Detailed Description	2146
16.178 ref_ptr_list	2147
16.179 slab_alloc	2147
16.180 slist	2151
16.181 static_container	2153
16.182 static_vector	2154
16.183 std_alloc	2155
16.184 l4/cxx/std_exc_io File Reference	2156
16.184.1 Detailed Description	2156
16.185 std_exc_io	2156
16.186 std_ops	2157
16.187 string	2157
16.188 l4/cxx/string.h File Reference	2160
16.188.1 Detailed Description	2163
16.189 string.h	2163

16.190 type_list	2163
16.191 type_traits	2164
16.192 unique_ptr	2168
16.193 l4/cxx/unique_ptr_list File Reference	2170
16.193.1 Detailed Description	2171
16.194 unique_ptr_list	2171
16.195 utils	2171
16.196 weak_ref	2172
16.197 amd64/l4/util/irq.h File Reference	2173
16.197.1 Detailed Description	2174
16.197.2 Function Documentation	2174
16.197.2.1 l4util_irq_acknowledge()	2174
16.198 irq.h	2175
16.199 arm/l4/util/irq.h File Reference	2175
16.199.1 Detailed Description	2176
16.200 irq.h	2176
16.201 l4/irq/irq.h File Reference	2177
16.201.1 Detailed Description	2178
16.202 irq.h	2178
16.203 l4/sys/irq.h File Reference	2179
16.203.1 Detailed Description	2181
16.204 irq.h	2181
16.205 x86/l4/util/irq.h File Reference	2183
16.205.1 Detailed Description	2184
16.205.2 Function Documentation	2184
16.205.2.1 l4util_irq_acknowledge()	2184
16.206 irq.h	2185
16.207 backend	2186
16.208 default_ops_impl.h	2189
16.209 fd_store.h	2190
16.210 fd_store_impl.h	2191
16.211 ns_fs.h	2191
16.212 ns_fs_impl.h	2192
16.213 ro_file.h	2197
16.214 ro_file_impl.h	2198
16.215 vcon_stream.h	2199
16.216 vcon_stream_impl.h	2200
16.217 vfs_impl.h	2202
16.218 vfs.h	2214
16.219 virtio-net	2221
16.220 l4virtio	2224
16.221 l4virtio	2227

16.222 l4virtio	2228
16.223 virtio	2239
16.224 virtio-block	2243
16.225 virtio-block	2246
16.226 virtio.h	2252
16.227 virtio_block.h	2255
16.228 virtio_input.h	2256
16.229 virtio_net.h	2257
16.230 virtqueue	2257
16.231 block_device_mgr.h	2262
16.232 debug.h	2267
16.233 l4/re/c/debug.h File Reference	2267
16.233.1 Detailed Description	2268
16.234 debug.h	2268
16.235 device.h	2269
16.236 errand.h	2270
16.237 gpt.h	2272
16.238 inout_memory.h	2272
16.239 part_device.h	2274
16.240 partition.h	2276
16.241 request_queue.h	2279
16.242 types.h	2280
16.243 types.h	2281
16.244 l4/sys/types.h File Reference	2282
16.244.1 Detailed Description	2284
16.244.2 Function Documentation	2284
16.244.2.1 l4_capability_next()	2284
16.245 types.h	2284
16.246 virtio_client.h	2287
16.247 l4/libedid/edid.h File Reference	2295
16.248 edid.h	2296
16.249 l4/libgfxbitmap/bitmap.h File Reference	2297
16.249.1 Detailed Description	2298
16.249.2 Macro Definition Documentation	2298
16.249.2.1 pSLIM_BMAP_START_LSB	2298
16.249.3 Typedef Documentation	2299
16.249.3.1 gfxbitmap_color_pix_t	2299
16.249.3.2 gfxbitmap_color_t	2299
16.249.4 Function Documentation	2299
16.249.4.1 gfxbitmap_bmap()	2299
16.249.4.2 gfxbitmap_convert_color()	2300
16.249.4.3 gfxbitmap_copy()	2300

16.249.4.4 gfbitmap_fill()	2300
16.249.4.5 gfbitmap_set()	2301
16.250 bitmap.h	2301
16.251 l4/libgfbitmap/font.h File Reference	2302
16.251.1 Detailed Description	2304
16.251.2 Enumeration Type Documentation	2304
16.251.2.1 anonymous enum	2304
16.251.3 Function Documentation	2304
16.251.3.1 gfbitmap_font_data()	2304
16.251.3.2 gfbitmap_font_get()	2305
16.251.3.3 gfbitmap_font_height()	2305
16.251.3.4 gfbitmap_font_init()	2305
16.251.3.5 gfbitmap_font_text()	2306
16.251.3.6 gfbitmap_font_text_scale()	2306
16.251.3.7 gfbitmap_font_width()	2307
16.252 font.h	2307
16.253 l4/libgfbitmap/support File Reference	2308
16.253.1 Detailed Description	2308
16.254 support	2309
16.255 l4/re/c/dataspace.h File Reference	2309
16.255.1 Detailed Description	2310
16.256 dataspace.h	2311
16.257 l4/re/c/dma_space.h File Reference	2312
16.257.1 Detailed Description	2313
16.257.2 Enumeration Type Documentation	2313
16.257.2.1 l4re_dma_space_direction	2313
16.257.2.2 l4re_dma_space_space_attrbs	2313
16.258 dma_space.h	2314
16.259 l4/re/c/event.h File Reference	2315
16.259.1 Detailed Description	2316
16.260 event.h	2316
16.261 l4/re/event.h File Reference	2317
16.261.1 Detailed Description	2318
16.262 event.h	2318
16.263 event_buffer.h	2319
16.264 l4/re/c/inhibitor.h File Reference	2319
16.264.1 Detailed Description	2320
16.264.2 Function Documentation	2320
16.264.2.1 l4re_inhibitor_acquire()	2320
16.264.2.2 l4re_inhibitor_next_lock_info()	2321
16.264.2.3 l4re_inhibitor_release()	2321
16.265 inhibitor.h	2322

16.266 l4/re/c/log.h File Reference	2322
16.266.1 Detailed Description	2323
16.267 log.h	2323
16.268 l4/re/c/mem_alloc.h File Reference	2324
16.268.1 Detailed Description	2325
16.269 mem_alloc.h	2326
16.270 l4/re/c/namespace.h File Reference	2326
16.270.1 Detailed Description	2327
16.271 namespace.h	2328
16.272 l4/re/c/rm.h File Reference	2328
16.272.1 Detailed Description	2330
16.273 rm.h	2330
16.274 l4/re/c/util/cap_alloc.h File Reference	2332
16.274.1 Detailed Description	2333
16.275 cap_alloc.h	2333
16.276 l4/re/c/util/kumem_alloc.h File Reference	2334
16.276.1 Detailed Description	2335
16.276.2 Function Documentation	2335
16.276.2.1 l4re_util_kumem_alloc()	2335
16.277 kumem_alloc.h	2335
16.278 l4/re/c/util/video/goos_fb.h File Reference	2336
16.278.1 Detailed Description	2336
16.279 goos_fb.h	2337
16.280 l4/re/c/video/colors.h File Reference	2338
16.280.1 Detailed Description	2339
16.281 colors.h	2339
16.282 l4/re/c/video/goos.h File Reference	2340
16.282.1 Detailed Description	2341
16.283 goos.h	2342
16.284 l4/re/c/video/view.h File Reference	2343
16.284.1 Detailed Description	2344
16.285 view.h	2345
16.286 console	2346
16.287 l4/re/dataspace File Reference	2346
16.287.1 Detailed Description	2347
16.288 dataspace	2348
16.289 l4/re/dataspace-sys.h File Reference	2349
16.289.1 Detailed Description	2350
16.290 dataspace-sys.h	2350
16.291 l4/re/dma_space File Reference	2350
16.292 dma_space	2352
16.293 l4/re/elf_aux.h File Reference	2353

16.293.1 Detailed Description	2354
16.294 elf_aux.h	2354
16.295 l4/re/env File Reference	2355
16.295.1 Detailed Description	2356
16.296 env	2356
16.297 l4/re/env.h File Reference	2358
16.297.1 Detailed Description	2359
16.297.2 Typedef Documentation	2359
16.297.2.1 l4re_env_t	2359
16.298 env.h	2360
16.299 l4/re/error_helper File Reference	2361
16.299.1 Detailed Description	2363
16.300 error_helper	2363
16.301 event-sys.h	2364
16.302 event_enums.h	2365
16.303 l4/re/impl/dataspace_impl.h File Reference	2371
16.303.1 Detailed Description	2372
16.304 dataspace_impl.h	2372
16.305 l4/re/impl/mem_alloc_impl.h File Reference	2374
16.305.1 Detailed Description	2375
16.306 mem_alloc_impl.h	2375
16.307 l4/re/impl/namespace_impl.h File Reference	2375
16.307.1 Detailed Description	2376
16.308 namespace_impl.h	2376
16.309 l4/re/impl/rm_impl.h File Reference	2378
16.309.1 Detailed Description	2379
16.310 rm_impl.h	2379
16.311 inhibitor	2380
16.312 inhibitor-sys.h	2381
16.313 l4/re/l4aux.h File Reference	2381
16.313.1 Detailed Description	2382
16.314 l4aux.h	2382
16.315 l4/re/log File Reference	2382
16.315.1 Detailed Description	2384
16.316 log	2384
16.317 l4/re/log-sys.h File Reference	2384
16.317.1 Detailed Description	2385
16.318 log-sys.h	2385
16.319 l4/re/mem_alloc File Reference	2385
16.319.1 Detailed Description	2387
16.320 mem_alloc	2387
16.321 l4/re/mem_alloc-sys.h File Reference	2387

16.321.1 Detailed Description	2388
16.322 mem_alloc-sys.h	2388
16.323 l4/re/mmio_space File Reference	2389
16.323.1 Detailed Description	2390
16.324 mmio_space	2390
16.325 l4/re/namespace File Reference	2390
16.325.1 Detailed Description	2392
16.326 namespace	2392
16.327 l4/re/namespace-sys.h File Reference	2393
16.327.1 Detailed Description	2393
16.328 namespace-sys.h	2394
16.329 l4/re/parent File Reference	2394
16.329.1 Detailed Description	2396
16.330 parent	2396
16.331 l4/re/parent-sys.h File Reference	2396
16.331.1 Detailed Description	2397
16.332 parent-sys.h	2397
16.333 l4/re/protocols.h File Reference	2397
16.333.1 Detailed Description	2398
16.333.2 Enumeration Type Documentation	2398
16.333.2.1 L4re_protocols	2398
16.334 protocols.h	2398
16.335 l4/re/random File Reference	2399
16.335.1 Detailed Description	2400
16.336 random	2400
16.337 l4/re/rm File Reference	2400
16.337.1 Detailed Description	2402
16.338 rm	2402
16.339 l4/re/rm-sys.h File Reference	2406
16.339.1 Detailed Description	2406
16.340 rm-sys.h	2406
16.341 l4/re/util/bitmap_cap_alloc File Reference	2407
16.341.1 Detailed Description	2407
16.342 bitmap_cap_alloc	2408
16.343 br_manager	2409
16.344 l4/re/util/cap File Reference	2411
16.344.1 Detailed Description	2412
16.345 cap	2412
16.346 l4/re/cap_alloc File Reference	2412
16.346.1 Detailed Description	2414
16.347 cap_alloc	2414
16.348 l4/re/util/cap_alloc File Reference	2416

16.348.1 Detailed Description	2418
16.349 cap_alloc	2418
16.350 I4/re/util/cap_alloc_impl.h File Reference	2419
16.350.1 Detailed Description	2420
16.351 cap_alloc_impl.h	2420
16.352 I4/re/util/counting_cap_alloc File Reference	2421
16.352.1 Detailed Description	2422
16.353 counting_cap_alloc	2423
16.354 dataspace_svr	2425
16.355 I4/re/debug File Reference	2427
16.355.1 Detailed Description	2428
16.356 debug	2429
16.357 debug	2429
16.358 env_ns	2431
16.359 event	2432
16.360 I4/re/util/event File Reference	2435
16.361 event	2436
16.362 event_buffer	2437
16.363 event_svr	2438
16.364 icu_svr	2440
16.365 I4/re/util/item_alloc File Reference	2442
16.365.1 Detailed Description	2443
16.366 item_alloc	2444
16.367 I4/re/util/kumem_alloc File Reference	2445
16.367.1 Detailed Description	2446
16.368 kumem_alloc	2446
16.369 name_space_svr	2446
16.370 object_registry	2449
16.371 poll_timeout_kipclock	2452
16.372 I4/re/util/region_mapping File Reference	2452
16.372.1 Detailed Description	2453
16.373 region_mapping	2454
16.374 region_mapping_svr_2	2459
16.375 I4/re/shared_cap File Reference	2461
16.375.1 Detailed Description	2463
16.376 shared_cap	2463
16.377 I4/re/util/shared_cap File Reference	2464
16.377.1 Detailed Description	2466
16.378 shared_cap	2466
16.379 I4/re/unique_cap File Reference	2467
16.379.1 Detailed Description	2469
16.380 unique_cap	2469

16.381 l4/re/util/unique_cap File Reference	2469
16.381.1 Detailed Description	2471
16.382 unique_cap	2471
16.383 vcon_svr	2472
16.384 goos_fb	2473
16.385 goos_svr	2474
16.386 colors	2476
16.387 goos	2477
16.388 l4/re/video/goos-sys.h File Reference	2480
16.388.1 Detailed Description	2481
16.389 goos-sys.h	2481
16.390 view	2482
16.391 l4/shmc/ringbuf.h File Reference	2482
16.391.1 Macro Definition Documentation	2483
16.391.1.1 L4SHMC_RINGBUF_DATA	2483
16.391.1.2 L4SHMC_RINGBUF_DATA_SIZE	2484
16.391.1.3 L4SHMC_RINGBUF_HEAD	2484
16.391.2 Function Documentation	2484
16.391.2.1 l4shmc_rb_attach_receiver()	2484
16.391.2.2 l4shmc_rb_attach_sender()	2485
16.391.2.3 l4shmc_rb_deinit_buffer()	2485
16.391.2.4 l4shmc_rb_init_buffer()	2485
16.391.2.5 l4shmc_rb_init_receiver()	2486
16.391.2.6 l4shmc_rb_receiver_copy_out()	2487
16.391.2.7 l4shmc_rb_receiver_notify_done()	2487
16.391.2.8 l4shmc_rb_receiver_read_next_size()	2487
16.391.2.9 l4shmc_rb_receiver_wait_for_data()	2488
16.391.2.10 l4shmc_rb_sender_alloc_packet()	2488
16.391.2.11 l4shmc_rb_sender_commit_packet()	2488
16.391.2.12 l4shmc_rb_sender_next_copy_in()	2489
16.391.2.13 l4shmc_rb_sender_put_data()	2489
16.392 ringbuf.h	2490
16.393 l4/shmc/shmc.h File Reference	2491
16.393.1 Detailed Description	2494
16.394 shmc.h	2494
16.395 l4/sigma0/sigma0.h File Reference	2496
16.395.1 Detailed Description	2498
16.396 sigma0.h	2498
16.397 l4/sys/__kernel_object_impl.h File Reference	2500
16.397.1 Detailed Description	2500
16.398 __kernel_object_impl.h	2500
16.399 __kip-32bit.h	2501

16.400	__kip-64bit.h	2502
16.401	l4/sys/__ktrace-impl.h File Reference	2503
16.401.1	Detailed Description	2505
16.402	__ktrace-impl.h	2505
16.403	__l4_fpage.h	2506
16.404	__task-arm.h	2510
16.405	__timeout.h	2510
16.406	l4/sys/__typeinfo.h File Reference	2513
16.406.1	Detailed Description	2515
16.407	__typeinfo.h	2515
16.408	__vcpu-arm.h	2525
16.409	l4/sys/__vm-arm.h File Reference	2526
16.409.1	Detailed Description	2528
16.410	__vm-arm.h	2528
16.411	__vm-svm.h	2528
16.412	__vm-vmx.h	2530
16.413	l4/sys/arm_smccc File Reference	2534
16.413.1	Detailed Description	2535
16.414	arm_smccc	2536
16.415	l4/sys/arm_smccc.h File Reference	2536
16.415.1	Detailed Description	2537
16.415.2	Function Documentation	2537
16.415.2.1	l4_arm_smccc_call()	2537
16.416	arm_smccc.h	2538
16.417	amd64/l4/sys/cache.h File Reference	2539
16.417.1	Detailed Description	2540
16.418	cache.h	2540
16.419	arm/l4/sys/cache.h File Reference	2541
16.419.1	Detailed Description	2542
16.420	cache.h	2542
16.421	l4/sys/cache.h File Reference	2543
16.421.1	Detailed Description	2545
16.422	cache.h	2545
16.423	x86/l4/sys/cache.h File Reference	2546
16.423.1	Detailed Description	2546
16.424	cache.h	2546
16.425	l4/sys/capability File Reference	2547
16.425.1	Detailed Description	2549
16.425.2	Macro Definition Documentation	2549
16.425.2.1	L4_DISABLE_COPY	2549
16.426	capability	2549
16.427	l4/sys/compiler.h File Reference	2551

16.427.1 Detailed Description	2552
16.428 compiler.h	2552
16.429 amd64/l4/sys/consts.h File Reference	2555
16.429.1 Detailed Description	2555
16.430 consts.h	2555
16.431 arm/l4/sys/consts.h File Reference	2556
16.431.1 Detailed Description	2556
16.432 consts.h	2556
16.433 l4/re/consts.h File Reference	2557
16.433.1 Detailed Description	2558
16.433.2 Enumeration Type Documentation	2558
16.433.2.1 anonymous enum	2558
16.434 consts.h	2558
16.435 l4/sys/consts.h File Reference	2559
16.435.1 Detailed Description	2560
16.436 consts.h	2561
16.437 x86/l4/sys/consts.h File Reference	2563
16.437.1 Detailed Description	2563
16.438 consts.h	2563
16.439 capability.h	2564
16.440 l4/re/consts File Reference	2566
16.440.1 Detailed Description	2567
16.441 consts	2567
16.442 consts	2567
16.443 ipc_array	2568
16.444 ipc_basics	2572
16.445 l4/sys/cxx/ipc_client File Reference	2575
16.445.1 Macro Definition Documentation	2577
16.445.1.1 L4_RPC_DEF	2577
16.446 ipc_client	2578
16.447 ipc_epiface	2578
16.448 l4/sys/cxx/ipc_iface File Reference	2582
16.448.1 Detailed Description	2584
16.448.2 Macro Definition Documentation	2584
16.448.2.1 L4_INLINE_RPC	2584
16.448.2.2 L4_INLINE_RPC_NF	2585
16.448.2.3 L4_INLINE_RPC_NF_OP	2585
16.448.2.4 L4_INLINE_RPC_OP	2586
16.448.2.5 L4_RPC	2586
16.448.2.6 L4_RPC_NF	2587
16.448.2.7 L4_RPC_NF_OP	2587
16.448.2.8 L4_RPC_OP	2588

16.449 ipc_iface	2588
16.450 ipc_legacy	2593
16.451 ipc_ret_array	2594
16.452 l4/cxx/ipc_server File Reference	2595
16.452.1 Detailed Description	2596
16.453 ipc_server	2597
16.454 ipc_server	2598
16.455 ipc_server_loop	2601
16.456 ipc_string	2604
16.457 l4/sys/cxx/ipc_types File Reference	2606
16.458 ipc_types	2608
16.459 ipc_varg	2614
16.460 l4/sys/cxx/smart_capability_1x File Reference	2620
16.461 smart_capability_1x	2621
16.462 l4/sys/cxx/types File Reference	2623
16.462.1 Macro Definition Documentation	2624
16.462.1.1 L4_TYPES_FLAGS_OPS_DEF	2624
16.463 types	2624
16.464 l4/sys/debugger File Reference	2627
16.464.1 Detailed Description	2628
16.465 debugger	2628
16.466 l4/sys/debugger.h File Reference	2629
16.466.1 Detailed Description	2630
16.467 debugger.h	2630
16.468 l4/sys/err.h File Reference	2633
16.468.1 Detailed Description	2634
16.469 err.h	2635
16.470 l4/sys/exception File Reference	2635
16.470.1 Detailed Description	2636
16.471 exception	2637
16.472 l4/sys/factory File Reference	2637
16.472.1 Detailed Description	2639
16.473 factory	2639
16.474 l4/sys/factory.h File Reference	2641
16.474.1 Detailed Description	2642
16.475 factory.h	2642
16.476 l4/sys/icu File Reference	2647
16.476.1 Detailed Description	2648
16.477 icu	2648
16.478 l4/sys/icu.h File Reference	2648
16.478.1 Detailed Description	2651
16.479 icu.h	2651

16.480 iommu	2654
16.481 ipc.h	2655
16.482 arm/l4/l4/sys/ipc.h File Reference	2655
16.482.1 Detailed Description	2656
16.483 ipc.h	2656
16.484 l4/sys/ipc.h File Reference	2657
16.484.1 Detailed Description	2659
16.484.2 Function Documentation	2659
16.484.2.1 l4_ipc_to_errno()	2659
16.485 ipc.h	2659
16.486 x86/l4/l4/sys/ipc.h File Reference	2663
16.486.1 Detailed Description	2663
16.487 ipc.h	2664
16.488 l4/sys/ipc_gate File Reference	2664
16.488.1 Detailed Description	2666
16.489 ipc_gate	2666
16.490 l4/sys/ipc_gate.h File Reference	2666
16.490.1 Detailed Description	2668
16.491 ipc_gate.h	2669
16.492 l4/sys/irq File Reference	2670
16.492.1 Detailed Description	2671
16.493 irq	2671
16.494 l4/sys/kdebug.h File Reference	2673
16.494.1 Detailed Description	2675
16.494.2 Enumeration Type Documentation	2675
16.494.2.1 l4_kdebug_ops_t	2675
16.494.3 Function Documentation	2676
16.494.3.1 __kdebug_3_text()	2676
16.494.3.2 __kdebug_op()	2677
16.494.3.3 __kdebug_op_1()	2678
16.494.3.4 __kdebug_text()	2680
16.494.3.5 enter_kdebug()	2681
16.494.3.6 outchar()	2682
16.494.3.7 outdec()	2682
16.494.3.8 outhex12()	2683
16.494.3.9 outhex16()	2683
16.494.3.10 outhex20()	2684
16.494.3.11 outhex32()	2685
16.494.3.12 outhex64()	2685
16.494.3.13 outhex8()	2686
16.494.3.14 outnstring()	2686
16.494.3.15 outstring()	2687

16.494.3.16 outumword()	2688
16.495 kdebug.h	2689
16.496 l4/sys/kernel_object.h File Reference	2691
16.496.1 Detailed Description	2692
16.497 kernel_object.h	2692
16.498 l4/sys/kip File Reference	2693
16.498.1 Detailed Description	2693
16.499 kip	2694
16.500 kobject	2695
16.501 l4/sys/ktrace.h File Reference	2696
16.501.1 Detailed Description	2697
16.502 ktrace.h	2698
16.503 amd64/l4/sys/l4int.h File Reference	2698
16.503.1 Detailed Description	2699
16.504 l4int.h	2699
16.505 arm/l4/sys/l4int.h File Reference	2699
16.505.1 Detailed Description	2699
16.506 l4int.h	2700
16.507 l4/sys/l4int.h File Reference	2700
16.507.1 Detailed Description	2701
16.508 l4int.h	2701
16.509 x86/l4/sys/l4int.h File Reference	2701
16.509.1 Detailed Description	2702
16.510 l4int.h	2702
16.511 l4/sys/memdesc.h File Reference	2702
16.511.1 Detailed Description	2704
16.512 memdesc.h	2704
16.513 meta	2706
16.514 l4/sys/meta File Reference	2706
16.514.1 Detailed Description	2708
16.515 meta	2708
16.516 l4/sys/pager File Reference	2708
16.516.1 Detailed Description	2710
16.517 pager	2710
16.518 l4/sys/platform_control File Reference	2710
16.518.1 Detailed Description	2711
16.519 platform_control	2712
16.520 l4/sys/platform_control.h File Reference	2712
16.520.1 Detailed Description	2714
16.521 platform_control.h	2714
16.522 l4/sys/rcv_endpoint File Reference	2716
16.522.1 Detailed Description	2718

16.523 rcv_endpoint	2718
16.524 l4/sys/rcv_endpoint.h File Reference	2718
16.524.1 Detailed Description	2720
16.524.2 Enumeration Type Documentation	2720
16.524.2.1 L4_rcv_ep_ops	2720
16.525 rcv_endpoint.h	2720
16.526 l4/sys/scheduler File Reference	2721
16.526.1 Detailed Description	2722
16.527 scheduler	2722
16.528 l4/sys/scheduler.h File Reference	2723
16.528.1 Detailed Description	2724
16.529 scheduler.h	2725
16.530 l4/sys/semaphore File Reference	2727
16.530.1 Detailed Description	2729
16.531 semaphore	2729
16.532 l4/sys/semaphore.h File Reference	2729
16.532.1 Detailed Description	2731
16.533 semaphore.h	2731
16.534 l4/sys/smart_capability File Reference	2732
16.534.1 Detailed Description	2733
16.535 smart_capability	2733
16.536 l4/sys/task File Reference	2735
16.536.1 Detailed Description	2737
16.537 task	2737
16.538 task.h	2738
16.539 l4/sys/task.h File Reference	2738
16.539.1 Detailed Description	2739
16.540 task.h	2739
16.541 l4/cxx/thread File Reference	2742
16.541.1 Detailed Description	2743
16.542 thread	2744
16.543 l4/sys/thread File Reference	2744
16.543.1 Detailed Description	2746
16.544 thread	2746
16.545 l4/sys/typeinfo_svr File Reference	2748
16.545.1 Detailed Description	2749
16.546 typeinfo_svr	2750
16.547 amd64/l4/sys/utcb.h File Reference	2750
16.547.1 Detailed Description	2751
16.548 utcb.h	2752
16.549 arm/l4/sys/utcb.h File Reference	2753
16.549.1 Detailed Description	2754

16.550 utcb.h	2754
16.551 l4/sys/utcb.h File Reference	2755
16.551.1 Detailed Description	2757
16.552 utcb.h	2757
16.553 x86/l4/sys/utcb.h File Reference	2759
16.553.1 Detailed Description	2761
16.554 utcb.h	2761
16.555 l4/sys/vcon File Reference	2762
16.555.1 Detailed Description	2764
16.556 vcon	2764
16.557 l4/sys/vcon.h File Reference	2764
16.557.1 Detailed Description	2767
16.557.2 Enumeration Type Documentation	2767
16.557.2.1 L4_vcon_read_flags	2767
16.558 vcon.h	2767
16.559 l4/sys/vhw.h File Reference	2770
16.559.1 Detailed Description	2771
16.560 vhw.h	2772
16.561 vm	2773
16.562 l4/sys/vm File Reference	2773
16.562.1 Detailed Description	2774
16.563 vm	2775
16.564 l4/sys/assert.h File Reference	2775
16.564.1 Detailed Description	2777
16.564.2 Macro Definition Documentation	2777
16.564.2.1 l4_assert	2777
16.565 assert.h	2777
16.566 l4/util/assert.h File Reference	2778
16.566.1 Detailed Description	2779
16.567 assert.h	2779
16.568 arm/l4/sys/atomic.h File Reference	2781
16.568.1 Detailed Description	2781
16.569 atomic.h	2781
16.570 l4/cxx/atomic.h File Reference	2782
16.570.1 Detailed Description	2782
16.571 atomic.h	2783
16.572 l4/util/atomic.h File Reference	2783
16.572.1 Detailed Description	2786
16.573 atomic.h	2786
16.574 l4/util/backtrace.h File Reference	2790
16.574.1 Detailed Description	2791
16.574.2 Function Documentation	2791

16.574.2.1 l4util_backtrace()	2791
16.575 backtrace.h	2792
16.576 l4/util/base64.h File Reference	2792
16.576.1 Detailed Description	2793
16.577 base64.h	2793
16.578 l4/util/bitops.h File Reference	2793
16.578.1 Detailed Description	2795
16.579 bitops.h	2796
16.580 l4/util/elf.h File Reference	2799
16.580.1 Detailed Description	2805
16.581 elf.h	2805
16.582 l4/util/getopt.h File Reference	2815
16.582.1 Detailed Description	2815
16.583 getopt.h	2816
16.584 l4/util/keymap.h File Reference	2817
16.584.1 Detailed Description	2817
16.585 keymap.h	2818
16.586 l4/sys/kip.h File Reference	2818
16.586.1 Detailed Description	2819
16.586.2 Macro Definition Documentation	2820
16.586.2.1 l4_kip_for_each_feature	2820
16.586.3 Function Documentation	2820
16.586.3.1 l4_kip_kernel_has_feature()	2820
16.587 kip.h	2821
16.588 l4/util/kip.h File Reference	2822
16.589 kip.h	2823
16.590 l4/util/kprintf.h File Reference	2824
16.590.1 Detailed Description	2824
16.591 kprintf.h	2825
16.592 l4/util/l4mod.h File Reference	2825
16.592.1 Detailed Description	2826
16.592.2 Enumeration Type Documentation	2826
16.592.2.1 l4util_l4mod_mod_info_flag	2826
16.593 l4mod.h	2826
16.594 l4/util/list_alloc.h File Reference	2827
16.594.1 Detailed Description	2827
16.594.2 Function Documentation	2828
16.594.2.1 l4la_alloc()	2828
16.594.2.2 l4la_avail()	2828
16.594.2.3 l4la_dump()	2828
16.594.2.4 l4la_free()	2828
16.594.2.5 l4la_init()	2829

16.595 list_alloc.h	2829
16.596 llulc.h	2829
16.597 l4/util/lock.h File Reference	2830
16.597.1 Detailed Description	2831
16.598 lock.h	2831
16.599 l4/util/mb_info.h File Reference	2832
16.599.1 Detailed Description	2835
16.599.2 Macro Definition Documentation	2835
16.599.2.1 l4util_mb_for_each_mmap_entry	2835
16.599.2.2 L4UTIL_MB_MEMORY	2835
16.599.2.3 MB_ARD_MEMORY	2835
16.599.2.4 MB_ART_MEMORY	2836
16.600 mb_info.h	2836
16.601 l4/util/parse_cmd.h File Reference	2840
16.601.1 Detailed Description	2841
16.602 parse_cmd.h	2841
16.603 l4/util/rand.h File Reference	2841
16.603.1 Detailed Description	2842
16.604 rand.h	2843
16.605 l4/util/splitlog2.h File Reference	2843
16.605.1 Detailed Description	2844
16.606 splitlog2.h	2844
16.607 arm/l4/sys/thread.h File Reference	2845
16.607.1 Detailed Description	2845
16.608 thread.h	2845
16.609 l4/sys/thread.h File Reference	2846
16.609.1 Detailed Description	2848
16.610 thread.h	2848
16.611 l4/util/thread.h File Reference	2855
16.611.1 Detailed Description	2856
16.611.2 Macro Definition Documentation	2856
16.611.2.1 __L4UTIL_THREAD_FUNC	2856
16.612 thread.h	2856
16.613 vbus	2857
16.614 l4/vbus/vbus.h File Reference	2858
16.614.1 Detailed Description	2860
16.614.2 Enumeration Type Documentation	2860
16.614.2.1 anonymous enum	2860
16.614.2.2 l4vbus_icu_src_types	2861
16.615 vbus.h	2861
16.616 vbus_generic	2862
16.617 vbus_gpio	2862

16.618 vbus_gpio-ops.h	2864
16.619 vbus_gpio.h	2864
16.620 vbus_i2c.h	2865
16.621 vbus_inhibitor.h	2865
16.622 I4/vbus/vbus_interfaces.h File Reference	2865
16.622.1 Detailed Description	2867
16.622.2 Typedef Documentation	2867
16.622.2.1 I4vbus_iface_type_t	2867
16.622.3 Enumeration Type Documentation	2867
16.622.3.1 anonymous enum	2867
16.622.3.2 I4vbus_iface_type_t	2867
16.622.4 Function Documentation	2868
16.622.4.1 I4vbus_subinterface_supported()	2868
16.623 vbus_interfaces.h	2868
16.624 vbus_mcspi.h	2869
16.625 vbus_pci	2869
16.626 vbus_pci-ops.h	2870
16.627 vbus_pci.h	2871
16.628 vbus_pm-ops.h	2871
16.629 vbus_pm.h	2871
16.630 I4/vbus/vbus_types.h File Reference	2872
16.630.1 Detailed Description	2873
16.630.2 Enumeration Type Documentation	2873
16.630.2.1 I4vbus_device_flags_t	2873
16.630.2.2 I4vbus_resource_flags_t	2874
16.630.2.3 I4vbus_resource_type_t	2874
16.631 vbus_types.h	2874
16.632 vdevice-ops.h	2875
16.633 I4/vcpu/vcpu File Reference	2876
16.633.1 Detailed Description	2876
16.634 vcpu	2876
16.635 I4/sys/vcpu.h File Reference	2878
16.635.1 Detailed Description	2879
16.635.2 Function Documentation	2880
16.635.2.1 I4_vcpu_check_version()	2880
16.636 vcpu.h	2880
16.637 I4/vcpu/vcpu.h File Reference	2881
16.637.1 Detailed Description	2882
16.638 vcpu.h	2883
16.639 ipc-invoke.h	2885
16.640 ipc-l42-gcc3-nopic.h	2885

17 Examples	2887
17.1 hello/server/src/main.c	2887
17.2 examples/sys/ipc/ipc_example.c	2887
17.3 examples/sys/ipc/ipc.cfg	2889
17.4 examples/sys/start-with-exc/main.c	2889
17.5 examples/sys/singlestep/main.c	2891
17.6 examples/sys/aliens/main.c	2893
17.7 examples/sys/utcb-ipc/main.c	2896
17.8 examples/sys/ux-vhw/main.c	2898
17.9 examples/sys/isr/main.c	2899
17.10 examples/clntsrv/server.cc	2901
17.11 examples/clntsrv/client.cc	2902
17.12 examples/clntsrv/clntsrv.cfg	2903
17.13 examples/libs/l4re/c/ma+rm.c	2903
17.14 examples/libs/l4re/c++/mem_alloc/ma+rm.cc	2904
17.15 examples/libs/l4re/c++/shared_ds/ds_clnt.cc	2905
17.16 examples/libs/l4re/c++/shared_ds/ds_srv.cc	2907
17.17 examples/libs/l4re/c++/shared_ds/shared_ds.cfg	2909
17.18 examples/libs/l4re/streammap/server.cc	2909
17.19 examples/libs/l4re/streammap/client.cc	2910
17.20 examples/libs/l4re/streammap/streammap.cfg	2911
17.21 examples/libs/libirq/loop.c	2912
17.22 examples/libs/libirq/async_isr.c	2912
17.23 examples/sys/migrate/thread_migrate.cc	2913
17.24 examples/sys/migrate/thread_migrate.cfg	2915
17.25 tmpfs/lib/src/fs.cc	2915
17.26 examples/libs/shmc/prodcons.c	2922
Index	2925

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.

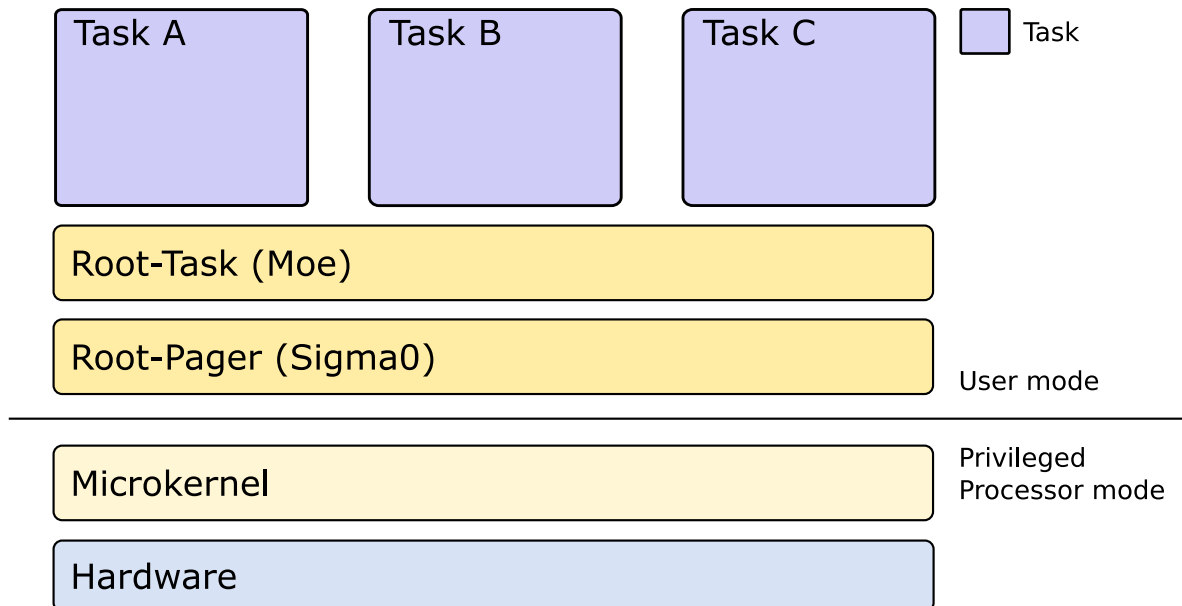


Figure 2.1 Basic Structure of an L4Re based system

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_svr::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_svr::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.

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.

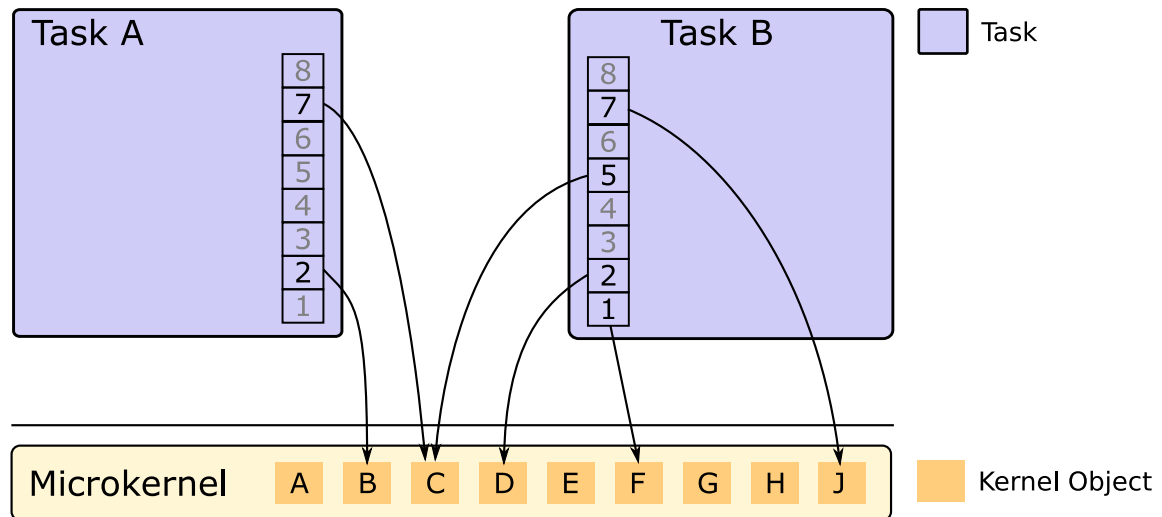


Figure 4.1 Capabilities and Local Naming in L4

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. 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
<code>T</code>	Input	<code>T</code>	<code>T</code>
<code>T const &</code>	Input	<code>T const &</code>	<code>T const &</code>
<code>T const *</code>	Input	<code>T const *</code>	<code>T const &</code>
<code>T &</code>	Output	<code>T &</code>	<code>T &</code>
<code>T *</code>	Output	<code>T *</code>	<code>T &</code>
<code>L4::Ipc::In_out<T &></code>	In/Out	<code>T &</code>	<code>T &</code>
<code>L4::Ipc::In_out<T *></code>	In/Out	<code>T *</code>	<code>T &</code>
<code>L4::Ipc::Cap<T></code>	Input	<code>L4::Ipc::Cap<T></code>	<code>L4::Ipc::Snd_fpage</code>
<code>L4::Ipc::Out<L4::Cap<T> ></code>	Output	<code>L4::Cap<T></code>	<code>L4::Ipc::Cap<T> &</code>
<code>L4::Ipc::Rcv_fpage</code>	Input	<code>L4::Ipc::Rcv_fpage</code>	<code>void</code>
<code>L4::Ipc::Small_buf</code>	Input	<code>L4::Ipc::Small_buf</code>	<code>void</code>

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<const T></code>	Input	<code>L4::Ipc::Array<const T></code>	<code>L4::Ipc::Array_ref<const T></code>
<code>L4::Ipc::Array<const T></code>	Input	<code>L4::Ipc::Array<const T></code>	<code>L4::Ipc::Array_in_buf<const T> &</code>
<code>L4::Ipc::Array<T> &</code>	Output	<code>L4::Ipc::Array<T> &</code>	<code>L4::Ipc::Array_ref<T> &</code>
<code>L4::Ipc::Array_ref<T> &</code>	Output	<code>L4::Ipc::Array_ref<T> &</code>	<code>L4::Ipc::Array_ref<T> &</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<T></code>	Input	<code>L4::Ipc::Opt<T></code>	<code>T</code>
<code>L4::Ipc::Opt<const T*></code>	Input	<code>L4::Ipc::Opt<const T*></code>	<code>T</code>
<code>L4::Ipc::Opt<T &></code>	Output	<code>T &</code>	<code>L4::Ipc::Opt<T> &</code>
<code>L4::Ipc::Opt<T *></code>	Output	<code>T *</code>	<code>L4::Ipc::Opt<T> &</code>
<code>L4::Ipc::Opt<Array↔_ref<T> &></code>	Output	<code>Array_ref<T> &</code>	<code>L4::Ipc::Opt<Array↔_ref<T>> &</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.

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=make test`. 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::Irq](#)
- [L4::Semaphore](#)
- [L4::Task](#)
- [L4::Thread](#)
- [L4::Vm](#)

The protocol IDs for objects in this list are given in [L4_msgtag_protocol](#). 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.
- [l4vio_net_p2p, a virtual network point-to-point link](#)
- [Uvmm, the virtual machine monitor](#)
- [NVMe server](#)
- [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.
- [Sigma0, the Root-Pager](#)
- [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.
- [AHCI driver](#)

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
 - 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`.

```
-- <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 `L4Re::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:

```
[--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.

- Execute Statement Option: **execute**, **e**
Execute the Lua statement `STATEMENT`.

5.4 lo, the lo Server

The lo 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 lo 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.

lo'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.

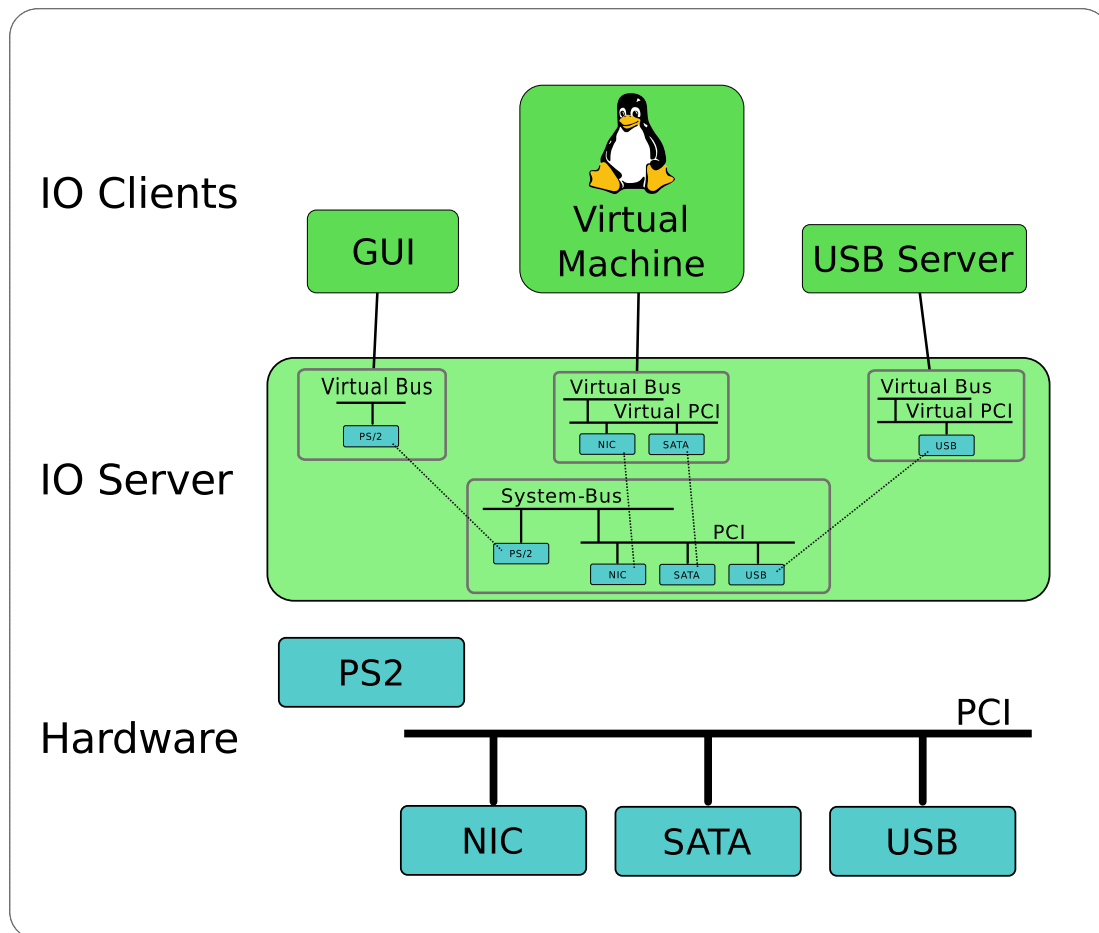


Figure 5.1 IO Service Architecture Overview

The Io 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 Io 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 Io 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);
end);
```

```

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_ccsspp

- 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
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
-	-	-

Common Name	Description	PCI ID string
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_gpiib	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
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 verboseness 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 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 dataspace 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")
```

5.6 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.
- `-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.

5.6.1 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).

5.7 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-driv`

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-driv");
```

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.
- `--nomsi`
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 dataspace other than the registered ones. Can be used multiple times for multiple dataspace.
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 dataspace 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_cl1` capability to communicate with the NVMe server.

```
local nvme_cl1 = 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(),
    cl1 = nvme_cl1:svr(),
  },
},
"rom/nvme-drv --client cl1 --device 88E59675-4DC8-469A-98E4-B7B021DC7FBE --ds-max 5");
```

5.8 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**
`df1-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");
```

5.9 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.
- `-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

5.10 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-driv --client cl1 --device 88E59675-4DC8-469A-98E4-B7B021DC7FBE --ds-max 5");
```


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.

Chapter 7

Deprecated List

Global [L4::lrq::unmask](#) ([l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept

Use [L4::lrq_eoi::unmask\(\)](#)

Struct [L4::lrq_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_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](#) (unsigned int mus) [L4_NOTHROW](#)

Use [l4_timeout_from_us\(\)](#).

Use [l4_timeout_from_us\(\)](#).

Use [l4_timeout_from_us\(\)](#).

Chapter 8

Topic Index

8.1 Topics

Here is a list of all topics with brief descriptions:

Base API	129
Basic Macros	132
C++ IPC Interface Definition.	136
Internal Helpers	136
Cache Consistency	137
Capabilities	140
Error codes	144
Fiasco extensions	146
Fiasco real time scheduling extensions	149
Kernel Debugger	150
Kernel Tracing	156
Flex pages	159
Integer Types	176
Kernel Interface Page	177
Fiasco-UX Virtual devices	184
Memory descriptors (C version)	185
Kernel Objects	190
DMA space	191
Factory	192
IPC-Gate API	200
IRQs	203
Interrupt controller	217
Kernel-provided semaphore	233
L4 kernel object type information	235
Platform Control C API	236
Scheduler	241
Task	249
Thread	259
Thread control	279
vCPU API	285
Virtual Console	289
Virtual Machines	303
VM API for SVM	304
VM API for TZ	305

VM API for VMX	305
Memory operations.	319
Memory related	321
Object Invocation	328
Error Handling	347
Message Items	352
Message Tag	356
Realtime API	370
Timeouts	370
Virtual Registers (UTCBS)	378
ARM Virtual Registers (UTCB)	382
Buffer Registers (BRs)	383
Message Registers (MRs)	384
Exception registers	384
Thread Control Registers (TCRs)	388
amd64 Virtual Registers (UTCB)	389
x86 Virtual Registers (UTCB)	389
EDID parsing functionality	390
IO interface	394
IRQ handling library	401
Interface for asynchronous ISR handlers.	401
Interface for asynchronous ISR handlers with a given IRQ capability.	403
Interface using direct functionality.	404
Interface using direct functionality.	408
L4 IPC Opcodes	410
L4 VIRTIO Interface	414
L4 VIRTIO Block Device	415
L4 VIRTIO Input Device	416
L4 VIRTIO Network Device	417
L4 VIRTIO Transport Layer	418
L4 Vbus functions	426
L4vbus GPIO functions	437
L4vbus PCI functions	446
L4vbus power management functions	452
L4Re C Interface	454
Capability allocator	456
DMA Space Interface	457
Dataspace interface	461
Debug interface	465
Event interface	466
Initial Environment	469
Kumem allocator utility	474
L4Re Util C Interface	474
Log interface	474
Memory allocator	477
Namespace interface	481
Region map interface	485
Video API	499
L4Re C++ Interface	506
Auxiliary data	508
C++ Exceptions	509
Console API	510
Debugging API	510
Event API	511
L4Re ELF Auxiliary Information	512
L4Re Protocol identifiers	514

L4Re Util C++ Interface	515
Kumem utilities	516
L4Re Capability API	517
Logging interface	520
Name-space API	520
Parent API	521
Region map API	522
Vbus API	523
L4SHM-based ring buffer implementation	524
Internal	524
Receiver	525
Sender	525
Server-Side IPC framework	525
Shared Memory Library	527
Chunks	532
Consumer	536
Producer	540
Signals	544
Consumer	547
Producer	551
Sigma0 API	552
Internal constants	557
Small C++ Template Library	558
Utility Functions	562
Atomic Instructions	569
Bit Manipulation	586
Bitmap graphics and fonts	593
Functions for rendering bitmap data in frame buffers	594
Functions for rendering bitmap fonts to frame buffers	594
CPU related functions	594
Comfortable Command Line Parsing	597
ELF binary format	599
Functions to manipulate the local IDT	623
IA32 Port I/O API	624
Internal functions	630
Kernel Interface Page API	631
Low-Level Thread Functions	633
Random number support	633
Timestamp Counter	634
vCPU Support Library	641
Extended vCPU support	649

Chapter 9

Namespace Index

9.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

cxx	Our C++ library	651
cxx::Bits	Internal helpers for the cxx package	655
L4	L4 low-level kernel interface	655
L4::lpc	IPC related functionality	666
L4::lpc::Msg	IPC Message related functionality	674
L4::lpc_svr	Helper classes for L4::Server instantiation	680
L4::Typeid	Definition of interface data-type helpers	681
L4::Types	L4 basic type helpers for C++	682
L4Re	L4Re C++ Interfaces	682
L4Re::Util	Documentation of the L4 Runtime Environment utility functionality in C++	697
L4Re::Vfs	Virtual file system for interfaces in POSIX libc	706
L4vbus	C++ interface of the Vbus API.	707
L4virtio	L4-VIRTIO Transport C++ API	708

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 >	1009
L4::lpc::Array_ref< ELEM_TYPE, Array_len_default >	1009
cxx::Bits::Base_avl_set< ITEM_TYPE, Lt_functor< ITEM_TYPE >, New_allocator, Bits::Avl_set_get_↔ key< ITEM_TYPE > >	791
cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, Lt_functor< KEY_TYPE >, New_allocator, Bits::Avl_map_get_key< KEY_TYPE > >	791
cxx::Bits::Base_avl_set< Pair< Region, Hdlr >, cxx::Lt_functor< Region >, Alloc, Bits::Avl_map_get_↔ key< Region > >	791
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >	748
cxx::Base_slab< sizeof(Type), L4_PAGESIZE, 2, New_allocator >	748
cxx::Base_slab_static< sizeof(Type), L4_PAGESIZE, 2, New_allocator >	754
cxx::Bits::Basic_list< Bits::Basic_list_policy< cxx::Base_slab::Slab_i, H_list_item > >	807
cxx::Bits::Basic_list< Bits::Basic_list_policy< Observer, H_list_item > >	807
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item > >	807
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item_t< T > > >	807
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, S_list_item > >	807
cxx::Bits::Basic_list< Bits::Basic_list_policy< Timeout, H_list_item > >	807
cxx::Bits::Basic_list< Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_base > > >	807
Block_device::Device_discard_feature	709
Block_device::Device_mgr< DEV, FACTORY >	710
Block_device::Dma_region_info	711
Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, bool >	718
Block_device::Partitioned_device< BASE_DEV >	724
Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, true >	719
Block_device::Inout_block	720
Block_device::Inout_memory< DEV >	721
Block_device::Mem_region_info	722
Block_device::Partition_info	722
Block_device::Partition_reader< DEV >	723
Block_device::Pending_request	725
Block_device::Pending_request::Owner	727
Block_device::Simple_request_queue	728
L4::Types::Bool< __lface_conflict< I, I2 >::value _lface_conflict< I, LIST::type >::value >	1313
L4::Types::Bool< false >	1313
L4::Types::False	1314

L4::Types::Same< A, B >	1326
L4::Types::Bool< I1::Proto !=PROTO_EMPTY &&I1::Proto==I2::Proto >	1313
L4::Types::Bool< lface_conflict< I, L2::type >::value _Conflict< L1::type, L2::type >::value >	1313
L4::Types::Bool< true >	1313
L4::Types::True	1328
L4::lpc::Msg::ls_valid_rpc_type< A * >	1058
L4::lpc::Msg::ls_valid_rpc_type< T >	1058
L4::Types::Bool< Typeid::Conflict< L1::type, L2::type >::value Conflict< L1, LIST... >::value Conflict< L2, LIST... >::value >	1313
L4::Types::Bool< Typeid::lface_conflict< I::type, L::type >::value lface_conflict< I, LIST... >::value >	1313
cxx::Bits::Bst< _Node, Bits::Avl_map_get_key< KEY_TYPE >, Lt_functor< KEY_TYPE > >	809
cxx::Bits::Bst< _Node, Bits::Avl_map_get_key< Region >, cxx::Lt_functor< Region > >	809
cxx::Bits::Bst< _Node, Bits::Avl_set_get_key< ITEM_TYPE >, Lt_functor< ITEM_TYPE > >	809
cxx::Bits::Bst< _Node, GET_KEY, COMPARE >	809
cxx::Bits::Bst< Entry, Names_get_key, Lt_functor< typename Get_key::Key_type > >	809
cxx::Bits::Bst< Node, Get_key, Lt_functor< typename Get_key::Key_type > >	809
L4::lpc::Msg::Cnt_val_ops< Detail::_Plain< T >::type, DIR, CLASS >	1043
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 >	1043
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 >	1043
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 >	1043
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 >	1043
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 >	1043
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 >	1043
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >	748
cxx::Slab< Type, Slab_size, Max_free, Alloc >	873
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >	754
cxx::Slab_static< Type, Slab_size, Max_free, Alloc >	877
cxx::Bitfield< T, LSB, MSB >	759
cxx::Bitfield< T, LSB, MSB >::Value_base< TT >	773
cxx::Bitfield< T, LSB, MSB >::Value< TT >	771
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >	774
cxx::Bitmap_base	779
cxx::Bitmap< BITS >	775
cxx::Bitmap_base::Bit	787
cxx::Bitmap_base::Char< BITS >	788
cxx::Bitmap_base::Word< BITS >	788
cxx::Bits::Avl_map_get_key< KEY_TYPE >	790
cxx::Bits::Avl_set_get_key< KEY_TYPE >	790
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >	791
cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc >	729
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >	729
cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >	735
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node	804
cxx::Bits::Basic_list< POLICY >	807
cxx::H_list< T, Bits::Basic_list_policy< T, H_list_item_t< T > > >	832
cxx::H_list_t< T >	843
cxx::H_list< Observer >	832

cxx::H_list< cxx::Base_slab::Slab_i >	832
cxx::H_list< Weak_ref_base, Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_base > > >	832
cxx::H_list< Timeout >	832
cxx::S_list< T, Bits::Basic_list_policy< T, S_list_item > >	870
cxx::H_list< T, POLICY >	832
cxx::H_list_t< Weak_ref_base >	843
cxx::S_list< T, POLICY >	870
cxx::Bits::Bst< Node, Get_key, Compare >	809
cxx::Avl_tree< _Node, Bits::Avl_map_get_key< KEY_TYPE >, Lt_functor< KEY_TYPE > >	739
cxx::Avl_tree< _Node, Bits::Avl_set_get_key< ITEM_TYPE >, Lt_functor< ITEM_TYPE > >	739
cxx::Avl_tree< Entry, Names_get_key >	739
cxx::Avl_tree< _Node, Bits::Avl_map_get_key< Region >, cxx::Lt_functor< Region > >	739
cxx::Avl_tree< _Node, GET_KEY, COMPARE >	739
cxx::Avl_tree< Node, Get_key, Compare >	739
cxx::Bits::Bst_node	823
cxx::Avl_tree_node	746
cxx::Bits::Direction	825
cxx::Bits::Smart_ptr_list< ITEM >	827
cxx::Bits::Smart_ptr_list_item< T, STORE_T >	830
cxx::Ref_obj_list_item< Connection >	864
cxx::Ref_obj_list_item< T >	864
cxx::H_list_item_t< ELEM_TYPE >	840
L4::lpc_svr::Timeout	1116
Block_device::Errand::Errand	711
Block_device::Errand::Poll_errand	715
cxx::List< D, Alloc >	846
cxx::List< D, Alloc >::Iter	848
cxx::List_alloc	849
cxx::List_item	852
cxx::List_item::Iter	856
cxx::List_item::T_iter< E >	857
cxx::List_item::T_iter< T, Poly >	857
cxx::Lt_functor< Obj >	859
cxx::New_allocator< _Type >	859
cxx::Nothrow	860
cxx::Pair< First, Second >	861
cxx::Pair_first_compare< Cmp, Typ >	863
cxx::Ref_ptr< T, CNT >	866
cxx::static_vector< T, IDX >	881
cxx::String	882
L4Re::Util::Names::Name	1562
L4virtio::Svr::Device_t< Ds_data >	1840
L4virtio::Svr::Block_dev_base< Ds_data >	1811
L4virtio::Svr::Device_t< Mem_region_info >	1840
L4virtio::Svr::Driver_mem_list_t< Ds_data >	1848
L4virtio::Svr::Driver_mem_list_t< Mem_region_info >	1848
L4virtio::Svr::Driver_mem_region_t< Ds_data >	1856
Elf32_Auxv	894
Elf32_Dyn	895
Elf32_Ehdr	896
Elf32_Phdr	898
Elf32_Rel	900
Elf32_Rela	900
Elf32_Shdr	901
Elf32_Sym	903

Elf64_Auxv	904
Elf64_Dyn	905
Elf64_Ehdr	906
Elf64_Phdr	908
Elf64_Rel	910
Elf64_Rela	910
Elf64_Shdr	911
Elf64_Sym	913
L4::Epiface_t0< IFACE, BASE >	966
L4::Epiface_t0< void, BASE >	966
L4Re::Event_buffer_t< PAYLOAD >	1455
L4Re::Util::Event_buffer_t< PAYLOAD >	1552
L4Re::Util::Event_buffer_consumer_t< PAYLOAD >	1548
L4::Types::Flags_ops_t< Flags >	1320
L4::Types::Flags_ops_t< Flags_t< DT, T > >	1320
L4::Types::Flags_t< DT, T >	1322
gfxbitmap_offset	914
cxx::H_list_item_t< Weak_ref_base >	840
cxx::Weak_ref_base	891
cxx::Weak_ref< T >	888
Block_device::Inout_memory< Device_type >	721
L4::Kobject_2t< Console, Video::Goos, Event, L4::PROTO_EMPTY >	1166
L4Re::Console	1410
L4::Kobject_2t< Debug_obj_t< BASE >, BASE, Debug_obj, L4::PROTO_EMPTY >	1166
L4::Kobject_x< Iommu, Proto_t< L4_PROTO_IOMMU >, Type_info::Demand_t< 1 > >	1179
L4::Iommu	1002
L4::Alloc_list	915
L4::Basic_registry	919
L4Re::Util::Object_registry	1569
L4::Cap_base	930
L4::Cap< void >	924
L4::Cap< L4::Rcv_endpoint >	924
L4::Cap< L4Re::Rm >	924
L4::Cap< L4Re::Namespace >	924
L4::Cap< L4Re::Dataspace >	924
L4::Cap< L4::Vcon >	924
L4::Cap< L4::Semaphore >	924
L4::Cap< L4::Irq >	924
L4::Cap< L4::Thread >	924
L4::Cap< L4::Factory >	924
L4::Cap< L4Re::Video::Goos >	924
L4::Cap< L4vbus::Vbus >	924
L4::Cap< L4virtio::Device >	924
L4::Cap< T >	924
L4::Smart_cap< T, SMART >	1246
L4::Epiface	958
L4::Epiface_t0< L4virtio::Device, L4::Epiface >	966
L4::Epiface_t0< IFACE, L4::Epiface >	966
L4::Epiface_t0< void, Epiface >	966
L4::Epiface_t0< L4::Kobject, L4::Epiface >	966
L4::Epiface_t0< RPC_IFACE, BASE >	966
L4::Epiface_t< Virtio_client< DEV >, L4virtio::Device >	963
L4::Epiface_t< Null_handler, L4::Kobject >	963
L4::Epiface_t< Block_dev< Ds_data >, L4virtio::Device >	963
L4::Irqep_t< Irq_object >	1148
L4::Epiface_t< Derived, IFACE, BASE, bool >	963

L4::Irqep_t< Derived, BASE, bool >	1148
L4::Server_object	1233
L4::Server_object_t< Kobject >	1237
L4::Irq_handler_object	1140
L4::Server_object_t< IFACE, L4::Server_object >	1237
L4::Server_object_t< IFACE, BASE >	1237
L4::Server_object_x< Derived, IFACE, BASE >	1243
L4::Server_object_x< Derived, IFACE, BASE >	1243
L4::Exception_tracer	970
L4::Base_exception	917
L4::Invalid_capability	997
L4::Runtime_error	1211
L4::Bounds_error	922
L4::Com_error	940
L4::Element_already_exists	952
L4::Element_not_found	955
L4::Out_of_memory	1185
L4::Unknown_error	1335
L4::Factory::Lstr	980
L4::Factory::Nil	982
L4::Factory::S	982
L4::IOModifier	1004
L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >	1008
L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >	1009
L4::lpc::Array< char const, unsigned long >	1005
L4::lpc::Array< ELEM_TYPE, LEN_TYPE >	1005
L4::lpc::As_value< T >	1010
L4::lpc::Buf_item	1011
L4::lpc::Call	1012
L4::lpc::Call_t< RIGHTS >	1013
L4::lpc::Call_zero_send_timeout	1014
L4::lpc::Cap< T >	1016
L4::lpc::Gen_fpage< T >	1018
L4::lpc::In_out< T >	1021
L4::lpc::Istream	1032
L4::lpc::Iostream	1022
L4::lpc::Msg::CInt_val_ops< MTYPE, DIR, CLASS >	1043
L4::lpc::Msg::Cls_buffer	1044
L4::lpc::Msg::Do_rcv_buffers	1053
L4::lpc::Msg::Cls_data	1045
L4::lpc::Msg::Do_in_data	1049
L4::lpc::Msg::Do_out_data	1051
L4::lpc::Msg::Cls_item	1046
L4::lpc::Msg::Do_in_items	1050
L4::lpc::Msg::Do_out_items	1052
L4::lpc::Msg::Dir_in	1047
L4::lpc::Msg::Do_in_data	1049
L4::lpc::Msg::Do_in_items	1050
L4::lpc::Msg::Do_rcv_buffers	1053
L4::lpc::Msg::Dir_out	1048
L4::lpc::Msg::Do_out_data	1051
L4::lpc::Msg::Do_out_items	1052
L4::lpc::Msg::Elem< Array< A, LEN > & >	1055
L4::lpc::Msg::Elem< Array< A, LEN > >	1056
L4::lpc::Msg::Elem< Array_ref< A, LEN > & >	1057

L4::lpc::Msg::Svr_arg_pack< IPC_TYPE >	1060
L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >	1060
L4::lpc::Msg_ptr< T >	1061
L4::lpc::Opt< T >	1062
L4::lpc::Ostream	1064
L4::lpc::lostream	1022
L4::lpc::Out< T >	1070
L4::lpc::Ret_array< T >	1071
L4::lpc::Send_only	1072
L4::lpc::Small_buf	1073
L4::lpc::Snd_item	1074
L4::lpc::Str_cp_in< T >	1075
L4::lpc::Varg	1076
L4::lpc::Varg_list_ref	1084
L4::lpc::Varg_list< MAX >	1082
L4::lpc::Varg_list_ref::iterator	1086
L4::lpc_svr::Compound_reply	1096
L4::lpc_svr::Default_loop_hooks	1098
L4::Server< L4::lpc_svr::Default_loop_hooks >	1228
L4::Server< LOOP_HOOKS >	1228
L4Re::Util::Registry_server< LOOP_HOOKS >	1578
L4Re::Util::Br_manager_hooks	1529
L4::lpc_svr::Default_setup_wait	1101
L4::lpc_svr::Default_timeout	1102
L4::lpc_svr::Default_loop_hooks	1098
L4Re::Util::Br_manager_hooks	1529
L4::lpc_svr::Direct_dispatch< R >	1104
L4::lpc_svr::Exc_dispatch< R, Exc >	1107
L4::lpc_svr::Direct_dispatch< R * >	1106
L4::lpc_svr::Ignore_errors	1109
L4::lpc_svr::Default_loop_hooks	1098
L4Re::Util::Br_manager_hooks	1529
L4Re::Util::Br_manager_timeout_hooks	1531
L4::lpc_svr::Server_iface	1110
L4::lpc_svr::Timeout_queue_hooks< Loop_hooks, L4Re::Util::Br_manager >	1124
L4::lpc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager >	1124
L4Re::Util::Br_manager_timeout_hooks	1531
L4::lpc_svr::Br_manager_no_buffers	1092
L4::lpc_svr::Default_loop_hooks	1098
L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >	1124
L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >	1124
L4Re::Util::Br_manager	1523
L4::lpc_svr::Timeout_queue_hooks< Loop_hooks, L4Re::Util::Br_manager >	1124
L4Re::Util::Br_manager_hooks	1529
L4::lpc_svr::Timeout_queue	1119
L4::Kip::Mem_desc	1153
L4::Kobject	1163
L4::Kobject_t< Arm_smccc, L4::Kobject, PROTO, Type_info::Demand_t<> >	1174
L4::Kobject_t< Debugger, Kobject, L4_PROTO_DEBUGGER >	1174
L4::Debugger	944
L4::Kobject_t< Exception, L4::Kobject, PROTO, Type_info::Demand_t<> >	1174
L4::Kobject_t< Factory, Kobject, L4_PROTO_FACTORY >	1174
L4::Factory	972
L4::Kobject_t< Mem_alloc, L4::Factory, L4::PROTO_EMPTY >	1174
L4Re::Mem_alloc	1470

L4::Kobject_t< Io_pager, L4::Kobject, PROTO, Type_info::Demand_t<> >	1174
L4::Kobject_t< Irq_eoi, L4::Kobject, PROTO, Type_info::Demand_t<> >	1174
L4::Kobject_t< Derived, L4::Kobject, L4::PROTO_ANY, Type_info::Demand_t<> >	1174
L4::Kobject_t< Meta, Kobject, L4_PROTO_META >	1174
L4::Meta	1180
L4::Kobject_t< Platform_control, Kobject, L4_PROTO_PLATFORM_CTL >	1174
L4::Platform_control	1192
L4::Kobject_t< Rcv_endpoint, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >	1174
L4::Rcv_endpoint	1203
L4::Kobject_2t< Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER >	1166
L4::Irq	1130
L4::Kobject_t< Ipc_gate, Rcv_endpoint, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >	1174
L4::Ipc_gate	1087
L4::Kobject_t< Task, Kobject, L4_PROTO_TASK, Type_info::Demand_t< 2 > >	1174
L4::Task	1251
L4::Kobject_t< Vm, Task, L4_PROTO_VM >	1174
L4::Vm	1346
L4::Vm	1346
L4::Kobject_t< Thread, Kobject, L4_PROTO_THREAD, Type_info::Demand_t< 1 > >	1174
L4::Thread	1262
L4::Kobject_t< Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE, L4::Type_info::Demand_t< 1 > >	1174
L4Re::Dataspace	1413
L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >	1169
L4vbus::Vbus	1728
L4::Kobject_t< Debug_obj, L4::Kobject, L4RE_PROTO_DEBUG >	1174
L4Re::Debug_obj	1426
L4::Kobject_t< Dma_space, L4::Kobject, PROTO, L4::Type_info::Demand_t< 1 > >	1174
L4::Kobject_t< Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR >	1174
L4Re::Inhibitor	1460
L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >	1169
L4::Kobject_t< Mmio_space, L4::Kobject, L4RE_PROTO_MMIO_SPACE >	1174
L4Re::Mmio_space	1476
L4::Kobject_t< Namespace, L4::Kobject, L4RE_PROTO_NAMESPACE, L4::Type_info::Demand_t< 1 > >	1174
L4Re::Namespace	1480
L4::Kobject_t< Cmd_control, L4::Kobject, L4::PROTO_ANY, Type_info::Demand_t<> >	1174
L4::Kobject_t< Parent, L4::Kobject, L4RE_PROTO_PARENT >	1174
L4Re::Parent	1489
L4::Kobject_t< Goos, L4::Kobject, L4RE_PROTO_GOOS >	1174
L4Re::Video::Goos	1631
L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >	1166
L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >	1169
L4::Kobject_demand< T >	1173
L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >	1174
L4::Kobject_typeid< T >	1175
L4::Kobject_typeid< void >	1178
L4::Kobject_x< Derived, ARGS >	1179
L4::Poll_timeout_counter	1196
L4::Poll_timeout_kipclock	1199
L4::Proto_t< P >	1202
L4::Registry_iface	1207
L4Re::Util::Object_registry	1569
L4::String	1250
L4::Thread::Attr	1276
L4::Thread::Modify_senders	1281

L4::Type_info	1287
L4::Type_info::Demand	1288
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 >	1292
L4::Type_info::Demand_union_t< Kobject_typeid< T1 >::Demand, Kobject_demand< T2... > >	1294
L4::Type_info::Demand_union_t< D1, D2 >	1294
L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >	1292
L4::Typeid::Detail::_Rpcs< OPCODE, O, Default_op< R > >::Rpc< Y >	1299
L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >	1299
L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >::Rpc< Y >	1301
L4::Typeid::Detail::Rpcs_end	1301
L4::Typeid::Detail::_Rpcs< void, 0, OPERATION >	1297
L4::Typeid::Rpc_nocode< OPERATION >	1304
L4::Typeid::Detail::_Rpcs< L4::Opcode, 0, RPCS... >	1297
L4::Typeid::Rpcs< RPCS >	1306
L4::Typeid::Detail::_Rpcs< OPCODE_TYPE, 0, RPCS... >	1297
L4::Typeid::Rpcs_code< OPCODE_TYPE >::F< RPCS >	1309
L4::Typeid::Detail::_Rpcs< l4_umword_t, 0, ARG... >	1297
L4::Typeid::Rpcs_sys< ARG >	1311
L4::Typeid::Detail::_Rpcs< OPCODE, O, X >	1297
L4::Typeid::P_dispatch< LIST >	1302
L4::Typeid::Raw_ipc< CLASS >	1303
L4::Typeid::Rpcs_code< OPCODE_TYPE >	1308
L4::Types::Bool< V >	1313
L4::Types::Flags< BITS_ENUM, UNDERLYING >	1316
L4::Types::Flags_ops_t< DT >	1320
L4::Types::Int_for_size< SIZE, bool >	1324
L4::Types::Int_for_type< T >	1325
L4::Uart	1330
l4_buf_regs_t	1351
l4_exc_regs_t	1352
l4_fpage_t	1355
l4_icu_info_t	1356
L4::lcu::Info	995
l4_icu_msi_info_t	1357
l4_kernel_info_mem_desc_t	1358
l4_kernel_info_t	1359
l4_msg_regs_t	1361
l4_msgtag_t	1362
l4_sched_cpu_set_t	1364
l4_sched_param_t	1368
l4_snd_fpage_t	1369
l4_thread_regs_t	1370
l4_timeout_s	1371
l4_timeout_t	1372
l4_vcon_attr_t	1373
l4_vcpu_arch_state_t	1374
l4_vcpu_ipc_regs_t	1375
l4_vcpu_regs_t	1376
l4_vcpu_state_t	1380
L4vcpu::Vcpu	1743
l4_vhw_descriptor	1383
l4_vhw_entry	1384
l4_vm_svm_vmcb_control_area	1385
l4_vm_svm_vmcb_state_save_area	1386
l4_vm_svm_vmcb_state_save_area_seg	1387
l4_vm_svm_vmcb_t	1387

L4_vm_tz_state	1388
L4drivers::Register_block< MAX_BITS, BLOCK >	1390
L4drivers::Register_block_base< MAX_BITS >	1395
L4drivers::Register_block_impl< BASE, MAX_BITS >	1396
L4drivers::Mmio_register_block< MAX_BITS >	1389
L4drivers::Register_block_tmpl< BLOCK >	1397
L4drivers::Ro_register_block< MAX_BITS, BLOCK >	1403
L4drivers::Ro_register_tmpl< BITS, BLOCK >	1405
L4drivers::Register_tmpl< BITS, BLOCK >	1399
L4Re::Cap_alloc	1407
L4Re::Dataspace::F	1424
L4Re::Dataspace::Stats	1425
L4Re::Default_event_payload	1429
L4Re::Env	1436
L4Re::Event_buffer_t< PAYLOAD >	1455
L4Re::Event_buffer_t< PAYLOAD >::Event	1459
L4Re::Rm::F	1512
L4Re::Rm::Range	1514
L4Re::Rm::Unique_region< T >	1515
L4Re::Smart_cap_auto< Unmap_flags >	1521
L4Re::Smart_count_cap< Unmap_flags >	1522
L4Re::Util::Cap_alloc_base	1535
L4Re::Util::Counter< COUNTER >	1536
L4Re::Util::Counter_atomic< COUNTER >	1536
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >	1537
L4Re::Util::Dataspace_svr	1543
L4Re::Util::Event_svr< SVR >	1556
L4Re::Util::Event_t< PAYLOAD >	1558
L4Re::Util::Item_alloc_base	1562
L4Re::Util::Names::Name_space	1566
L4Re::Util::Ref_cap< T >	1576
L4Re::Util::Ref_del_cap< T >	1577
L4Re::Util::Smart_cap_auto< Unmap_flags >	1584
L4Re::Util::Smart_count_cap< Unmap_flags >	1585
L4Re::Util::Vcon_svr< SVR >	1586
L4Re::Util::Video::Goos_svr	1587
L4Re::Vfs::Directory	1597
L4Re::Vfs::File	1602
L4Re::Vfs::Be_file	1590
L4Re::Vfs::File_system	1605
L4Re::Vfs::Be_file_system	1594
L4Re::Vfs::Fs	1607
L4Re::Vfs::Ops	1616
L4Re::Vfs::Generic_file	1611
L4Re::Vfs::File	1602
L4Re::Vfs::Mman	1614
L4Re::Vfs::Ops	1616
L4Re::Vfs::Regular_file	1619
L4Re::Vfs::File	1602
L4Re::Vfs::Special_file	1625
L4Re::Vfs::File	1602
L4Re::Video::Color_component	1627
L4Re::Video::Goos::Info	1637
L4Re::Video::Pixel_info	1639
L4Re::Video::View	1647
L4Re::Video::View::Info	1651

l4re_aux_t	1654
l4re_ds_stats_t	1655
l4re_elf_aux_mword_t	1655
l4re_elf_aux_t	1656
l4re_elf_aux_vma_t	1656
l4re_env_cap_entry_t	1657
l4re_env_t	1659
l4re_event_t	1661
l4re_video_color_component_t	1662
l4re_video_goos_info_t	1662
l4re_video_pixel_info_t	1664
l4re_video_view_info_t	1665
l4re_video_view_t	1666
l4shmc_ringbuf_head_t	1667
l4shmc_ringbuf_t	1668
l4util_idt_desc_t	1669
l4util_idt_header_t	1670
l4util_l4mod_info	1671
l4util_l4mod_mod	1672
l4util_mb_addr_range_t	1673
l4util_mb_apm_t	1674
l4util_mb_drive_t	1675
l4util_mb_info_t	1677
l4util_mb_mod_t	1678
l4util_mb_vbe_ctrl_t	1679
l4util_mb_vbe_mode_t	1680
L4vbus::Gpio_module::Pin_slice	1700
L4vbus::Pm< DEC >	1725
l4vbus_device_t	1739
l4vbus_resource_t	1740
L4vcpu::State	1741
L4virtio::Driver::Block_device::Handle	1771
L4virtio::Driver::Device	1772
L4virtio::Driver::Block_device	1762
L4virtio::Driver::Virtio_net_device	1785
L4virtio::Driver::Virtio_net_device::Packet	1794
L4virtio::Ptr< T >	1805
L4virtio::Svr::Bad_descriptor	1809
L4virtio::Svr::Block_request< Ds_data >	1820
L4virtio::Svr::Data_buffer	1822
L4virtio::Svr::Dev_config	1826
L4virtio::Svr::Dev_features	1838
L4virtio::Svr::Dev_status	1839
L4virtio::Svr::Device_t< DATA >	1840
L4virtio::Svr::Block_dev_base< Mem_region_info >	1811
L4virtio::Svr::Driver_mem_list_t< DATA >	1848
L4virtio::Svr::Driver_mem_region_t< DATA >	1856
L4virtio::Svr::Request_processor	1863
L4virtio::Svr::Virtqueue::Head_desc	1877
L4virtio::Virtqueue	1879
L4virtio::Driver::Virtqueue	1794
L4virtio::Svr::Virtqueue	1870
L4virtio::Virtqueue::Avail	1894
L4virtio::Virtqueue::Avail::Flags	1896
L4virtio::Virtqueue::Desc	1897
L4virtio::Virtqueue::Desc::Flags	1898
L4virtio::Virtqueue::Used	1900

L4virtio::Virtqueue::Used::Flags	1901
L4virtio::Virtqueue::Used_elem	1902
l4virtio_block_config_t	1903
l4virtio_block_discard_t	1904
l4virtio_block_header_t	1905
l4virtio_config_hdr_t	1906
l4virtio_config_queue_t	1907
l4virtio_input_absinfo_t	1909
l4virtio_input_config_t	1909
l4virtio_input_devids_t	1910
l4virtio_input_event_t	1910
l4virtio_net_config_t	1911
l4virtio_net_header_t	1911
cxx::New_allocator< _Node >	859
cxx::New_allocator< Node >	859
cxx::New_allocator< Slab_i >	859
Block_device::Impl::Partitioned_device_discard_mixin< Partitioned_device< Device >, Device >	718
L4vbus::Pm< Device >	1725
L4vbus::Device	1684
L4vbus::Gpio_module	1693
L4vbus::Gpio_pin	1700
L4vbus::Icu	1709
L4vbus::Pci_dev	1713
L4vbus::Pci_host_bridge	1719
L4virtio::Ptr< void >	1805
cxx::Ref_ptr< Device_type >	866
cxx::Ref_ptr< L4Re::Vfs::File >	866
cxx::Ref_ptr< Mount_tree >	866
L4drivers::Register_block_base< 16 >	1395
L4drivers::Register_block_base< 32 >	1395
L4drivers::Register_block_base< 64 >	1395
L4drivers::Register_block_base< 8 >	1395
L4drivers::Register_block_base< MAX_BITS >	1395
L4drivers::Register_block_impl< Mmio_register_block< 32 >, 32 >	1396
L4drivers::Register_block_tmpl< Register_block_base< MAX_BITS > >	1397
L4drivers::Register_block_tmpl< Register_block_base< MAX_BITS > const >	1397
cxx::Bits::Smart_ptr_list< Connection >	827
L4::lpc::Msg::Svr_val_ops< Array_ref< A, LEN >, Dir_in, CLASS >	1060
L4::lpc::Msg::Svr_val_ops< Array_ref< A, LEN >, Dir_out, CLASS >	1060
L4::lpc::Msg::Svr_val_ops< L4::lpc::Snd_fpage, Dir_in, CLASS >	1060
L4::lpc::Msg::Svr_val_ops< typename ELEM::svr_type, typename Direction< A * >::type, typename Class< typename Detail::_Plain< A * >::type >::type >	1060
L4::lpc::Msg::Svr_val_ops< typename ELEM::svr_type, typename Direction< A >::type, typename Class< typename Detail::_Plain< A >::type >::type >	1060
L4::lpc::Msg::Svr_val_ops< typename ELEM::svr_type, typename Direction< A const * >::type, typename Class< typename Detail::_Plain< A const * >::type >::type >	1060
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 >	1060
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 >	1060
L4::lpc::Msg::Svr_val_ops< typename ELEM::svr_type, typename Direction< T >::type, typename Class< typename Detail::_Plain< T >::type >::type >	1060
L4Re::Rm::Unique_region< char * >	1515
L4Re::Rm::Unique_region< l4_addr_t >	1515
L4Re::Rm::Unique_region< l4_uint8_t * >	1515
L4Re::Rm::Unique_region< L4virtio::Device::Config_hdr * >	1515
L4Re::Rm::Unique_region< l4virtio_config_hdr_t * >	1515
L4Re::Rm::Unique_region< unsigned char * >	1515

L4Re::Rm::Unique_region< void * >	1515
cxx::Bitmap_base::Word< Bits >	788
cxx::Bitmap_base::Word< Size >	788

Chapter 11

Data Structure Index

11.1 Data Structures

Here are the data structures with brief descriptions:

Block_device::Device_discard_feature	
Partial interface for devices that offer discard functionality	709
Block_device::Device_mgr< DEV, FACTORY >	
Basic class that scans devices and handles client connections	710
Block_device::Dma_region_info	
Base class used by the driver implementation to derive its own DMA mapping tracking structure	711
Block_device::Errand::Errand	
Wrapper for a small task executed asynchronously in the server loop	711
Block_device::Errand::Poll_errand	
Wrapper for a regularly repeated task	715
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	718
Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, true >	
Mixin implementing discard for partition devices	719
Block_device::Inout_block	
Description of an inout block to be sent to the device	720
Block_device::Inout_memory< DEV >	
Helper class that temporarily allocates memory that can be used for in/out operations with the device	721
Block_device::Mem_region_info	
Additional info stored in each L4virtio::Svr::Driver_mem_region_t used for tracking dataspace-wide DMA mappings	722
Block_device::Partition_info	
Information about a single partition	722
Block_device::Partition_reader< DEV >	
Partition table reader for block devices	723
Block_device::Partitioned_device< BASE_DEV >	
A partition device for the given device interface	724
Block_device::Pending_request	
Interface for pending requests that can be queued	725
Block_device::Pending_request::Owner	
Base class for object that can be owner of a pending request	727
Block_device::Simple_request_queue	
Simple request queue implementation based on a linked list	728
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >	
AVL tree based associative container	729

cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >	
AVL set for simple compareable items	735
cxx::Avl_tree< Node, Get_key, Compare >	
A generic AVL tree	739
cxx::Avl_tree_node	
Node of an AVL tree	746
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >	
Basic slab allocator	748
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i	
Type of a slab	754
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >	
Merged slab allocator (allocators for objects of the same size are merged together)	754
cxx::Bitfield< T, LSB, MSB >	
Definition for a member (part) of a bit field	759
cxx::Bitfield< T, LSB, MSB >::Value< TT >	
Internal helper type	771
cxx::Bitfield< T, LSB, MSB >::Value_base< TT >	
Internal helper type	773
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >	
Internal helper type	774
cxx::Bitmap< BITS >	
A static bit map	775
cxx::Bitmap_base	
Basic bitmap abstraction	779
cxx::Bitmap_base::Bit	
A writeable bit in a bitmap	787
cxx::Bitmap_base::Char< BITS >	
Helper abstraction for a byte contained in the bitmap	788
cxx::Bitmap_base::Word< BITS >	
Helper abstraction for a word contained in the bitmap	788
cxx::Bits::Avl_map_get_key< KEY_TYPE >	
Key-getter for Avl_map	790
cxx::Bits::Avl_set_get_key< KEY_TYPE >	
Internal, key-getter for Avl_set nodes	790
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >	
Internal: AVL set with internally managed nodes	791
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node	
A smart pointer to a tree item	804
cxx::Bits::Basic_list< POLICY >	
Internal: Common functions for all head-based list implementations	807
cxx::Bits::Bst< Node, Get_key, Compare >	
Basic binary search tree (BST)	809
cxx::Bits::Bst_node	
Basic type of a node in a binary search tree (BST)	823
cxx::Bits::Direction	
The direction to go in a binary search tree	825
cxx::Bits::Smart_ptr_list< ITEM >	
List of smart-pointer-managed objects	827
cxx::Bits::Smart_ptr_list_item< T, STORE_T >	
List item for an arbitrary item in a Smart_ptr_list	830
cxx::H_list< T, POLICY >	
General double-linked list of unspecified cxx::H_list_item elements	832
cxx::H_list_item_t< ELEM_TYPE >	
Basic element type for a double-linked H_list	840
cxx::H_list_t< T >	
Double-linked list of typed H_list_item_t elements	843
cxx::List< D, Alloc >	
Doubly linked list, with internal allocation	846

cxx::List< D, Alloc >::Iter	
Iterator	848
cxx::List_alloc	
Standard list-based allocator	849
cxx::List_item	
Basic list item	852
cxx::List_item::Iter	
Iterator for a list of ListItem-s	856
cxx::List_item::T_iter< T, Poly >	
Iterator for derived classes from ListItem	857
cxx::Lt_functor< Obj >	
Generic comparator class that defaults to the less-than operator	859
cxx::New_allocator< _Type >	
Standard allocator based on <code>operator new ()</code>	859
cxx::Nothrow	
Helper type to distinguish the <code>operator new</code> version that does not throw exceptions	860
cxx::Pair< First, Second >	
Pair of two values	861
cxx::Pair_first_compare< Cmp, Typ >	
Comparison functor for Pair	863
cxx::Ref_obj_list_item< T >	
Item for list linked via cxx::Ref_ptr with default reference counting	864
cxx::Ref_ptr< T, CNT >	
A reference-counting pointer with automatic cleanup	866
cxx::S_list< T, POLICY >	
Simple single-linked list	870
cxx::Slab< Type, Slab_size, Max_free, Alloc >	
Slab allocator for object of type <code>Type</code>	873
cxx::Slab_static< Type, Slab_size, Max_free, Alloc >	
Merged slab allocator (allocators for objects of the same size are merged together)	877
cxx::static_vector< T, IDX >	
Simple encapsulation for a dynamically allocated array	881
cxx::String	
Allocation free string class with explicit length field	882
cxx::Weak_ref< T >	
Typed weak reference to an object of type <code>T</code>	888
cxx::Weak_ref_base	
Generic (base) weak reference to some object	891
Elf32_Auxv	
Auxiliary vector (32-bit)	894
Elf32_Dyn	
ELF32 dynamic entry	895
Elf32_Ehdr	
ELF32 header	896
Elf32_Phdr	
ELF32 program header	898
Elf32_Rel	
ELF32 relocation entry w/o addend	900
Elf32_Rela	
ELF32 relocation entry w/ addend	900
Elf32_Shdr	
ELF32 section header	901
Elf32_Sym	
ELF32 symbol table entry	903
Elf64_Auxv	
Auxiliary vector (64-bit)	904
Elf64_Dyn	
ELF64 dynamic entry	905

Elf64_Ehdr	
ELF64 header	906
Elf64_Phdr	
ELF64 program header	908
Elf64_Rel	
ELF64 relocation entry w/o addend	910
Elf64_Rela	
ELF64 relocation entry w/ addend	910
Elf64_Shdr	
ELF64 section header	911
Elf64_Sym	
ELF64 symbol table entry	913
gfxbitmap_offset	
Offsets in pmap[] and bmap[]	914
L4::Alloc_list	
A simple list-based allocator	915
L4::Arm_smccc	
Wrapper for function calls that follow the ARM SMC/HVC calling convention	915
L4::Base_exception	
Base class for all exceptions, thrown by the L4Re framework	917
L4::Basic_registry	
This registry returns the corresponding server object based on the label of an lpc_gate	919
L4::Bounds_error	
Access out of bounds	922
L4::Cap< T >	
C++ interface for capabilities	924
L4::Cap_base	
Base class for all kinds of capabilities	930
L4::Com_error	
Error conditions during IPC	940
L4::Debugger	
C++ kernel debugger API	944
L4::Element_already_exists	
Exception for duplicate element insertions	952
L4::Element_not_found	
Exception for a failed lookup (element not found)	955
L4::Epiface	
Base class for interface implementations	958
L4::Epiface_t< Derived, IFACE, BASE, bool >	
Epiface implementation for Kobject-based interface implementations	963
L4::Epiface_t0< RPC_IFACE, BASE >	
Epiface mixin for generic Kobject-based interfaces	966
L4::Exception	
Exception interface	969
L4::Exception_tracer	
Back-trace support for exceptions	970
L4::Factory	
C++ Factory interface, see Factory for the C interface	972
L4::Factory::Lstr	
Special type to add a pascal string into the factory create stream	980
L4::Factory::Nil	
Special type to add a void argument into the factory create stream	982
L4::Factory::S	
Stream class for the create() argument stream	982
L4::Icu	
C++ Icu interface, see Interrupt controller for the C interface	986
L4::Icu::Info	
This class encapsulates information about an ICU	995

L4::Invalid_capability	Indicates that an invalid object was invoked	997
L4::lo_pager	lo_pager interface	1000
L4::lommu	Interface for IO-MMUs used for DMA remapping	1002
L4::LOModifier	Modifier class for the IO stream	1004
L4::lpc::Array< ELEM_TYPE, LEN_TYPE >	Array data type for dynamically sized arrays in RPCs	1005
L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >	Server-side copy in buffer for Array	1008
L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >	Array reference data type for arrays located in the message	1009
L4::lpc::As_value< T >	Pass the argument as plain data value	1010
L4::lpc::Buf_item	RPC warpper for a receive item	1011
L4::lpc::Call	RPC attribute for a standard RPC call	1012
L4::lpc::Call_t< RIGHTS >	RPC attribute for an RPC call with required rights	1013
L4::lpc::Call_zero_send_timeout	RPC attribute for an RPC call, with zero send timeout	1014
L4::lpc::Cap< T >	Capability type for RPC interfaces (see L4::Cap<T>)	1016
L4::lpc::Gen_fpage< T >	Generic RPC wrapper for L4 flex-pages	1018
L4::lpc::In_out< T >	Mark an argument as in-out argument	1021
L4::lpc::lostream	Input/Output stream for IPC [un]marshalling	1022
L4::lpc::lstream	Input stream for IPC unmarshalling	1032
L4::lpc::Msg::CInt_val_ops< MTYPE, DIR, CLASS >	Defines client-side handling of 'MTYPE' as RPC argument	1043
L4::lpc::Msg::Cls_buffer	Marker type for receive buffer values	1044
L4::lpc::Msg::Cls_data	Marker type for data values	1045
L4::lpc::Msg::Cls_item	Marker type for item values	1046
L4::lpc::Msg::Dir_in	Marker type for input values	1047
L4::lpc::Msg::Dir_out	Marker type for output values	1048
L4::lpc::Msg::Do_in_data	Marker for Input data	1049
L4::lpc::Msg::Do_in_items	Marker for Input items	1050
L4::lpc::Msg::Do_out_data	Marker for Output data	1051
L4::lpc::Msg::Do_out_items	Marker for Output items	1052
L4::lpc::Msg::Do_rcv_buffers	Marker for receive buffers	1053
L4::lpc::Msg::Elem< Array< A, LEN > & >	Array as output argument	1055

L4::lpc::Msg::Elem< Array< A, LEN > >	
Array as input arguments	1056
L4::lpc::Msg::Elem< Array_ref< A, LEN > & >	
Array_ref as output argument	1057
L4::lpc::Msg::Is_valid_rpc_type< T >	
Type trait defining a valid RPC parameter type	1058
L4::lpc::Msg::Svr_arg_pack< IPC_TYPE >	
Server-side RPC arguments data structure used to provide arguments to the server-side implementation of an RPC function	1060
L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >	
Defines server-side handling for MTYPE server arguments	1060
L4::lpc::Msg_ptr< T >	
Pointer to an element of type T in an lpc::Istream	1061
L4::lpc::Opt< T >	
Attribute for defining an optional RPC argument	1062
L4::lpc::Ostream	
Output stream for IPC marshalling	1064
L4::lpc::Out< T >	
Mark an argument as a output value in an RPC signature	1070
L4::lpc::Ret_array< T >	
Dynamically sized output array of type T	1071
L4::lpc::Send_only	
RPC attribute for a send-only RPC	1072
L4::lpc::Small_buf	
A receive item for receiving a single object capability	1073
L4::lpc::Snd_item	
RPC wrapper for a send item	1074
L4::lpc::Str_cp_in< T >	
Abstraction for extracting a zero-terminated string from an lpc::Istream	1075
L4::lpc::Varg	
Variably sized RPC argument	1076
L4::lpc::Varg_list< MAX >	
Self-contained list of variable-sized RPC parameters	1082
L4::lpc::Varg_list_ref	
List of variable-sized RPC parameters as received by the server	1084
L4::lpc::Varg_list_ref::Iterator	
Iterator for Valists	1086
L4::lpc_gate	
The C++ IPC gate interface, see IPC-Gate API for the C interface	1087
L4::lpc_svr::Br_manager_no_buffers	
Empty implementation of Server_iface	1092
L4::lpc_svr::Compound_reply	
Mix in for LOOP_HOOKS to always use compound reply and wait	1096
L4::lpc_svr::Default_loop_hooks	
Default LOOP_HOOKS	1098
L4::lpc_svr::Default_setup_wait	
Mix in for LOOP_HOOKS for setup_wait no op	1101
L4::lpc_svr::Default_timeout	
Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout	1102
L4::lpc_svr::Direct_dispatch< R >	
Direct dispatch helper, for forwarding dispatch calls to a registry <i>R</i>	1104
L4::lpc_svr::Direct_dispatch< R * >	
Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry <i>R</i>	1106
L4::lpc_svr::Exc_dispatch< R, Exc >	
Dispatch helper wrapping try {} catch {} around the dispatch call	1107
L4::lpc_svr::Ignore_errors	
Mix in for LOOP_HOOKS to ignore IPC errors	1109

L4::lpc_svr::Server_iface	Interface for server-loop related functions	1110
L4::lpc_svr::Timeout	Callback interface for Timeout_queue	1116
L4::lpc_svr::Timeout_queue	Timeout queue to be used in L4re server loop	1119
L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >	Loop hooks mixin for integrating a timeout queue into the server loop	1124
L4::Irq	C++ Irq interface, see IRQs for the C interface	1130
L4::Irq_eoi	Interface for sending an unmask message to an object	1138
L4::Irq_handler_object	Server object base class for handling IRQ messages	1140
L4::Irq_mux	IRQ multiplexer for shared IRQs	1144
L4::Irqp_t< Derived, BASE, bool >	Epiface implementation for interrupt handlers	1148
L4::Kip::Mem_desc	Memory descriptors stored in the kernel interface page	1153
L4::Kobject	Base class for all kinds of kernel objects and remote objects, referenced by capabilities	1163
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)	1166
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)	1169
L4::Kobject_demand< T >	Get the combined server-side resource requirements for all type T	1173
L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >	Helper class to create an L4Re interface class that is derived from a single base class	1174
L4::Kobject_typeid< T >	Meta object for handling access to type information of Kobjects	1175
L4::Kobject_typeid< void >	Minimalistic ID for void interface	1178
L4::Kobject_x< Derived, ARGS >	Generic Kobject inheritance template	1179
L4::Meta	Meta interface that shall be implemented by each L4Re object and gives access to the dynamic type information for L4Re objects	1180
L4::Out_of_memory	Exception signalling insufficient memory	1185
L4::Pager	Pager interface including the Io_pager interface	1188
L4::Platform_control	L4 C++ interface for controlling platform-wide properties, see Platform Control C API for the C interface	1192
L4::Poll_timeout_counter	Evaluate an expression for a maximum number of times	1196
L4::Poll_timeout_kipclock	A polling timeout based on the L4Re clock	1199
L4::Proto_t< P >	Data type for defining protocol numbers	1202
L4::Rcv_endpoint	Interface for kernel objects that allow to receive IPC from them	1203
L4::Registry_iface	Abstract interface for object registries	1207

L4::Runtime_error	
Exception for an abstract runtime error	1211
L4::Scheduler	
C++ interface of the Scheduler kernel object, see Scheduler for the C interface	1215
L4::Semaphore	
C++ Kernel-provided semaphore interface, see Kernel-provided semaphore for the C interface	1222
L4::Server< LOOP_HOOKS >	
Basic server loop for handling client requests	1228
L4::Server_object	
Abstract server object to be used with L4::Server and L4::Basic_registry	1233
L4::Server_object_t< IFACE, BASE >	
Base class (template) for server implementing server objects	1237
L4::Server_object_x< Derived, IFACE, BASE >	
Helper class to implement p_dispatch based server objects	1243
L4::Smart_cap< T, SMART >	
Smart capability class	1246
L4::String	
A null-terminated string container class	1250
L4::Task	
C++ interface of the Task kernel object, see Task for the C interface	1251
L4::Thread	
C++ L4 kernel thread interface, see Thread for the C interface	1262
L4::Thread::Attr	
Thread attributes used for control()	1276
L4::Thread::Modify_senders	
Class wrapping a list of rules which modify the sender label of IPC messages inbound to this thread	1281
L4::Triggerable	
Interface that allows an object to be triggered by some source	1283
L4::Type_info	
Dynamic Type Information for L4Re Interfaces	1287
L4::Type_info::Demand	
Data type for expressing the needed receive buffers at the server-side of an interface	1288
L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >	
Template type statically describing demand of receive buffers	1292
L4::Type_info::Demand_union_t< D1, D2 >	
Template type statically describing the combination of two Demand object	1294
L4::Typeid::Detail::_Rpc< OPCODE, O, X >	
Empty list of RPCs	1297
L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y >	
Find the given RPC in the list	1299
L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >	
Non-empty list of RPCs	1299
L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y >	
Find the given RPC in the list	1301
L4::Typeid::Detail::Rpc_end	
Internal end-of-list marker	1301
L4::Typeid::P_dispatch< LIST >	
Use for protocol based dispatch stage	1302
L4::Typeid::Raw_ipc< CLASS >	
RPCs list for passing raw incoming IPC to the server object	1303
L4::Typeid::Rpc_nocode< OPERATION >	
List of RPCs of an interface using a single operation without an opcode	1304
L4::Typeid::Rpc< RPCS >	
Standard list of RPCs of an interface	1306
L4::Typeid::Rpc_code< OPCODE_TYPE >	
List of RPCs of an interface using a special opcode type	1308
L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >	
.	1309

L4::Typeid::Rpcsys< ARG >	
List of RPCs typically used for kernel interfaces	1311
L4::Types::Bool< V >	
Boolean meta type	1313
L4::Types::False	
False meta value	1314
L4::Types::Flags< BITS_ENUM, UNDERLYING >	
Template for defining typical Flags bitmaps	1316
L4::Types::Flags_ops_t< DT >	
Mixin class to define a set of friend bitwise operators on DT	1320
L4::Types::Flags_t< DT, T >	
Template type to define a flags type with bitwise operations	1322
L4::Types::Int_for_size< SIZE, bool >	
Metafunction to get an unsigned integral type for the given size	1324
L4::Types::Int_for_type< T >	
Metafunction to get an integral type of the same size as T	1325
L4::Types::Same< A, B >	
Compare two data types for equality	1326
L4::Types::True	
True meta value	1328
L4::Uart	
Uart driver abstraction	1330
L4::Unknown_error	
Exception for an unknown condition	1335
L4::Vcon	
C++ L4 Vcon interface, see Virtual Console for the C interface	1337
L4::Vm	
Virtual machine host address space	1346
l4_buf_regs_t	
Encapsulation of the buffer-registers block in the UTCB	1351
l4_exc_regs_t	
UTCB structure for exceptions	1352
l4_fpage_t	
L4 flexpage type	1355
l4_icu_info_t	
Info structure for an ICU	1356
l4_icu_msi_info_t	
Info to use for a specific MSI	1357
l4_kernel_info_mem_desc_t	
Memory descriptor data structure	1358
l4_kernel_info_t	
L4 Kernel Interface Page	1359
l4_msg_regs_t	
Encapsulation of the message-register block in the UTCB	1361
l4_msgtag_t	
Message tag data structure	1362
l4_sched_cpu_set_t	
CPU sets	1364
l4_sched_param_t	
Scheduler parameter set	1368
l4_snd_fpage_t	
Send-flex-page types	1369
l4_thread_regs_t	
Encapsulation of the thread-control-register block of the UTCB	1370
l4_timeout_s	
Basic timeout specification	1371
l4_timeout_t	
Timeout pair	1372

l4_vcon_attr_t	Vcon attribute structure	1373
l4_vcpu_arch_state_t	Architecture-specific vCPU state	1374
l4_vcpu_ipc_regs_t	VCPU message registers	1375
l4_vcpu_regs_t	VCPU registers	1376
l4_vcpu_state_t	State of a vCPU	1380
l4_vhw_descriptor	Virtual hardware devices description	1383
l4_vhw_entry	Description of a device	1384
l4_vm_svm_vmcb_control_area	VMCB structure for SVM VMs	1385
l4_vm_svm_vmcb_state_save_area	State save area structure for SVM VMs	1386
l4_vm_svm_vmcb_state_save_area_seg	State save area segment selector struct	1387
l4_vm_svm_vmcb_t	Control structure for SVM VMs	1387
l4_vm_tz_state	State structure for TrustZone VMs	1388
L4drivers::Mmio_register_block< MAX_BITS >	An MMIO block with up to 64-bit register access (32-bit default) and little endian byte order	1389
L4drivers::Register_block< MAX_BITS, BLOCK >	Handles a reference to a register block of the given maximum access width	1390
L4drivers::Register_block_base< MAX_BITS >	Abstract register block interface	1395
L4drivers::Register_block_impl< BASE, MAX_BITS >	Implementation helper for register blocks	1396
L4drivers::Register_block_tmpl< BLOCK >	Helper template that translates to the Register_block_base interface	1397
L4drivers::Register_tmpl< BITS, BLOCK >	Single hardware register inside a Register_block_base interface	1399
L4drivers::Ro_register_block< MAX_BITS, BLOCK >	Handles a reference to a read only register block of the given maximum access width	1403
L4drivers::Ro_register_tmpl< BITS, BLOCK >	Single read only register inside a Register_block_base interface	1405
L4Re::Cap_alloc	Capability allocator interface	1407
L4Re::Console	Console class	1410
L4Re::Dataspace	Interface for memory-like objects	1413
L4Re::Dataspace::F	Dataspace flags definitions	1424
L4Re::Dataspace::Stats	Information about the dataspace	1425
L4Re::Debug_obj	Debug interface	1426
L4Re::Default_event_payload	Default event stream payload	1429
L4Re::Dma_space	Managed DMA Address Space	1430
L4Re::Env	C++ interface of the initial environment that is provided to an L4 task	1436

L4Re::Event	
Event class	1448
L4Re::Event_buffer_t< PAYLOAD >	
Event buffer class	1455
L4Re::Event_buffer_t< PAYLOAD >::Event	
Event structure used in buffer	1459
L4Re::Inhibitor	
Set of inhibitor locks, which inhibit specific actions when held	1460
L4Re::Log	
Log interface class	1465
L4Re::Mem_alloc	
Memory allocation interface	1470
L4Re::Mmio_space	
Interface for memory-like address space accessible via IPC	1476
L4Re::Namespace	
Name-space interface	1480
L4Re::Ned::Cmd_control	
Direct control interface for Ned	1487
L4Re::Parent	
Parent interface	1489
L4Re::Random	
Low-bandwidth interface for random number generators	1493
L4Re::Rm	
Region map	1498
L4Re::Rm::F	
Rm flags definitions	1512
L4Re::Rm::Range	
A range of virtual addresses	1514
L4Re::Rm::Unique_region< T >	
Unique region	1515
L4Re::Smart_cap_auto< Unmap_flags >	
Helper for Unique_cap and Unique_del_cap	1521
L4Re::Smart_count_cap< Unmap_flags >	
Helper for Ref_cap and Ref_del_cap	1522
L4Re::Util::Br_manager	
Buffer-register (BR) manager for L4::Server	1523
L4Re::Util::Br_manager_hooks	
Predefined server-loop hooks for a server loop using the Br_manager	1529
L4Re::Util::Br_manager_timeout_hooks	
Predefined server-loop hooks for a server with using the Br_manager and a timeout queue	1531
L4Re::Util::Cap_alloc_base	
Capability allocator	1535
L4Re::Util::Counter< COUNTER >	
Counter for Counting_cap_alloc with variable data width	1536
L4Re::Util::Counter_atomic< COUNTER >	
Thread safe version of counter for Counting_cap_alloc	1536
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >	
Internal reference-counting cap allocator	1537
L4Re::Util::Dataspace_svr	
Dataspace server class	1543
L4Re::Util::Event_buffer_consumer_t< PAYLOAD >	
An event buffer consumer	1548
L4Re::Util::Event_buffer_t< PAYLOAD >	
Event_buffer utility class	1552
L4Re::Util::Event_svr< SVR >	
Convenience wrapper for implementing an event server	1556
L4Re::Util::Event_t< PAYLOAD >	
Convenience wrapper for getting access to an event object	1558

L4Re::Util::Item_alloc_base	
Item allocator	1562
L4Re::Util::Names::Name	
Name class	1562
L4Re::Util::Names::Name_space	
Abstract server-side implementation of the L4::Namespace interface	1566
L4Re::Util::Object_registry	
A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread	1569
L4Re::Util::Ref_cap< T >	
Automatic capability that implements automatic free and unmap of the capability selector	1576
L4Re::Util::Ref_del_cap< T >	
Automatic capability that implements automatic free and unmap+delete of the capability selector	1577
L4Re::Util::Registry_server< LOOP_HOOKS >	
A server loop object which has a Object_registry included	1578
L4Re::Util::Smart_cap_auto< Unmap_flags >	
Helper for Unique_cap and Unique_del_cap	1584
L4Re::Util::Smart_count_cap< Unmap_flags >	
Helper for Ref_cap and Ref_del_cap	1585
L4Re::Util::Vcon_svr< SVR >	
Console server template class	1586
L4Re::Util::Video::Goos_svr	
Goos server class	1587
L4Re::Vfs::Be_file	
Boiler plate class for implementing an open file for L4Re::Vfs	1590
L4Re::Vfs::Be_file_system	
Boilerplate class for implementing a L4Re::Vfs::File_system	1594
L4Re::Vfs::Directory	
Interface for a POSIX file that is a directory	1597
L4Re::Vfs::File	
The basic interface for an open POSIX file	1602
L4Re::Vfs::File_system	
Basic interface for an L4Re::Vfs file system	1605
L4Re::Vfs::Fs	
POSIX File-system related functionality	1607
L4Re::Vfs::Generic_file	
The common interface for an open POSIX file	1611
L4Re::Vfs::Mman	
Interface for POSIX memory management	1614
L4Re::Vfs::Ops	
Interface for the POSIX backends of an application	1616
L4Re::Vfs::Regular_file	
Interface for a POSIX file that provides regular file semantics	1619
L4Re::Vfs::Special_file	
Interface for a POSIX file that provides special file semantics	1625
L4Re::Video::Color_component	
A color component	1627
L4Re::Video::Goos	
Class that abstracts framebuffers	1631
L4Re::Video::Goos::Info	
Information structure of a Goos	1637
L4Re::Video::Pixel_info	
Pixel information	1639
L4Re::Video::View	
View of a framebuffer	1647
L4Re::Video::View::Info	
Information structure of a view	1651

l4re_aux_t	Auxiliary descriptor	1654
l4re_ds_stats_t	Information about the data space	1655
l4re_elf_aux_mword_t	Auxiliary vector element for a single unsigned data word	1655
l4re_elf_aux_t	Generic header for each auxiliary vector element	1656
l4re_elf_aux_vma_t	Auxiliary vector element for a reserved virtual memory area	1656
l4re_env_cap_entry_t	Entry in the L4Re environment array for the named initial objects	1657
l4re_env_t	Initial environment data structure	1659
l4re_event_t	Event structure used in buffer	1661
l4re_video_color_component_t	Color component structure	1662
l4re_video_goos_info_t	Goos information structure	1662
l4re_video_pixel_info_t	Pixel_info structure	1664
l4re_video_view_info_t	View information structure	1665
l4re_video_view_t	C representation of a goos view	1666
l4shmc_ringbuf_head_t	Head field of a ring buffer	1667
l4shmc_ringbuf_t	Ring buffer	1668
l4util_idt_desc_t	IDT entry	1669
l4util_idt_header_t	Header of an IDT table	1670
l4util_l4mod_info	Base module structure	1671
l4util_l4mod_mod	A single module	1672
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	1673
l4util_mb_apm_t	APM BIOS info	1674
l4util_mb_drive_t	Drive Info structure	1675
l4util_mb_info_t	MultiBoot Info description	1677
l4util_mb_mod_t	The structure type "mod_list" is used by the multiboot_info structure	1678
l4util_mb_vbe_ctrl_t	VBE controller information	1679
l4util_mb_vbe_mode_t	VBE mode information	1680
L4vbus::Device	Device on a L4vbus::Vbus	1684
L4vbus::Gpio_module	A Gpio_module groups multiple GPIO pins together	1693

L4vbus::Gpio_module::Pin_slice	
A slice of the pins provided by this module	1700
L4vbus::Gpio_pin	
A GPIO pin	1700
L4vbus::Icu	
Vbus Interrupt controller API	1709
L4vbus::Pci_dev	
A PCI device	1713
L4vbus::Pci_host_bridge	
A Pci host bridge	1719
L4vbus::Pm< DEC >	
Power-management API mixin	1725
L4vbus::Vbus	
The virtual bus (Vbus) interface	1728
l4vbus_device_t	
Detailed information about a vbus device	1739
l4vbus_resource_t	
Description of a single vbus resource	1740
L4vcpu::State	
C++ implementation of state word in the vCPU area	1741
L4vcpu::Vcpu	
C++ implementation of the vCPU save state area	1743
L4virtio::Device	
IPC interface for virtio over L4 IPC	1754
L4virtio::Driver::Block_device	
Simple class for accessing a virtio block device synchronously	1762
L4virtio::Driver::Block_device::Handle	
Handle to an ongoing request	1771
L4virtio::Driver::Device	
Client-side implementation for a general virtio device	1772
L4virtio::Driver::Virtio_net_device	
Simple class for accessing a virtio net device	1785
L4virtio::Driver::Virtio_net_device::Packet	
Structure for a network packet (header including data) with maximum size, assuming that no extra features have been negotiated	1794
L4virtio::Driver::Virtqueue	
Driver-side implementation of a Virtqueue	1794
L4virtio::Ptr< T >	
Pointer used in virtio descriptors	1805
L4virtio::Svr::Bad_descriptor	
Exception used by Queue to indicate descriptor errors	1809
L4virtio::Svr::Block_dev_base< Ds_data >	
Base class for virtio block devices	1811
L4virtio::Svr::Block_request< Ds_data >	
A request to read or write data	1820
L4virtio::Svr::Data_buffer	
Abstract data buffer	1822
L4virtio::Svr::Dev_config	
Abstraction for L4-Virtio device config memory	1826
L4virtio::Svr::Dev_features	
Type for device feature bitmap	1838
L4virtio::Svr::Dev_status	
Type of the device status register	1839
L4virtio::Svr::Device_t< DATA >	
Server-side L4-VIRTIO device stub	1840
L4virtio::Svr::Driver_mem_list_t< DATA >	
List of driver memory regions assigned to a single L4-VIRTIO transport instance	1848

L4virtio::Svr::Driver_mem_region_t< DATA >	
Region of driver memory, that shall be managed locally	1856
L4virtio::Svr::Request_processor	
Encapsulate the state for processing a VIRTIO request	1863
L4virtio::Svr::Virtqueue	
Virtqueue implementation for the device	1870
L4virtio::Svr::Virtqueue::Head_desc	
VIRTIO request, essentially a descriptor from the available ring	1877
L4virtio::Virtqueue	
Low-level Virtqueue	1879
L4virtio::Virtqueue::Avail	
Type of available ring, this is read-only for the host	1894
L4virtio::Virtqueue::Avail::Flags	
Flags of the available ring	1896
L4virtio::Virtqueue::Desc	
Descriptor in the descriptor table	1897
L4virtio::Virtqueue::Desc::Flags	
Type for descriptor flags	1898
L4virtio::Virtqueue::Used	
Used ring	1900
L4virtio::Virtqueue::Used::Flags	
Flags for the used ring	1901
L4virtio::Virtqueue::Used_elem	
Type of an element of the used ring	1902
l4virtio_block_config_t	
Device configuration for block devices	1903
l4virtio_block_discard_t	
Structure used for the write zeroes and discard commands	1904
l4virtio_block_header_t	
Header structure of a request for a block device	1905
l4virtio_config_hdr_t	
L4-VIRTIO config header, provided in shared data space	1906
l4virtio_config_queue_t	
Queue configuration entry	1907
l4virtio_input_absinfo_t	
Information about the absolute axis in the underlying evdev implementation	1909
l4virtio_input_config_t	
Device configuration for input devices	1909
l4virtio_input_devids_t	
Device ID information for the device	1910
l4virtio_input_event_t	
Single event in event or status queue	1910
l4virtio_net_config_t	
Device configuration for network devices	1911
l4virtio_net_header_t	
Header structure of a request for a network device	1911

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	1917
pkg/drivers-frst/include/ hw_mmio_register_block	1917
pkg/drivers-frst/include/ hw_register_block	1918
pkg/drivers-frst/include/ io_regblock.h	1923
pkg/drivers-frst/include/ io_regblock_port.h	1926
pkg/drivers-frst/include/ Makefile	1940
pkg/drivers-frst/include/ poll_timeout_counter.h	1926
pkg/drivers-frst/include/ARCH-amd64/ asm_access.h	1913
pkg/drivers-frst/include/ARCH-arm/ asm_access.h	1913
pkg/drivers-frst/include/ARCH-arm64/ asm_access.h	1914
pkg/drivers-frst/include/ARCH-mips/ asm_access.h	1915
pkg/drivers-frst/include/ARCH-ppc32/ asm_access.h	1915
pkg/drivers-frst/include/ARCH-sparc/ asm_access.h	1916
pkg/drivers-frst/include/ARCH-x86/ asm_access.h	1916
pkg/drivers-frst/uart/include/ Makefile	1940
pkg/drivers-frst/uart/include/ uart_16550.h	1927
pkg/drivers-frst/uart/include/ uart_16550_dw.h	1928
pkg/drivers-frst/uart/include/ uart_base.h	1929
pkg/drivers-frst/uart/include/ uart_cadence.h	1930
pkg/drivers-frst/uart/include/ uart_dcc-v6.h	1930
pkg/drivers-frst/uart/include/ uart_dm.h	1931
pkg/drivers-frst/uart/include/ uart_dummy.h	1931
pkg/drivers-frst/uart/include/ uart_geni.h	1932
pkg/drivers-frst/uart/include/ uart_imx.h	1932
pkg/drivers-frst/uart/include/ uart_leon3.h	1933
pkg/drivers-frst/uart/include/ uart_linflex.h	1934
pkg/drivers-frst/uart/include/ uart_lpuart.h	1934
pkg/drivers-frst/uart/include/ uart_mvebu.h	1935
pkg/drivers-frst/uart/include/ uart_of.h	1935
pkg/drivers-frst/uart/include/ uart_omap35x.h	1936
pkg/drivers-frst/uart/include/ uart_pl011.h	1936
pkg/drivers-frst/uart/include/ uart_s3c2410.h	1937
pkg/drivers-frst/uart/include/ uart_sa1000.h	1938
pkg/drivers-frst/uart/include/ uart_sh.h	1938
pkg/l4re-core/ned/lib/include/ cmd_control	1939

pkg/l4re-core/ned/lib/include/Makefile	1940
amd64/l4/sys/__kip-arch.h	1997
amd64/l4/sys/__vcpu-arch.h	
AMD64-specific vCPU interface	1998
amd64/l4/sys/cache.h	
Cache functions	2539
amd64/l4/sys/consts.h	
Common L4 constants, amd64 version	2555
amd64/l4/sys/ktrace_events.h	2006
amd64/l4/sys/l4int.h	
Fixed sized integer types, amd64 version	2698
amd64/l4/sys/linkage.h	
Linkage	2016
amd64/l4/sys/segment.h	
Segment handling	1974
amd64/l4/sys/utcb.h	
UTCB definitions for amd64	2750
amd64/l4/sys/vm.h	2021
amd64/l4/util/bitops_arch.h	
Amd64 bit manipulation functions	2023
amd64/l4/util/cpu.h	
CPU related functions	2030
amd64/l4/util/idt.h	
IDT related functions	1940
amd64/l4/util/irq.h	
Some PIC and hardware interrupt related functions	2173
amd64/l4/util/l4_macros.h	
Main function	2035
amd64/l4/util/mbi_argv.h	
Command line handling	2038
amd64/l4/util/perform.h	
Performance Monitoring using P5/P6 Measurement Counters	1944
amd64/l4/util/port_io.h	
Port I/O functions	1988
amd64/l4/util/rdtsc.h	
Timestamp counter related functions	1956
amd64/l4/util/spin.h	
Spinning for amd64	1965
amd64/l4/util/util.h	
Utilities, amd64 version	1967
amd64/l4f/l4/sys/ipc.h	2655
amd64/l4f/l4/sys/segment.h	
L4f specific fs/gs manipulation	1980
amd64/l4f/l4/util/port_io.h	
Port I/O functions	1989
arm/l4/sys/__kip-arch.h	1998
arm/l4/sys/__vcpu-arch.h	
ARM-specific vCPU interface	2001
arm/l4/sys/atomic.h	
Atomic memory modifications	2781
arm/l4/sys/cache.h	
Cache functions	2541
arm/l4/sys/consts.h	
Common L4 constants, arm version	2556
arm/l4/sys/ktrace_events.h	2009
arm/l4/sys/l4int.h	
Fixed sized integer types, arm version	2699

arm/l4/sys/linkage.h	
Linkage	2017
arm/l4/sys/mem_op.h	
Memory access functions (ARM specific)	2018
arm/l4/sys/task.h	2738
arm/l4/sys/thread.h	
ARM-specific thread related definitions	2845
arm/l4/sys/utcb.h	
UTCB definitions for ARM	2753
arm/l4/sys/vm	2773
arm/l4/sys/vm.h	
ARM virtualization interface	2021
arm/l4/util/bitops_arch.h	
ARM specific implementation of bitops functions	2026
arm/l4/util/cpu.h	
CPU related functions	2033
arm/l4/util/irq.h	
ARM specific implementation of irq functions	2175
arm/l4/util/l4_macros.h	
Main function	2036
arm/l4/util/mbi_argv.h	
Multiboot	2040
arm/l4f/l4/sys/ipc.h	
L4 IPC System Calls, ARM	2655
arm/l4f/l4/sys/syscall_defs.h	
Syscall entry definitions	2043
contrib/libio-io/l4/io/io.h	2044
contrib/libio-io/l4/io/types.h	2280
l4/cxx/alloc.h	
Alloc list	2047
l4/cxx/arith	2048
l4/cxx/atomic.h	
Atomic template	2782
l4/cxx/avl_map	
AVL map	2049
l4/cxx/avl_set	
AVL set	2052
l4/cxx/avl_tree	
AVL tree	2056
l4/cxx/basic_ostream	
Basic IO stream	2062
l4/cxx/basic_vector.h	
Basic vector	2066
l4/cxx/bitfield	2068
l4/cxx/bitmap	2070
l4/cxx/dlist	2091
l4/cxx/exceptions	
Base exceptions	2094
l4/cxx/hlist	2098
l4/cxx/iostream	
IO Stream	2101
l4/cxx/ipc_helper	
IPC helper	2102
l4/cxx/ipc_server	
IPC server loop	2595
l4/cxx/ipc_stream	
IPC stream	2104
l4/cxx/ipc_timeout_queue	2123

l4/cxx/l4iostream	
L4 IO stream	2125
l4/cxx/l4types.h	
L4 Types	2126
l4/cxx/list	2128
l4/cxx/list_alloc	2132
l4/cxx/main_thread	
Main thread	2137
l4/cxx/minmax	2139
l4/cxx/observer	2139
l4/cxx/pair	
Pair implementation	2140
l4/cxx/ref_ptr	2142
l4/cxx/ref_ptr_list	
Implementation of a list of ref-ptr-managed objects	2145
l4/cxx/slab_alloc	2147
l4/cxx/slist	2151
l4/cxx/static_container	2153
l4/cxx/static_vector	2154
l4/cxx/std_alloc	2155
l4/cxx/std_exc_io	
Base exceptions std stream operator	2156
l4/cxx/std_ops	2157
l4/cxx/string	2157
l4/cxx/string.h	
String	2160
l4/cxx/thread	
Thread implementation	2742
l4/cxx/type_list	2163
l4/cxx/type_traits	2164
l4/cxx/unique_ptr	2168
l4/cxx/unique_ptr_list	
Implementation of a list of unique-ptr-managed objects	2170
l4/cxx/utils	2171
l4/cxx/weak_ref	2172
l4/cxx/bits/bst.h	
AVL tree	2072
l4/cxx/bits/bst_base.h	
AVL tree	2077
l4/cxx/bits/bst_iter.h	
AVL tree	2080
l4/cxx/bits/list_basics.h	2083
l4/cxx/bits/smart_ptr_list.h	
Implementation of a list of smart-pointer-managed objects	2085
l4/cxx/bits/type_traits.h	2088
l4/irq/irq.h	
IRQ handling routines	2177
l4/l4re_vfs/backend	2186
l4/l4re_vfs/vfs.h	2214
l4/l4re_vfs/impl/default_ops_impl.h	2189
l4/l4re_vfs/impl/fd_store.h	2190
l4/l4re_vfs/impl/fd_store_impl.h	2191
l4/l4re_vfs/impl/ns_fs.h	2191
l4/l4re_vfs/impl/ns_fs_impl.h	2192
l4/l4re_vfs/impl/ro_file.h	2197
l4/l4re_vfs/impl/ro_file_impl.h	2198
l4/l4re_vfs/impl/vcon_stream.h	2199
l4/l4re_vfs/impl/vcon_stream_impl.h	2200

l4/l4re_vfs/impl/vfs_impl.h	2202
l4/l4virtio/l4virtio	2227
l4/l4virtio/virtio.h	2252
l4/l4virtio/virtio_block.h	2255
l4/l4virtio/virtio_input.h	2256
l4/l4virtio/virtio_net.h	2257
l4/l4virtio/virtqueue	2257
l4/l4virtio/client/l4virtio	2224
l4/l4virtio/client/virtio-block	2243
l4/l4virtio/client/virtio-net	2221
l4/l4virtio/server/l4virtio	2228
l4/l4virtio/server/virtio	2239
l4/l4virtio/server/virtio-block	2246
l4/libblock-device/block_device_mgr.h	2262
l4/libblock-device/debug.h	2267
l4/libblock-device/device.h	2269
l4/libblock-device/errand.h	2270
l4/libblock-device/gpt.h	2272
l4/libblock-device/inout_memory.h	2272
l4/libblock-device/part_device.h	2274
l4/libblock-device/partition.h	2276
l4/libblock-device/request_queue.h	2279
l4/libblock-device/types.h	2281
l4/libblock-device/virtio_client.h	2287
l4/libedid/edid.h	2295
l4/libgfxbitmap/bitmap.h	
Bitmap renderer header file	2297
l4/libgfxbitmap/font.h	
Bitmap font renderer header file	2302
l4/libgfxbitmap/support	
Terminal support functionality	2308
l4/re/cap_alloc	
Abstract capability-allocator interface	2412
l4/re/console	2346
l4/re/consts	
Constants	2566
l4/re/consts.h	
Constants	2557
l4/re/dataspace	
Dataspace interface	2346
l4/re/dataspace-sys.h	
Dataspace protocol definition	2349
l4/re/debug	
Debug interface	2427
l4/re/dma_space	2350
l4/re/elf_aux.h	
Auxiliary information for binaries	2353
l4/re/env	
Environment interface	2355
l4/re/env.h	
Environment interface	2358
l4/re/error_helper	
Error helper	2361
l4/re/event	2432
l4/re/event-sys.h	2364
l4/re/event.h	
Events	2317
l4/re/event_enums.h	2365

l4/re/inhibitor	2380
l4/re/inhibitor-sys.h	2381
l4/re/l4aux.h	
Auxiliary definitions	2381
l4/re/log	
Log interface	2382
l4/re/log-sys.h	
Log protocol definition	2384
l4/re/mem_alloc	
Memory allocator interface	2385
l4/re/mem_alloc-sys.h	
Memory allocator protocol definitions	2387
l4/re/mmio_space	
Interface definition to emit MMIO-like accesses via IPC	2389
l4/re/namespace	
Namespace interface	2390
l4/re/namespace-sys.h	
Namespace protocol definitions	2393
l4/re/parent	
Parent interface	2394
l4/re/parent-sys.h	
Parent protocol definition	2396
l4/re/protocols.h	
L4Re Protocol Constants (C version)	2397
l4/re/random	
Random number generator interface definition	2399
l4/re/rm	
Region mapper interface	2400
l4/re/rm-sys.h	
Region mapper protocol definitions	2406
l4/re/shared_cap	
Shared_cap / Shared_del_cap	2461
l4/re/unique_cap	
Unique_cap / Unique_del_cap	2467
l4/re/c/dataspace.h	
Data space C interface	2309
l4/re/c/debug.h	
Debug C interface	2267
l4/re/c/dma_space.h	
DMA space C interface	2312
l4/re/c/event.h	
Event C interface	2315
l4/re/c/event_buffer.h	2319
l4/re/c/inhibitor.h	
Inhibitor C interface	2319
l4/re/c/log.h	
Log C interface	2322
l4/re/c/mem_alloc.h	
Memory allocator C interface	2324
l4/re/c/namespace.h	
Namespace functions, C interface	2326
l4/re/c/rm.h	
Region map interface, C interface	2328
l4/re/c/util/cap_alloc.h	
Capability allocator C interface	2332
l4/re/c/util/kumem_alloc.h	
Kumem allocator utility C interface	2334

l4/re/c/util/video/goos_fb.h	
Framebuffer utility functionality	2336
l4/re/c/video/colors.h	2338
l4/re/c/video/goos.h	2340
l4/re/c/video/view.h	2343
l4/re/impl/dataspace_impl.h	
Dataspace client stub implementation	2371
l4/re/impl/mem_alloc_impl.h	
Memory allocator client stub implementation	2374
l4/re/impl/namespace_impl.h	
Namespace client stub implementation	2375
l4/re/impl/rm_impl.h	
Region map client stub implementation	2378
l4/re/util/bitmap_cap_alloc	
Bitmap capability allocator	2407
l4/re/util/br_manager	2409
l4/re/util/cap	
Capability utility functions	2411
l4/re/util/cap_alloc	
Capability allocator	2416
l4/re/util/cap_alloc_impl.h	
Capability allocator implementation	2419
l4/re/util/counting_cap_alloc	
Reference-counting capability allocator	2421
l4/re/util/dataspace_svr	2425
l4/re/util/debug	2429
l4/re/util/env_ns	2431
l4/re/util/event	2435
l4/re/util/event_buffer	2437
l4/re/util/event_svr	2438
l4/re/util/icu_svr	2440
l4/re/util/item_alloc	
Item allocator	2442
l4/re/util/kumem_alloc	
Kumem allocator helper	2445
l4/re/util/meta	2706
l4/re/util/name_space_svr	2446
l4/re/util/object_registry	2449
l4/re/util/poll_timeout_kipclock	2452
l4/re/util/region_mapping	
Region handling	2452
l4/re/util/region_mapping_svr_2	2459
l4/re/util/shared_cap	
Shared_cap / Shared_del_cap	2464
l4/re/util/unique_cap	
Unique_cap / Unique_del_cap	2469
l4/re/util/vcon_svr	2472
l4/re/util/video/goos_fb	2473
l4/re/util/video/goos_svr	2474
l4/re/video/colors	2476
l4/re/video/goos	2477
l4/re/video/goos-sys.h	
Goos protocol definition	2480
l4/re/video/view	2482
l4/shmc/ringbuf.h	2482
l4/shmc/shmc.h	
Shared memory library header file	2491

l4/sigma0/sigma0.h	
Sigma0 interface	2496
l4/sys/__kernel_object_impl.h	
Low-level kernel debugger functions	2500
l4/sys/__kip-32bit.h	2501
l4/sys/__kip-64bit.h	2502
l4/sys/__ktrace-impl.h	
L4 kernel event tracing	2503
l4/sys/__l4_fpage.h	2506
l4/sys/__task-arm.h	2510
l4/sys/__timeout.h	2510
l4/sys/__typeinfo.h	
Type information handling	2513
l4/sys/__vcpu-arm.h	2525
l4/sys/__vm-arm.h	
Virtualization interface	2526
l4/sys/__vm-svm.h	2528
l4/sys/__vm-vmx.h	2530
l4/sys/arm_smccc	
ARM secure monitor call functions	2534
l4/sys/arm_smccc.h	
ARM secure monitor call functions	2536
l4/sys/assert.h	
Low-level assert implementation	2775
l4/sys/cache.h	
Cache-consistency functions	2543
l4/sys/capability	
L4::Cap related definitions	2547
l4/sys/compiler.h	
L4 compiler related defines	2551
l4/sys/consts.h	
Common constants	2559
l4/sys/debugger	
The debugger interface specifies common debugging related definitions	2627
l4/sys/debugger.h	
Debugger related definitions	2629
l4/sys/err.h	
Error codes	2633
l4/sys/exception	
Exception C++ interface	2635
l4/sys/factory	
Common factory related definitions	2637
l4/sys/factory.h	
Common factory related definitions	2641
l4/sys/icu	
Interrupt controller	2647
l4/sys/icu.h	
Interrupt controller	2648
l4/sys/iommu	2654
l4/sys/ipc.h	
Common IPC interface	2657
l4/sys/ipc_gate	
The C++ IPC gate interface	2664
l4/sys/ipc_gate.h	
The C IPC gate interface, see L4::lpc_gate for the C++ interface	2666
l4/sys/irq	
C++ Irq interface	2670

l4/sys/irq.h	
C Irq interface	2179
l4/sys/kdebug.h	
Functionality for invoking the kernel debugger	2673
l4/sys/kernel_object.h	
Kernel object system calls	2691
l4/sys/kip	2693
l4/sys/kip.h	
Kernel Info Page access functions	2818
l4/sys/kobject	2695
l4/sys/ktrace.h	
L4 kernel event tracing	2696
l4/sys/l4int.h	
Fixed sized integer types, generic version	2700
l4/sys/memdesc.h	
Memory description functions	2702
l4/sys/meta	
Meta interface for getting dynamic type information about objects behind capabilities	2706
l4/sys/pager	
Pager and lo_pager C++ interface	2708
l4/sys/platform_control	
Platform control object	2710
l4/sys/platform_control.h	
Platform control object	2712
l4/sys/rcv_endpoint	
The C++ Receive endpoint interface	2716
l4/sys/rcv_endpoint.h	
Receive endpoint C interface	2718
l4/sys/scheduler	
Scheduler object functions	2721
l4/sys/scheduler.h	
Scheduler object functions	2723
l4/sys/semaphore	
Semaphore class definition	2727
l4/sys/semaphore.h	
C semaphore interface	2729
l4/sys/smart_capability	
L4::Capability class	2732
l4/sys/task	
Common task related definitions	2735
l4/sys/task.h	
Common task related definitions	2738
l4/sys/thread	
Common thread related definitions	2744
l4/sys/thread.h	
Common thread related definitions	2846
l4/sys/typeinfo_svr	
Type information server template	2748
l4/sys/types.h	
Common L4 ABI Data Types	2282
l4/sys/utcb.h	
UTCB definitions	2755
l4/sys/vcon	
C++ Virtual console interface	2762
l4/sys/vcon.h	
Virtual console interface	2764
l4/sys/vcpu.h	
VCPU API	2878

l4/sys/vhw.h	
Descriptors for virtual hardware (under UX)	2770
l4/sys/vm	
Virtualization interface	2773
l4/sys/cxx/capability.h	2564
l4/sys/cxx/consts	2567
l4/sys/cxx/ipc_array	2568
l4/sys/cxx/ipc_basics	2572
l4/sys/cxx/ipc_client	2575
l4/sys/cxx/ipc_epiface	2578
l4/sys/cxx/ipc_iface	
Interface Definition Language	2582
l4/sys/cxx/ipc_legacy	2593
l4/sys/cxx/ipc_ret_array	2594
l4/sys/cxx/ipc_server	2598
l4/sys/cxx/ipc_server_loop	2601
l4/sys/cxx/ipc_string	2604
l4/sys/cxx/ipc_types	2606
l4/sys/cxx/ipc_varg	2614
l4/sys/cxx/smart_capability_1x	2620
l4/sys/cxx/types	2623
l4/util/assert.h	
Some useful assert-style macros	2778
l4/util/atomic.h	
Atomic operations header and generic implementations	2783
l4/util/backtrace.h	
Backtrace	2790
l4/util/base64.h	
Base 64 encoding and decoding functions adapted from Bob Trower 08/04/01	2792
l4/util/bitops.h	
Bit manipulation functions	2793
l4/util/elf.h	
ELF definition	2799
l4/util/getopt.h	
Getopt	2815
l4/util/keymap.h	
Event to ASCII key mapping	2817
l4/util/kip.h	2822
l4/util/kprintf.h	
Printf using the kernel debugger	2824
l4/util/l4_macros.h	
Some useful generic macros, L4f version	2037
l4/util/l4mod.h	
L4mod structures and constants	2825
l4/util/list_alloc.h	
Simple list-based allocator	2827
l4/util/lulc.h	2829
l4/util/lock.h	
Simple lock implementation	2830
l4/util/mb_info.h	
Multiboot info structure as defined by GRUB	2832
l4/util/parse_cmd.h	
Comfortable command-line parsing	2840
l4/util/rand.h	
Simple Pseudo-Random Number Generator	2841
l4/util/splitlog2.h	
Split a range in log2 aligned and size-aligned chunks	2843

l4/util/thread.h	
Low-level Thread Functions	2855
l4/util/util.h	1970
l4/vbus/vbus	2857
l4/vbus/vbus.h	
Description of the vbus C API	2858
l4/vbus/vbus_generic	2862
l4/vbus/vbus_gpio	2862
l4/vbus/vbus_gpio-ops.h	2864
l4/vbus/vbus_gpio.h	2864
l4/vbus/vbus_i2c.h	2865
l4/vbus/vbus_inhibitor.h	2865
l4/vbus/vbus_interfaces.h	
This header contains the definition of VBUS sub-interfaces and convenience functions to work with the interface IDs	2865
l4/vbus/vbus_mcspi.h	2869
l4/vbus/vbus_pci	2869
l4/vbus/vbus_pci-ops.h	2870
l4/vbus/vbus_pci.h	2871
l4/vbus/vbus_pm-ops.h	2871
l4/vbus/vbus_pm.h	2871
l4/vbus/vbus_types.h	
This header file contains descriptions of vbus related data types and constants	2872
l4/vbus/vdevice-ops.h	2875
l4/vcpu/vcpu	
VCPU support library (C++ interface)	2876
l4/vcpu/vcpu.h	
VCPU support library (C interface)	2881
x86/l4/sys/__kip-arch.h	1998
x86/l4/sys/__vcpu-arch.h	
X86-specific vCPU interface	2004
x86/l4/sys/cache.h	
Cache functions	2546
x86/l4/sys/consts.h	
Common L4 constants, x86 version	2563
x86/l4/sys/ipc-invoke.h	2885
x86/l4/sys/ktrace_events.h	2012
x86/l4/sys/l4int.h	
Fixed sized integer types, x86 version	2701
x86/l4/sys/linkage.h	
Linkage	2017
x86/l4/sys/segment.h	
Segment handling	1983
x86/l4/sys/utcb.h	
UTCB definitions for X86	2759
x86/l4/sys/vm.h	2023
x86/l4/util/bitops_arch.h	
X86 bit manipulation functions	2027
x86/l4/util/cpu.h	
CPU related functions	2033
x86/l4/util/idt.h	
IDT related functions	1942
x86/l4/util/irq.h	
Some PIC and hardware interrupt related functions	2183
x86/l4/util/l4_macros.h	
Main function	2037
x86/l4/util/mbi_argv.h	
Command line handling	2041

x86/l4/util/ perform.h	
Performance Monitoring using P5/P6 Measurement Counters	1950
x86/l4/util/ port_io.h	
X86 port I/O	1990
x86/l4/util/ rdtsc.h	
Timestamp counter related functions	1960
x86/l4/util/ spin.h	
Spinning for x86	1966
x86/l4/util/ util.h	
Utilities for x86	1971
x86/l4f/l4/sys/ ipc-l42-gcc3-nopic.h	2885
x86/l4f/l4/sys/ ipc.h	
L4 IPC System Calls, x86	2663
x86/l4f/l4/sys/ segment.h	
L4f specific segment manipulation	1987
x86/l4f/l4/util/ port_io.h	
Port I/O functions	1994

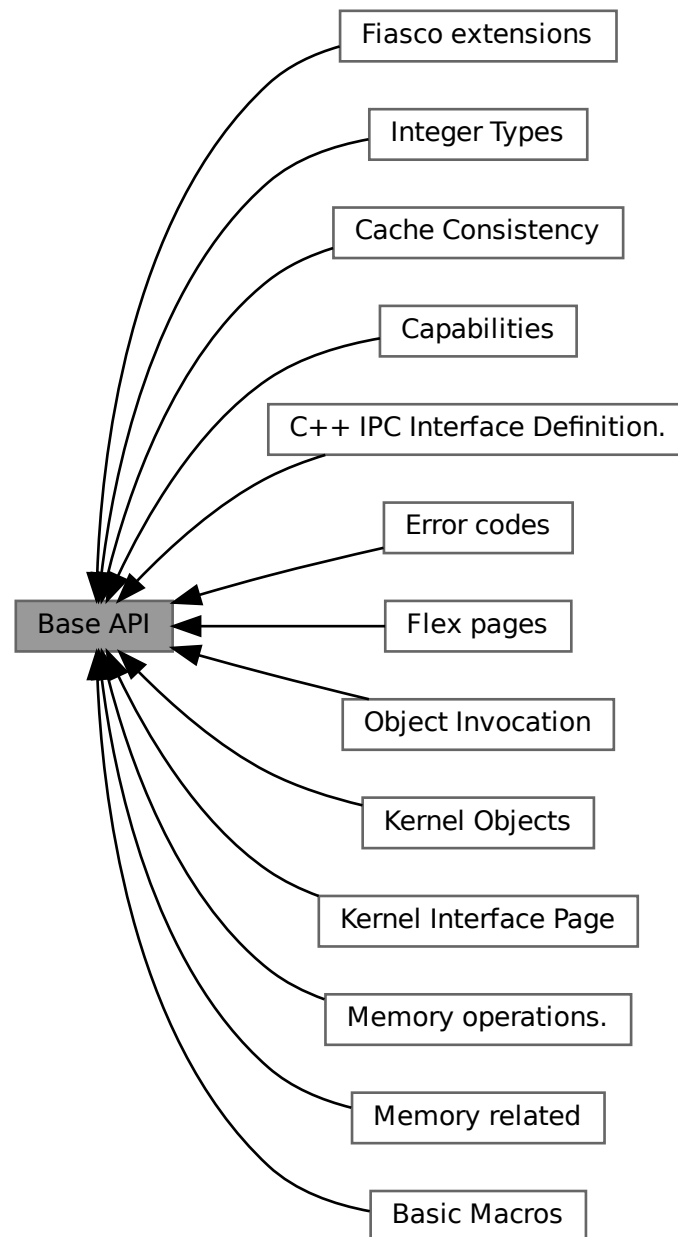
Chapter 13

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_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 221 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 218 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 274 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 342 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

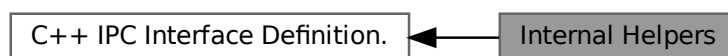
- namespace [L4::Typeid](#)
Definition of interface data-type helpers.

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

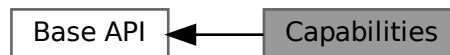
Implementations may choose to write back dirty lines nonetheless if this is more efficient.

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 [358](#) 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

L4_CAP_SHIFT	Capability index shift.
L4_CAP_SIZE	
Generated for L4Re by Doxygen	Deprecated Superseded by L4_CAP_OFFSET .
L4_CAP_OFFSET	Offset of two consecutive capability selectors.
L4_CAP_MASK	Mask to get only the relevant bits of an l4_cap_idx_t .

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 419 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 411 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
----------------	---------------------

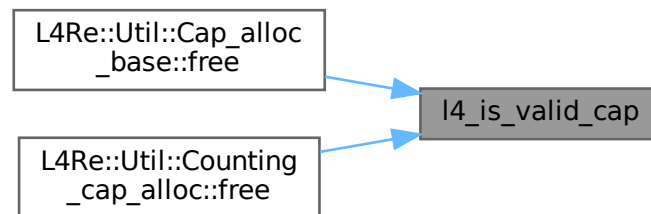
Return values

<code>0</code>	The capability selector is not valid.
<code>>0</code>	The capability selector is valid.

Definition at line 415 of file [types.h](#).

Referenced by [L4Re::Util::Cap_alloc_base::free\(\)](#), and [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::free\(\)](#).

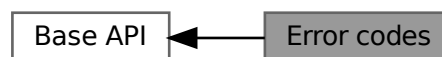
Here is the caller graph for this function:



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_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

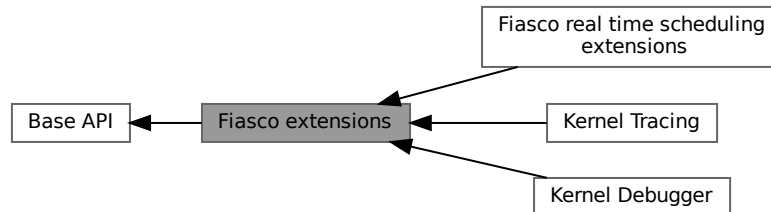
L4_EOK	Ok.
L4_EPERM	No permission.
L4_ENOENT	No such entity.
L4_EIO	I/O error.
L4_ENXIO	No such device or address.
L4_E2BIG	Argument value too big.
L4_EAGAIN	Try again.
L4_ENOMEM	No memory.
L4_EACCESS	Permission denied.
L4_EFAULT	Invalid memory address.
L4_EBUSY	Object currently busy, try later.
L4_EEXIST	Already exists.
L4_ENODEV	No such thing.
L4_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.
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.

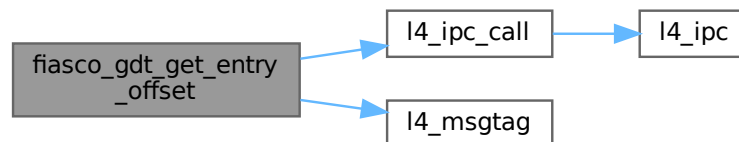
Parameters

<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 l4_utcb .

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 l4_utcb .

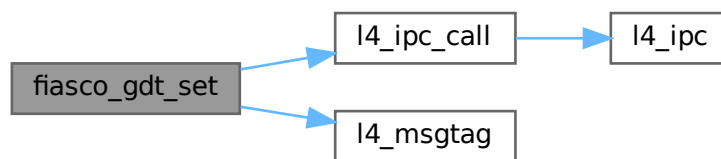
Return values

<0	At least one provided GDT descriptor is considered unsafe by the kernel, and not all selected GDT descriptors have been updated.
<i>L4_EOK</i>	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 l4_utcb .

Return values

<code>-L4_ENOSYS</code>	The kernel configuration doesn't support this feature.
<code>-L4_EINVAL</code>	Invalid descriptor or invalid entry number.
<code>L4_EOK</code>	Success.

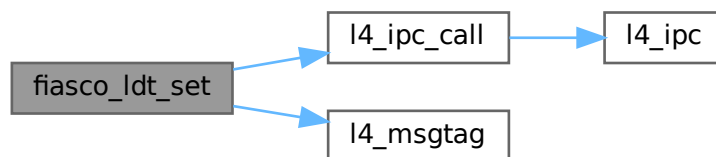
Note

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.

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_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 386 of file `debugger.h`.

References `l4_utcb()`.

Here is the call graph for this function:



13.1.7.4.2.2 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

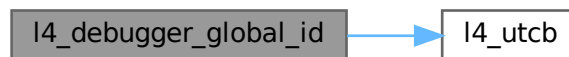
<i>~0UL</i>	Capability is not valid.
<i>otherwise</i>	Global debugger id.

This is a debugging facility, the call might be invalid.

Definition at line 351 of file [debugger.h](#).

References [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.7.4.2.3 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

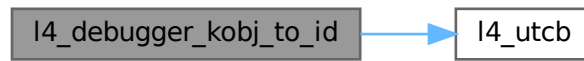
<i>~0UL</i>	The capability or the kobject pointer are invalid.
<i>otherwise</i>	The globally unique id.

This is a debugging facility, the call might be invalid.

Definition at line 357 of file [debugger.h](#).

References [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.7.4.2.4 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 370 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_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

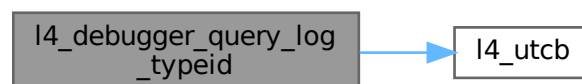
positive ID, or negative error code

This is a debugging facility, the call might be invalid.

Definition at line 363 of file [debugger.h](#).

References [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.7.4.2.6 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.

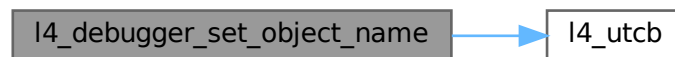
Examples

[examples/libs/shmc/prodcons.c](#), and [examples/sys/aliens/main.c](#).

Definition at line 344 of file [debugger.h](#).

References [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.7.4.2.7 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

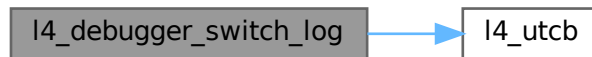
Returns

Syscall return tag

Definition at line 379 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
-------------	--------------

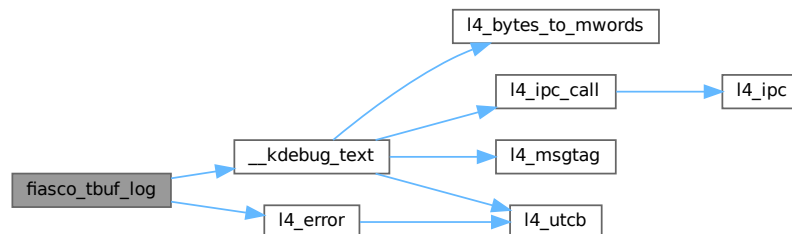
Returns

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

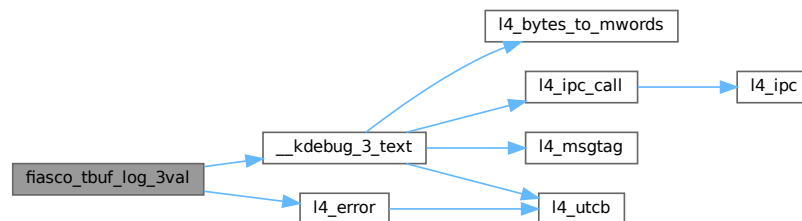
Returns

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
-------------	-------------

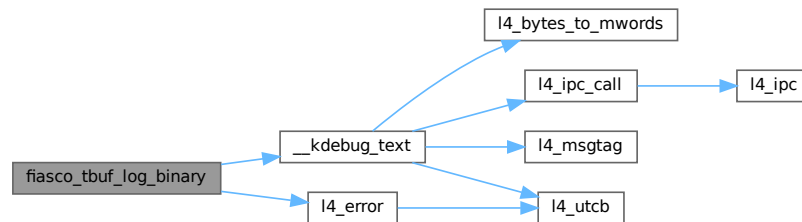
Returns

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 L4_WHOLE_ADDRESS_SPACE , 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 L4::Task::map() , the 'S' right is only transferred if the flex-page, the source and destination task capabilities have the 'S' right.

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 L4_CAP_FPAGE_W
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 L4_CAP_FPAGE_S
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 L4_CAP_FPAGE_R , L4_CAP_FPAGE_W , and L4_CAP_FPAGE_S
L4_CAP_FPAGE_RWSD	Full rights for capability flex-pages. See also L4_CAP_FPAGE_R , L4_CAP_FPAGE_W , L4_CAP_FPAGE_S , and L4_CAP_FPAGE_D
L4_CAP_FPAGE_RWD	Read, write, and delete right for capability flex-pages. See also L4_CAP_FPAGE_R , L4_CAP_FPAGE_W , and L4_CAP_FPAGE_D
L4_CAP_FPAGE_RSD	Read, 'S', and delete right for capability flex-pages. See also L4_CAP_FPAGE_R , L4_CAP_FPAGE_S , and L4_CAP_FPAGE_D

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), L4_WHOLE_ADDRESS_SPACE to specify the whole address space (with <code>address</code> 0). The minimum log2 size of a memory flex page is defined by L4_LOG2_PAGESIZE according to the size of the smallest virtual page supported by the MMU.
<i>rights</i>	Access rights, see L4_fpage_rights

Returns

Memory flex page

Definition at line 668 of file [__l4_fpage.h](#).

References [L4_FPAGE_MEMORY](#).

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.

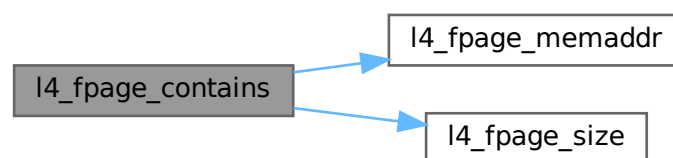
Return values

<code>==0</code>	The range is not completely in the fpage.
<code>!=0</code>	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.

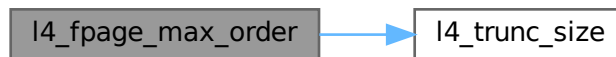
Note

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

<code>f</code>	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

<i>f</i>	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

<i>f</i>	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), L4_WHOLE_IOADDRESS_SPACE 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 L4_cap_fpage_rights

Returns

Flex page for a set of kernel objects.

Note

[L4_CAP_FPAGE_R](#) is always required, otherwise no capability is mapped.

Examples

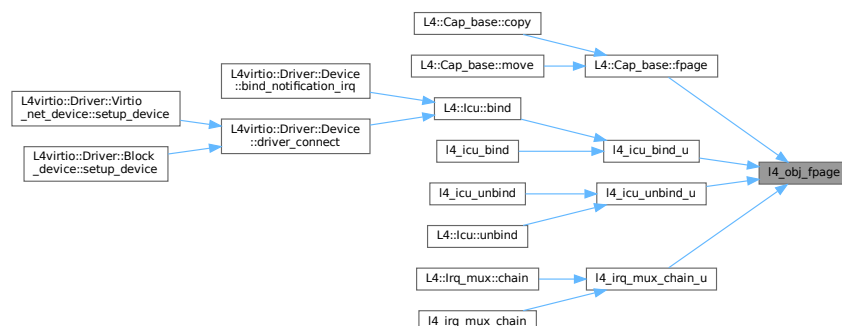
[examples/sys/utcb-ipc/main.c](#).

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

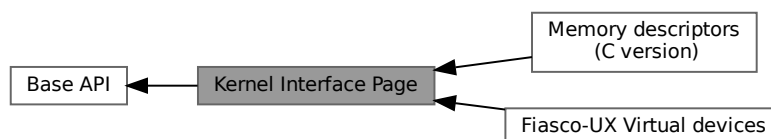
Include File

```
#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_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.

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_OFFSET_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_OFFSET_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.

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).
------------	--

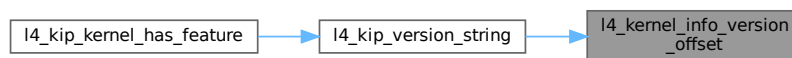
Returns

offset of version_strings relative to the KIP base address, in bytes.

Definition at line 211 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.

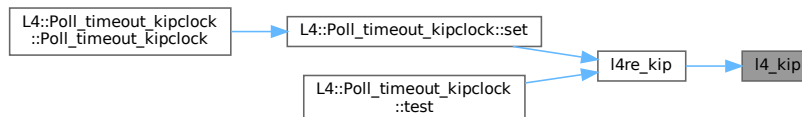
Returns

Pointer to Kernel Info Page (KIP) structure.

Definition at line 192 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

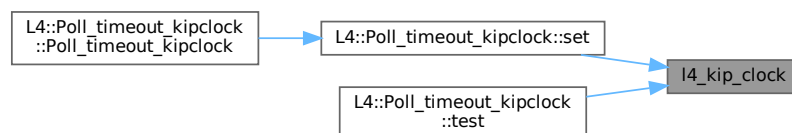
[examples/libs/shmc/prodcons.c](#).

Definition at line 215 of file [kip.h](#).

References [L4_KIP_OFFS_READ_US](#).

Referenced by [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.

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 234 of file [kip.h](#).

References [l4_mb\(\)](#).

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 225 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 203 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

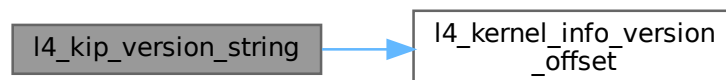
Kernel version string.

Definition at line 207 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:



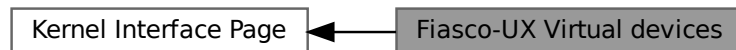
Here is the caller graph for this function:



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

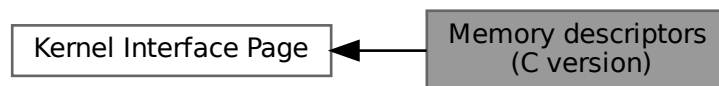
L4_TYPE_VHW_NONE	None entry.
L4_TYPE_VHW_FRAMEBUFFER	Framebuffer device.
L4_TYPE_VHW_INPUT	Input device.
L4_TYPE_VHW_NET	Network device.

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 l4_mem_archspecific_sub_type_common_t

```
enum l4_mem_archspecific_sub_type_common_t
```

Memory sub types for l4_mem_type_archspecific descriptors.

Enumerator

<code>I4_mem_archspecific_acpi_tables</code>	Firmware ACPI tables.
<code>I4_mem_archspecific_acpi_nvs</code>	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

<code>I4_mem_info_acpi_rsdp</code>	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

<code>I4_mem_type_undefined</code>	Undefined, unused descriptor.
<code>I4_mem_type_conventional</code>	Conventional memory.
<code>I4_mem_type_reserved</code>	Reserved memory for kernel etc.
<code>I4_mem_type_dedicated</code>	Dedicated memory (some device memory)
<code>I4_mem_type_shared</code>	Shared memory (not implemented)
<code>I4_mem_type_info</code>	Info from the boot loader.
<code>I4_mem_type_bootloader</code>	Memory owned by the boot loader.
<code>I4_mem_type_archspecific</code>	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 I4_kernel_info_get_mem_desc_end()**

```
I4_umword_t I4_kernel_info_get_mem_desc_end (
    I4_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

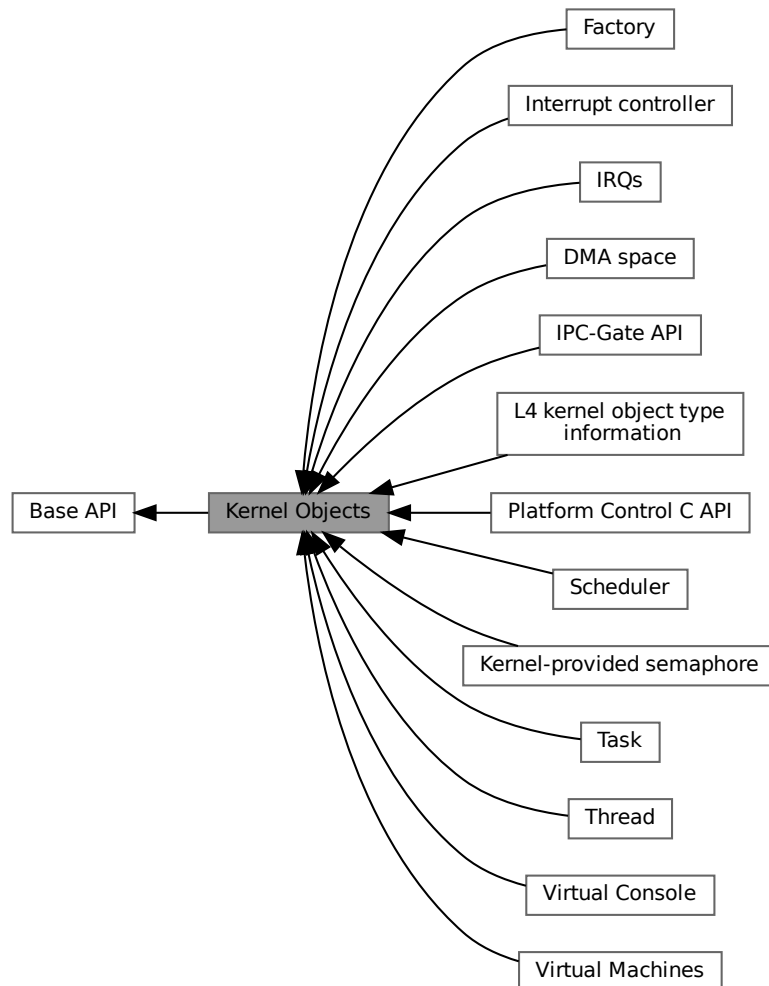
<i>md</i>	Pointer to memory descriptor
<i>start</i>	Start of region
<i>end</i>	End of region
<i>type</i>	Type of region
<i>virt</i>	1 if virtual region, 0 if physical region
<i>sub_type</i>	Sub type.

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 (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.

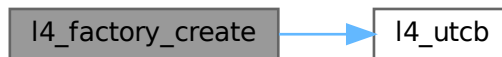
Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	The factory instance requires L4_CAP_FPAGE_S rights on <code>factory</code> and L4_CAP_FPAGE_S is not present.
<i><0</i>	Error code.

Definition at line 591 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 L4_CAP_FPAGE_S rights on <code>factory</code> and L4_CAP_FPAGE_S is not present.
<i><0</i>	Error code.

Note

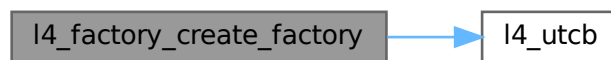
The limit of the new factory is subtracted from the available amount of the factory used for creation.

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 445 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 L4_INVALID_CAP 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 L4_CAP_FPAGE_S 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\(\)](#).

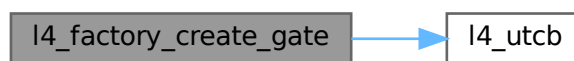
See also

[IPC-Gate API](#)

Definition at line 453 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 L4_CAP_FPAGE_S rights on <i>factory</i> and L4_CAP_FPAGE_S is not present.
<i><0</i>	Error code.

See also

[IRQs](#)

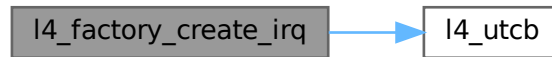
Examples

[examples/sys/isr/main.c](#).

Definition at line 461 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.
	<i>utcb_area</i>	Flexpage that describes an area of kernel-user memory that can be used for UTCBs and vCPU state-save-areas of the new task.

Returns

Syscall return tag.

Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	The factory instance requires L4_CAP_FPAGE_S rights on <i>factory</i> and L4_CAP_FPAGE_S is not present.
<i><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.

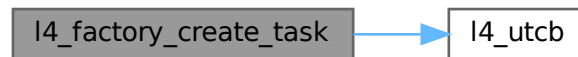
See also

[Task](#)

Definition at line 431 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 L4_CAP_FPAGE_S rights on <i>factory</i> and L4_CAP_FPAGE_S is not present.
<i><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 438 of file [factory.h](#).

References [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.11.3.2.7 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 L4_CAP_FPAGE_S rights on <i>factory</i> and L4_CAP_FPAGE_S is not present.
<i>< 0</i>	Error code.

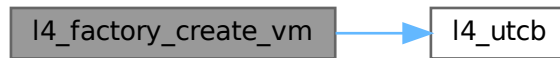
See also

[Virtual Machines](#)

Definition at line 468 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.

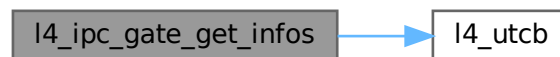
Precondition

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 L4_CAP_FPAGE_S 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](#).

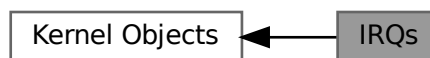
References [l4_utcb\(\)](#).

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 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

-L4_EPERM	No L4_CAP_FPAGE_S rights on the capability used to invoke this operation. -L4_EBUSY A detach operation on this IRQ object by another thread was in progress.
-----------	--

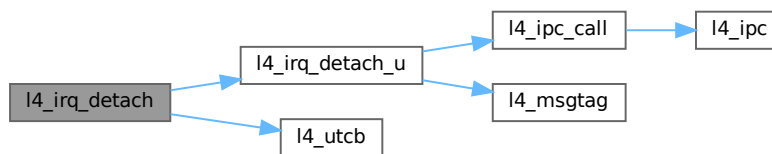
Examples

[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 l4_utcb .

Returns

Syscall return tag

Return values

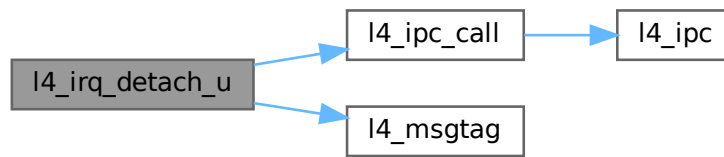
-L4_EPERM	No L4_CAP_FPAGE_S rights on the capability used to invoke this operation. -L4_EBUSY A detach operation on this IRQ object by another thread was in progress.
------------------	---

Definition at line 254 of file [irq.h](#).

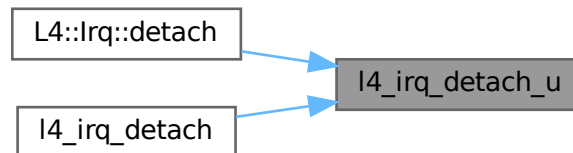
References [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_PROTO_IRQ_SENDER](#), and [l4_msg_regs_t::mr](#).

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

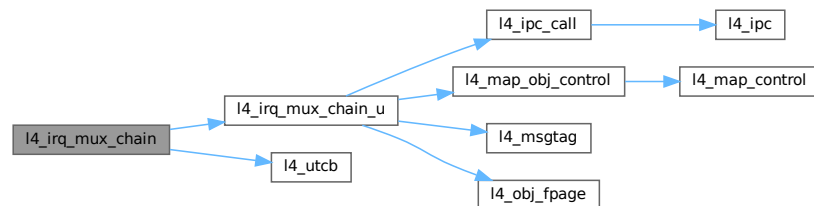
The chaining feature of IRQ objects allows to deal with shared IRQs. For chaining IRQs there must be a master IRQ object, bound to the real IRQ source. Note, the master IRQ must not have a thread bound to it.

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 l4_utcb .

Returns

Syscall return tag

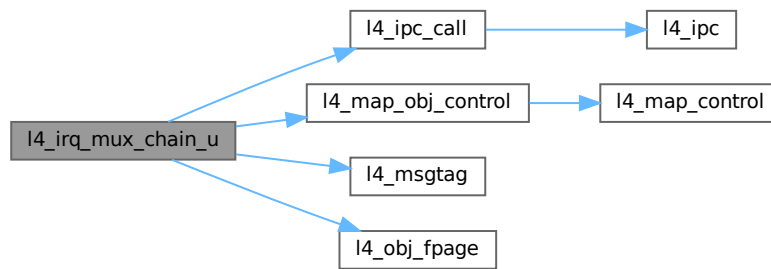
The chaining feature of IRQ objects allows to deal with shared IRQs. For chaining IRQs there must be an IRQ multiplexer ([l4_irq_mux](#)) bound to the real IRQ source. This function allows to add slave IRQs to this multiplexer.

Definition at line 242 of file [irq.h](#).

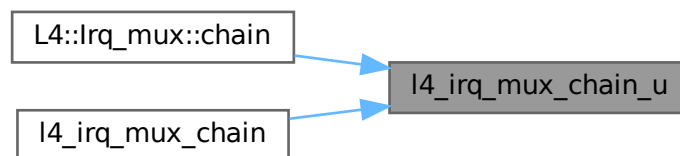
References [L4_CAP_FPAGE_RWS](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_map_obj_control\(\)](#), [l4_msgtag\(\)](#), [l4_obj_fpage\(\)](#), [L4_PROTO_IRQ_MUX](#), [l4_msg_regs_t::mr](#), and [l4_fpage_t::raw](#).

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

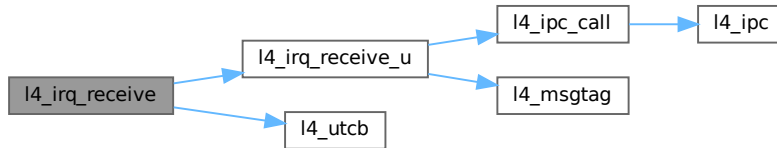
Syscall return tag

Definition at line 311 of file `irq.h`.

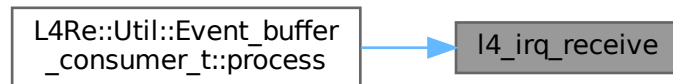
References `l4_irq_receive_u()`, and `l4_utcb()`.

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 l4_utcb .

Returns

Syscall return tag

Note

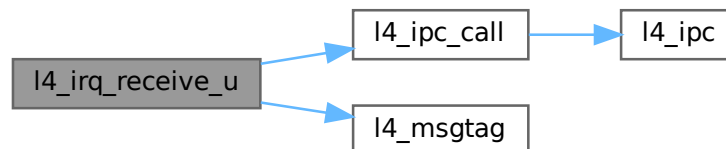
If this is the function normally used for your IRQs consider using [L4::Semaphore](#) instead of [L4::Irq](#).

Definition at line 269 of file [irq.h](#).

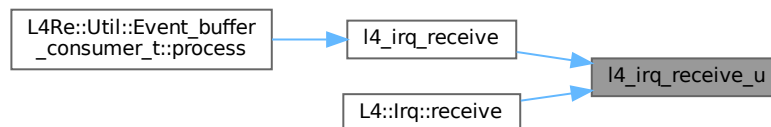
References [l4_ipc_call\(\)](#), [l4_msgtag\(\)](#), [L4_PROTO_IRQ](#), and [l4_msg_regs_t::mr](#).

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.

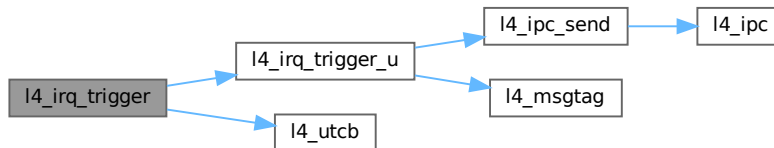
Use [l4_ipc_error\(\)](#) to check for (send) errors.

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 l4_utcb .

Returns

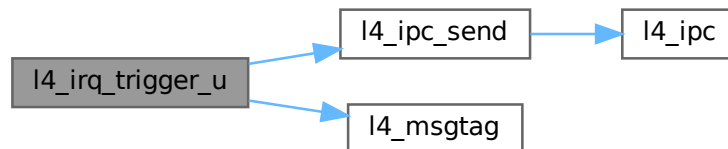
Syscall return tag for a send-only operation, this means there is no return value except [L4_MSGTAG_ERROR](#) indicating success or failure of the send operation. Use [l4_ipc_error\(\)](#) to check for errors and **do not** use [l4_error\(\)](#).

Definition at line 262 of file [irq.h](#).

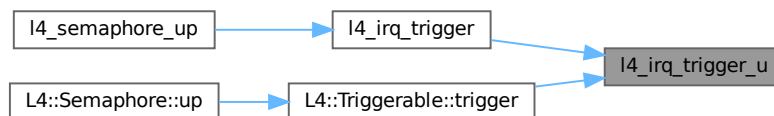
References [L4_IPC_BOTH_TIMEOUT_0](#), [l4_ipc_send\(\)](#), [l4_msgtag\(\)](#), and [L4_PROTO_IRQ](#).

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

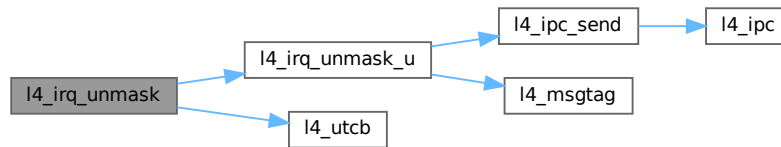
Note

[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 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 l4_utcb .

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.

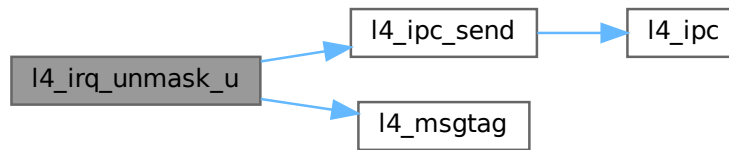
Deprecated Use `L4::Irq_eoi::unmask()`

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

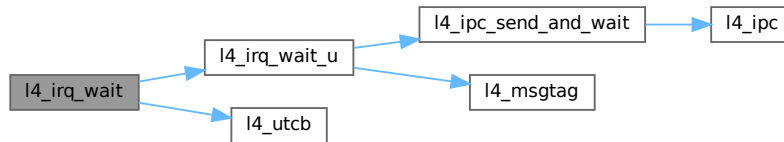
Examples

[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 l4_utcb .

Returns

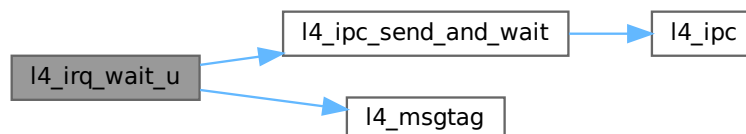
Syscall return tag

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:



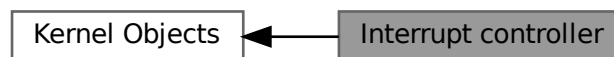
Here is the caller graph for this function:



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 l4_icu_bind() and l4_icu_unbind() 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	<i>irq</i> is bound to an interrupt source.
-L4_EPERM	The ICU instance requires L4_CAP_FPAGE_W on <i>irq</i> and L4_CAP_FPAGE_W 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.

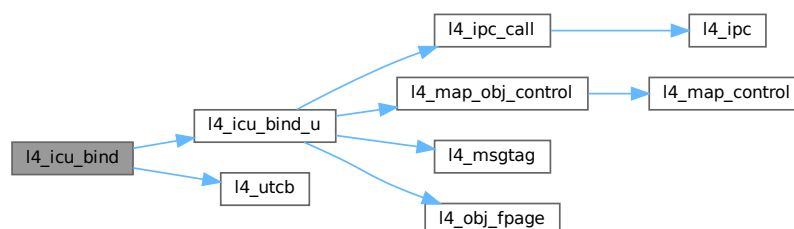
Examples

[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 <i>irq</i> 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 l4_utcb .

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	<i>irq</i> is bound to an interrupt source.
-L4_EPERM	The ICU instance requires L4_CAP_FPAGE_W on <i>irq</i> and L4_CAP_FPAGE_W was not present.

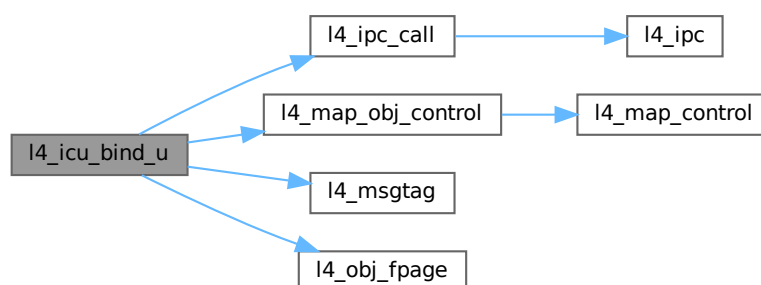
In case the *irq* is already bound to an interrupt source, it is unbound first. In case the *irq* is bound and the interrupt source is bound to a different [L4::irq](#) object, only the unbinding happens. An [L4::irq](#) object that is bound to an interrupt source will get unbound if the [L4::irq](#) object is deleted.

Definition at line 404 of file [icu.h](#).

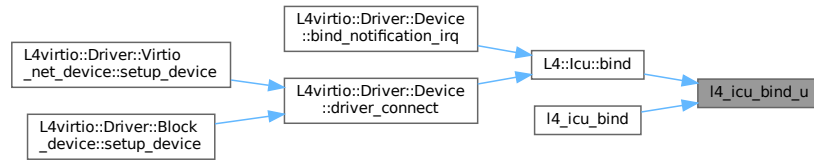
References [L4_CAP_FPAGE_RWS](#), [L4_ICU_OP_BIND](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_map_obj_control\(\)](#), [l4_msgtag\(\)](#), [l4_obj_fpage\(\)](#), [L4_PROTO_IRQ](#), [l4_msg_regs_t::mr](#), and [l4_fpage_t::raw](#).

Referenced by [L4::icu::bind\(\)](#), and [l4_icu_bind\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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.

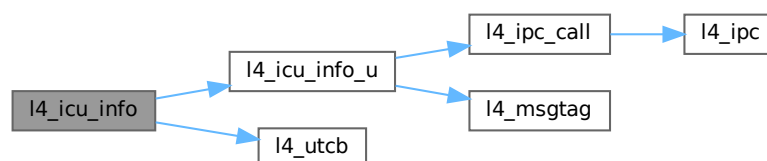
Returns

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 l4_utcb .

Returns

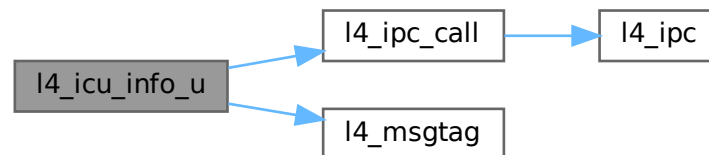
Syscall return tag

Definition at line 428 of file [icu.h](#).

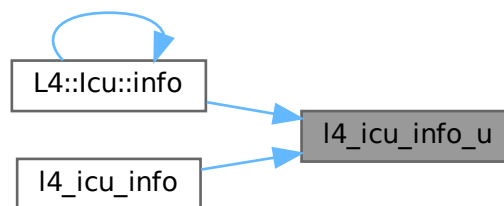
References [L4_ICU_OP_INFO](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_PROTO_IRQ](#), and [l4_msg_regs_t::mr](#).

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 l4_utcb .

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.

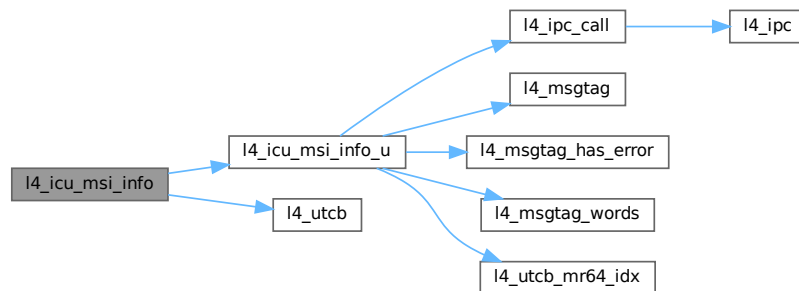
Returns

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 l4_utcb .
	<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 l4_icu_msi_info_t structure receiving the address and the data value to trigger this MSI.

Returns

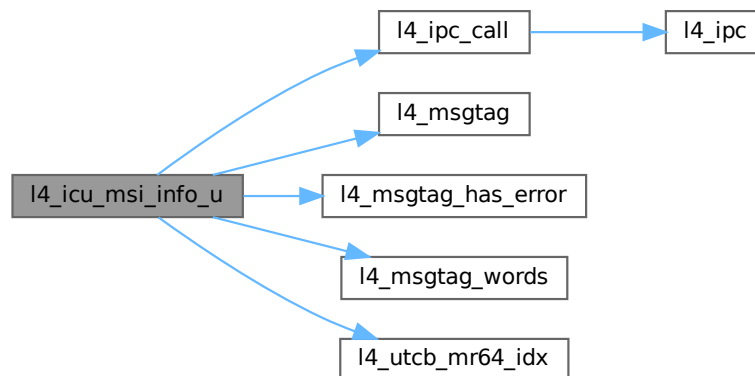
Syscall return tag

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 L4_irq_mode .

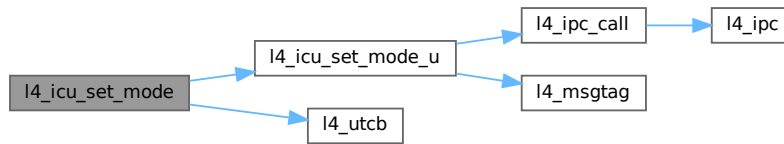
Returns

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 L4_irq_mode .
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb .

Returns

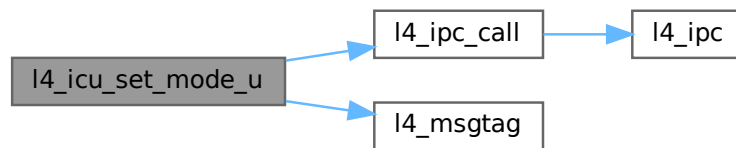
Syscall return tag

Definition at line 465 of file [icu.h](#).

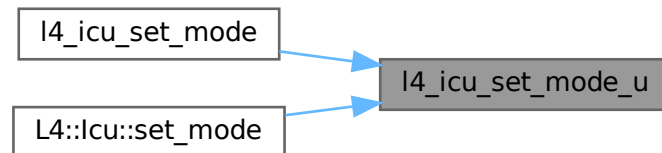
References [L4_ICU_OP_SET_MODE](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_PROTO_IRQ](#), and [l4_msg_regs_t::mr](#).

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.

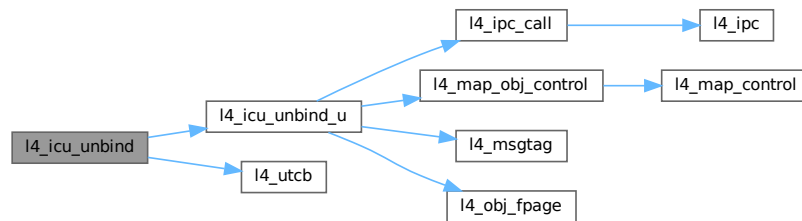
Returns

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 l4_utcb .

Returns

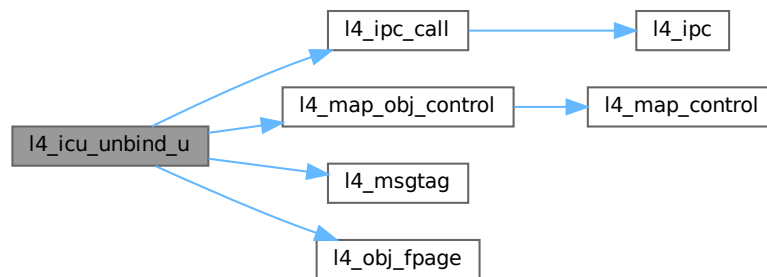
Syscall return tag

Definition at line 416 of file [icu.h](#).

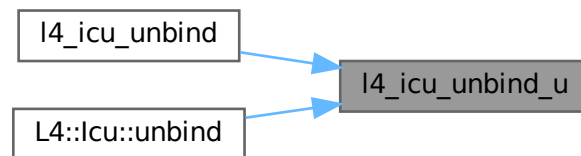
References [L4_CAP_FPAGE_RWS](#), [L4_ICU_OP_UNBIND](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_map_obj_control\(\)](#), [l4_msgtag\(\)](#), [l4_obj_fpage\(\)](#), [L4_PROTO_IRQ](#), [l4_msg_regs_t::mr](#), and [l4_fpage_t::raw](#).

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::l4::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 L4_CAP_FPAGE_S 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

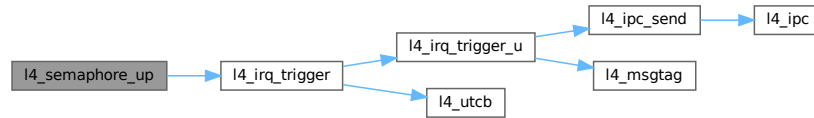
Send-only IPC message return tag. Use [l4_ipc_error\(\)](#) to check for errors, do **not** use [l4_error\(\)](#).

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

T	The type (L4Re interface) for which the information shall be returned.
-------------------	---

Returns

A pointer to the [L4::Type_info](#) structure for T.

Definition at line 692 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_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.

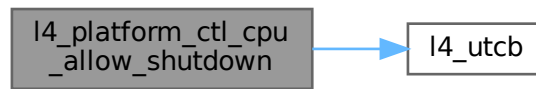
Returns

Syscall return tag

Definition at line 247 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

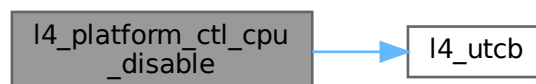
System call message tag

This function is currently only supported on the ARM EXYNOS platform.

Definition at line 286 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

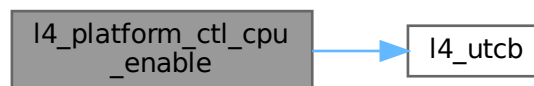
System call message tag

This function is currently only supported on the ARM EXYNOS platform.

Definition at line 279 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_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.

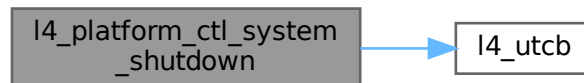
Returns

Syscall return tag

Definition at line 226 of file [platform_control.h](#).

References [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.11.9.2.5 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.

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.

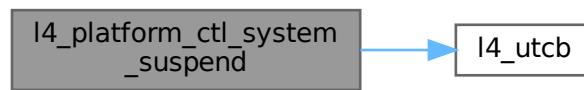
Returns

Syscall return tag

Definition at line 219 of file [platform_control.h](#).

References [l4_utcb\(\)](#).

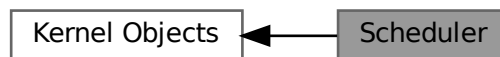
Here is the call graph for this function:



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 μ 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

L4_SCHEDULER_CLASS_FIXED_PRIO	Fixed-priority scheduler.
L4_SCHEDULER_CLASS_WFQ	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

L4_SCHEDULER_INFO_OP	Query infos about the scheduler.
L4_SCHEDULER_RUN_THREAD_OP	Run a thread on this scheduler.
L4_SCHEDULER_IDLE_TIME_OP	Query idle time for the scheduler.

Definition at line 262 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 272 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 282 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 μ 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 μ s.

Return values

0	Success.
-L4_EINVAL	Invalid CPU requested in <code>cpu</code> set.

This function retrieves the idle time in μ 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.

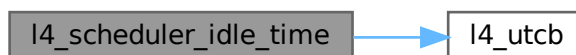
Note

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 396 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>cpus.granularity</i> (see l4_sched_cpu_set_t). <i>cpus.map</i> Bitmap of online CPUs. Pass NULL if this information is not required.

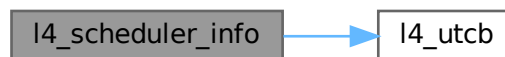
Return values

0	Success.
-L4_ERANGE	The given CPU offset is larger than the maximum number of CPUs.

Definition at line 374 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 l4_sched_cpu_set_t). <i>cpus.map</i> Bitmap of online CPUs. Pass NULL if this information is not required.
out	<i>sched_classes</i>	A bitmap of available scheduling classes (see L4_scheduler_classes). Optional, can be NULL.

Return values

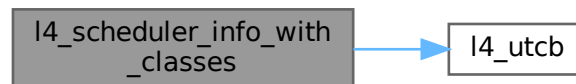
0	Success.
-L4_ERANGE	The given CPU offset is larger than the maximum number of CPUs.

This function delivers the same information as [l4_scheduler_info](#) plus the available scheduler classes (see [L4_scheduler_classes](#)).

Definition at line 381 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.

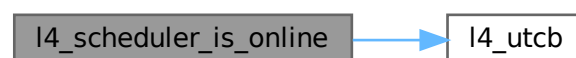
Return values

<i>true</i>	The CPU is online.
<i>false</i>	The CPU is offline

Definition at line 403 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 389 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

L4_FP_ALL_SPACES	<p>Flag to tell the unmap operation to revoke permissions from all child mappings including the mapping in the invoked task.</p> <p>Note</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>See also</p> <p>L4::Task::unmap() l4_task_unmap()</p>
L4_FP_DELETE_OBJ	<p>Flag that indicates that an unmap operation on object capabilities shall try to delete the corresponding objects immediately. This flag implies the L4_FP_ALL_SPACES 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 L4_FP_ALL_SPACES alone.</p> <p>See also</p> <p>L4::Task::unmap() l4_task_unmap()</p>
L4_FP_OTHER_SPACES	<p>Counterpart to L4_FP_ALL_SPACES; revoke permissions from child mappings only.</p> <p>See also</p> <p>L4::Task::unmap() l4_task_unmap()</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>ku_mem</i>	Flexpage describing the virtual area the memory goes to.

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

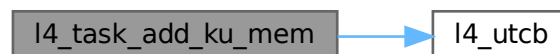
The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page (`L4_PAGE_SIZE`). A portable implementation should not depend on allocations greater than 16KiB to succeed.

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 459 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_a</i>	Capability selector to compare
<i>cap_b</i>	Capability selector to compare

Generated for L4Re by Doxygen

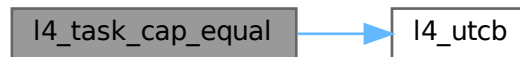
Returns

label contains 1 if equal, 0 if not equal

Definition at line 452 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.

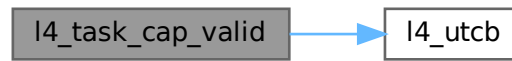
Precondition

cap must be a valid capability index (i.e. not L4_INVALID_CAP or the like).

Definition at line 446 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.

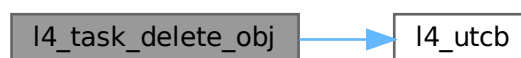
If *obj* does not have the delete permission, no error will be reported and only the capability *obj* is removed. (Note that, depending on the object's reference counter, this might still imply initiation of deletion.)

This operation is equivalent to [l4_task_unmap\(\)](#) with [L4_FP_DELETE_OBJ](#) flag.

Definition at line [425](#) 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 l4_fpage_cacheability_opt_t for memory mappings, L4_obj_fpage_ctl for object mappings, and L4_MAP_ITEM_GRANT ; also see l4_map_control() and l4_map_obj_control()).

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 L4_CAP_FPAGE_W 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

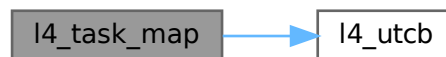
For peculiarities when using grant, see [L4_MAP_ITEM_GRANT](#).

Definition at line 395 of file [task.h](#).

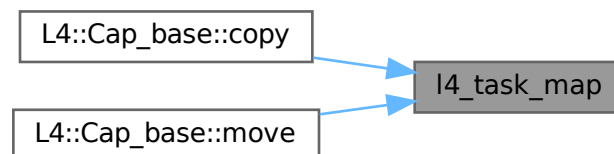
References [l4_utcb\(\)](#).

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.

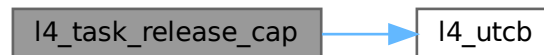
Note

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 440 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 l4_unmap_flags_t

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 402 of file [task.h](#).

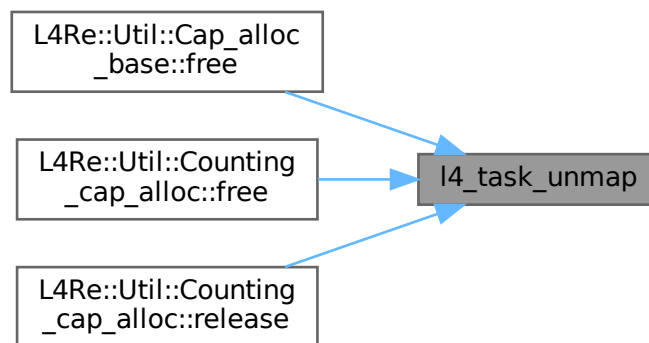
References [l4_utcb\(\)](#).

Referenced by [L4Re::Util::Cap_alloc_base::free\(\)](#), [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::free\(\)](#), and [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::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 l4_unmap_flags_t

Returns

Syscall return tag

Revoke rights specified in an array of flexpages, see [l4_task_unmap](#) for details.

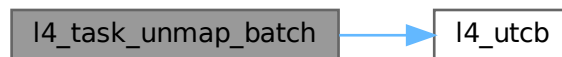
Precondition

The caller needs to take care that `num_fpages` is not bigger than `L4_UTCB_GENERIC_DATA_SIZE - 2`.

Definition at line [409](#) 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

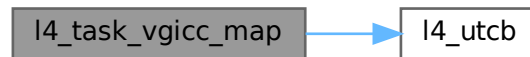
Returns

Syscall return tag

Definition at line 57 of file [__task-arm.h](#).

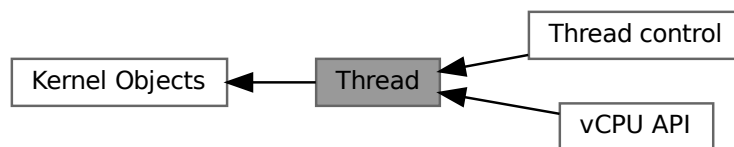
References [l4_utcb\(\)](#).

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 }

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.

- `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 μ 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 718 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 L4_thread_control_flags .
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 741 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.

Definition at line 756 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

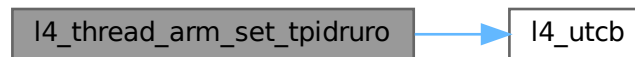
Returns

System call return tag

Definition at line 59 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 L4_thread_ex_regs_flags .

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.

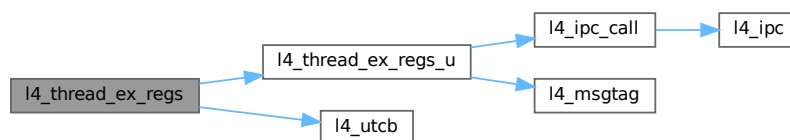
Examples

[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 907 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 <code>~0UL</code> to leave the instruction pointer unchanged, return previous instruction pointer.
in, out	<i>sp</i>	New stack pointer, use <code>~0UL</code> to leave the stack pointer unchanged, returns previous stack pointer.
in, out	<i>flags</i>	Ex-regs flags, see L4_thread_ex_regs_flags , 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.

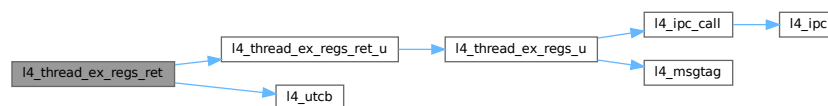
The thread is started using `l4_scheduler_run_thread()`. However, if at the time `l4_scheduler_run_thread()` is called, the instruction pointer of the thread is invalid, a later call to `l4_thread_ex_regs()` with a valid instruction pointer might start the thread.

Returned values are valid only if function returns successfully.

Definition at line 914 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 <code>~0UL</code> to leave the instruction pointer unchanged, return previous instruction pointer.
in, out	<i>sp</i>	New stack pointer, use <code>~0UL</code> to leave the stack pointer unchanged, returns previous stack pointer.
in, out	<i>flags</i>	Ex-regs flags, see L4_thread_ex_regs_flags , 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 l4_utcb .

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.

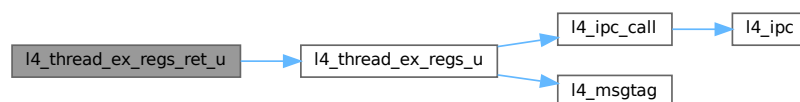
The thread is started using [L4::Scheduler::run_thread\(\)](#). However, if at the time [L4::Scheduler::run_thread\(\)](#) is called, the instruction pointer of the thread is invalid, a later call to [ex_regs\(\)](#) with a valid instruction pointer might start the thread.

Definition at line 780 of file [thread.h](#).

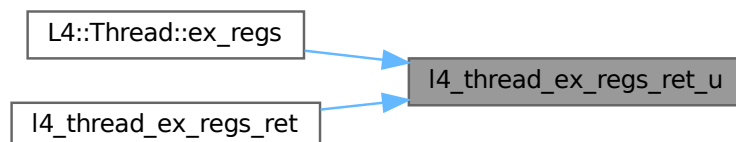
References [l4_thread_ex_regs_u\(\)](#), and [l4_msg_regs_t::mr](#).

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 L4_thread_ex_regs_flags .
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb .

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.

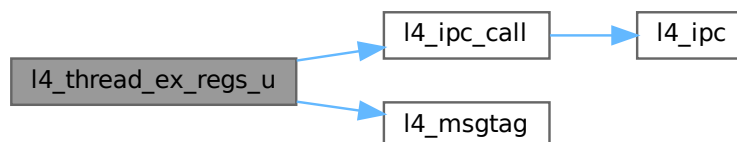
The thread is started using [L4::Scheduler::run_thread\(\)](#). However, if at the time [L4::Scheduler::run_thread\(\)](#) is called, the instruction pointer of the thread is invalid, a later call to `ex_regs()` with a valid instruction pointer might start the thread.

Definition at line 769 of file [thread.h](#).

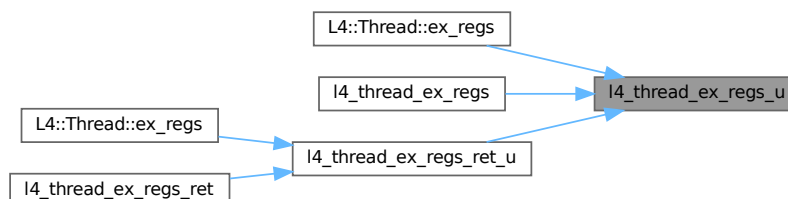
References [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_PROTO_THREAD](#), [L4_THREAD_EX_REGS_OP](#), and [l4_msg_regs_t::mr](#).

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 l4_thread_modify_sender_start() or previous l4_thread_modify_sender_add() 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.

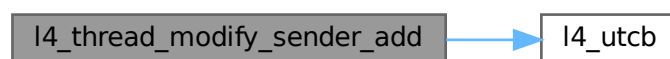
See also

[l4_thread_modify_sender_start](#)
[l4_thread_modify_sender_commit](#)

Definition at line 1087 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.

See also

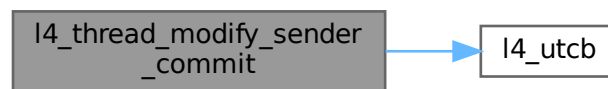
[l4_thread_modify_sender_start](#)

[l4_thread_modify_sender_add](#)

Definition at line 1098 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

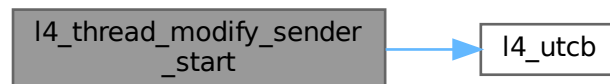
[l4_thread_modify_sender_add](#)

[l4_thread_modify_sender_commit](#)

Definition at line 1081 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:

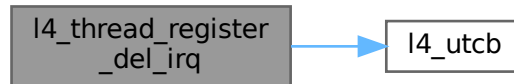
- deletion of one or several IPC gates bound to this thread.

When the deletion event is delivered, there is no indication about which IPC gate was deleted.

Definition at line 1008 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 μ s.

Parameters

	<i>thread</i>	Thread to get the consumed time from.
out	<i>us</i>	Consumed time in μ s.

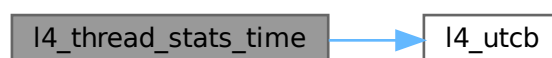
Returns

system call return tag

Definition at line 976 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 967 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 l4_task_add_ku_mem()).

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`.

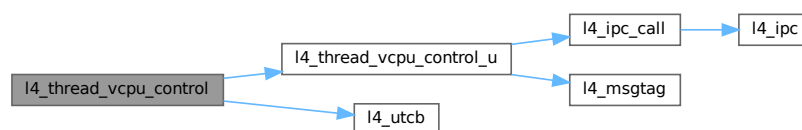
Note

Disabling of the vCPU feature is optional and currently not supported.

Definition at line 1025 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 <code>l4_task_add_ku_mem()</code>).

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.

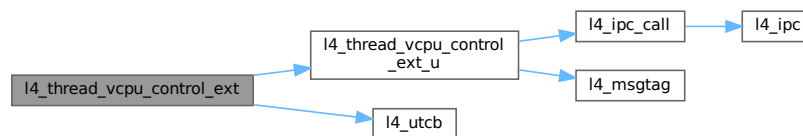
Disabling of the extended vCPU feature is currently not supported.

Upgrading from non-extended vCPU feature to extended vCPU feature is currently not supported.

Definition at line 1040 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 L4::Task::add_ku_mem()).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb .

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

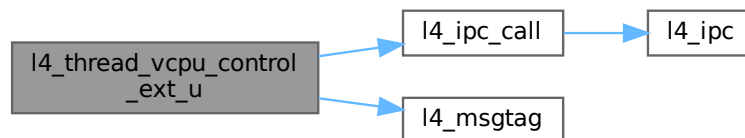
Enabling the extended vCPU feature for a thread running on a different CPU core is currently not supported.
 Disabling of the extended vCPU feature is currently not supported.
 Upgrading from non-extended vCPU feature to extended vCPU feature is currently not supported.

Definition at line 1030 of file [thread.h](#).

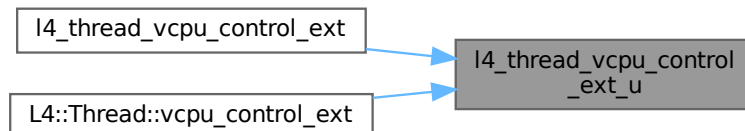
References [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_PROTO_THREAD](#), and [l4_msg_regs_t::mr](#).

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 l4_vcpu_state_t . It must be a valid kernel-user-memory address (see L4::Task::add_ku_mem()).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb .

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

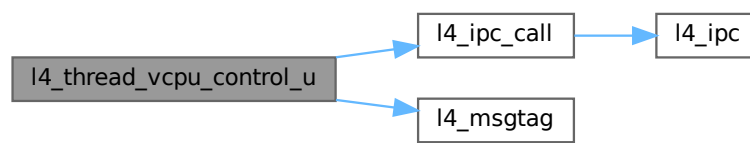
Disabling of the vCPU feature is optional and currently not supported.

Definition at line 1015 of file [thread.h](#).

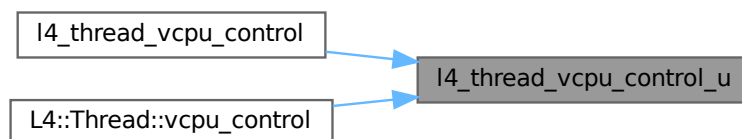
References [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_PROTO_THREAD](#), [L4_THREAD_VCPU_CONTROL_OP](#), and [l4_msg_regs_t::mr](#).

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 l4_thread_vcpu_resume_start()

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 L4_CAP_FPAGE_S 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.

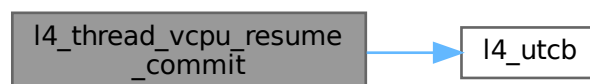
See also

[l4_vcpu_state_t](#)

Definition at line 988 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

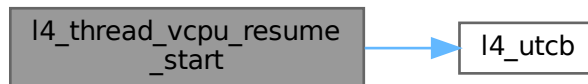
Message tag to be used for [l4_sndfpage_add\(\)](#) and [l4_thread_vcpu_resume_commit\(\)](#)

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 982 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.

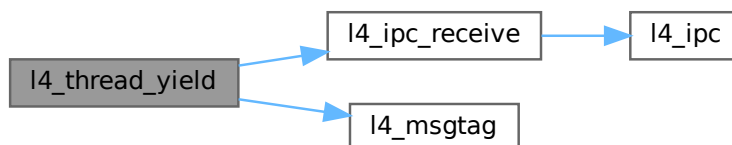
Returns

system call return tag

Definition at line 856 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](#).

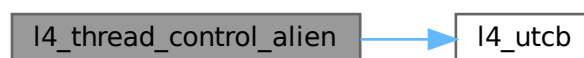
Examples

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line [946](#) 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.

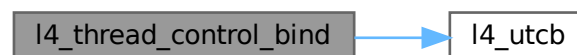
Examples

[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 940 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 L4_CAP_FPAGE_S right on <code>thread</code> or the task capability set in l4_thread_control_bind() .
<i>-L4_EINVAL</i>	Malformed thread control parameters.

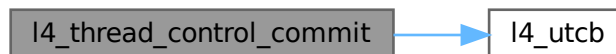
Examples

[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 [958](#) 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).

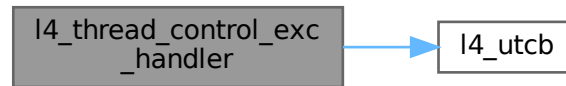
Examples

[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 933 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 927 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.

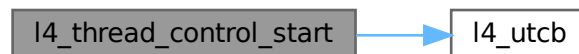
Examples

[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 921 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

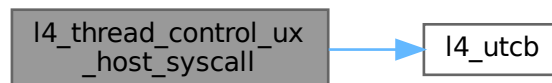
Running on Fiasco-UX

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 952 of file [thread.h](#).

References [l4_utcb\(\)](#).

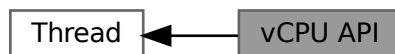
Here is the call graph for this function:



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](#) = 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

L4_VCPU_F_IRQ	<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>Note</p> <p>On L4::Thread::vcpu_resume_commit() this flag is automatically enabled in l4_vcpu_state_t::state if L4_VCPU_F_USER_MODE is enabled.</p> <p>When the kernel mode is entered, this flags is automatically disabled in l4_vcpu_state_t::state.</p>
L4_VCPU_F_PAGE_FAULTS	<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>Note</p> <p>IPC redirection for page faults controlled by this flag works independent of "kernel" and "user" mode.</p> <p>On L4::Thread::vcpu_resume_commit() this flag is automatically enabled in l4_vcpu_state_t::state if L4_VCPU_F_USER_MODE is enabled.</p> <p>When the kernel mode is entered, this flags is automatically disabled in l4_vcpu_state_t::state.</p>
Generated for L4Re by Doxygen	

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>Note</p> <p>IPC redirection for exceptions controlled by this flag works independent of "kernel" and "user" mode.</p> <p>On L4::Thread::vcpu_resume_commit() this flag is automatically enabled in l4_vcpu_state_t::state if L4_VCPU_F_USER_MODE is enabled.</p>
L4_VCPU_F_USER_MODE	<p>User task will be used. If set, the vCPU switches to user mode on next L4::Thread::vcpu_resume_commit(). If clear, the vCPU stays in "kernel" mode.</p> <p>Note</p> <p>When the kernel mode is entered, this flags is automatically disabled in l4_vcpu_state_t::state.</p>
L4_VCPU_F_FPU_ENABLED	<p>FPU enabled. This flag is only relevant if L4_VCPU_F_USER_MODE 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

```
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 189 of file [vcpu.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 l4_vcon_write call.
L4_VCON_READ_SIZE	Maximum size that can be read with one l4_vcon_read* 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.
<i>out</i>	<i>attr</i>	Attribute structure.

Returns

Syscall return tag

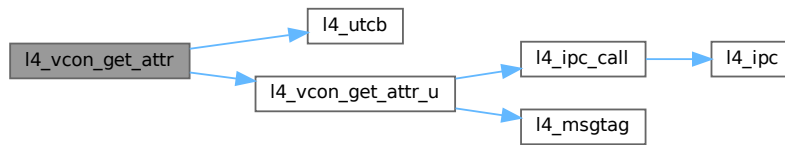
Examples

[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 l4_utcb .

Returns

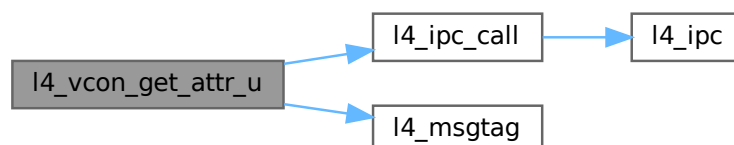
Syscall return tag.

Definition at line 426 of file [vcon.h](#).

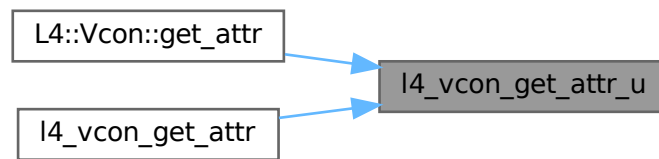
References [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_PROTO_LOG](#), [L4_VCON_GET_ATTR_OP](#), and [l4_msg_regs_t::mr](#).

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 L4_CAP_FPAGE_W right on the <code>vcon</code> capability and this right is not present.
<code>>size</code>	More bytes to read, <code>size</code> bytes are in the buffer <code>buf</code> .
<code><=size</code>	Number of bytes read.

Note

Size must not exceed [L4_VCON_READ_SIZE](#).

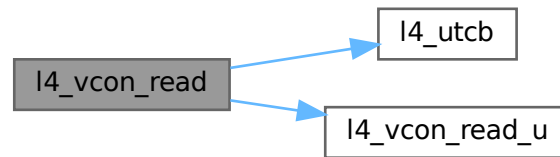
Examples

[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 l4_utcb .

Return values

<code>-L4_EPERM</code>	The Vcon instance requires the L4_CAP_FPAGE_W right on the capability used to invoke this operation and this right is not present.
<code>>size</code>	More bytes to read, <i>size</i> bytes are in the buffer <i>buf</i> .
<code><=size</code>	Number of bytes read.

Note

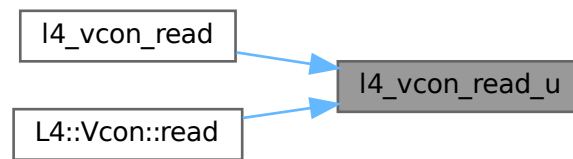
Size must not exceed [L4_VCON_READ_SIZE](#).

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 L4_CAP_FPAGE_W right on the <code>vcon</code> capability and this right is not present.
<code>>size</code>	More bytes to read, <code>size</code> bytes are in the buffer <code>buf</code> .
<code><=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

<i>vcon</i>	Vcon object.
<i>buf</i>	Pointer to data buffer.
<i>size</i>	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\(\)](#).

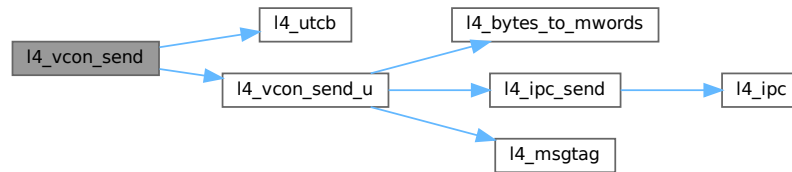
Examples

[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 l4_utcb .

Returns

Syscall return tag

Note

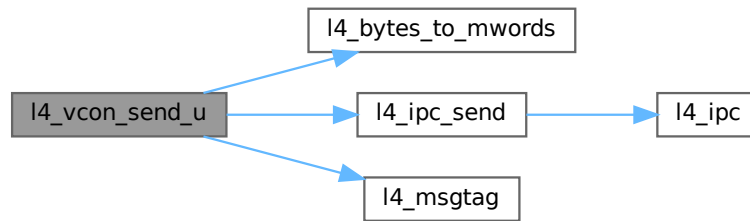
Size must not exceed [L4_VCON_WRITE_SIZE](#), a proper value of the *size* parameter is NOT checked. Also, this function is a send only operation, this means there is no return value except for a failed send operation. Use [l4_ipc_error\(\)](#) to check for send errors, do not use [l4_error\(\)](#), as [l4_error\(\)](#) will always return an error.

Definition at line 311 of file [vcon.h](#).

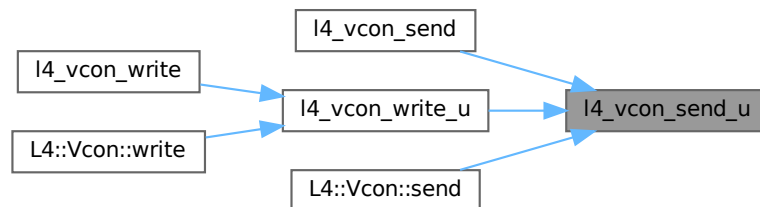
References [l4_bytes_to_mwords\(\)](#), [L4_IPC_NEVER](#), [l4_ipc_send\(\)](#), [l4_msgtag\(\)](#), [L4_MSGTAG_SCHEDULE](#), [L4_PROTO_LOG](#), [L4_VCON_WRITE_OP](#), and [l4_msg_regs_t::mr](#).

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

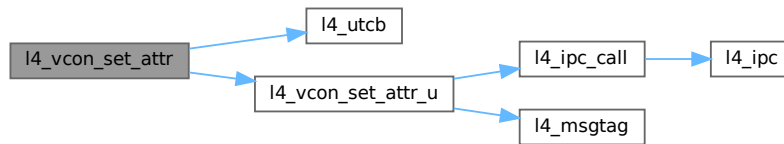
Examples

[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

<code>in, out</code>	<code>attr</code>	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 l4_utcb .

Returns

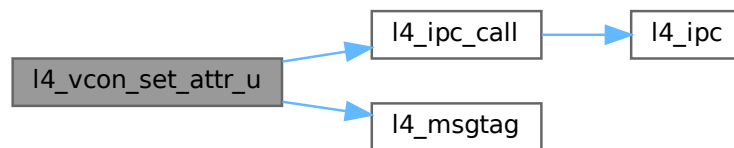
Syscall return tag.

Definition at line 406 of file [vcon.h](#).

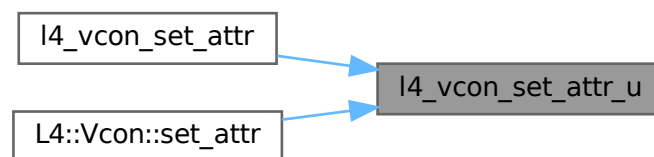
References [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_PROTO_LOG](#), [L4_VCON_SET_ATTR_OP](#), and [l4_msg_regs_t::mr](#).

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.

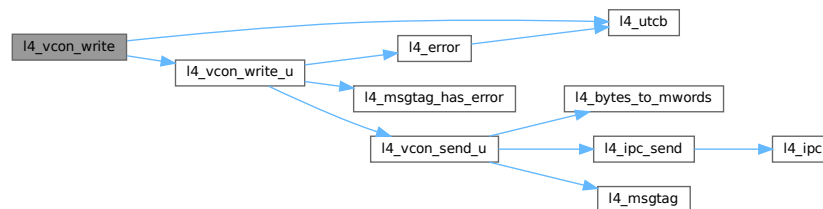
Return values

< 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 l4_utcb .

Return values

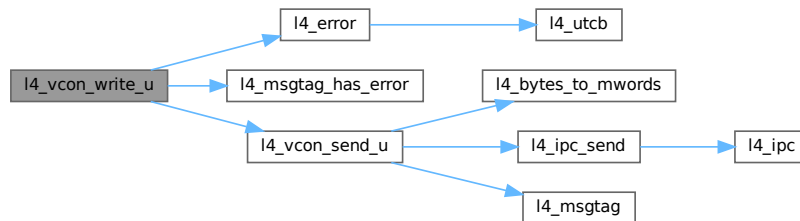
< 0	Error.
≥ 0	Number of bytes written to the virtual console.

Definition at line 330 of file [vcon.h](#).

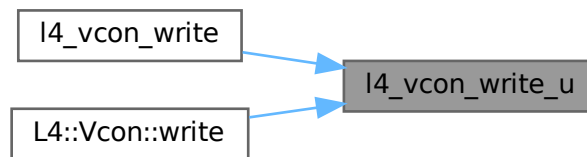
References [l4_error\(\)](#), [l4_msgtag_has_error\(\)](#), [l4_vcon_send_u\(\)](#), and [L4_VCON_WRITE_SIZE](#).

Referenced by [l4_vcon_write\(\)](#), and [L4::Vcon::write\(\)](#).

Here is the call graph for this function:



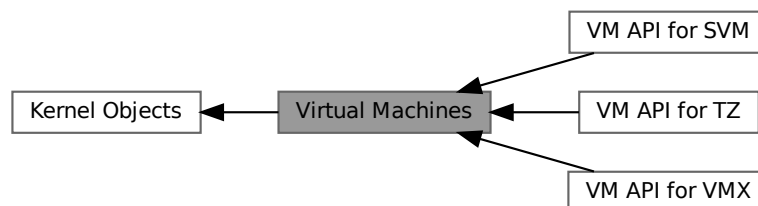
Here is the caller graph for this function:



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.

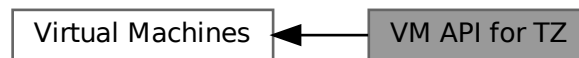
13.1.11.14.2.1 Detailed Description

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:



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_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_VMCS_CR2](#) = 0x683e , [L4_VM_VMX_VMCS_XCR0](#) = 0x2840 , [L4_VM_VMX_VMCS_MSR_SYSCALL_MASK](#)
= 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 VMCS to the user VMCS.
- void [l4_vm_vmx_ptr_load](#) (void *vmcs, void *user_vmcs) [L4_NOTHROW](#)
Loads the user_vmcs as the current VMCS.
- [l4_uint32_t l4_vm_vmx_get_cr2_index](#) (void const *vmcs) [L4_NOTHROW](#)
Get the 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 VMCS field.
- [l4_uint16_t l4_vm_vmx_read_16](#) (void *vmcs, unsigned field) [L4_NOTHROW](#)
Read a 16bit VMCS field.
- [l4_uint32_t l4_vm_vmx_read_32](#) (void *vmcs, unsigned field) [L4_NOTHROW](#)
Read a 32bit VMCS field.
- [l4_uint64_t l4_vm_vmx_read_64](#) (void *vmcs, unsigned field) [L4_NOTHROW](#)
Read a 64bit VMCS field.
- [l4_uint64_t l4_vm_vmx_read](#) (void *vmcs, unsigned field) [L4_NOTHROW](#)
Read any VMCS field.
- void [l4_vm_vmx_write_nat](#) (void *vmcs, unsigned field, [l4_umword_t](#) val) [L4_NOTHROW](#)
Write to a natural width VMCS field.
- void [l4_vm_vmx_write_16](#) (void *vmcs, unsigned field, [l4_uint16_t](#) val) [L4_NOTHROW](#)

Write to a 16bit VMCS field.

- void [l4_vm_vmx_write_32](#) (void *vmcs, unsigned field, [l4_uint32_t](#) val) [L4_NOTHROW](#)

Write to a 32bit VMCS field.

- void [l4_vm_vmx_write_64](#) (void *vmcs, unsigned field, [l4_uint64_t](#) val) [L4_NOTHROW](#)

Write to a 64bit VMCS field.

- void [l4_vm_vmx_write](#) (void *vmcs, unsigned field, [l4_uint64_t](#) val) [L4_NOTHROW](#)

Write to an arbitrary VMCS field.

13.1.11.14.4.1 Detailed Description

Virtual machine API for VMX.

13.1.11.14.4.2 Enumeration Type Documentation

anonymous enum

`anonymous enum`

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. Note You usually need to check this value against the value you get from l4_vm_vmx_get_cr2_index() 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 105 of file [__vm-vmx.h](#).

[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_CTLIS_REG	True pin-based control caps.
L4_VM_VMX_TRUE_PROCBASED_CTLIS_REG	True processor based control caps.
L4_VM_VMX_TRUE_EXIT_CTLIS_REG	True exit control caps.
L4_VM_VMX_TRUE_ENTRY_CTLIS_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_CTLIS2_REG	Processor based control 2 caps.
L4_VM_VMX_EPT_VPID_CAP_REG	EPT and VPID caps.
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_CTLIS_DFL1_REG	Default 1 bits in pin-based controls.
L4_VM_VMX_PROCBASED_CTLIS_DFL1_REG	Default 1 bits in processor-based controls.
L4_VM_VMX_EXIT_CTLIS_DFL1_REG	Default 1 bits in exit controls.
L4_VM_VMX_ENTRY_CTLIS_DFL1_REG	Default 1 bits in entry controls.
L4_VM_VMX_NUM_DFL1_REGS	Total number of default on registers.

Definition at line 62 of file [__vm-vmx.h](#).

13.1.11.14.4.3 Function Documentation**l4_vm_vmx_clear()**

```
void l4_vm_vmx_clear (
    void * vmcs,
    void * user_vmcs ) [inline]
```

Saves cached state from the kernel VMCS to the user VMCS.

Parameters

<i>vmcs</i>	Pointer to the kernel VMCS.
<i>user_vmcs</i>	Pointer to the user VMCS.

This function is comparable to VMX vmclear.

Definition at line 433 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_ptr_load\(\)](#).

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 357 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 342 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 L4_vm_vmx_caps_regs).

Returns

The value of the capability register.

Definition at line 529 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 L4_vm_vmx_dfl1_regs).

Returns

The value of the capability register.

Definition at line 537 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 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 545 of file [__vm-vmx.h](#).

`l4_vm_vmx_ptr_load()`

```
void l4_vm_vmx_ptr_load (
    void * vmcs,
    void * user_vmcs ) [inline]
```

Loads the *user_vmcs* as the current VMCS.

Parameters

<i>vmcs</i>	Pointer to the kernel VMCS.
<i>user_vmcs</i>	Pointer to the user VMCS.

This function is comparable to VMX vmprld.

Definition at line 445 of file [__vm-vmx.h](#).

References [l4_vm_vmx_clear\(\)](#).

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 VMCS field.

Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.

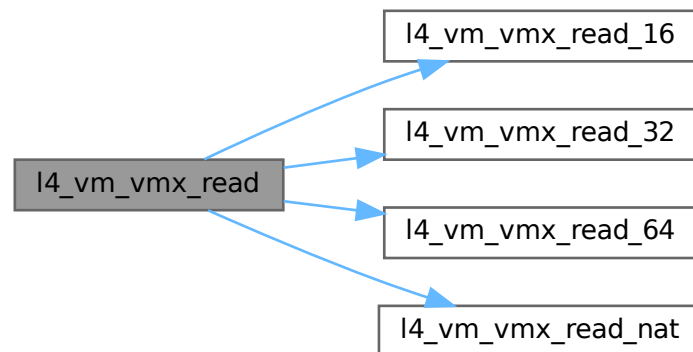
Returns

The value of the VMCS field with the given index.

Definition at line 481 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 16bit 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 VMCS field with the given index.

Definition at line 466 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 32bit 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 VMCS field with the given index.

Definition at line 471 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 64bit 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 VMCS field with the given index.

Definition at line 476 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 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 VMCS field with the given index.

Definition at line 461 of file `__vm-vmx.h`.

Referenced by [l4_vm_vmx_read\(\)](#).

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 VMCS field.

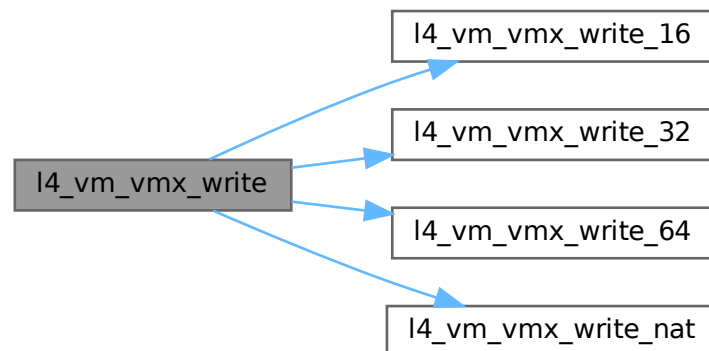
Parameters

<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 516 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 16bit VMCS field.

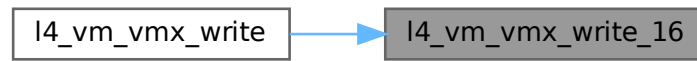
Parameters

<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 500 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 32bit VMCS field.

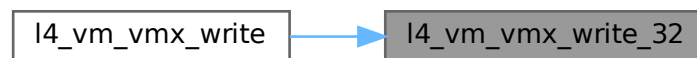
Parameters

<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 505 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 64bit 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 510 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 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 495 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

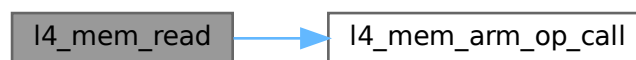
Read value.

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

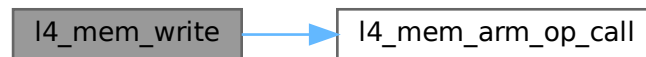
<i>virtaddress</i>	Virtual address in the calling task.
<i>width</i>	Width of access in bytes in log2 (i.e. allowed values: 0, 1, 2)
<i>value</i>	Value to write.

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_SUPERPAGE_SIZE`
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.

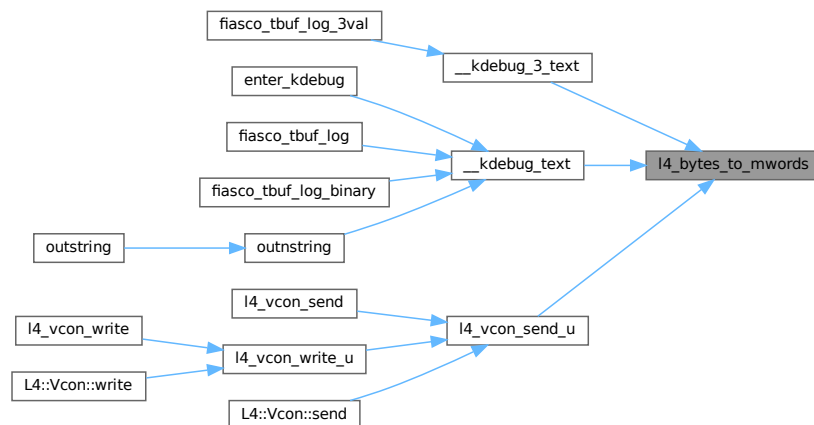
Parameters

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

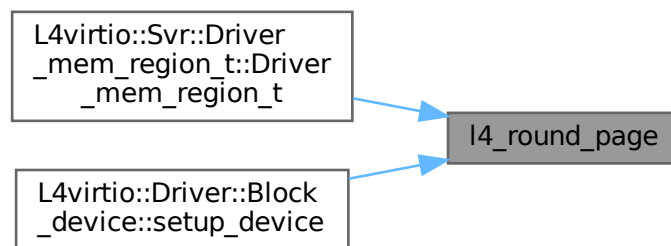
<i>address</i>	The address to round up.
----------------	--------------------------

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\(\)](#), 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.

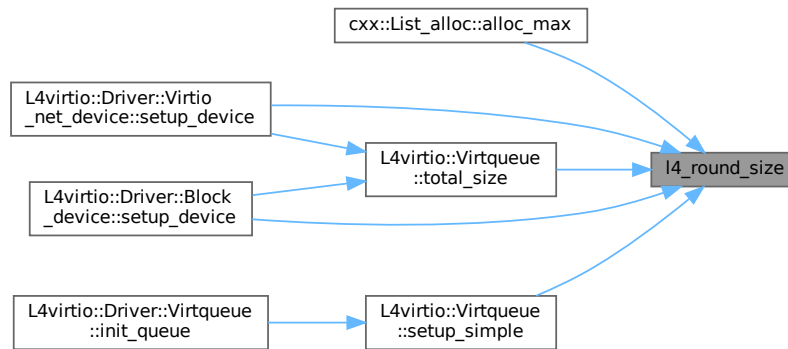
Parameters

<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

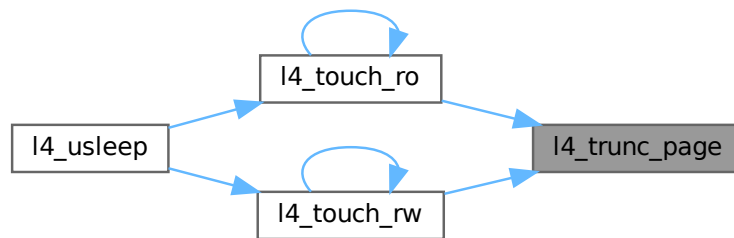
[examples/libs/l4re/c++/mem_alloc/ma+rm.cc](#), and [examples/libs/l4re/c/ma+rm.c](#).

Definition at line [437](#) of file [consts.h](#).

References [L4_PAGEMASK](#).

Referenced by [l4_touch_ro\(\)](#), and [l4_touch_rw\(\)](#).

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*.

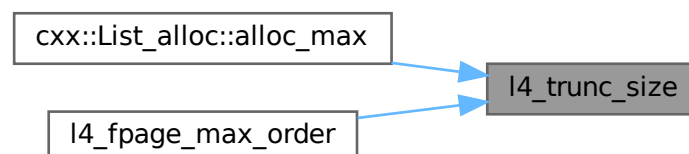
Parameters

<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\(\)](#), and [l4_fpage_max_order\(\)](#).

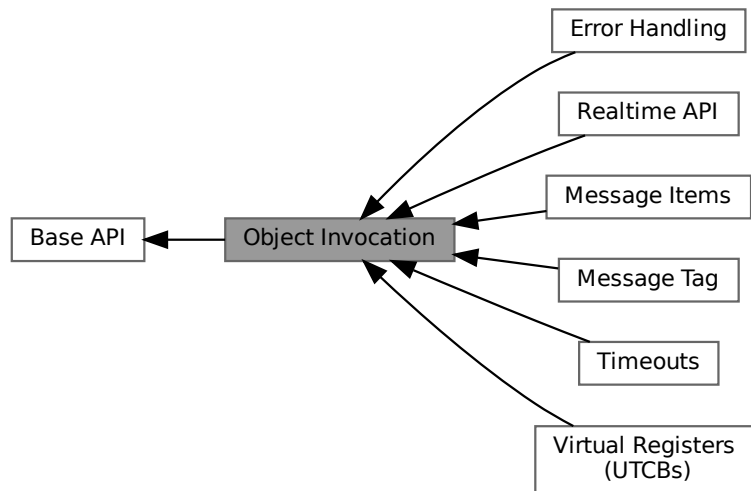
Here is the caller graph for this function:



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](#) (unsigned ms) [L4_NOTHROW](#)
Sleep for a certain amount of milliseconds.
- [l4_msgtag_t l4_ipc_sleep_us](#) (unsigned 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 Enumeration Type Documentation

13.1.14.2.1 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	Empty set of flags. Deprecated 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).
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 L4_SYSF_RECV . In l4_vcpu_state_t::user_task this flag means that the kernel has cached the user task capability internally, see l4_thread_vcpu_resume_commit() .
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 L4_SYSF_SEND .
L4_SYSF_OPEN_WAIT	Open-wait flag. This flag indicates that the receive operation (see L4_SYSF_RECV) 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 L4_SYSF_SEND and L4_SYSF_RECV .
L4_SYSF_WAIT	Wait flags (combines receive and open wait). Combines L4_SYSF_RECV and L4_SYSF_OPEN_WAIT .
L4_SYSF_SEND_AND_WAIT	Send-and-wait flags. Combines L4_SYSF_SEND and L4_SYSF_WAIT .
L4_SYSF_REPLY_AND_WAIT	Reply-and-wait flags. Combines L4_SYSF_SEND , L4_SYSF_REPLY , and L4_SYSF_WAIT .

Definition at line 61 of file [consts.h](#).

13.1.14.3 Function Documentation

13.1.14.3.1 [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. L4_INVALID_CAP 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 l4_utcb .
	<i>flags</i>	Invocation flags (see l4_syscall_flags_t).
	<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 l4_timeout_t).

Returns

return tag

See also

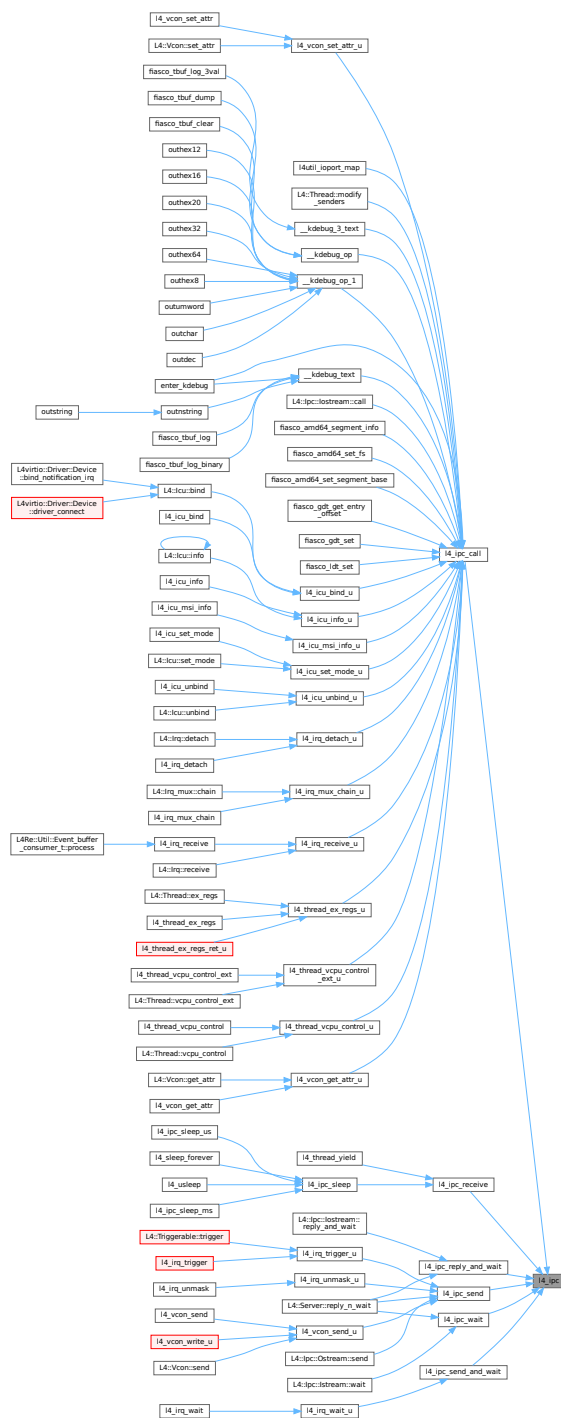
[L4 Inter-Process Communication \(IPC\)](#)

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:



13.1.14.3.2 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 L4_INVALID_CAP 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 l4_utcb .
<i>tag</i>	Message tag to describe the message to be sent.
<i>timeout</i>	Timeout pair for send an receive phase (see l4_timeout_t).

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.

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 550 of file [ipc.h](#).

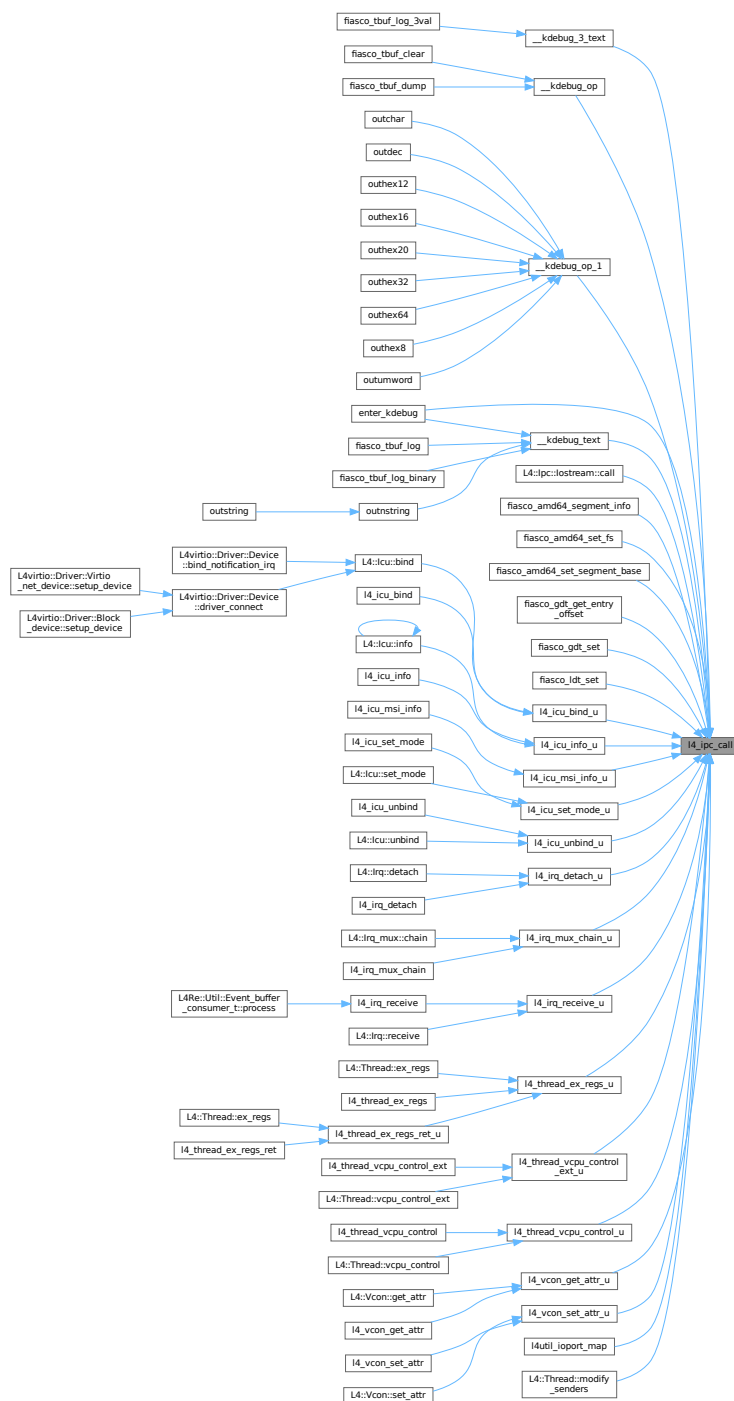
References [l4_ipc\(\)](#), and [L4_SYSF_CALL](#).

Referenced by [__kdebug_3_text\(\)](#), [__kdebug_op\(\)](#), [__kdebug_op_1\(\)](#), [__kdebug_text\(\)](#), [L4::lpc::lostream::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:



13.1.14.3.3 l4_ipc_receive()

```
l4_msgtag_t l4_ipc_receive (
    l4_cap_idx_t object,
    l4_utcb_t * utcb,
    l4_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 L4_INVALID_CAP 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 l4_utcb .
<i>timeout</i>	Timeout pair (see l4_timeout_t , 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

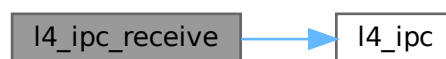
[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 592 of file [ipc.h](#).

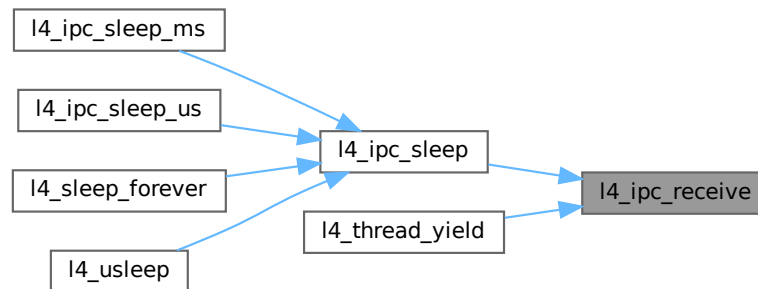
References [l4_ipc\(\)](#), [L4_SYSF_RECV](#), and [l4_msgtag_t::raw](#).

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:



13.1.14.3.4 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 l4_utcb .
	<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 l4_timeout_t).

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

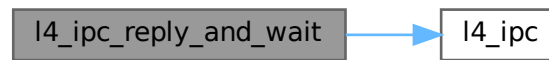
[examples/sys/ipc/ipc_example.c](#).

Definition at line 558 of file [ipc.h](#).

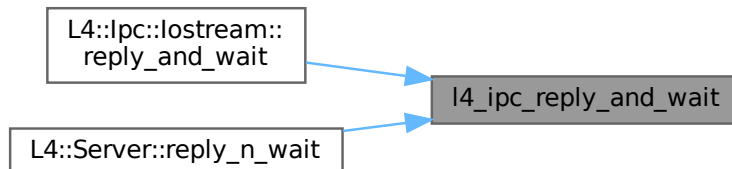
References [L4_INVALID_CAP](#), [l4_ipc\(\)](#), and [L4_SYSF_REPLY_AND_WAIT](#).

Referenced by [L4::ipc::lostream::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:



13.1.14.3.5 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 L4_INVALID_CAP 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 l4_utcb .
<i>tag</i>	Descriptor for the message to be sent.
<i>timeout</i>	Timeout pair (see l4_timeout_t) 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 [575](#) of file [ipc.h](#).

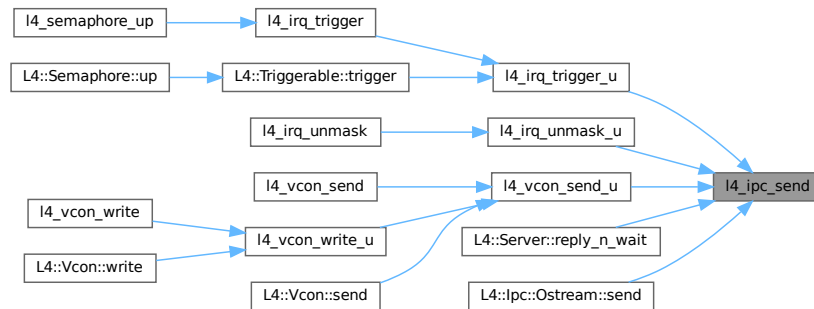
References [l4_ipc\(\)](#), and [L4_SYSF_SEND](#).

Referenced by [l4_irq_trigger_u\(\)](#), [l4_irq_unmask_u\(\)](#), [l4_vcon_send_u\(\)](#), [L4::Server< LOOP_HOOKS >::reply_n_wait\(\)](#), and [L4::lpc::Ostream::send\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.14.3.6 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 L4_INVALID_CAP 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 l4_utcb .
	<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 l4_timeout_t).

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

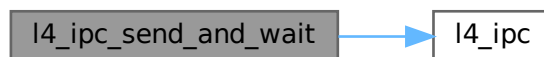
[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 566 of file [ipc.h](#).

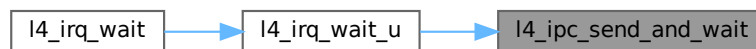
References [l4_ipc\(\)](#), and [L4_SYSF_SEND_AND_WAIT](#).

Referenced by [l4_irq_wait_u\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.14.3.7 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 l4_timeout_t , 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

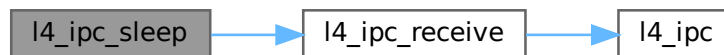
[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 601 of file [ipc.h](#).

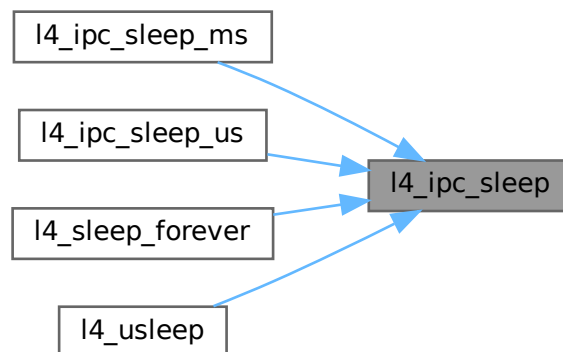
References [L4_INVALID_CAP](#), and [l4_ipc_receive\(\)](#).

Referenced by [l4_ipc_sleep_ms\(\)](#), [l4_ipc_sleep_us\(\)](#), [l4_sleep_forever\(\)](#), and [l4_usleep\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.14.3.8 l4_ipc_sleep_ms()

```

l4_msgtag_t l4_ipc_sleep_ms (
    unsigned 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\(\)](#).

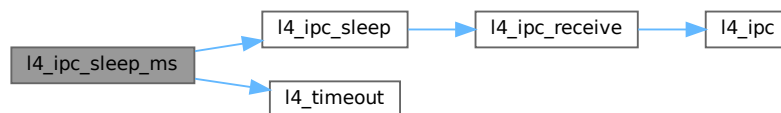
See also

[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 605 of file [ipc.h](#).

References [l4_ipc_sleep\(\)](#), [L4_IPC_TIMEOUT_NEVER](#), and [l4_timeout\(\)](#).

Here is the call graph for this function:

**13.1.14.3.9 l4_ipc_sleep_us()**

```
l4_msgtag_t l4_ipc_sleep_us (
    unsigned us ) [inline]
```

Sleep for a certain amount of microseconds.

Parameters

<code>us</code>	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\(\)](#).

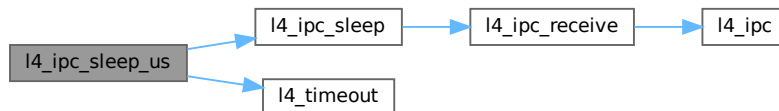
See also

[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 612 of file [ipc.h](#).

References [l4_ipc_sleep\(\)](#), [L4_IPC_TIMEOUT_NEVER](#), and [l4_timeout\(\)](#).

Here is the call graph for this function:



13.1.14.3.10 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 l4_utcb .
out	<i>label</i>	Label assigned to the source object (IPC gate or IRQ).
	<i>timeout</i>	Timeout pair (see l4_timeout_t , 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 583 of file [ipc.h](#).

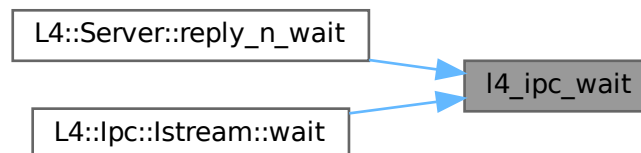
References [L4_INVALID_CAP](#), [l4_ipc\(\)](#), [L4_SYSF_WAIT](#), and [l4_msgtag_t::raw](#).

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:



13.1.14.3.11 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 675 of file [ipc.h](#).

References [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.14.4 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.

13.1.14.4.1 Detailed Description

Error handling for `L4` object invocation.

Include File

```
#include <l4/sys/ipc.h>
```

13.1.14.4.2 Enumeration Type Documentation

13.1.14.4.2.1 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.
<code>L4_IPC_SETIMEOUT</code>	Timeout during send operation.
<code>L4_IPC_RECANCELED</code>	Receive operation canceled.
<code>L4_IPC_SECANCELED</code>	Send operation canceled.
<code>L4_IPC_REMAPFAILED</code>	Map flexpage failed in receive operation.
<code>L4_IPC_SEMAPFAILED</code>	Map flexpage failed in send operation.
<code>L4_IPC_RESNDPFTO</code>	Send-pagefault timeout in receive operation.
<code>L4_IPC_SESNDPFTO</code>	Send-pagefault timeout in send operation.
<code>L4_IPC_RERCVPFTO</code>	Receive-pagefault timeout in receive operation.
<code>L4_IPC_SERCVPFTO</code>	Receive-pagefault timeout in send operation.
<code>L4_IPC_REABORTED</code>	Receive operation aborted.
<code>L4_IPC_SEABORTED</code>	Send operation aborted.
<code>L4_IPC_REMSGCUT</code>	Received message truncated. Usually returned when the typed items to be sent by the IPC partner exceed the buffer registers of the respective types.
<code>L4_IPC_SEMSGCUT</code>	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 75 of file [ipc.h](#).

13.1.14.4.3 Function Documentation

13.1.14.4.3.1 `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 636 of file [ipc.h](#).

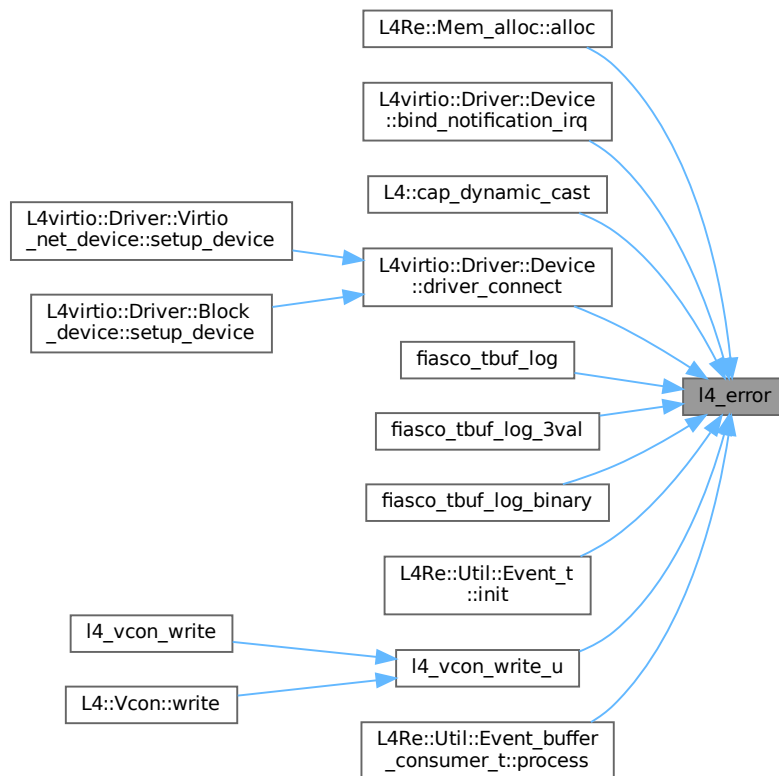
References [l4_utcb\(\)](#).

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:



13.1.14.4.3.2 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 619 of file [ipc.h](#).

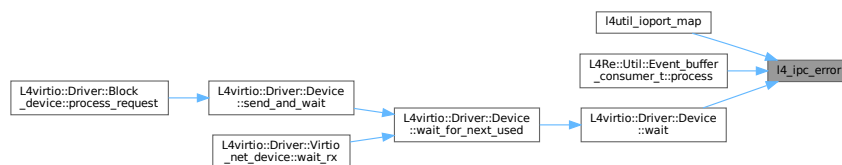
References [l4_thread_regs_t::error](#), [L4_IPC_ERROR_MASK](#), and [l4_msgtag_has_error\(\)](#).

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:



13.1.14.4.3.3 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 648 of file [ipc.h](#).

References [l4_thread_regs_t::error](#), and [L4_IPC_ERROR_MASK](#).

13.1.14.4.3.4 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 645 of file [ipc.h](#).

References [l4_thread_regs_t::error](#).

13.1.14.4.3.5 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

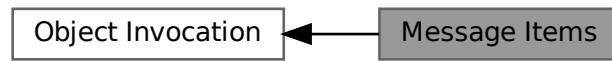
Boolean value.

Definition at line 642 of file [ipc.h](#).

13.1.14.5 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.

13.1.14.5.1 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.

13.1.14.5.2 Enumeration Type Documentation

13.1.14.5.2.1 `l4_msg_item_consts_t`

```
enum l4_msg_item_consts_t
```

Constants for message items.

Enumerator

<code>L4_ITEM_MAP</code>	Identify a message item as <i>map item</i> .
--------------------------	--

Enumerator

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 Spaces and Mappings for more details on map/grant.</p> <p>Note</p> <p>The grant operation is not performed if the resulting rights of the receiver mapping would not contain the L4_CAP_FPAGE_R bit (for object capabilities) or none of the L4_FPAGE_RWX 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 L4 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 L4_RCV_ITEM_SINGLE_CAP.</p> <p>When this flag is set, then,</p> <ul style="list-style-type: none"> • 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, • 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 L4_CAP_FPAGE_W and L4_CAP_FPAGE_S permissions that would have been mapped is transferred, • otherwise a regular mapping is done for this item.

Definition at line 224 of file [consts.h](#).

13.1.14.5.3 Function Documentation

13.1.14.5.3.1 l4_map_control()

```
l4_umword_t l4_map_control (
    l4_umword_t spot,
    unsigned char cache,
    unsigned grant ) [inline]
```

Create the first word for a map item for the memory space.

Parameters

<i>spot</i>	Hot spot address, used to determine what is actually mapped when send and receive flex page have differing sizes.
<i>cache</i>	Cacheability hints for memory flex pages. See Cacheability options .
<i>grant</i>	Indicates if it is a map or a grant item. Allowed values: L4_MAP_ITEM_MAP , L4_MAP_ITEM_GRANT .

Returns

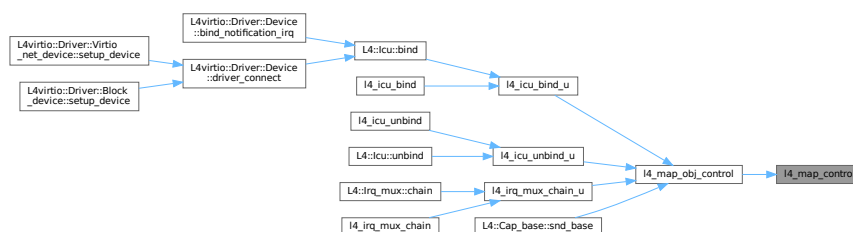
The value to be used as first word in a map item for memory.

Definition at line 707 of file __l4_fpage.h.

References [L4_FPAGE_CONTROL_MASK](#), and [L4_ITEM_MAP](#).

Referenced by [l4_map_obj_control\(\)](#).

Here is the caller graph for this function:



13.1.14.5.3.2 l4_map_obj_control()

```
l4_umword_t l4_map_obj_control (
    l4_umword_t spot,
    unsigned grant ) [inline]
```

Create the first word for a map item for the object space.

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: <code>L4_MAP_ITEM_MAP</code> , <code>L4_MAP_ITEM_GRANT</code> .

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.

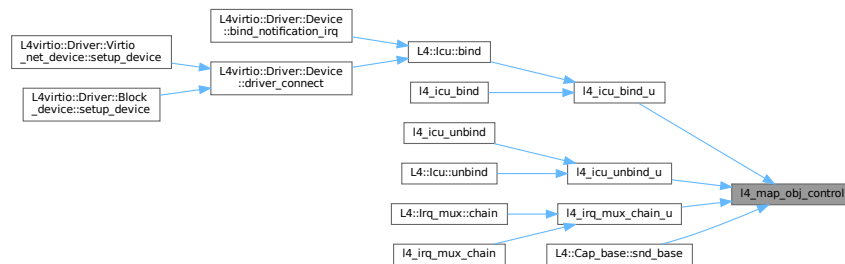
References [l4_map_control\(\)](#).

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:



13.1.14.6 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 }
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.

13.1.14.6.1 Detailed Description

API related to the message tag data type.

Include File

```
#include <l4/sys/types.h>
```

13.1.14.6.2 Typedef Documentation

13.1.14.6.2.1 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.

13.1.14.6.3 Enumeration Type Documentation

13.1.14.6.3.1 L4_msgtag_flags

```
enum L4_msgtag_flags
```

Flags for message tags.

Enumerator

L4_MSGTAG_ERROR	Error indicator flag.
L4_MSGTAG_TRANSFER_FPU	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).
L4_MSGTAG_SCHEDULE	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 performance.
L4_MSGTAG_FLAGS	Mask for all flags.

Definition at line 95 of file [types.h](#).

13.1.14.6.3.2 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.

Definition at line 49 of file [types.h](#).

13.1.14.6.3.3 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 L4_platform_ctl_ops for allowed operations.
-----------------------	---

Definition at line 179 of file [platform_control.h](#).

13.1.14.6.4 Function Documentation

13.1.14.6.4.1 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

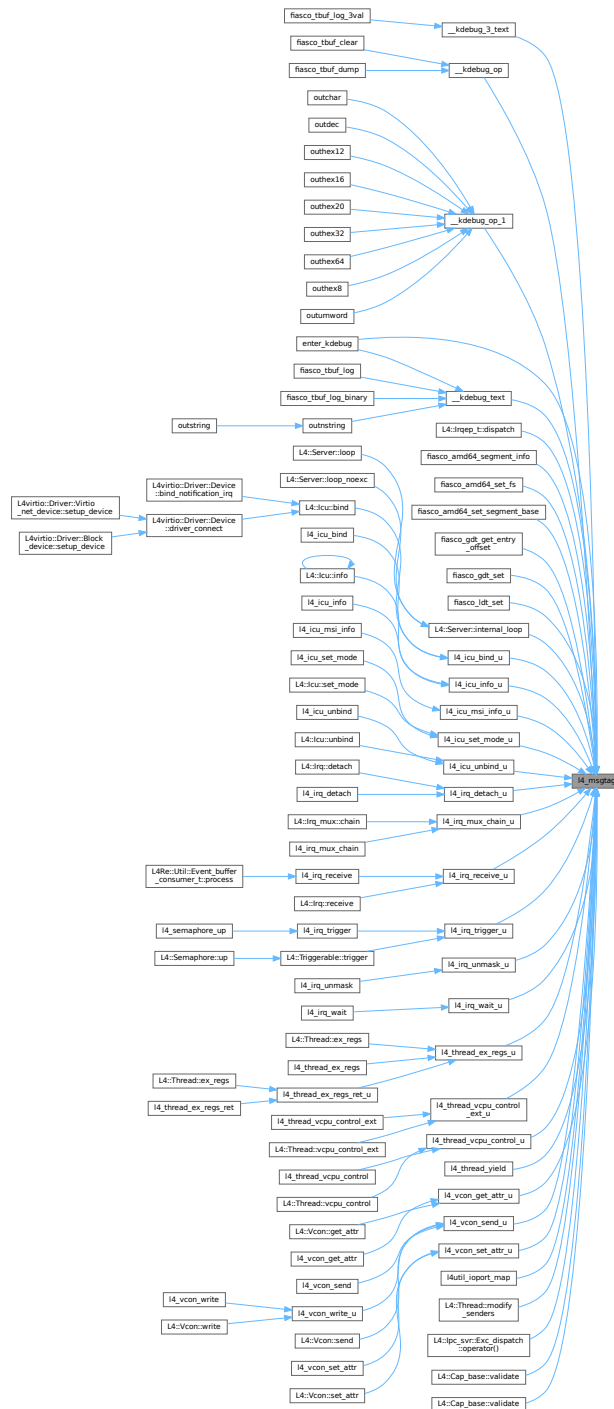
Examples

[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 427 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\(\)](#).

Get the flags.



Get the flags.

Get the flags.

Get the flags.

Parameters

<i>t</i>	The tag
----------	---------

Returns

Flags

Definition at line 451 of file [types.h](#).

13.1.14.6.4.3 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

>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

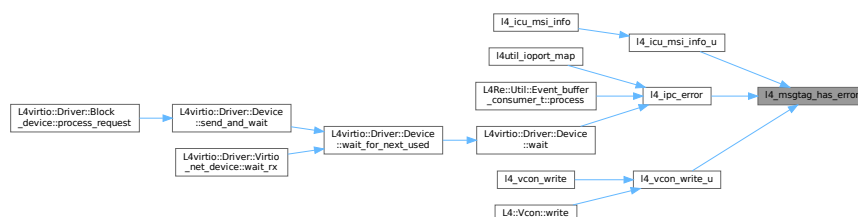
[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 456 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:



13.1.14.6.4.4 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

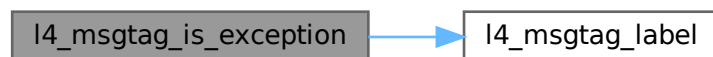
Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 470 of file [types.h](#).

References [l4_msgtag_label\(\)](#), and [L4_PROTO_EXCEPTION](#).

Here is the call graph for this function:

**13.1.14.6.4.5 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
----------	---------

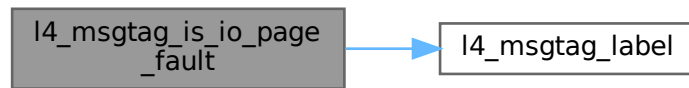
Returns

Boolean value

Definition at line 476 of file [types.h](#).

References [l4_msgtag_label\(\)](#), and [L4_PROTO_IO_PAGE_FAULT](#).

Here is the call graph for this function:



13.1.14.6.4.6 l4_msgtag_is_page_fault()

```
unsigned l4_msgtag_is_page_fault (  
    l4_msgtag_t t ) [inline]
```

Test for page-fault protocol.

Parameters

<i>t</i>	The tag
----------	---------

Returns

Boolean value

Definition at line 461 of file [types.h](#).

References [l4_msgtag_label\(\)](#), and [L4_PROTO_PAGE_FAULT](#).

Here is the call graph for this function:



13.1.14.6.4.7 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 464 of file [types.h](#).

References [l4_msgtag_label\(\)](#), and [L4_PROTO_PREEMPTION](#).

Here is the call graph for this function:

**13.1.14.6.4.8 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 473 of file [types.h](#).

References [l4_msgtag_label\(\)](#), and [L4_PROTO_SIGMA0](#).

Here is the call graph for this function:



13.1.14.6.4.9 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
----------	---------

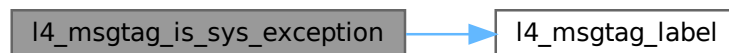
Returns

Boolean value

Definition at line 467 of file [types.h](#).

References [l4_msgtag_label\(\)](#), and [L4_PROTO_SYS_EXCEPTION](#).

Here is the call graph for this function:



13.1.14.6.4.10 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 447 of file [types.h](#).

Referenced by [l4util_ioport_map\(\)](#).

Here is the caller graph for this function:



13.1.14.6.4.11 `l4_msgtag_label()`

```
long l4_msgtag_label (
    l4_msgtag_t t ) [inline]
```

Get the protocol of tag.

Parameters

<code>t</code>	The tag
----------------	---------

Returns

Label

Examples

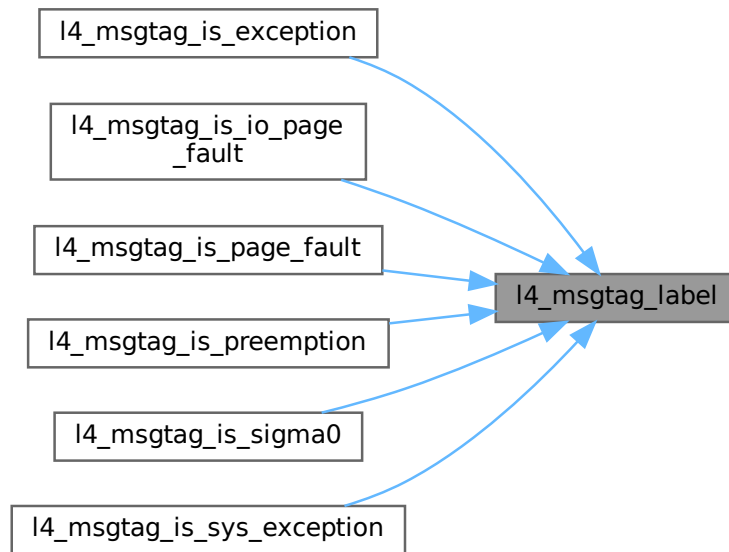
[examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line [439](#) 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:



13.1.14.6.4.12 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

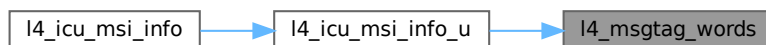
Examples

[examples/sys/utcb-ipc/main.c](#).

Definition at line 443 of file [types.h](#).

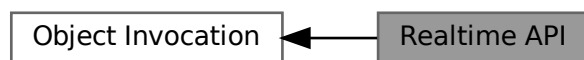
Referenced by [l4_icu_msi_info_u\(\)](#).

Here is the caller graph for this function:



13.1.14.7 Realtime API

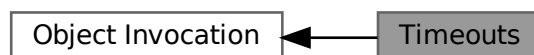
Collaboration diagram for Realtime API:



13.1.14.8 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 (~0U)`
The waiting period in microseconds which is interpreted as "never" by l4_timeout_from_us().
- `#define L4_TIMEOUT_US_MAX (~0U - 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_timeout_s l4_timeout_rel (unsigned man, unsigned exp) L4_NOTHROW`
Get relative timeout consisting of mantissa and exponent.
- `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_timeout_t l4_timeout (l4_timeout_s snd, l4_timeout_s rcv) L4_NOTHROW`
Combine send and receive timeout in a timeout.
- `void l4_snd_timeout (l4_timeout_s snd, l4_timeout_t *to) L4_NOTHROW`
Set send timeout in given to timeout.
- `void l4_rcv_timeout (l4_timeout_s rcv, l4_timeout_t *to) L4_NOTHROW`
Set receive timeout in given to timeout.
- `l4_kernel_clock_t l4_timeout_rel_get (l4_timeout_s to) L4_NOTHROW`
Get clock value of out timeout.
- `unsigned l4_timeout_is_absolute (l4_timeout_s to) L4_NOTHROW`
Return whether the given timeout is absolute or not.
- `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.

13.1.14.8.1 Detailed Description

All kinds of timeouts and time related functions.

13.1.14.8.2 Macro Definition Documentation

13.1.14.8.2.1 L4_IPC_TIMEOUT_0

```
#define L4_IPC_TIMEOUT_0 ((l4_timeout_s){0x0400})
```

Timeout constants.

0 timeout

Definition at line 79 of file [__timeout.h](#).

13.1.14.8.3 Typedef Documentation

13.1.14.8.3.1 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.

13.1.14.8.3.2 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.

13.1.14.8.4 Function Documentation

13.1.14.8.4.1 l4_ipc_timeout()

```
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 208 of file [__timeout.h](#).

References [l4_timeout_t::p](#), [l4_timeout_t::rcv](#), [l4_timeout_t::snd](#), and [l4_timeout_s::t](#).

13.1.14.8.4.2 l4_rcv_timeout()

```
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>	L4 timeout

Definition at line 236 of file [__timeout.h](#).

13.1.14.8.4.3 l4_snd_timeout()

```
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>	L4 timeout

Definition at line 229 of file [__timeout.h](#).

References [l4_timeout_t::p](#), and [l4_timeout_t::snd](#).

13.1.14.8.4.4 l4_timeout()

```
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

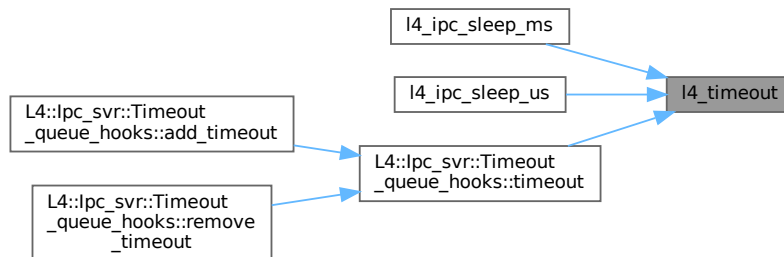
L4 timeout

Definition at line 219 of file `__timeout.h`.

References `l4_timeout_t::p`, `l4_timeout_t::rcv`, and `l4_timeout_t::snd`.

Referenced by `l4_ipc_sleep_ms()`, `l4_ipc_sleep_us()`, and `L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::timeout()`.

Here is the caller graph for this function:



13.1.14.8.4.5 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

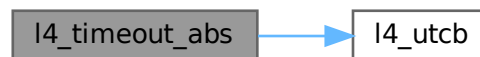
timeout value

Definition at line 383 of file [utcb.h](#).

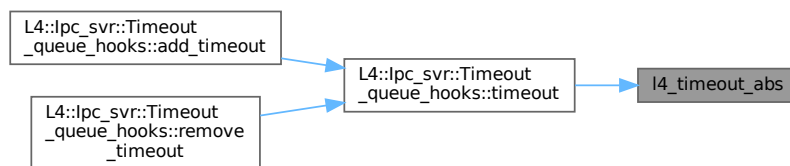
References [l4_utcb\(\)](#).

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:

**13.1.14.8.4.6 l4_timeout_get()**

```

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>	L4 timeout

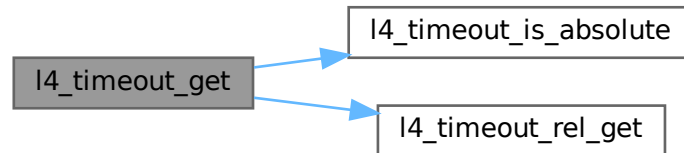
Returns

Clock sum

Definition at line 266 of file [__timeout.h](#).

References [l4_timeout_is_absolute\(\)](#), and [l4_timeout_rel_get\(\)](#).

Here is the call graph for this function:



13.1.14.8.4.7 l4_timeout_is_absolute()

```

unsigned l4_timeout_is_absolute (
    l4_timeout_s to ) [inline]
  
```

Return whether the given timeout is absolute or not.

Parameters

<i>to</i>	L4 timeout
-----------	------------

Returns

!= 0 if absolute, 0 if relative

Definition at line 259 of file [__timeout.h](#).

References [l4_timeout_s::t](#).

Referenced by [l4_timeout_get\(\)](#).

Here is the caller graph for this function:



13.1.14.8.4.8 l4_timeout_rel()

```
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 243 of file [__timeout.h](#).

13.1.14.8.4.9 l4_timeout_rel_get()

```
l4_kernel_clock_t l4_timeout_rel_get (
    l4_timeout_s to ) [inline]
```

Get clock value of out timeout.

Parameters

<i>to</i>	L4 timeout
-----------	----------------------------

Returns

Clock value

Definition at line 250 of file [__timeout.h](#).

Referenced by [l4_timeout_get\(\)](#).

Here is the caller graph for this function:



13.1.14.8.4.10 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
------------	--

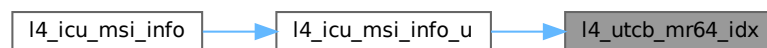
Returns

Index to 64bit message register alias

Definition at line 386 of file [utcb.h](#).

Referenced by [l4_icu_msi_info_u\(\)](#).

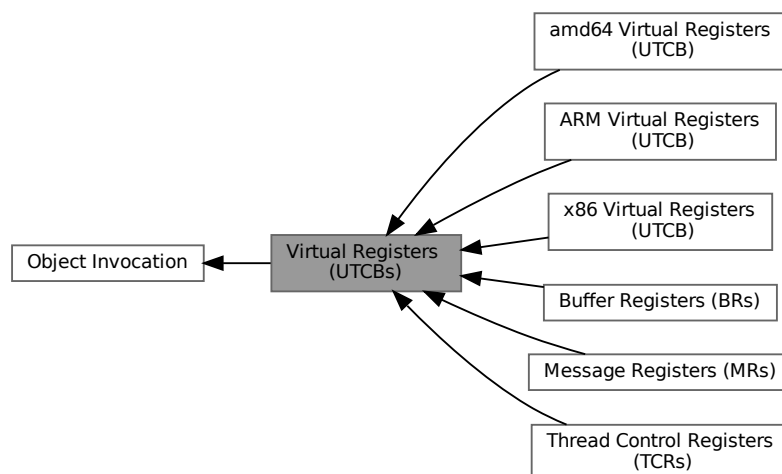
Here is the caller graph for this function:



13.1.14.9 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.

13.1.14.9.1 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.

13.1.14.9.2 Typedef Documentation

13.1.14.9.2.1 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](#).

13.1.14.9.3 Function Documentation

13.1.14.9.3.1 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:



13.1.14.9.3.2 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](#).

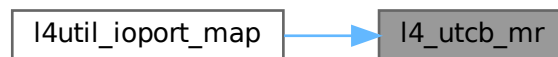
References [l4_utcb\(\)](#).

Referenced by [l4util_ioport_map\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.14.9.3.3 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](#).

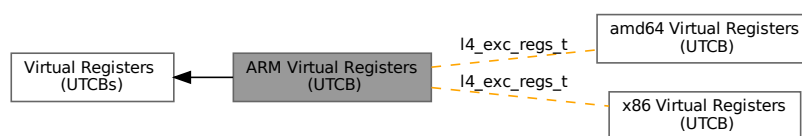
References [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.14.9.4 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

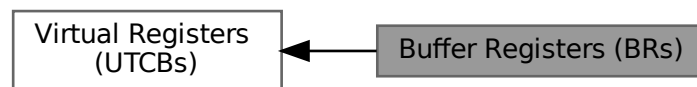
- enum [L4_utcb_consts_arm](#)

UTCB constants for ARM.

13.1.14.9.4.1 Detailed Description

13.1.14.9.5 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.

13.1.14.9.5.1 Detailed Description

13.1.14.9.5.2 Enumeration Type Documentation

[l4_buffer_desc_consts_t](#)

enum [l4_buffer_desc_consts_t](#)

Constants for buffer descriptors.

Enumerator

L4_BDR_MEM_SHIFT	Bit offset for the memory-buffer index.
L4_BDR_IO_SHIFT	Bit offset for the IO-buffer index.
L4_BDR_OBJ_SHIFT	Bit offset for the capability-buffer index.

Definition at line 292 of file [consts.h](#).

13.1.14.9.6 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

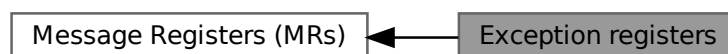
- typedef union [l4_msg_regs_t](#) **l4_msg_regs_t**
Encapsulation of the message-register block in the UTCB.

13.1.14.9.6.1 Detailed Description

13.1.14.9.6.2 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 ex_regs.
!=0	Exception was triggered through ex_regs.

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 116 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 106 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

u	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 91 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.

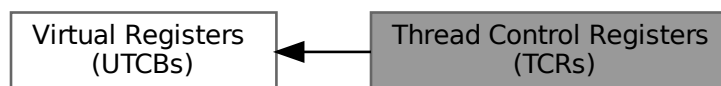
Parameters

<i>u</i>	UTCB
<i>pc</i>	The program counter to set.

Definition at line 96 of file [utcb.h](#).

13.1.14.9.7 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.

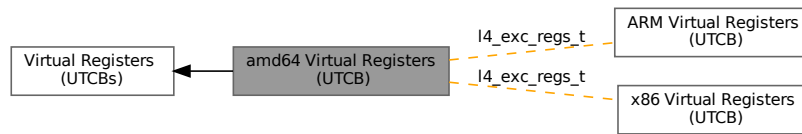
Typedefs

- typedef struct [l4_thread_regs_t](#) **l4_thread_regs_t**
Encapsulation of the thread-control-register block of the UTCB.

13.1.14.9.7.1 Detailed Description

13.1.14.9.8 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.

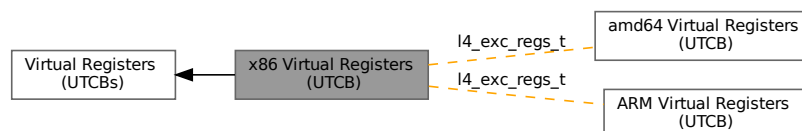
Enumerations

- enum [L4_utcb_consts_amd64](#)
UTCB constants for AMD64.

13.1.14.9.8.1 Detailed Description

13.1.14.9.9 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.

13.1.14.9.1 Detailed Description

13.1.14.9.2 Enumeration Type Documentation

[L4_utcb_consts_x86](#)

enum [L4_utcb_consts_x86](#)

UTCB constants for x86.

Enumerator

L4_UTCB_EXCEPTION_REGS_SIZE	Number if message registers used for exception IPC.
L4_UTCB_GENERIC_DATA_SIZE	Total number of message register (MRs) available.
L4_UTCB_GENERIC_BUFFERS_SIZE	Total number of buffer registers (BRs) available.
L4_UTCB_MSG_REGS_OFFSET	Offset of MR[0] relative to the UTCB pointer.
L4_UTCB_BUF_REGS_OFFSET	Offset of BR[0] relative to the UTCB pointer.
L4_UTCB_THREAD_REGS_OFFSET	Offset of TCR[0] relative to the UTCB pointer.
L4_UTCB_INHERIT_FPU	BDR flag to accept reception of FPU state.
L4_UTCB_OFFSET	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_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.

13.2.1 Detailed Description

13.2.2 Enumeration Type Documentation

13.2.2.1 Libedid_consts

enum [Libedid_consts](#)

EDID constants.

Enumerator

Libedid_block_size	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 l4io_request_iomem_region()
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 l4io_resource_types_t).
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 l4io_request_iomem
<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 l4io_iomem_flags_t
<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 l4io_iomem_flags_t

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

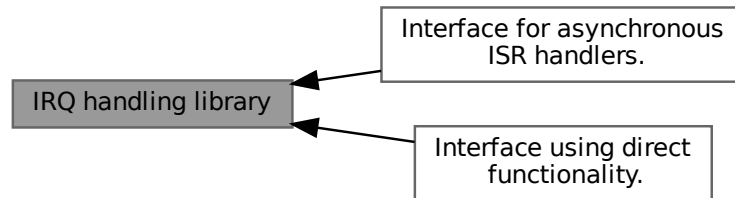
	<i>devhandle</i>	Device handle.
<i>in, out</i>	<i>reshandle</i>	Resource handle from which IO memory should be requested. Upon successful completion 'reshandle' points to the device's next resource.

Return values

0	An error occurred. The value of 'reshandle' is undefined.
>0	The virtual address of the IO memory mapping.

13.4 IRQ handling library

Collaboration diagram for IRQ handling library:



Modules

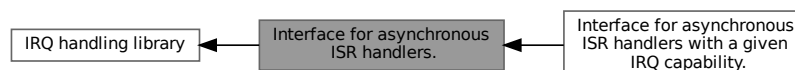
- [Interface for asynchronous ISR handlers.](#)
This interface has just two (main) functions.
- [Interface using direct functionality.](#)

13.4.1 Detailed Description

13.4.2 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.

13.4.2.1 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.

13.4.2.2 Function Documentation

13.4.2.2.1 `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](#).

13.4.2.2.2 `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>	L4 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

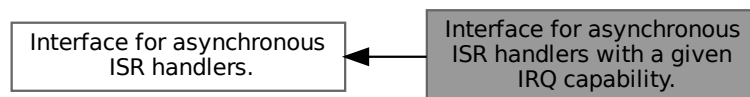
Examples

[examples/libs/libirq/async_isr.c](#).

13.4.2.3 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.

13.4.2.3.1 Detailed Description

This group is just an enhanced version to [l4irq_request\(\)](#) which takes a capability object instead of a plain number.

13.4.2.3.2 Function Documentation

13.4.2.3.2.1 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>	L4 thread priority of the ISR handler. Give -1 for same priority as creator.
<i>mode</i>	Interrupt type,

See also

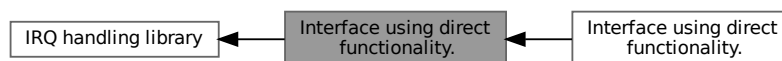
[L4_irq_mode](#)

Returns

Pointer to `l4irq_t` structure, 0 on error

13.4.3 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.

13.4.3.1 Detailed Description

13.4.3.2 Function Documentation

13.4.3.2.1 `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](#).

13.4.3.2.2 `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.

13.4.3.2.3 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.

13.4.3.2.4 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.

13.4.3.2.5 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

13.4.3.2.6 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`.

13.4.3.2.7 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

13.4.3.2.8 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](#).

13.4.3.2.9 l4irq_wait_any()

```
long l4irq_wait_any (
    l4irq_t ** irq )
```

Wait for any attached IRQ.

Return values

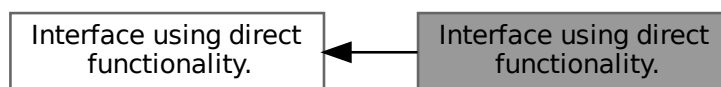
<i>irq</i>	Received interrupt.
------------	---------------------

Returns

0 on success, != 0 on error

13.4.3.3 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.

13.4.3.3.1 Detailed Description

13.4.3.3.2 Function Documentation

13.4.3.3.2.1 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.

13.4.3.3.2.2 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.

13.4.3.3.2.3 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.

13.4.3.3.2.4 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.

13.5 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 }

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.

13.5.1 Detailed Description

List of protocol specific opcodes used for communication with [L4Re](#) and Kernel objects.

13.5.2 Enumeration Type Documentation

13.5.2.1 L4_icu_opcode

enum `L4_icu_opcode`

Opcodes to the ICU interface.

Enumerator

<code>L4_ICU_OP_BIND</code>	Bind opcode. See also l4_icu_bind()
<code>L4_ICU_OP_UNBIND</code>	Unbind opcode. See also l4_icu_unbind()

Enumerator

L4_ICU_OP_INFO	Info opcode. See also l4_icu_info()
L4_ICU_OP_MSI_INFO	Msi-info opcode. See also l4_icu_msi_info()
L4_ICU_OP_UNMASK	Unmask opcode. See also l4_icu_unmask()
L4_ICU_OP_MASK	Mask opcode. See also l4_icu_mask()
L4_ICU_OP_SET_MODE	Set-mode opcode. See also l4_icu_set_mode()

Definition at line 106 of file [icu.h](#).

13.5.2.2 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](#).

13.5.2.3 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

Definition at line 166 of file [platform_control.h](#).

13.5.2.4 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 310 of file [task.h](#).

13.5.2.5 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 690 of file [thread.h](#).

13.5.2.6 L4_vcon_ops

enum [L4_vcon_ops](#)

Operations on vcon objects.

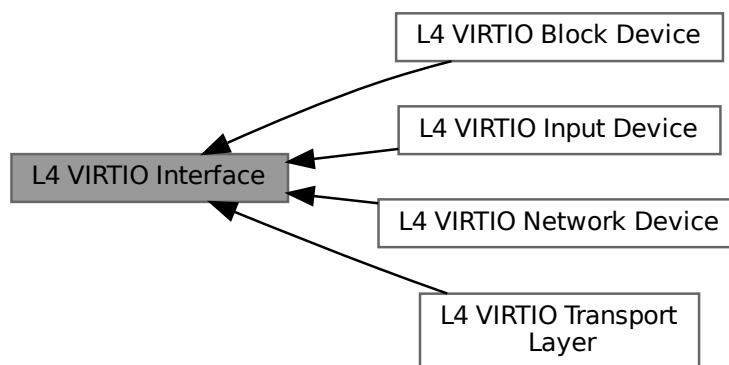
Enumerator

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](#).

13.6 L4 VIRTIO Interface

Collaboration diagram for L4 VIRTIO Interface:



Modules

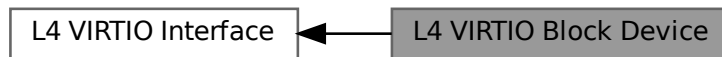
- [L4 VIRTIO Block Device](#)
- [L4 VIRTIO Input Device](#)
- [L4 VIRTIO Network Device](#)
- [L4 VIRTIO Transport Layer](#)

[L4](#) specific VIRTIO Transport layer.

13.6.1 Detailed Description

13.6.2 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.

13.6.2.1 Detailed Description

13.6.2.2 Enumeration Type Documentation

13.6.2.2.1 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 31 of file [virtio_block.h](#).

13.6.2.2.2 L4virtio_block_status

```
enum L4virtio_block_status
```

Status of a finished block request.

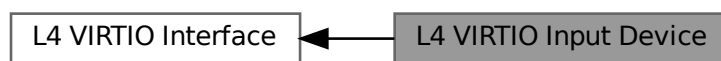
Enumerator

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 44 of file [virtio_block.h](#).

13.6.3 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.

13.6.3.1 Detailed Description

13.6.4 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

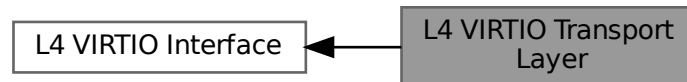
- enum [L4virtio_net_feature_bits](#)
Network device feature bits.

13.6.4.1 Detailed Description

13.6.5 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_RPMMSG` = 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_CRYPTIO` = 20 , `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`
= 224 }
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_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`

13.6.5.1 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.

13.6.5.2 Typedef Documentation

13.6.5.2.1 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.

13.6.5.3 Enumeration Type Documentation

13.6.5.3.1 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_MASK</code>	Mask to get command bits.

Definition at line 114 of file [virtio.h](#).

13.6.5.3.2 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 105 of file [virtio.h](#).

13.6.5.3.3 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](#).

13.6.5.3.4 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_SOCKET	Unofficial socket device.

Definition at line 61 of file [virtio.h](#).

13.6.5.3.5 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 82 of file [virtio.h](#).

13.6.5.3.6 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 L4virtio .
L4VIRTIO_FEATURE_CMD_CONFIG	Status and queue config are set via cmd field instead of via IPC.

Definition at line 93 of file [virtio.h](#).

13.6.5.4 Function Documentation

13.6.5.4.1 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.

13.6.5.4.2 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 235 of file [virtio.h](#).

References [l4virtio_config_hdr_t::queues_offset](#).

13.6.5.4.3 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 246 of file [virtio.h](#).

13.6.5.4.4 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.

13.6.5.4.5 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><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.

13.6.5.4.6 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><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.

13.6.5.4.7 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.
---	-------------

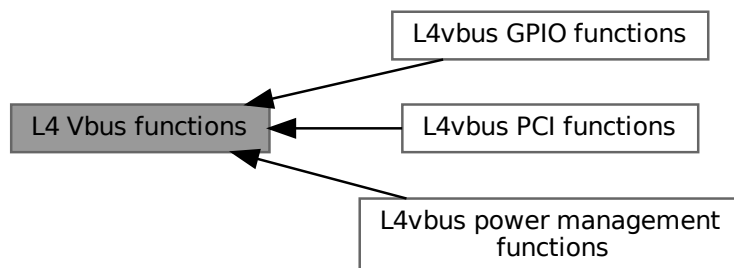
Note

All other registers are accessed via shared memory.

13.7 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, int 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.

13.7.1 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.

13.7.2 Enumeration Type Documentation

13.7.2.1 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 L4Re::Dma_space .
L4VBUS_DMAD_KERNEL_DMA_SPACE	The given DMA space is a kernel DMA space (L4::Task)

Definition at line 176 of file [vbus.h](#).

13.7.3 Function Documentation

13.7.3.1 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

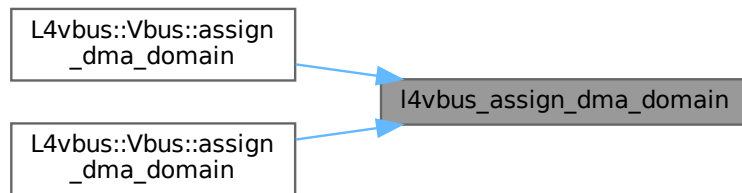
<i>vbus</i>	Capability of the system bus
<i>domain_id</i>	DMA domain ID (resource address of DMA domain found on the vBUS). If the value is ~0U the DMA space of the whole vBUS is used.
<i>flags</i>	A combination of L4vbus_dma_domain_assign_flags .
<i>dma_space</i>	The DMA space capability to bind or unbind, this must either be an L4Re::Dma_space or a kernel DMA space (L4::Task 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:



13.7.3.2 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.

13.7.3.3 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 <code>dev</code> could be found.

Referenced by [L4vbus::Device::device\(\)](#).

Here is the caller graph for this function:



13.7.3.4 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

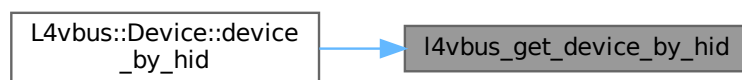
<i>in, out</i>	<i>child</i>	Handle of the device from where in the device tree the search should start. To start searching from the beginning <code>child</code> must be initialized using the default (L4VBUS_NULL). If a matching device is found, its handle is returned through this parameter.
	<i>hid</i>	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

≥ 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.

Referenced by [L4vbus::Device::device_by_hid\(\)](#).

Here is the caller graph for this function:



13.7.3.5 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

13.7.3.6 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

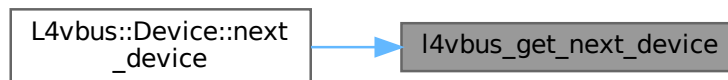
	<i>vbus</i>	Capability of the system bus
	<i>parent</i>	Handle to the parent device (use L4VBUS_ROOT_BUS for the system bus)
in, out	<i>child</i>	Handle of the device that precedes the device that shall be returned. To start from the beginning, <i>child</i> must be initialized with L4VBUS_NULL . If a 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:



13.7.3.7 l4vbus_get_resource()

```

int l4vbus_get_resource (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    int 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 l4vbus_get_device_by_hid() and l4vbus_get_next_device() 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 l4vbus_device_t structure that is returned by L4vbus::Device::device_by_hid() and L4vbus::Device::next_device() .
out	<i>res</i>	Descriptor of the resource.

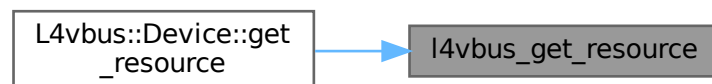
This function returns the resource descriptor of an individual device resource selected by the `res_idx` parameter.

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:



13.7.3.8 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

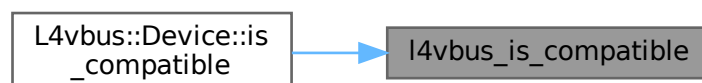
<i>vbus</i>	Capability of the system bus
<i>dev</i>	device handle for which the CID shall be tested
<i>cid</i>	the compatibility ID to test

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:



13.7.3.9 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

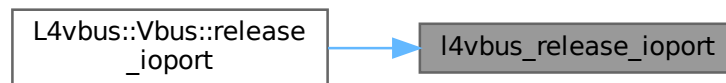
	<i>vbus</i>	Capability of the system bus.
in	<i>res</i>	The IO port resource to be released from the bus.

Returns

>=0 on success, <0 on error.

Referenced by [L4vbus::Vbus::release_ioport\(\)](#).

Here is the caller graph for this function:



13.7.3.10 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:



13.7.3.11 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

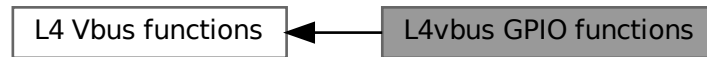
Referenced by [L4vbus::Icu::vicu\(\)](#).

Here is the caller graph for this function:



13.7.4 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.

13.7.4.1 Detailed Description

13.7.4.2 Enumeration Type Documentation

13.7.4.2.1 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](#).

13.7.4.2.2 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](#).

13.7.4.3 Function Documentation

13.7.4.3.1 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:



13.7.4.3.2 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:



13.7.4.3.3 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 L4vbus_gpio_pull_modes

Returns

0 if OK, error code otherwise

Referenced by [L4vbus::Gpio_pin::config_pull\(\)](#).

Here is the caller graph for this function:



13.7.4.3.4 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

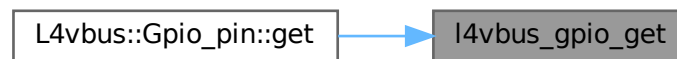
<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin number to read from

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:



13.7.4.3.5 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:



13.7.4.3.6 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:



13.7.4.3.7 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:

**13.7.4.3.8 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

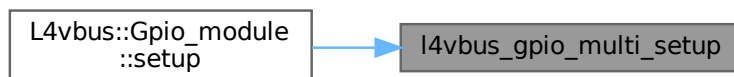
<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>offset</i>	Pin corresponding to the LSB in <i>mask</i> . Note: allowed may be hardware specific.
<i>mask</i>	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation.
<i>mode</i>	GPIO function, see L4vbus_gpio_generic_func 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:



13.7.4.3.9 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:



13.7.4.3.10 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 L4vbus_gpio_generic_func 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:



13.7.4.3.11 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

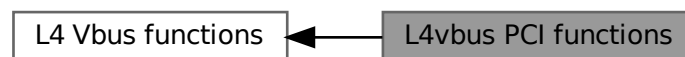
Referenced by [L4vbus::Gpio_pin::to_irq\(\)](#).

Here is the caller graph for this function:



13.7.5 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.

13.7.5.1 Detailed Description

13.7.5.2 Function Documentation

13.7.5.2.1 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

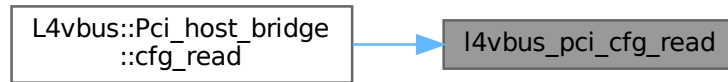
	<i>vbus</i>	Capability of the system bus
	<i>handle</i>	Device handle of the PCI root bridge
	<i>bus</i>	Bus number
	<i>devfn</i>	Device id (upper 16bit) and function (lower 16bit)
	<i>reg</i>	Register in configuration space to read
out	<i>value</i>	Value that has been read
	<i>width</i>	Width to read 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:



13.7.5.2.2 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:

**13.7.5.2.3 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:



13.7.5.2.4 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:



13.7.5.2.5 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

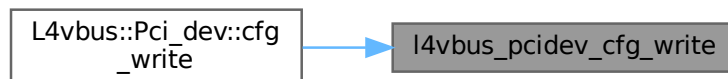
<i>vbus</i>	Capability of the system bus
<i>handle</i>	Device handle of the PCI device
<i>reg</i>	Register in configuration space to write
<i>value</i>	Value to write
<i>width</i>	Width to write in bits (e.g. 8, 16, 32)

Returns

0 on success, else failure

Referenced by [L4vbus::Pci_dev::cfg_write\(\)](#).

Here is the caller graph for this function:



13.7.5.2.6 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

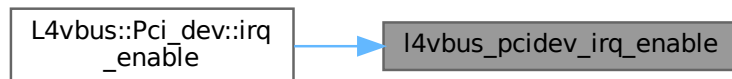
	<i>vbus</i>	Capability of the system bus
	<i>handle</i>	Device handle of the PCI device
out	<i>trigger</i>	False if interrupt is level-triggered
out	<i>polarity</i>	True if interrupt is of low polarity

Returns

On success: Interrupt line to be used, else failure

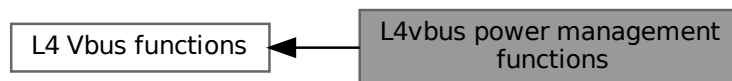
Referenced by [L4vbus::Pci_dev::irq_enable\(\)](#).

Here is the caller graph for this function:



13.7.6 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.

13.7.6.1 Detailed Description

13.7.6.2 Function Documentation

13.7.6.2.1 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:

**13.7.6.2.2 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\(\)](#).

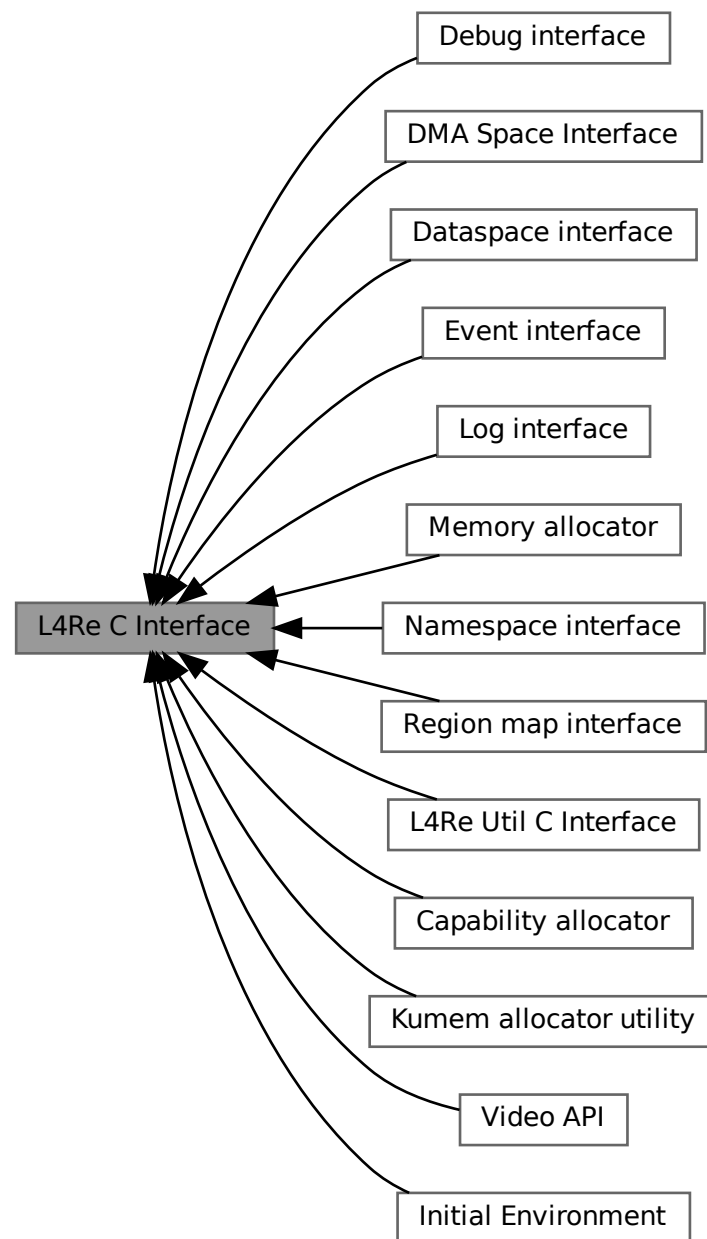
Here is the caller graph for this function:



13.8 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.

- [Region map interface](#)

Region map C interface.

- [Video API](#)

Files

- file [inhibitor.h](#)

Inhibitor C interface.

13.8.1 Detailed Description

Documentation for the [L4Re](#) C Interface.

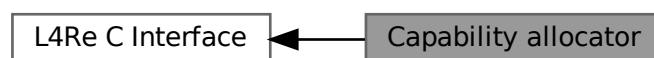
The interface functions closely align with the C++ functions and add no further functionalities.

For new programs it is advised to use the C++ interface.

13.8.2 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.

13.8.2.1 Detailed Description

Capability allocator C interface.

13.8.2.2 Function Documentation

13.8.2.2.1 l4re_util_cap_last()

```
long l4re_util_cap_last (
    void )
```

Return last capability index the allocator can return.

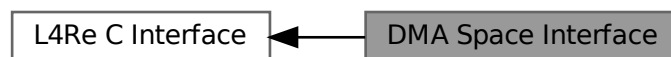
Returns

last/biggest capability index the allocator can return

13.8.3 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.

13.8.3.1 Detailed Description

DMA Space C interface.

13.8.3.2 Typedef Documentation

13.8.3.2.1 [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](#).

13.8.3.3 Function Documentation

13.8.3.3.1 [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) DMA space 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 L4Re::Dma_space::Phys_space 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 L4Re::Dma_space::Space_attrib .

Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EPERM</i>	No L4_CAP_FPAGE_W right on the <i>Dma_space</i> capability.
<i>-L4_EINVAL</i>	
<i>-L4_ENOENT</i>	

Precondition

requires capability rights: {RW}

13.8.3.3.2 [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 L4_CAP_FPAGE_W right on the <i>Dma_space</i> capability.
<i>-L4_ENOENT</i>	

Precondition

requires capability rights: {RW}

13.8.3.3.3 [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()`).

13.8.3.3.4 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.

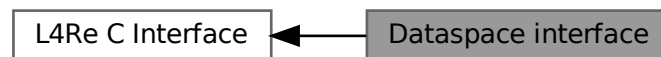
Precondition

requires capability rights: {R}

13.8.4 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.

13.8.4.1 Detailed Description

Dataspace C interface.

13.8.4.2 Enumeration Type Documentation

13.8.4.2.1 l4re_ds_map_flags

```
enum l4re\_ds\_map\_flags
```

Flags to specify the memory mapping type of a request.

Enumerator

L4RE_DS_F_NORMAL	request normal memory mapping
L4RE_DS_F_CACHEABLE	request normal memory mapping
L4RE_DS_F_BUFFERABLE	request bufferable (write buffered) mappings
L4RE_DS_F_UNCACHEABLE	request uncacheable memory mappings
L4RE_DS_F_CACHING_MASK	mask for caching flags
L4RE_DS_F_CACHING_SHIFT	shift value for caching flags

Definition at line 58 of file [dataspace.h](#).

13.8.4.3 Function Documentation

13.8.4.3.1 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><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.

13.8.4.3.2 `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>>=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 <code>L4_CAP_FPAGE_W</code> right on dataspace capability.
<i><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.

13.8.4.3.3 `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 L4_CAP_FPAGE_W right on the destination dataspace.
<i>-L4_EINVAL</i>	Invalid parameter supplied.
<i><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.

13.8.4.3.4 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>>=0</i>	Flags of the dataspace
<i><0</i>	IPC errors

See also

[L4Re::Dataspace::F::Flags](#)

13.8.4.3.5 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.
<i>out</i>	<i>stats</i>	Dataspace information

Return values

0	Success
<0	IPC errors

13.8.4.3.6 l4re_ds_size()

```
l4re_ds_size_t l4re_ds_size (  
    l4re_ds_t ds )
```

Get size of a dataspace.

Parameters

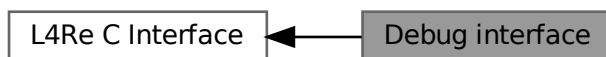
<i>ds</i>	Dataspace capability.
-----------	-----------------------

Returns

Size of the dataspace in bytes.

13.8.5 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.

13.8.5.1 Detailed Description

13.8.5.2 Function Documentation

13.8.5.2.1 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](#)

13.8.6 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.

13.8.6.1 Detailed Description

Event C interface.

13.8.6.2 Function Documentation

13.8.6.2.1 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](#)

13.8.6.2.2 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](#)

13.8.6.2.3 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](#)

13.8.6.2.4 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.
<i>out</i>	<i>info</i>	Information buffer.

Returns

0 for success, <0 on error

See also

[L4Re::Event::get_stream_info](#)

13.8.6.2.5 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</i> ↔ <i>_id</i>	Stream ID.
out	<i>info</i>	Information buffer.

Returns

0 for success, <0 on error

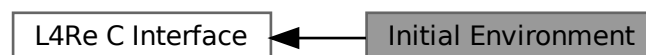
See also

[L4Re::Event::get_stream_info_for_id](#)

13.8.7 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.

13.8.7.1 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](#).

13.8.7.2 Function Documentation

13.8.7.2.1 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:



13.8.7.2.2 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.
-------------	---

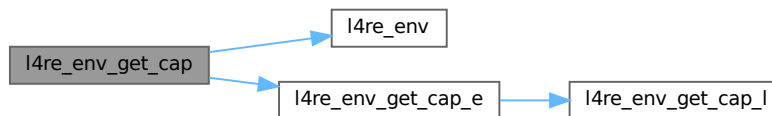
Returns

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:



13.8.7.2.3 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:



13.8.7.2.4 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

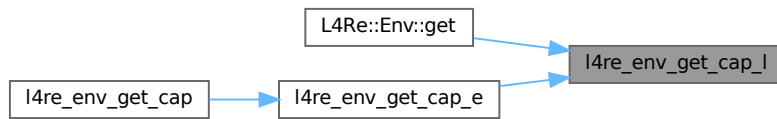
A pointer to an [l4re_env_cap_entry_t](#) if the object exists or NULL if not.

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:



13.8.7.2.5 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](#).

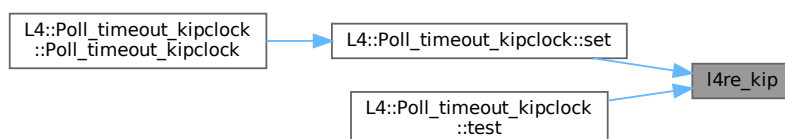
References [l4_kip\(\)](#).

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:



13.8.8 Kumem allocator utility

Kumem allocator utility C interface.

Collaboration diagram for Kumem allocator utility:

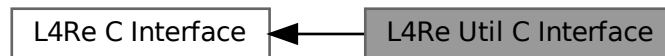


Kumem allocator utility C interface.

13.8.9 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.

13.8.10 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.

13.8.10.1 Detailed Description

Log C interface.

13.8.10.2 Function Documentation

13.8.10.2.1 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:



13.8.10.2.2 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

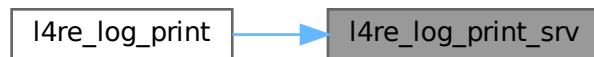
<i>logcap</i>	Log capability (service).
<i>string</i>	Text to print, null terminated.

See also

[L4Re::Log::print](#)

Referenced by [l4re_log_print\(\)](#).

Here is the caller graph for this function:



13.8.10.2.3 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:



13.8.10.2.4 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\(\)](#).

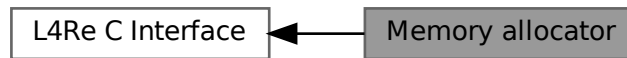
Here is the caller graph for this function:



13.8.11 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.

13.8.11.1 Detailed Description

Memory allocator C interface.

13.8.11.2 Enumeration Type Documentation

13.8.11.2.1 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](#).

13.8.11.3 Function Documentation

13.8.11.3.1 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 l4re_ma_flags

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:



13.8.11.3.2 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 l4re_ma_flags
<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><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:



13.8.11.3.3 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 l4re_ma_flags
<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

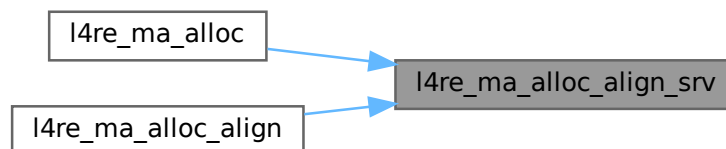
See also

[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:



13.8.12 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.

13.8.12.1 Detailed Description

Namespace C interface.

13.8.12.2 Enumeration Type Documentation

13.8.12.2.1 [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](#).

13.8.12.3 Function Documentation

13.8.12.3.1 [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 l4_error_code_t .

Definition at line 105 of file [namespace.h](#).

References [l4re_ns_query_to_srv\(\)](#).

Here is the call graph for this function:



13.8.12.3.2 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>>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><0</i>	IPC errors, see l4_error_code_t .

Referenced by [l4re_ns_query_srv\(\)](#).

Here is the caller graph for this function:



13.8.12.3.3 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 L4Re::Namespace::Register_flags . 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 L4_CAP_FPAGE_W right on the invoked capability.
<i>-L4_ENOMEM</i>	Server has insufficient resources.
<i>-L4_EINVAL</i>	Invalid parameter.
<i><0</i>	IPC errors, see l4_error_code_t .

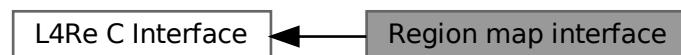
Precondition

requires capability rights: {RW}

13.8.13 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` = `0x20000` ,
`L4RE_RM_F_IN_AREA` = `0x40000` , `L4RE_RM_F_EAGER_MAP` = `0x80000` , `L4RE_RM_F_ATTACH_FLAGS`
= `0xf0000` }

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.

13.8.13.1 Detailed Description

Region map C interface.

13.8.13.2 Enumeration Type Documentation

13.8.13.2.1 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.
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_ATTACH_FLAGS	Mask of all attach flags.

Definition at line 40 of file [rm.h](#).

13.8.13.3 Function Documentation

13.8.13.3.1 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

in, out	<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 L4Re::Rm::F::Search_addr 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 L4Re::Rm::F::In_area is given the value is used as a selector for the area (see L4Re::Rm::reserve_area) 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 L4Re::Rm::F::Attach_flags and L4Re::Rm::F::Region_flags . 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 L4Re::Dataspace::map 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 (L4_PAGESHIFT). This is only meaningful if the L4Re::Rm::F::Search_addr flag is used.

Return values

0	Success
-L4_ENOENT	No area could be found (see L4Re::Rm::F::In_area)
-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 286 of file `rm.h`.

References [l4re_rm_attach_srv\(\)](#), and [l4re_env_t::rm](#).

Here is the call graph for this function:



13.8.13.3.2 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:



13.8.13.3.3 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

L4Re::Rm::Detach_result	On success.
<code>-L4_ENOENT</code>	No region found.
<code><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 296 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:



13.8.13.3.4 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.
<i>out</i>	<i>ds</i>	Returns dataspace that is affected.

Return values

L4Re::Rm::Detach_result	On success.
<code>-L4_ENOENT</code>	No region found.
<code><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 309 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:



13.8.13.3.5 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 <code>L4_INVALID_CAP</code> 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 316 of file [rm.h](#).

References [l4re_rm_detach_srv\(\)](#), and [l4re_env_t::rm](#).

Here is the call graph for this function:

**13.8.13.3.6 l4re_rm_detach_srv()**

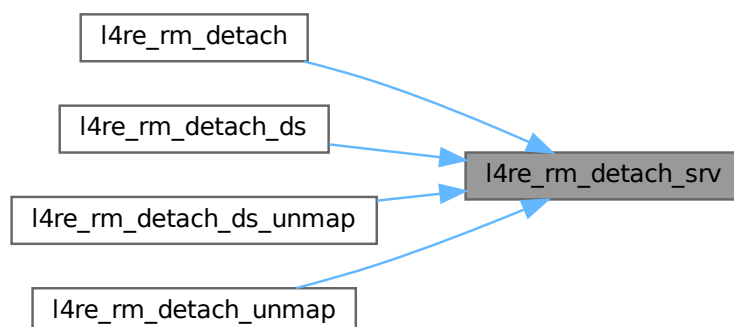
```
int l4re_rm_detach_srv (  
    l4_cap_idx_t rm,  
    l4_addr_t addr,  
    l4re_ds_t * ds,  
    l4_cap_idx_t task )
```

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:



13.8.13.3.7 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 303 of file [rm.h](#).

References [l4re_rm_detach_srv\(\)](#), and [l4re_env_t::rm](#).

Here is the call graph for this function:

**13.8.13.3.8 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.

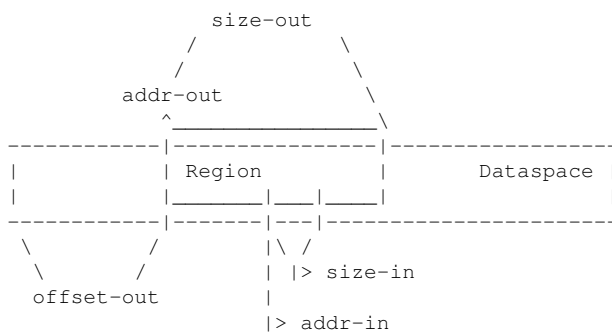
Parameters

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::In_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 323 of file [rm.h](#).

References [l4re_rm_find_srv\(\)](#), and [l4re_env_t::rm](#).

Here is the call graph for this function:



13.8.13.3.9 l4re_rm_find_srv()

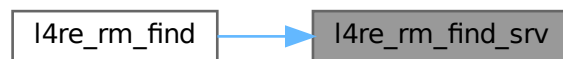
```
int l4re_rm_find_srv (
    l4_cap_idx_t rm,
    l4_addr_t * addr,
    unsigned long * size,
    l4re_rm_offset_t * offset,
    l4re_rm_flags_t * flags,
    l4re_ds_t * m )
```

See also

[L4Re::Rm::find](#)

Referenced by [l4re_rm_find\(\)](#).

Here is the caller graph for this function:



13.8.13.3.10 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.

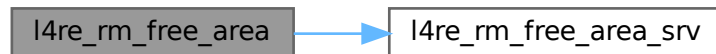
[L4Re::Rm::free_area](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 280 of file `rm.h`.

References [l4re_rm_free_area_srv\(\)](#), and [l4re_env_t::rm](#).

Here is the call graph for this function:



13.8.13.3.11 l4re_rm_free_area_srv()

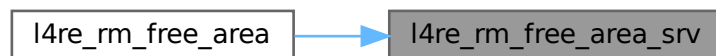
```
int l4re_rm_free_area_srv (
    l4_cap_idx_t rm,
    l4_addr_t addr )
```

See also

[L4Re::Rm::free_area](#)

Referenced by [l4re_rm_free_area\(\)](#).

Here is the caller graph for this function:



13.8.13.3.12 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 L4Re::Rm::F::Region_flags and L4Re::Rm::F::Attach_flags).
	<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><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 272 of file [rm.h](#).

References [l4re_rm_reserve_area_srv\(\)](#), and [l4re_env_t::rm](#).

Here is the call graph for this function:



13.8.13.3.13 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:



13.8.13.3.14 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 331 of file [rm.h](#).

References [l4re_rm_show_lists_srv\(\)](#), and [l4re_env_t::rm](#).

Here is the call graph for this function:



13.8.14 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.

13.8.14.1 Detailed Description

13.8.14.2 Typedef Documentation

13.8.14.2.1 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.

13.8.14.3 Enumeration Type Documentation

13.8.14.3.1 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](#).

13.8.14.3.2 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](#).

13.8.14.4 Function Documentation

13.8.14.4.1 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

≥ 0 : The index of the created buffer (used to assign views and for deletion). < 0 : on error

13.8.14.4.2 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.

13.8.14.4.3 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 l4re_video_goos_create_buffer())

13.8.14.4.4 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.

13.8.14.4.5 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.

13.8.14.4.6 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\(\)](#).

13.8.14.4.7 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

13.8.14.4.8 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

13.8.14.4.9 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.

13.8.14.4.10 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

13.8.14.4.11 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.

13.8.14.4.12 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.

13.8.14.4.13 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.

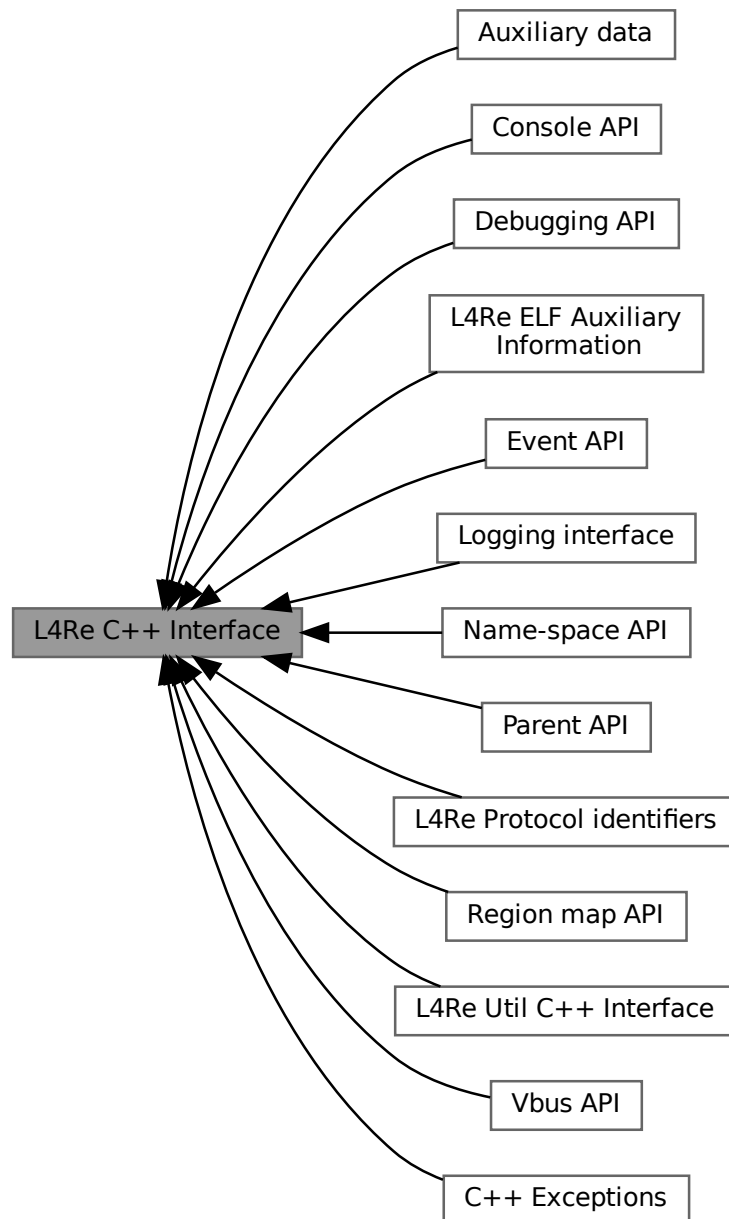
Parameters

<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.

13.9 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.

13.9.1 Detailed Description

Documentation of the [L4](#) Runtime Environment C++ API.

13.9.2 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.

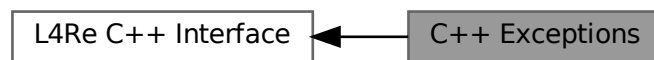
Enumerations

- enum [l4re_aux_ldr_flags_t](#)
Flags for program loading.

13.9.2.1 Detailed Description

13.9.3 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

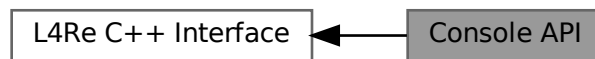
- `#define L4_CXX_EXCEPTION_BACKTRACE 20`
Number of instruction pointers in backtrace.

13.9.3.1 Detailed Description

13.9.4 Console API

[Console](#) interface.

Collaboration diagram for Console API:



Data Structures

- class [L4Re::Console](#)
[Console](#) class.

13.9.4.1 Detailed Description

[Console](#) interface.

13.9.5 Debugging API

Debugging Interface.

Collaboration diagram for Debugging API:



Data Structures

- class [L4Re::Debug_obj](#)
Debug interface.

13.9.5.1 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.

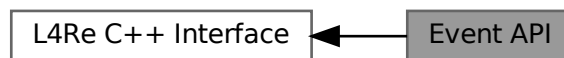
See also

[L4::Debug_obj](#) for more information.

13.9.6 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.

13.9.6.1 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](#).

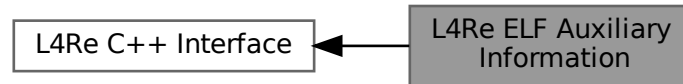
This interface is usually not used directly. Instead use [L4Re::Util::Event_t](#) for clients. An example server portion is implemented in [L4Re::Util::Event_svr](#).

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.

13.9.7 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](#) }

13.9.7.1 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.

13.9.7.2 Macro Definition Documentation

13.9.7.2.1 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](#).

13.9.7.2.2 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., l4re_elf_aux_vma_t)
<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., L4RE_ELF_AUX_T_VMA
<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](#).

13.9.7.3 Enumeration Type Documentation

13.9.7.3.1 anonymous enum

anonymous enum

Enumerator

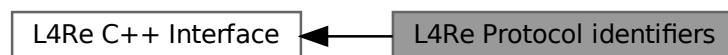
L4RE_ELF_AUX_T_NONE	Tag for an invalid element in the auxiliary vector.
L4RE_ELF_AUX_T_VMA	Tag for descriptor for a reserved virtual memory area.
L4RE_ELF_AUX_T_STACK_SIZE	Tag for descriptor that defines the stack size for the first application thread.
L4RE_ELF_AUX_T_STACK_ADDR	Tag for descriptor that 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.

Definition at line 70 of file [elf_aux.h](#).

13.9.8 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.

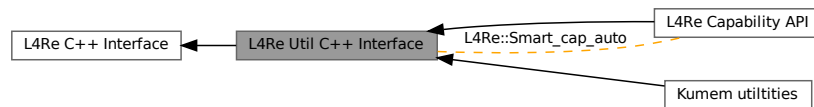
13.9.8.1 Detailed Description

Fix [L4Re](#) Protocol Constants.

13.9.9 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 >](#)
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.

13.9.9.1 Detailed Description

Documentation of the [L4](#) Runtime Environment utility functionality in C++.

13.9.9.2 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.

13.9.9.2.1 Detailed Description

13.9.9.2.2 Function Documentation

13.9.9.2.2.1 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:

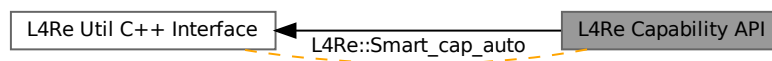


Here is the caller graph for this function:



13.9.9.3 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.

13.9.9.3.1 Detailed Description

13.9.9.3.2 Function Documentation

13.9.9.3.2.1 [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

T	Type of capability the slot is used for.
-------------------	--

Definition at line 218 of file [cap_alloc](#).

References [L4Re::Util::cap_alloc](#).

13.9.9.3.2.2 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 227 of file [cap_alloc](#).

References [L4Re::Util::cap_alloc](#).

13.9.9.3.3 Variable Documentation

13.9.9.3.3.1 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 actual implementation of the allocator depends on the configuration of the system.

Per default we use [Counting_cap_alloc](#), a reference-counting capability allocator, that keeps a reference counter for each managed capability selector.

Note

This capability allocator is not thread-safe.

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\(\)](#).

13.9.10 Logging interface

Interface for log output.

Collaboration diagram for Logging interface:



Data Structures

- class [L4Re::Log](#)
Log interface class.

13.9.10.1 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.

13.9.11 Name-space API

API for name spaces that store capabilities.

Collaboration diagram for Name-space API:



Data Structures

- class [L4Re::Namespace](#)
Name-space interface.

13.9.11.1 Detailed Description

API for name spaces that store capabilities.

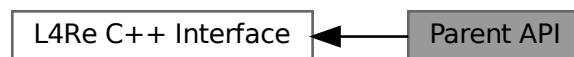
This is a basic abstraction for managing a mapping from human-readable names to capabilities. In particular, a name can also be mapped to a capability that refers to another name space object. By this means name spaces can be constructed hierarchically.

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.

13.9.12 Parent API

[Parent](#) interface.

Collaboration diagram for Parent API:



Data Structures

- class [L4Re::Parent](#)
[Parent](#) interface.

13.9.12.1 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.

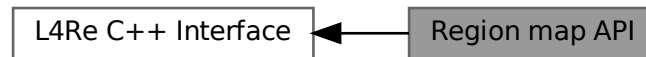
See also

[L4Re::Parent](#) for information about the concrete interface.

13.9.13 Region map API

Virtual address-space management.

Collaboration diagram for Region map API:



Data Structures

- class [L4Re::Rm](#)
Region map.

13.9.13.1 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.

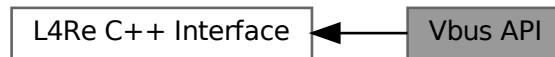
See also

[L4Re::Dataspace](#), [L4Re::Rm](#),
[Memory management - Data Spaces and the Region Map](#)

13.9.14 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.

13.9.14.1 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>
```

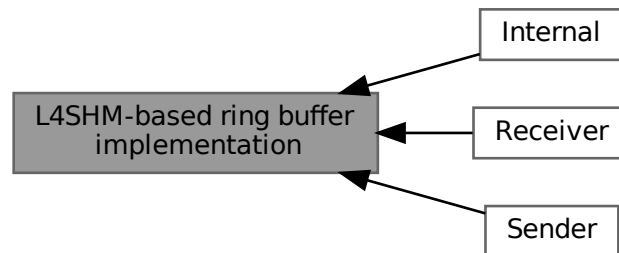
Include File

```
#include <l4/vbus/vbus_pci>
```

13.10 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](#)

13.10.1 Detailed Description

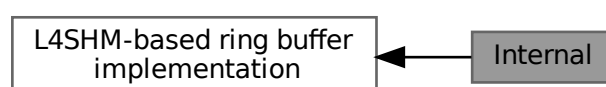
The library provides a non-locking (strictly 1:1) shared-memory-based ring buffer implementation based on the L4SHM library.

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.

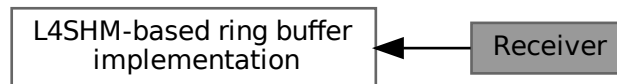
13.10.2 Internal

Collaboration diagram for Internal:



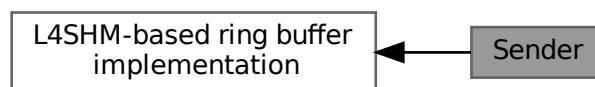
13.10.3 Receiver

Collaboration diagram for Receiver:



13.10.4 Sender

Collaboration diagram for Sender:



13.11 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::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.
- 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.

Enumerations

- enum [L4::lpc_svr::Reply_mode](#) { [L4::lpc_svr::Reply_compound](#) , [L4::lpc_svr::Reply_separate](#) }
Reply mode for server loop.

13.11.1 Detailed Description

Server-Side framework for implementing object-oriented servers.

,

13.11.2 Enumeration Type Documentation

13.11.2.1 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

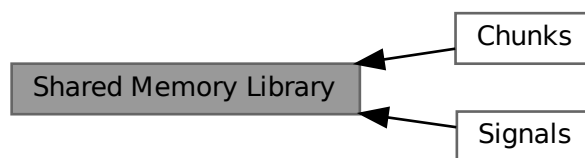
Reply_compound	Server shall use a compound reply and wait (fast).
Reply_separate	Server shall call reply and wait separately.

Definition at line 50 of file [ipc_server_loop](#).

13.12 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.

13.12.1 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).

13.12.2 Function Documentation

13.12.2.1 `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.

13.12.2.2 `l4shmc_area_size()`

```
long l4shmc_area_size (
    l4shmc_area_t const * shmarea )
```

Get size of shared memory area.

Parameters

<code>shmarea</code>	Shared memory area.
----------------------	---------------------

Return values

>0	Size of the shared memory area.
<0	Error.

13.12.2.3 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.

13.12.2.4 l4shmc_attach()

```
long l4shmc_attach (
    char const * shm_name,
    l4shmc_area_t * shmarea )
```

Attach to a shared memory area.

Parameters

	<i>shm_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](#).

13.12.2.5 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.

13.12.2.6 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](#).

13.12.2.7 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](#).

13.12.2.8 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.

13.12.2.9 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](#).

13.12.2.10 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

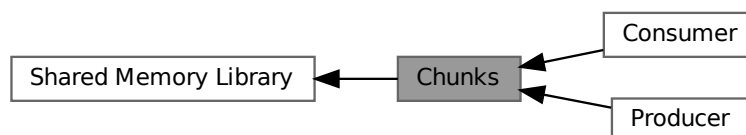
0	Success.
<0	Error.

Examples

[examples/libs/shmc/prodcons.c](#).

13.12.3 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.

13.12.3.1 Detailed Description

13.12.3.2 Function Documentation

13.12.3.2.1 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](#).

13.12.3.2.2 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.

13.12.3.2.3 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](#).

13.12.3.2.4 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.
---	--

Return values

<code>!=0</code>	Pointer to signal otherwise.
------------------	------------------------------

13.12.3.2.5 `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

<code>0</code>	Success.
<code><0</code>	Error.

Examples

[examples/libs/shmc/prodcons.c](#).

13.12.3.2.6 `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

<code>0</code>	Success.
----------------	----------

Return values

<0	Error.
------	--------

13.12.3.2.7 l4shmc_iterate_chunk()

```
long l4shmc_iterate_chunk (
    l4shmc_area_t const * shmarea,
    char const ** chunk_name,
    long offs )
```

Iterate over names of all existing chunks.

Parameters

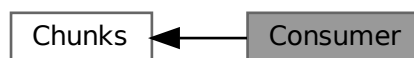
<i>shmarea</i>	Shared memory area.
<i>chunk_name</i>	Where the name of the current chunk will be stored
<i>offs</i>	0 to start iteration, return value of previous call to l4shmc_iterate_chunk() to get next chunk

Return values

0	No more chunks available.
<0	Error.
>0	Iterator value for the next call.

13.12.3.3 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.

13.12.3.3.1 Detailed Description

13.12.3.3.2 Function Documentation

13.12.3.3.2.1 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](#).

13.12.3.3.2.2 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](#).

13.12.3.3.2.3 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.

13.12.3.3.2.4 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.

13.12.3.3.2.5 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.

13.12.3.3.2.6 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><0</i>	Error.

Examples

[examples/libs/shmc/prodcons.c](#).

13.12.3.3.2.7 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

<i>0</i>	Success.
<i><0</i>	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.

13.12.3.3.2.8 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.

13.12.3.4 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.

13.12.3.4.1 Detailed Description

13.12.3.4.2 Function Documentation

13.12.3.4.2.1 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

0	Success.
<0	Error.

13.12.3.4.2.2 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

0	Success.
<0	Error.

Examples

[examples/libs/shmc/prodcons.c](#).

13.12.3.4.2.3 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](#).

13.12.3.4.2.4 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.

13.12.3.4.2.5 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.

13.12.3.4.2.6 l4shmc_is_chunk_clear()

```
long l4shmc_is_chunk_clear (
    l4shmc_chunk_t const * chunk ) [inline]
```

Check whether chunk is free.

Parameters

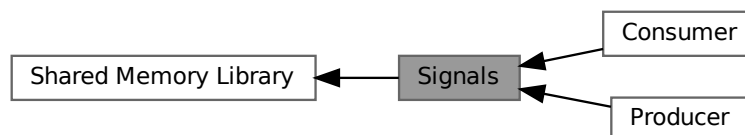
<i>chunk</i>	Chunk to check.
--------------	-----------------

Return values

<i>!=0</i>	Chunk is clear.
0	Chunk is not clear.

13.12.4 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.

13.12.4.1 Detailed Description

13.12.4.2 Function Documentation

13.12.4.2.1 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](#).

13.12.4.2.2 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](#).

13.12.4.2.3 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.

13.12.4.2.4 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.

13.12.4.2.5 l4shmc_signal_cap()

```
l4_cap_idx_t l4shmc_signal_cap (
    l4shmc_signal_t const * signal ) [inline]
```

Get the signal capability of a signal.

Parameters

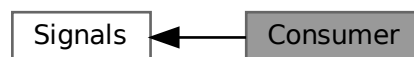
<i>signal</i>	Signal.
---------------	---------

Returns

Capability of the signal object.

13.12.4.3 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.

13.12.4.3.1 Detailed Description

13.12.4.3.2 Function Documentation

13.12.4.3.2.1 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.

13.12.4.3.2.2 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.

13.12.4.3.2.3 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.

13.12.4.3.2.4 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.

13.12.4.3.2.5 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](#).

13.12.4.3.2.6 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.

13.12.4.3.2.7 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.

13.12.4.4 Producer

Collaboration diagram for Producer:

**Functions**

- long [l4shmc_trigger](#) (l4shmc_signal_t *signal)
Trigger a signal.

13.12.4.4.1 Detailed Description

13.12.4.4.2 Function Documentation

13.12.4.4.2.1 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.

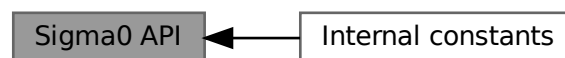
Examples

[examples/libs/shmc/prodcons.c](#).

13.13 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.

13.13.1 Detailed Description

Sigma0 API bindings.

Convenience bindings for the Sigma0 protocol.

13.13.2 Enumeration Type Documentation

13.13.2.1 l4sigma0_return_flags_t

```
enum l4sigma0_return_flags_t
```

Return flags of libsigma0 functions.

Enumerator

<code>L4SIGMA0_OK</code>	Ok.
<code>L4SIGMA0_NOTALIGNED</code>	Phys, virt or size not aligned.
<code>L4SIGMA0_IPCERROR</code>	IPC error.
<code>L4SIGMA0_NOFPAGE</code>	No fpage received.
<code>L4SIGMA0_SMALLERFPAGE</code>	Superpage requested but smaller flexpage received.

Definition at line 78 of file [sigma0.h](#).

13.13.3 Function Documentation

13.13.3.1 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.

13.13.3.2 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.
	<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^{sz} bytes.

Return values

<i>0</i>	Success.
<i>-L4SIGMA0_IPCERROR</i>	IPC error.
<i>-L4SIGMA0_NOFPAGE</i>	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.

13.13.3.3 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](#).

13.13.3.4 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.

13.13.3.5 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.

13.13.3.6 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

0	Success.
-L4SIGMA0_NOTALIGNED	phys, virt, or size are not aligned.
-L4SIGMA0_IPCERROR	IPC error.
-L4SIGMA0_NOFPAGE	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.

13.13.4 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.

13.13.4.1 Detailed Description

Internal sigma0 definitions.

13.14 Small C++ Template Library

Namespaces

- namespace [cxx](#)
Our C++ library.

Data Structures

- class [L4::Alloc_list](#)
A simple list-based allocator.
- class [cxx::Bitmap_base](#)
Basic bitmap abstraction.
- class [cxx::Bitmap< BITS >](#)
A static bit map.
- 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 *oeprator 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 T1 >`
`T1 cxx::min (T1 a, T1 b)`
Get the minimum of a and b.
- `template<typename T1 >`
`T1 cxx::max (T1 a, T1 b)`
Get the maximum of a and b.
- `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.

13.14.1 Detailed Description

13.14.2 Function Documentation

13.14.2.1 clamp()

```
template<typename T1 >
T1 cxx::clamp (
    T1 v,
    T1 lo,
    T1 hi ) [inline]
```

Limit *v* to the range given by *lo* and *hi*.

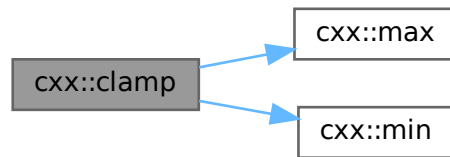
Parameters

<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 58 of file [minmax](#).

References [cxx::max\(\)](#), and [cxx::min\(\)](#).

Here is the call graph for this function:



13.14.2.2 `max()`

```

template<typename T1 >
T1 cxx::max (
    T1 a,
    T1 b ) [inline]
  
```

Get the maximum of *a* and *b*.

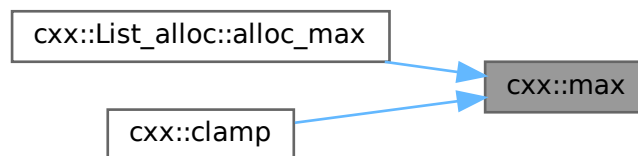
Parameters

<i>a</i>	the first value.
<i>b</i>	the second value.

Definition at line 46 of file [minmax](#).

Referenced by [cxx::List_alloc::alloc_max\(\)](#), and [cxx::clamp\(\)](#).

Here is the caller graph for this function:



13.14.2.3 `min()`

```

template<typename T1 >
T1 cxx::min (
  
```

```
T1 a,  
T1 b ) [inline]
```

Get the minimum of *a* and *b*.

Parameters

<i>a</i>	the first value.
<i>b</i>	the second value.

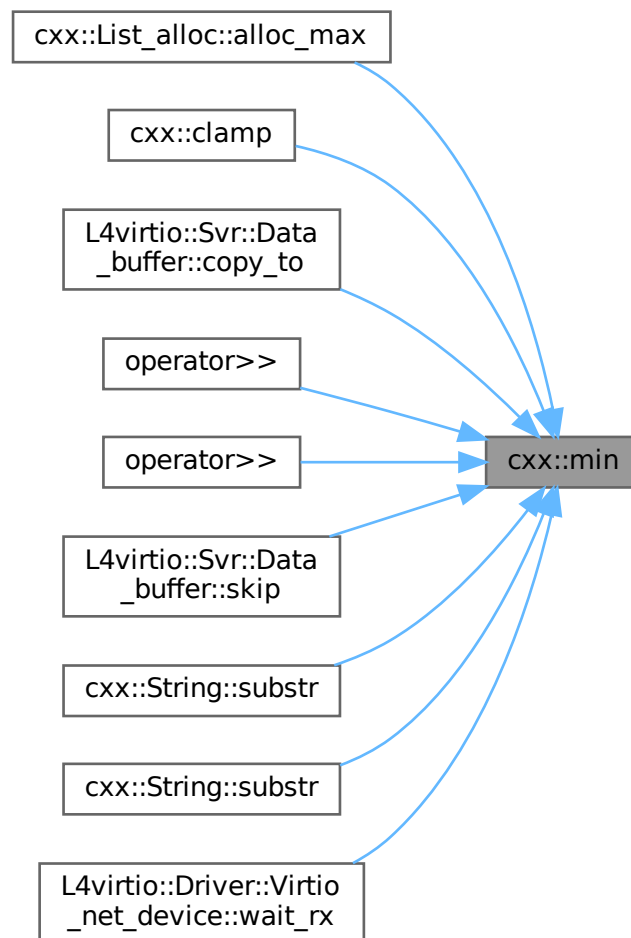
Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 35 of file [minmax](#).

Referenced by [cxx::List_alloc::alloc_max\(\)](#), [cxx::clamp\(\)](#), [L4virtio::Svr::Data_buffer::copy_to\(\)](#), [operator>>\(\)](#), [operator>>\(\)](#), [L4virtio::Svr::Data_buffer::skip\(\)](#), [cxx::String::substr\(\)](#), [cxx::String::substr\(\)](#), and [L4virtio::Driver::Virtio_net_device::wait](#).

Here is the caller graph for this function:



13.14.2.4 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

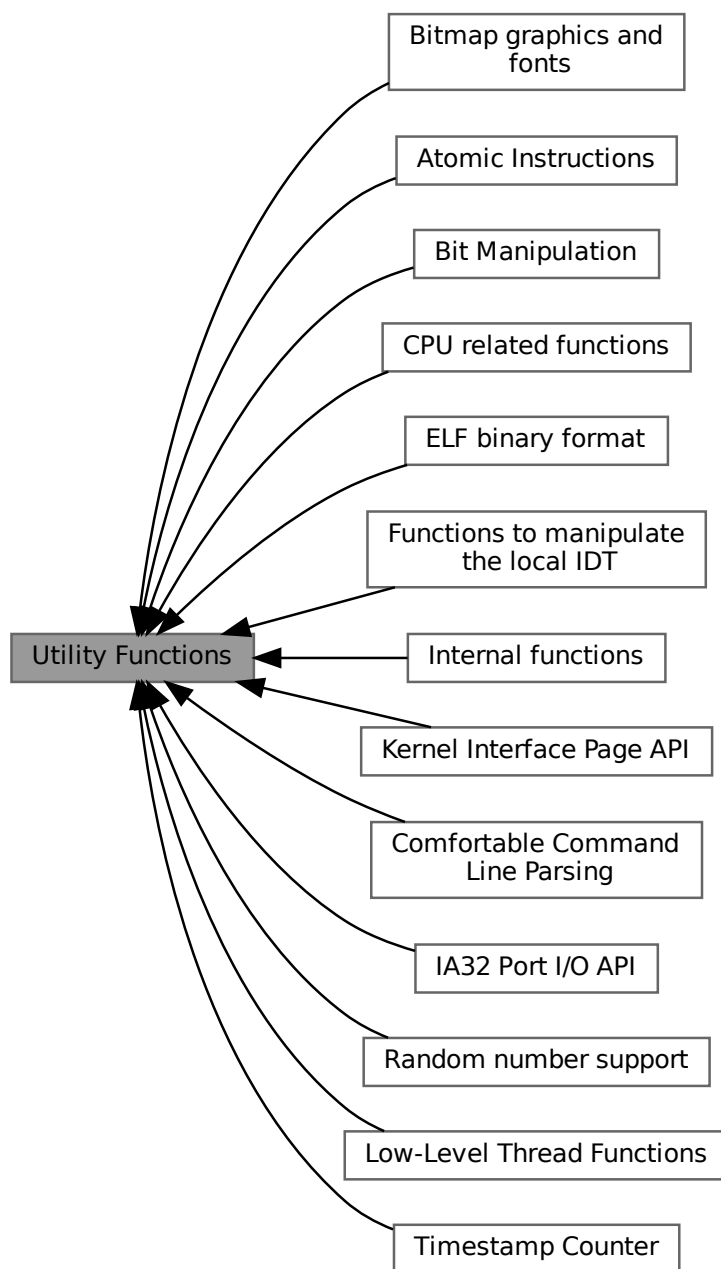
the address given by *mem*.

Definition at line 39 of file [std_alloc](#).

13.15 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

- void [l4_sleep_forever](#) (void) [L4_NOTHROW](#)
Go sleep and never wake up.
- 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](#) (unsigned int mus) [L4_NOTHROW](#)
Calculate l4 timeouts.
- void [l4_sleep](#) (int ms) [L4_NOTHROW](#)
Suspend thread for a period of ms milliseconds.
- void [l4_usleep](#) (int us) [L4_NOTHROW](#)
Suspend thread for a period of us microseconds.
- 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)

13.15.1 Detailed Description

Utilities, generic file.

13.15.2 Function Documentation

13.15.2.1 [l4_sleep\(\)](#)

```
void l4_sleep (
    int ms )
```

Suspend thread for a period of *ms* milliseconds.

Parameters

<i>ms</i>	Time in milliseconds
-----------	----------------------

13.15.2.2 `l4_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.

Definition at line 96 of file [util.h](#).

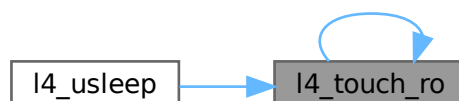
References [L4_PAGESIZE](#), [l4_touch_ro\(\)](#), and [l4_trunc_page\(\)](#).

Referenced by [l4_touch_ro\(\)](#), and [l4_usleep\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.15.2.3 l4_touch_rw()

```
void l4_touch_rw (
    const void * addr,
    unsigned size ) [inline]
```

Touch data areas to force mapping (read-write)

Parameters

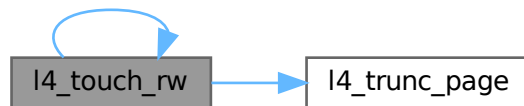
<i>addr</i>	Start of memory area to touch.
<i>size</i>	Size of area to touch.

Definition at line 109 of file [util.h](#).

References [L4_PAGESIZE](#), [l4_touch_rw\(\)](#), and [l4_trunc_page\(\)](#).

Referenced by [l4_touch_rw\(\)](#), and [l4_usleep\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.15.2.4 l4_usleep()

```
void l4_usleep (
    int us )
```

Suspend thread for a period of *us* microseconds.

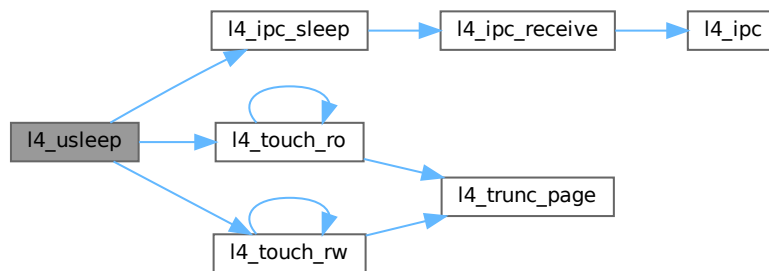
Parameters

<i>us</i>	Time in microseconds
-----------	----------------------

WARNING: This function is mostly bogus since the timer resolution of current [L4](#) implementations is about 1ms!

References [L4_IPC_NEVER](#), [l4_ipc_sleep\(\)](#), [l4_touch_ro\(\)](#), and [l4_touch_rw\(\)](#).

Here is the call graph for this function:

13.15.2.5 `l4util_micros2l4to()`

```
l4_timeout_s l4util_micros2l4to (
    unsigned int mus )
```

Calculate l4 timeouts.

Parameters

<i>mus</i>	time in microseconds. Special cases: <ul style="list-style-type: none"> • 0 - > timeout 0 • ~0U -> timeout NEVER
------------	--

Returns

the corresponding `l4_timeout` value

Deprecated Use `l4_timeout_from_us()`.

13.15.2.6 `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:



13.15.2.7 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](#).

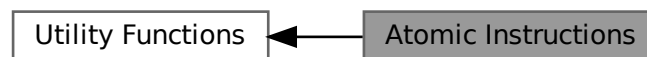
Referenced by [l4util_splitlog2_hdl\(\)](#).

Here is the caller graph for this function:



13.15.3 Atomic Instructions

Collaboration diagram for Atomic Instructions:



Files

- file [atomic.h](#)
atomic operations header and generic implementations

Functions

- int [l4util_cmpxchg64](#) (volatile [l4_uint64_t](#) *dest, [l4_uint64_t](#) cmp_val, [l4_uint64_t](#) new_val)
Atomic compare and exchange (64 bit version)
- 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)

13.15.3.1 Detailed Description

13.15.3.2 Function Documentation

13.15.3.2.1 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 460 of file [atomic.h](#).

13.15.3.2.2 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 512 of file [atomic.h](#).

13.15.3.2.3 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 464 of file [atomic.h](#).

13.15.3.2.4 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 516 of file [atomic.h](#).

13.15.3.2.5 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 456 of file [atomic.h](#).

13.15.3.2.6 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 508 of file [atomic.h](#).

13.15.3.2.7 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 488 of file [atomic.h](#).

13.15.3.2.8 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 536 of file [atomic.h](#).

13.15.3.2.9 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 492 of file [atomic.h](#).

13.15.3.2.10 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 540 of file [atomic.h](#).

13.15.3.2.11 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 484 of file [atomic.h](#).

13.15.3.2.12 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 532 of file [atomic.h](#).

13.15.3.2.13 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 468 of file [atomic.h](#).

13.15.3.2.14 l4util_atomic_inc()

```
void l4util_atomic_inc (
    volatile long * dest ) [inline]
```

Atomic increment.

Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 411 of file [atomic.h](#).

13.15.3.2.15 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

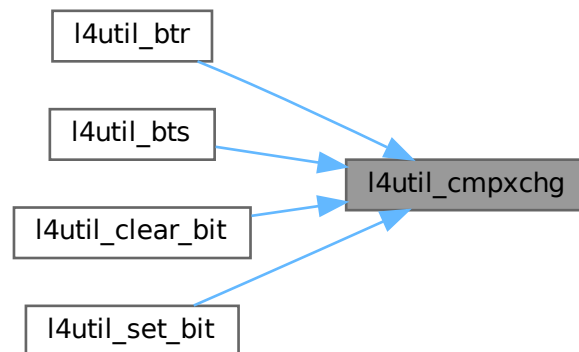
0 if comparison failed, 1 otherwise

Compare the value in *dest* with *cmp_val*, if equal set *dest* to *new_val*

Definition at line 367 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:



13.15.3.2.16 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 351 of file [atomic.h](#).

13.15.3.2.17 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 343 of file [atomic.h](#).

13.15.3.2.18 l4util_cmpxchg64()

```
int l4util_cmpxchg64 (
    volatile l4_uint64_t * dest,
    l4_uint64_t cmp_val,
    l4_uint64_t new_val ) [inline]
```

Atomic compare and exchange (64 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, 1 otherwise

Compare the value in *dest* with *cmp_val*, if equal set *dest* to *new_val*

Definition at line 335 of file [atomic.h](#).

13.15.3.2.19 l4util_cmpxchg8()

```
int l4util_cmpxchg8 (
    volatile l4_uint8_t * dest,
    l4_uint8_t cmp_val,
    l4_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 359 of file [atomic.h](#).

13.15.3.2.20 l4util_dec16()

```
void l4util_dec16 (
    volatile l4_uint16_t * dest ) [inline]
```

Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 419 of file [atomic.h](#).

13.15.3.2.21 l4util_dec16_res()

```
l4_uint16_t l4util_dec16_res (
    volatile l4_uint16_t * dest ) [inline]
```

Parameters

<i>dest</i>	destination operand
-------------	---------------------

Returns

res

Definition at line 444 of file [atomic.h](#).

13.15.3.2.22 l4util_dec32()

```
void l4util_dec32 (
    volatile l4_uint32_t * dest ) [inline]
```

Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 423 of file [atomic.h](#).

13.15.3.2.23 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 448 of file [atomic.h](#).

13.15.3.2.24 l4util_dec8()

```
void l4util_dec8 (  
    volatile l4_uint8_t * dest ) [inline]
```

Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 415 of file [atomic.h](#).

13.15.3.2.25 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 440 of file [atomic.h](#).

13.15.3.2.26 l4util_inc16()

```
void l4util_inc16 (  
    volatile l4_uint16_t * dest ) [inline]
```

Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 403 of file [atomic.h](#).

13.15.3.2.27 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 432 of file [atomic.h](#).

13.15.3.2.28 l4util_inc32()

```
void l4util_inc32 (
    volatile l4_uint32_t * dest ) [inline]
```

Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 407 of file [atomic.h](#).

13.15.3.2.29 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 436 of file [atomic.h](#).

13.15.3.2.30 l4util_inc8()

```
void l4util_inc8 (
    volatile l4_uint8_t * dest ) [inline]
```


Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 399 of file [atomic.h](#).

13.15.3.2.31 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 428 of file [atomic.h](#).

13.15.3.2.32 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 500 of file [atomic.h](#).

13.15.3.2.33 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 548 of file [atomic.h](#).

13.15.3.2.34 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 504 of file [atomic.h](#).

13.15.3.2.35 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 552 of file [atomic.h](#).

13.15.3.2.36 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 496 of file [atomic.h](#).

13.15.3.2.37 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 [544](#) of file [atomic.h](#).

13.15.3.2.38 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 [476](#) of file [atomic.h](#).

13.15.3.2.39 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 [524](#) of file [atomic.h](#).

13.15.3.2.40 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 480 of file [atomic.h](#).

13.15.3.2.41 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 528 of file [atomic.h](#).

13.15.3.2.42 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 472 of file [atomic.h](#).

13.15.3.2.43 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 520 of file [atomic.h](#).

13.15.3.2.44 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 393 of file [atomic.h](#).

13.15.3.2.45 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 381 of file [atomic.h](#).

13.15.3.2.46 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 375 of file [atomic.h](#).

13.15.3.2.47 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

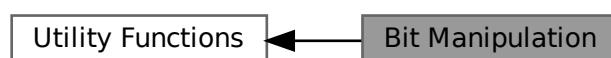
Returns

old value at destination

Definition at line 387 of file [atomic.h](#).

13.15.4 Bit Manipulation

Collaboration diagram for Bit Manipulation:



Files

- file [bitops.h](#)
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.

13.15.4.1 Detailed Description

13.15.4.2 Function Documentation

13.15.4.2.1 [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](#).

13.15.4.2.2 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](#).

13.15.4.2.3 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](#).

13.15.4.2.4 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:



13.15.4.2.5 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:

**13.15.4.2.6 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:



13.15.4.2.7 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](#).

13.15.4.2.8 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](#).

13.15.4.2.9 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](#).

13.15.4.2.10 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](#).

13.15.4.2.11 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:



13.15.4.2.12 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

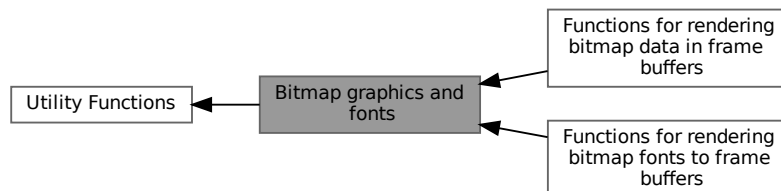
Value of bit *b*.

Definition at line 246 of file [bitops.h](#).

13.15.5 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](#)

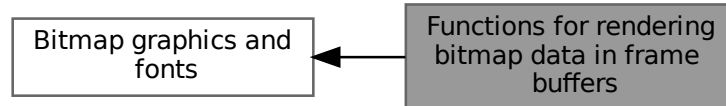
13.15.5.1 Detailed Description

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.

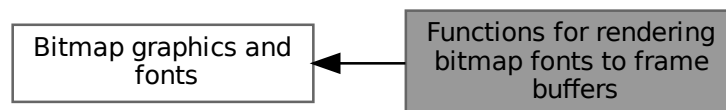
13.15.5.2 Functions for rendering bitmap data in frame buffers

Collaboration diagram for Functions for rendering bitmap data in frame buffers:



13.15.5.3 Functions for rendering bitmap fonts to frame buffers

Collaboration diagram for Functions for rendering bitmap fonts to frame buffers:



13.15.6 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.

13.15.6.1 Detailed Description

13.15.6.2 Function Documentation

13.15.6.2.1 l4util_cpu_capabilities()

```
unsigned int l4util_cpu_capabilities (  
    void ) [inline]
```

Returns the CPU capabilities if the "cpuid" instruction is available.

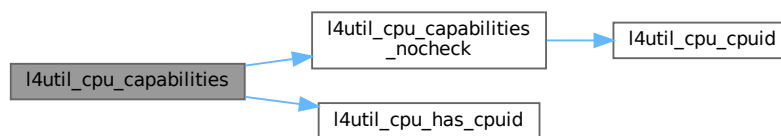
Returns

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:



13.15.6.2.2 l4util_cpu_capabilities_nocheck()

```
unsigned int l4util_cpu_capabilities_nocheck (  
    void ) [inline]
```

Returns the CPU capabilities.

Returns

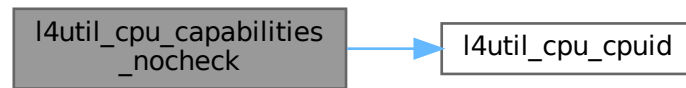
CPU capabilities.

Definition at line 86 of file [cpu.h](#).

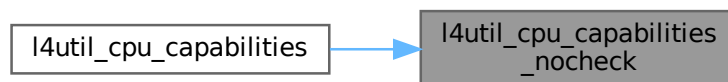
References [l4util_cpu_cpuid\(\)](#).

Referenced by [l4util_cpu_capabilities\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.15.6.2.3 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](#).

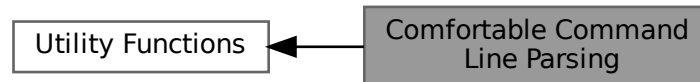
Referenced by [l4util_cpu_capabilities\(\)](#).

Here is the caller graph for this function:



13.15.7 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.

13.15.7.1 Detailed Description

13.15.7.2 Function Documentation

13.15.7.2.1 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 occurence 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.

13.15.8 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_NUM** = 192 }
- 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_CIASSs** { **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_PT**s {
 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_NT**s_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_NT**s_obj { NT_VERSION = 1 }

Legal values for the note segment descriptor types for object files.

- enum **Elf_DT**s {
 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_DF**s {
 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_1**s {
 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

13.15.8.1 Detailed Description

Functions and types related to ELF binaries.

13.15.8.2 Macro Definition Documentation

13.15.8.2.1 ELF32_R_TYPE

```
#define ELF32_R_TYPE(  
    i ) ((unsigned char)(i))
```

See also

[Elf_R_386s](#).

Definition at line [659](#) of file [elf.h](#).

13.15.8.2.2 ELF32_ST_BIND

```
#define ELF32_ST_BIND(  
    i ) ((i)>>4)
```

See also

[Elf_STBs](#).

Definition at line [889](#) of file [elf.h](#).

13.15.8.2.3 ELF32_ST_TYPE

```
#define ELF32_ST_TYPE(  
    i ) ((i)&0xf)
```

See also

[Elf_STTs](#).

Definition at line [892](#) of file [elf.h](#).

13.15.8.2.4 ELF64_R_TYPE

```
#define ELF64_R_TYPE(  
    i ) ((i)&0xffffffffL)
```

See also

[Elf_R_386s](#).

Definition at line [667](#) of file [elf.h](#).

13.15.8.2.5 ELF64_ST_BIND

```
#define ELF64_ST_BIND(  
    i )  ((i)>>4)
```

See also

[Elf_STBs](#)

Definition at line 898 of file [elf.h](#).

13.15.8.2.6 ELF64_ST_TYPE

```
#define ELF64_ST_TYPE(  
    i )  ((i)&0xf)
```

See also

[Elf_STTs](#)

Definition at line 901 of file [elf.h](#).

13.15.8.3 Enumeration Type Documentation

13.15.8.3.1 anonymous enum

anonymous enum

Enumerator

EI_NIDENT	Number of characters.
-----------	-----------------------

Definition at line 113 of file [elf.h](#).

13.15.8.3.2 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 766 of file [elf.h](#).

13.15.8.3.3 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_PHEMT	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	L4Re AUX section.
AT_L4_ENV	L4Re ENV section.

Definition at line [935](#) of file [elf.h](#).

13.15.8.3.4 Elf_CLASSES

enum [Elf_CLASSES](#)

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 [293](#) of file [elf.h](#).

13.15.8.3.5 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 302 of file [elf.h](#).

13.15.8.3.6 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.
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 590 of file [elf.h](#).

13.15.8.3.7 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 619 of file [elf.h](#).

13.15.8.3.8 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 577 of file [elf.h](#).

13.15.8.3.9 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

Enumerator

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 531 of file [elf.h](#).

13.15.8.3.10 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 726 of file [elf.h](#).

13.15.8.3.11 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 [269](#) of file [elf.h](#).

13.15.8.3.12 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.

Enumerator

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.
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

Enumerator

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.

Definition at line 179 of file [elf.h](#).

13.15.8.3.13 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 164 of file [elf.h](#).

13.15.8.3.14 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 261 of file [elf.h](#).

13.15.8.3.15 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 284 of file [elf.h](#).

13.15.8.3.16 Elf_NTs_core

enum [Elf_NTs_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.
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 fprxregset struct.

Definition at line 482 of file [elf.h](#).

13.15.8.3.17 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 503 of file [elf.h](#).

13.15.8.3.18 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.
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 311 of file [elf.h](#).

13.15.8.3.19 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 472 of file [elf.h](#).

13.15.8.3.20 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.
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 446 of file [elf.h](#).

13.15.8.3.21 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.
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 673 of file [elf.h](#).

13.15.8.3.22 Elf_R_AARCH64_s

enum [Elf_R_AARCH64_s](#)

AARCH64 relocations.

Enumerator

R_AARCH64_NONE	No reloc.
----------------	-----------

Definition at line 821 of file [elf.h](#).

13.15.8.3.23 Elf_R_ARM_senum [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 [773](#) of file [elf.h](#).**13.15.8.3.24 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.

Enumerator

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 828 of file [elf.h](#).

13.15.8.3.25 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 758 of file [elf.h](#).

13.15.8.3.26 Elf_SHFs

enum [Elf_SHFs](#)

Section attribute flags.

Enumerator

SHF_WRITE	writable 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.

Enumerator

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 401 of file [elf.h](#).

13.15.8.3.27 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 330 of file [elf.h](#).

13.15.8.3.28 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

Enumerator

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 372 of file [elf.h](#).

13.15.8.3.29 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.

Definition at line 908 of file [elf.h](#).

13.15.8.3.30 Elf_STTs

enum [Elf_STTs](#)

Symbol Types.

See also

[ELF32_ST_TYPE](#), [ELF64_ST_TYPE](#)

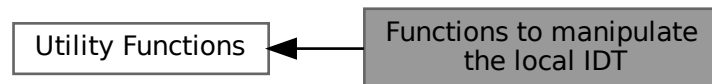
Enumerator

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 [921](#) of file [elf.h](#).

13.15.9 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.

13.15.9.1 Detailed Description

13.15.10 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

13.15.10.1 Detailed Description

13.15.10.2 Function Documentation

13.15.10.2.1 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](#).

13.15.10.2.2 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](#).

13.15.10.2.3 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:



13.15.10.2.4 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](#).

13.15.10.2.5 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](#).

13.15.10.2.6 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](#).

13.15.10.2.7 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](#).

13.15.10.2.8 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](#).

13.15.10.2.9 l4util_out8()

```
void l4util_out8 (  
    14_uint8_t value,  
    14_uint16_t port ) [inline]
```

Write byte to I/O port.

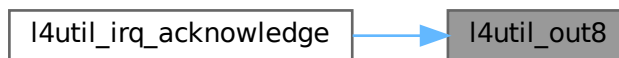
Parameters

<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:

**13.15.10.2.10 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](#).

13.15.10.2.11 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](#).

13.15.10.2.12 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](#).

13.15.11 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

13.15.11.1 Detailed Description

13.15.12 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.

13.15.12.1 Detailed Description

13.15.12.2 Macro Definition Documentation

13.15.12.2.1 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](#).

13.15.12.3 Function Documentation

13.15.12.3.1 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.

13.15.12.3.2 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\(\)](#).

13.15.12.3.3 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](#).

13.15.13 Low-Level Thread Functions

Collaboration diagram for Low-Level Thread Functions:



13.15.14 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.

13.15.14.1 Detailed Description

13.15.14.2 Function Documentation

13.15.14.2.1 l4util_rand()

```
l4_uint32_t l4util_rand (
    void )
```

Deliver next random number.

Returns

A new random number

13.15.14.2.2 l4util_srand()

```
void l4util_srand (
    l4_uint32_t seed )
```

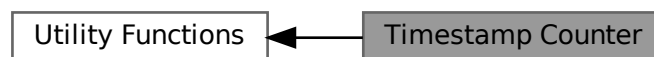
Initialize random number generator.

Parameters

<i>seed</i>	Value to initialize
-------------	---------------------

13.15.15 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 least 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.

13.15.15.1 Detailed Description

13.15.15.2 Function Documentation

13.15.15.2.1 l4_busy_wait_ns()

```
void l4_busy_wait_ns (
    l4\_uint64\_t ns ) [inline]
```

Wait busy for a small amount of time.

Parameters

<i>ns</i>	nano seconds to wait
-----------	----------------------

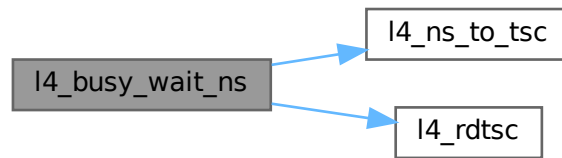
Attention

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:



13.15.15.2.2 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
-----------	-----------------------

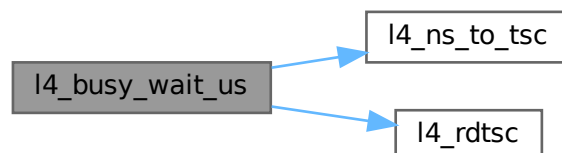
Attention

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:



13.15.15.2.3 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:



13.15.15.2.4 l4_get_hz()

```
l4_uint32_t l4_get_hz (
    void )
```

Get CPU frequency in Hz.

Returns

frequency in Hz

13.15.15.2.5 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
-----------	--------------

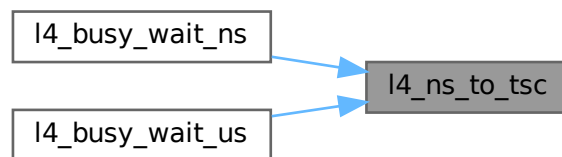
Returns

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:

**13.15.15.2.6 l4_rdpmc()**

```
l4_uint64_t l4_rdpmc (
    int ecx ) [inline]
```

Return current value of CPU-internal performance measurement counter.

Parameters

<i>ecx</i>	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](#).

13.15.15.2.7 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](#).

13.15.15.2.8 l4_rdtsc()

```
l4_cpu_time_t l4_rdtsc (
    void ) [inline]
```

Read current value of CPU-internal timestamp counter.

Returns

64-bit timestamp

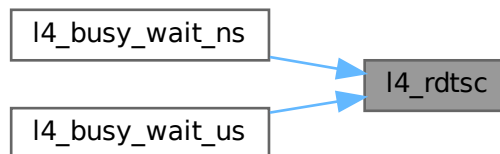
Examples

[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:



13.15.15.2.9 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](#).

13.15.15.2.10 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:



13.15.15.2.11 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](#).

13.15.15.2.12 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](#).

13.15.15.2.13 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

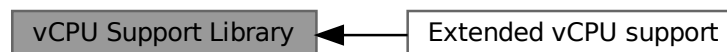
time value in micro seconds

Definition at line 221 of file [rdtsc.h](#).

13.16 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.

13.16.1 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.

13.16.2 Function Documentation

13.16.2.1 l4vcpu_irq_disable()

```
void l4vcpu_irq_disable (
    l4\_vcpu\_state\_t * vcpu ) [inline]
```

Disable a vCPU for event delivery.

Parameters

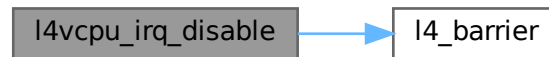
<i>vcpu</i>	Pointer to vCPU area.
-------------	-----------------------

Definition at line 212 of file [vcpu.h](#).

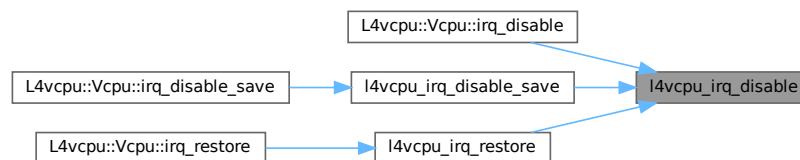
References [l4_barrier\(\)](#).

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:



13.16.2.2 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:



13.16.2.3 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

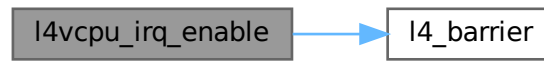
<i>vcpu</i>	Pointer to vCPU area.
<i>utcb</i>	Utc b pointer of the calling vCPU.
<i>do_event_work_cb</i>	Call-back function that is called in case an event (such as an interrupt) is pending.
<i>setup_ipc</i>	Function call-back that is called right before any IPC operation, and before event delivery is enabled.

Definition at line 243 of file [vcpu.h](#).

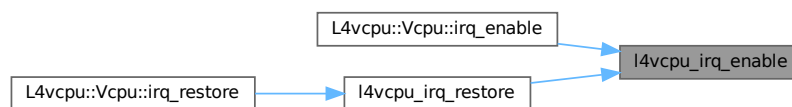
References [l4_barrier\(\)](#), [L4_IPC_BOTH_TIMEOUT_0](#), [L4_LIKELY](#), [L4_VCPU_F_IRQ](#), and [L4_VCPU_SF_IRQ_PENDING](#).

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:



13.16.2.4 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

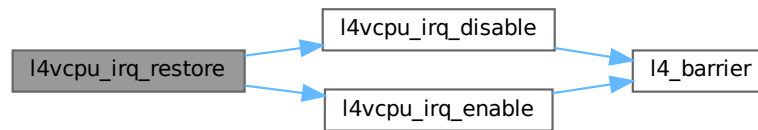
<i>vcpu</i>	Pointer to vCPU area.
<i>s</i>	IRQ state to be restored.
<i>utcb</i>	Utcbl pointer of the calling vCPU.
<i>do_event_work_cb</i>	Call-back function that is called in case an event (such as an interrupt) is pending after enabling.
<i>setup_ipc</i>	Function call-back that is called right before any IPC operation, and before event delivery is enabled.

Definition at line 268 of file `vcpu.h`.

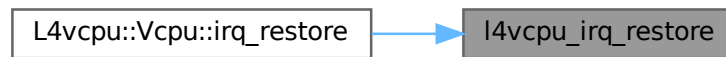
References `L4_VCPU_F_IRQ`, `l4vcpu_irq_disable()`, and `l4vcpu_irq_enable()`.

Referenced by `L4vcpu::Vcpu::irq_restore()`.

Here is the call graph for this function:



Here is the caller graph for this function:



13.16.2.5 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.

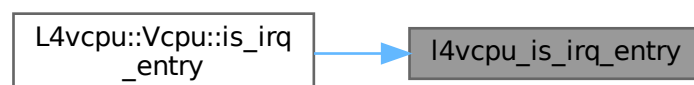
Parameters

<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:



13.16.2.6 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:



13.16.2.7 l4vcpu_print_state()

```
void l4vcpu_print_state (
    const l4_vcpu_state_t * vcpu,
    const char * prefix )
```

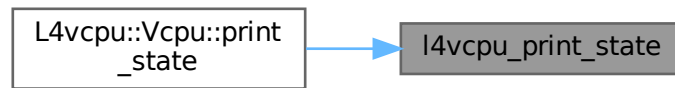
Print the state of a vCPU.

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:



13.16.2.8 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>	Utcbl 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.

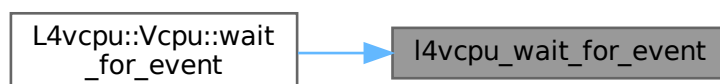
Note that event delivery remains disabled after this function returns.

Definition at line 281 of file [vcpu.h](#).

References [L4_IPC_NEVER](#).

Referenced by [L4vcpu::Vcpu::wait_for_event\(\)](#).

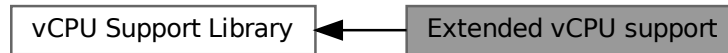
Here is the caller graph for this function:



13.16.3 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.

13.16.3.1 Detailed Description

Extended vCPU handling functionality.

13.16.3.2 Function Documentation

13.16.3.2.1 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 bit map.
- 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 T1 >`
`T1 min (T1 a, T1 b)`
Get the minimum of a and b.
- `template<typename T1 >`
`T1 max (T1 a, T1 b)`
Get the maximum of a and b.
- `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.

Note

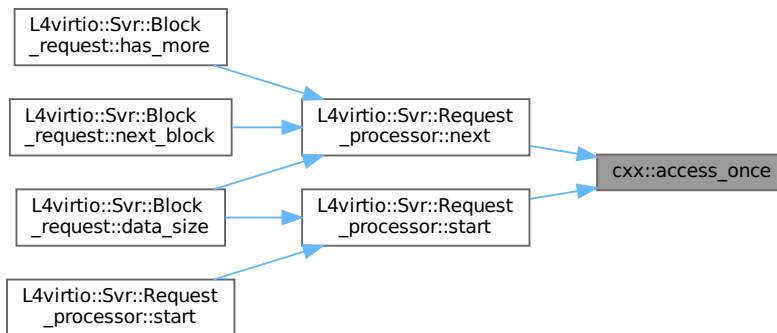
The read might still be moved relative to other code.

The value might be read from a hardware cache, not from RAM.

Definition at line 39 of file [utils](#).

Referenced by [L4virtio::Svr::Request_processor::next\(\)](#), and [L4virtio::Svr::Request_processor::start\(\)](#).

Here is the caller graph for this function:

**14.1.2.2 write_now()**

```

template<typename T , typename VAL >
void cxx::write_now (
    T * a,
    VAL && val ) [inline]

```

Write a value at an address exactly once.

The compiler is disallowed to skip the write, for example:

```

*a = val;
write_now(a, val); // compiler may not skip this line

```

The compiler is also disallowed to repeat the write.

The above implies that the compiler is also disallowed to move the write out of or into loops.

Note

The write might still be moved relative to other code.

The value might be written just to a hardware cache for the moment, not immediately to RAM.

Definition at line 70 of file [utils](#).

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 [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

- 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.
- `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.

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 Kobject_t and Kobject_x .
<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 382 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 Smart_cap 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<T></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 L4::Meta 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 125 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 413 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 Smart_cap 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 ^{order} .
<i>order</i>	order (2 ^{order}) to round up to.

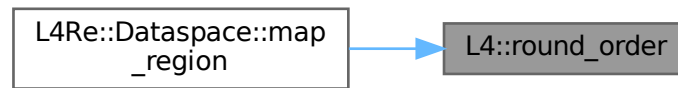
Returns

val rounded up to the next 2^{order}.

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 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 l4_utcb .

Definition at line 41 of file [ipc_helper](#).

References [l4_msgtag_t::has_error\(\)](#).

Referenced by [throw_ipc_exception\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.3.3.8 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.

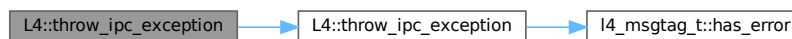
Parameters

<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 l4_utcb .

Definition at line 58 of file [ipc_helper](#).

References [throw_ipc_exception\(\)](#).

Here is the call graph for this function:

**14.3.3.9 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.

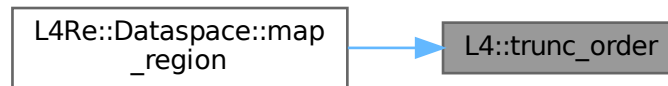
Returns

`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.
- class [Buf_item](#)
RPC warpper for a receive item.
- 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 wrapper for L4 flex-pages.
- struct [In_out](#)
Mark an argument as in-out argument.

- class [lostream](#)
Input/Output stream for IPC [un]marshalling.
- class [lstream](#)
Input stream for IPC unmarshalling.
- class [Msg_ptr](#)
Pointer to an element of type T in an [lpc::lstream](#).
- 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.
- 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_item](#)
RPC wrapper for a send item.
- class [Str_cp_in](#)
Abstraction for extracting a zero-terminated string from an [lpc::lstream](#).
- 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.
- typedef [Gen_fpage](#)< [Snd_item](#) > **Snd_fpage**
Send flex-page.
- typedef [Gen_fpage](#)< [Buf_item](#) > **Rcv_fpage**
Rcv flex-page.

Functions

- 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.
- `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.

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 <code>lpc::Istream</code> .
<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 lpc::Ostream .
<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 lpc::Istream .
out	<i>size</i>	The number of elements found in the stream.

This routine provides a possibility to extract an array from an [lpc::Istream](#), 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::Ipc::Cap<T> for the given capability and rights.

Template Parameters

<i>T</i>	(IMPLICIT) type of the referenced interface
----------	---

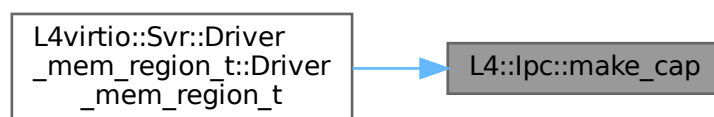
Parameters

<i>cap</i>	source capability (L4::Cap<T>)
<i>rights</i>	rights mask that shall be applied on transfer.

Definition at line 628 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 666 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::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 Parameters

<i>T</i>	(IMPLICIT) type of the referenced interface
----------	---

Parameters

<i>cap</i>	source capability (L4::Cap<T>)
------------	--------------------------------

Examples

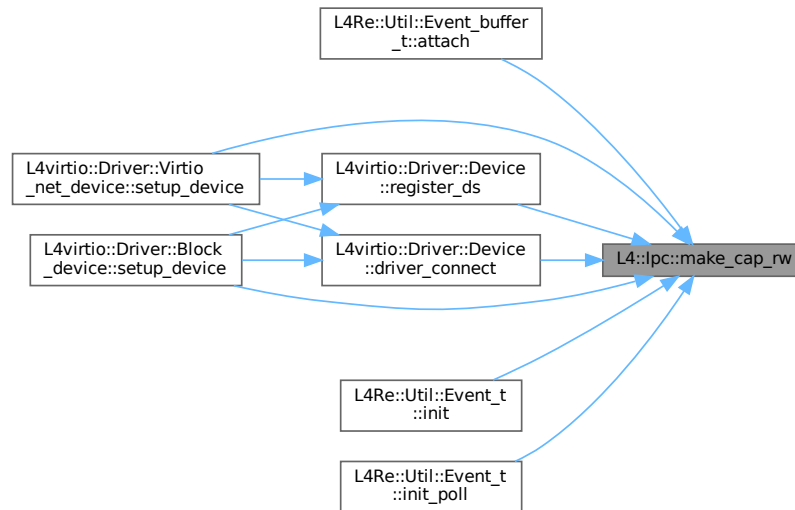
[examples/libs/l4re/c++/mem_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#), and [examples/libs/l4re/c++/shared_](#)

Definition at line 638 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

<code>T</code>	(IMPLICIT) type of the referenced interface
----------------	---

Parameters

<code>cap</code>	source capability (<code>L4::Cap<T></code>)
------------------	---

Definition at line 648 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

s	An Istream .
----------	------------------------------

Returns

The value of type T.

The stream position is progressed accordingly.

Definition at line 1294 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

	v	Pointer to the array that shall receive the values from the lpc::Istream .
in, out	size	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::Istream](#) (

See also

[Str_cp_in](#).)

Definition at line 224 of file [ipc_stream](#).

14.5 L4::ipc::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 [Rpc](#)
Standard list of RPCs of an interface.
- struct [Rpc_code](#)
List of RPCs of an interface using a special opcode type.
- struct [Rpc_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 (<code>L4::Cap</code> , <code>L4Re::Util::Unique_cap</code> 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`.

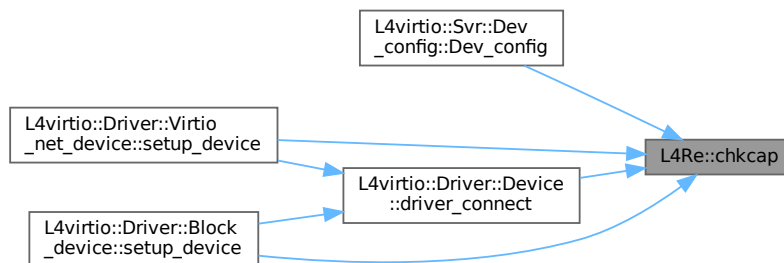
References `L4_UNLIKELY`, and `throw_error()`.

Referenced by [L4virtio::Svr::Dev_config::Dev_config\(\)](#), [L4virtio::Driver::Device::driver_connect\(\)](#), [L4virtio::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()

```

14_msgtag_t L4Re::chkipc (
    14_msgtag_t tag,
    char const * extra = "",
    14_utcb_t * utcb = 14_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

<code>L4::Runtime_exception</code>	with the translated IPC error code
------------------------------------	------------------------------------

This function does not check the message tag's label value.

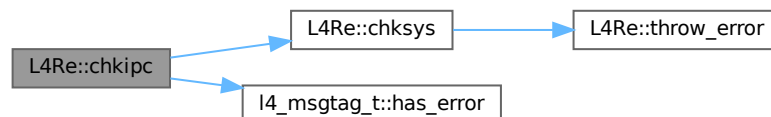
Note

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

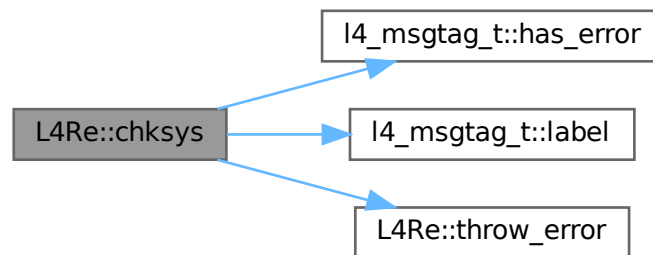
<i>t</i>	Message tag.
<i>extra</i>	Optional text for exception (default "")
<i>utcb</i>	Option UTCB
<i>ret</i>	Optional value for exception, default is error value (err)

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

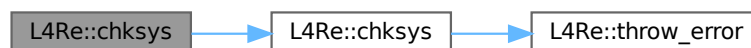
<i>t</i>	Message tag.
<i>utcb</i>	UTCB.
<i>extra</i>	Optional text for exception (default "")

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*.

Definition at line 68 of file [error_helper](#).

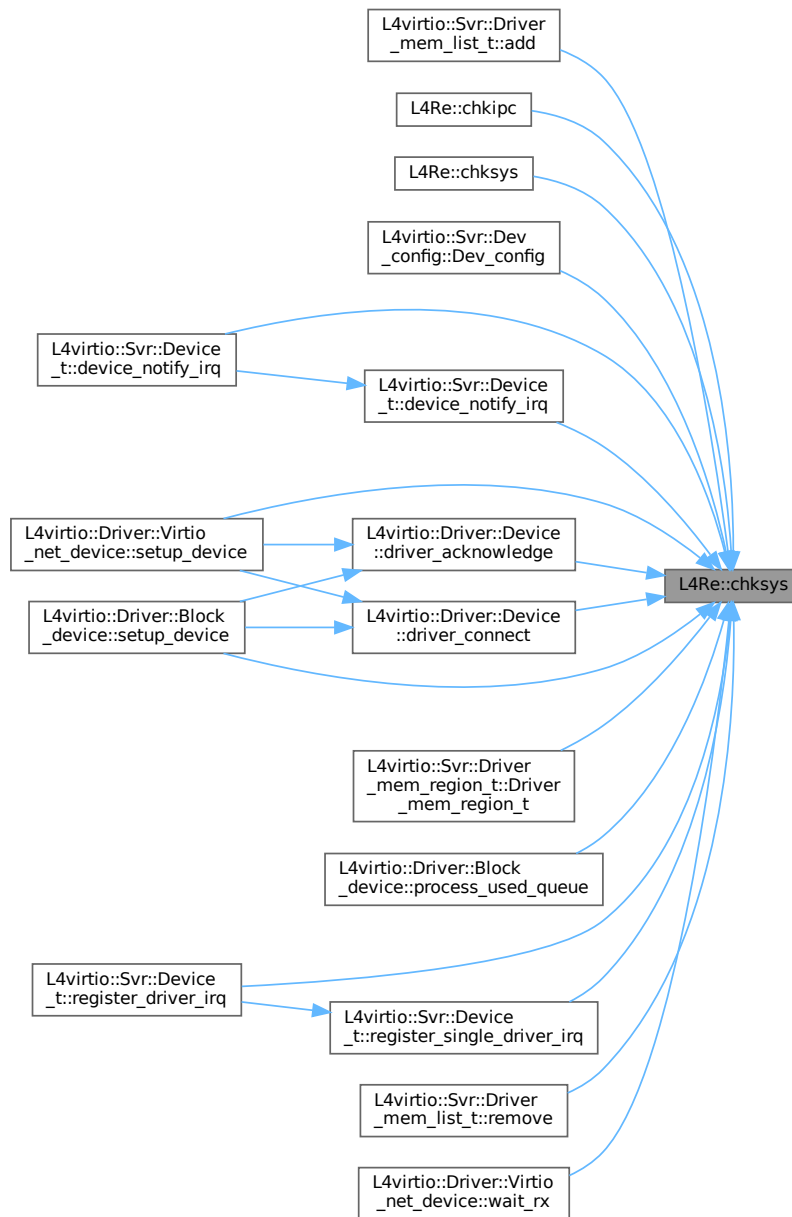
References [L4_UNLIKELY](#), and [throw_error\(\)](#).

Referenced by [L4virtio::Svr::Driver_mem_list_t< DATA >::add\(\)](#), [chkipc\(\)](#), [chksys\(\)](#), [L4virtio::Svr::Dev_config::Dev_config\(\)](#), [L4virtio::Svr::Device_t< DATA >::device_notify_irq\(\)](#), [L4virtio::Svr::Device_t< DATA >::device_notify_irq\(\)](#), [L4virtio::Driver::Device::driver_acknowledge\(\)](#), [L4virtio::Driver::Device::driver_connect\(\)](#), [L4virtio::Svr::Driver_mem_region_t< DATA >::driver_notify_irq\(\)](#), [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::Driver_mem_list_t< DATA >::remove\(\)](#), [L4virtio::Driver::Virtio_net_device::setup_device\(\)](#), [L4virtio::Driver::Block_device::setup_device\(\)](#), and [L4virtio::Driver::Virtio_net_device::setup_device\(\)](#).

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

<code>T</code>	Type of the object the capability refers to.
----------------	--

Parameters

<code>ca</code>	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

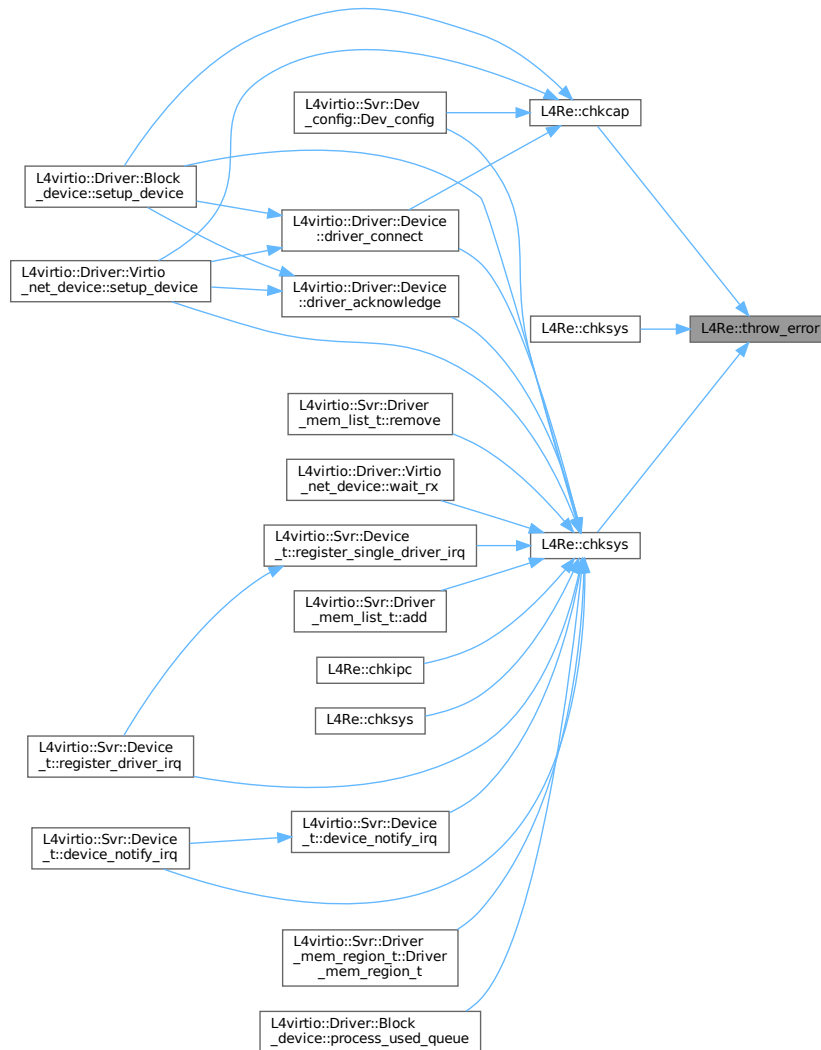
never return.

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

<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 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

T	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

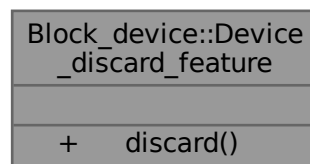
Data Structure Documentation

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 [111](#) of file [device.h](#).

The documentation for this struct was generated from the following file:

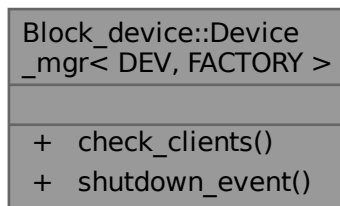
- [l4/libblock-device/device.h](#)

15.2 Block_device::Device_mgr< DEV, FACTORY > 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 >:



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>>
class Block_device::Device_mgr< DEV, FACTORY >
```

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.

Definition at line 76 of file [block_device_mgr.h](#).

The documentation for this class was generated from the following file:

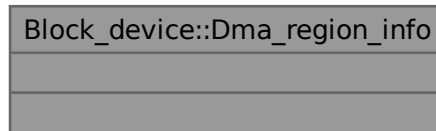
- l4/libblock-device/block_device_mgr.h

15.3 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:



15.3.1 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:

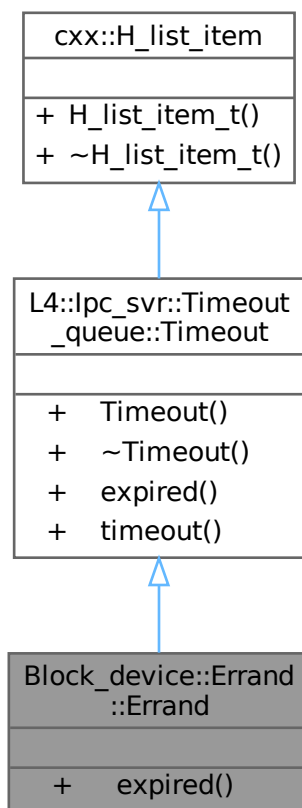
- I4/libblock-device/types.h

15.4 Block_device::Errand::Errand Class Reference

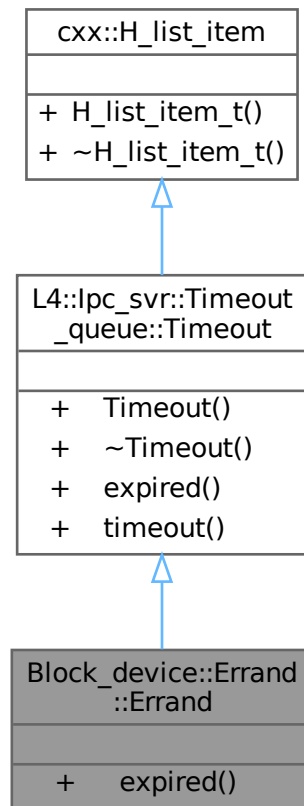
Wrapper for a small task executed asynchronously in the server loop.

```
#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.

15.4.1 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](#).

15.4.2 Member Function Documentation

15.4.2.1 `expired()`

```
void Block_device::Errand::Errand::expired ( ) [inline], [final], [virtual]
```

callback function to be called when timeout happened

Note

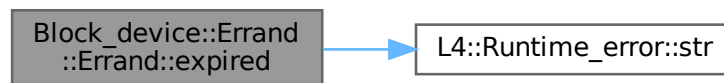
The timeout object is already dequeued when this function is called, this means the timeout may be safely queued again within the [expired\(\)](#) function.

Implements [L4::ipc_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:

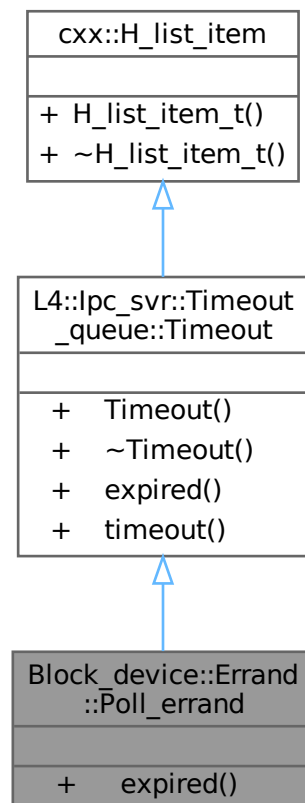
- `I4/libblock-device/errand.h`

15.5 Block_device::Errand::Poll_errand Class Reference

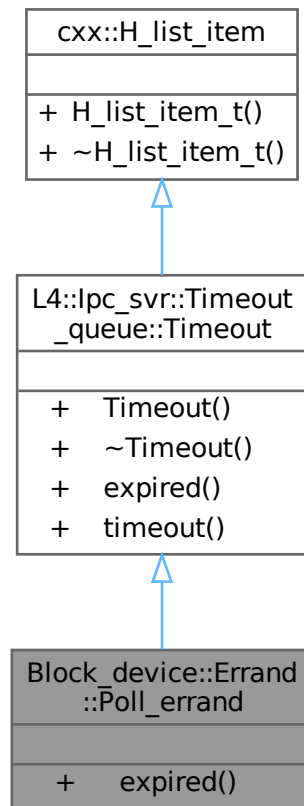
Wrapper for a regularly repeated task.

```
#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.

15.5.1 Detailed Description

Wrapper for a regularly repeated task.

Definition at line 42 of file [errand.h](#).

15.5.2 Member Function Documentation

15.5.2.1 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:

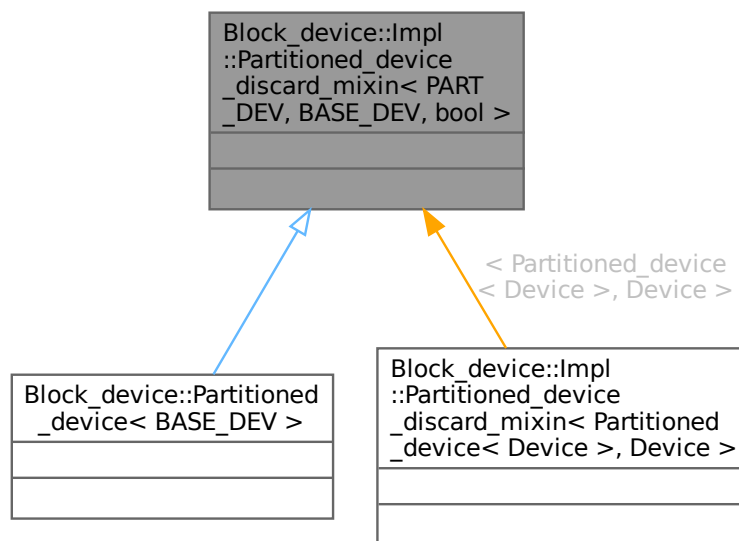
- [l4/libblock-device/errand.h](#)

15.6 Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, bool > Class Template Reference

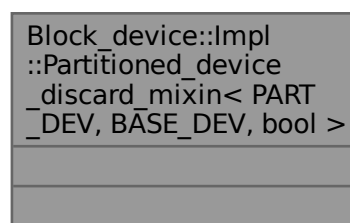
Dummy class used when the device class is not derived from [Device_discard_feature](#).

```
#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 >:



15.6.1 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

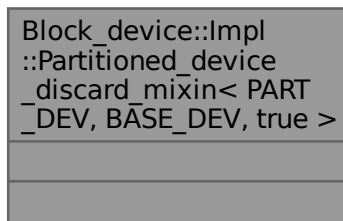
15.7 `Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, true >` Class Template Reference

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 >`:



15.7.1 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:

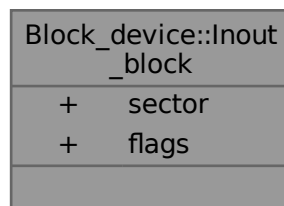
- l4/libblock-device/part_device.h

15.8 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.

15.8.1 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:

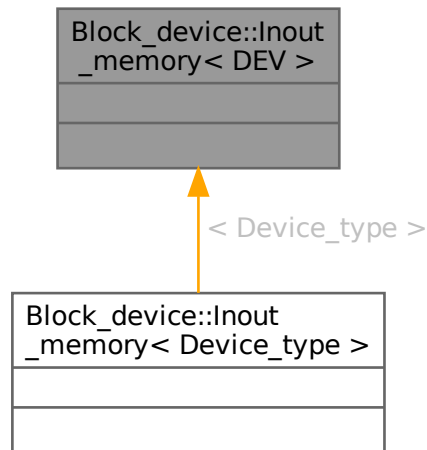
- l4/libblock-device/types.h

15.9 Block_device::Inout_memory< DEV > Class Template Reference

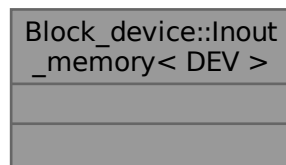
Helper class that temporarily allocates memory that can be used for in/out operations with the device.

```
#include <inout_memory.h>
```

Inheritance diagram for Block_device::Inout_memory< DEV >:



Collaboration diagram for Block_device::Inout_memory< DEV >:



15.9.1 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:

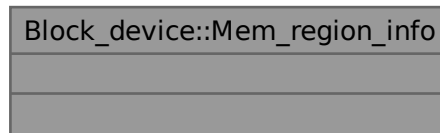
- `I4/libblock-device/inout_memory.h`

15.10 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:



15.10.1 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:

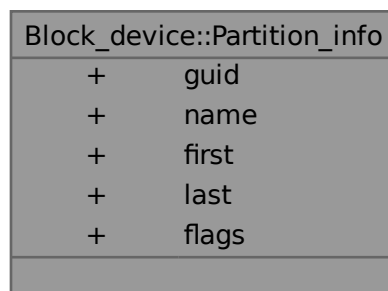
- l4/libblock-device/types.h

15.11 Block_device::Partition_info Struct Reference

Information about a single partition.

```
#include <partition.h>
```

Collaboration diagram for Block_device::Partition_info:



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.

15.11.1 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:

- [l4/libblock-device/partition.h](#)

15.12 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 >:



15.12.1 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:

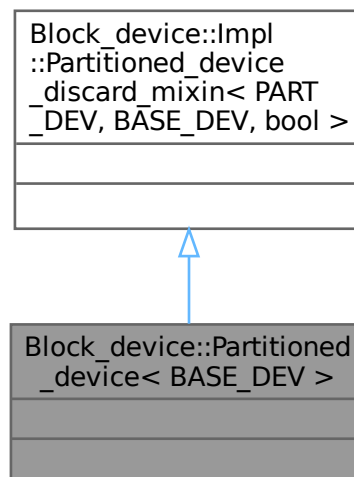
- I4/libblock-device/partition.h

15.13 Block_device::Partitioned_device< BASE_DEV > Class Template Reference

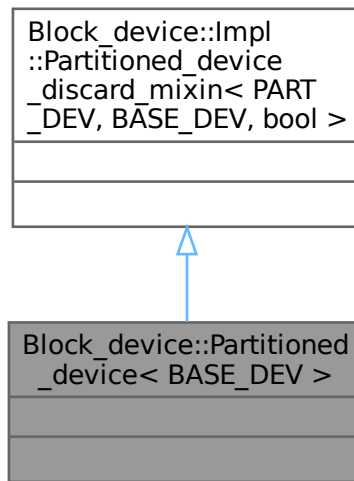
A partition device for the given device interface.

```
#include <part_device.h>
```

Inheritance diagram for Block_device::Partitioned_device< BASE_DEV >:



Collaboration diagram for Block_device::Partitioned_device< BASE_DEV >:



15.13.1 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`

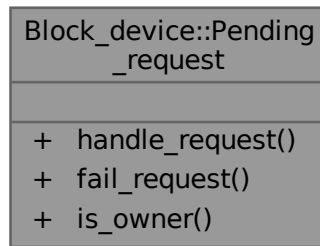
15.14 Block_device::Pending_request Struct Reference

Interface for pending requests that can be queued.

```
#include <request_queue.h>
```

Inherited by Block_device::Virtio_client< DEV >::Generic_pending_request.

Collaboration diagram for Block_device::Pending_request:



Data Structures

- struct [Owner](#)

Base class for object that can be owner of a 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.
- virtual bool [is_owner](#) ([Owner](#) *owner)=0
Check if somebody is owner of this request.

15.14.1 Detailed Description

Interface for pending requests that can be queued.

Definition at line 20 of file [request_queue.h](#).

15.14.2 Member Function Documentation

15.14.2.1 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.

15.14.2.2 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.

15.14.2.3 is_owner()

```
virtual bool Block_device::Pending_request::is_owner (
    Owner * owner ) [pure virtual]
```

Check if somebody is owner of this request.

Parameters

<i>owner</i>	Pointer to owner to check against.
--------------	------------------------------------

Returns

True, if the given object owns the request.

The documentation for this struct was generated from the following file:

- l4/libblock-device/request_queue.h

15.15 Block_device::Pending_request::Owner Struct Reference

Base class for object that can be owner of a pending request.

```
#include <request_queue.h>
```

Inherited by Block_device::Virtio_client< DEV >.

Collaboration diagram for Block_device::Pending_request::Owner:



15.15.1 Detailed Description

Base class for object that can be owner of a pending request.

The queue does not use the type itself or keep track of the owner. The implementation needs to provide a function to check a given object for ownership of the request.

Definition at line 29 of file [request_queue.h](#).

The documentation for this struct was generated from the following file:

- I4/libblock-device/request_queue.h

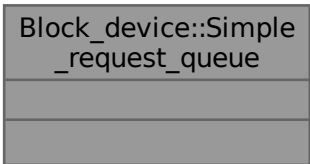
15.16 Block_device::Simple_request_queue Class Reference

Simple request queue implementation based on a linked list.

```
#include <request_queue.h>
```

Inherits Block_device::Request_queue.

Collaboration diagram for Block_device::Simple_request_queue:



```
classDiagram
    class Block_device__Simple_request_queue {
    }
    class Block_device__Request_queue {
    }
    Block_device__Simple_request_queue --|> Block_device__Request_queue
```

Block_device::Simple_request_queue

15.16.1 Detailed Description

Simple request queue implementation based on a linked list.

Definition at line 104 of file [request_queue.h](#).

The documentation for this class was generated from the following file:

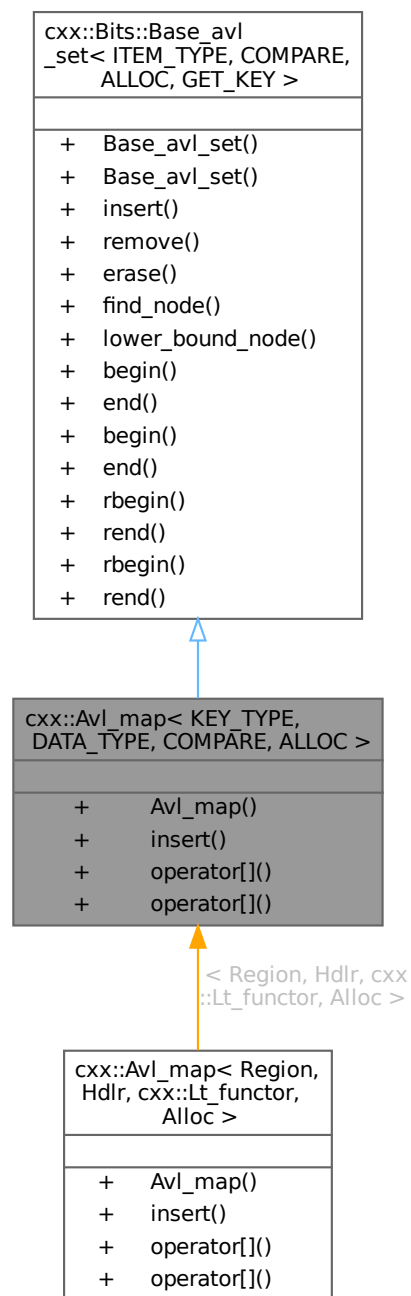
- I4/libblock-device/request_queue.h

15.17 cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC > Class Template Reference

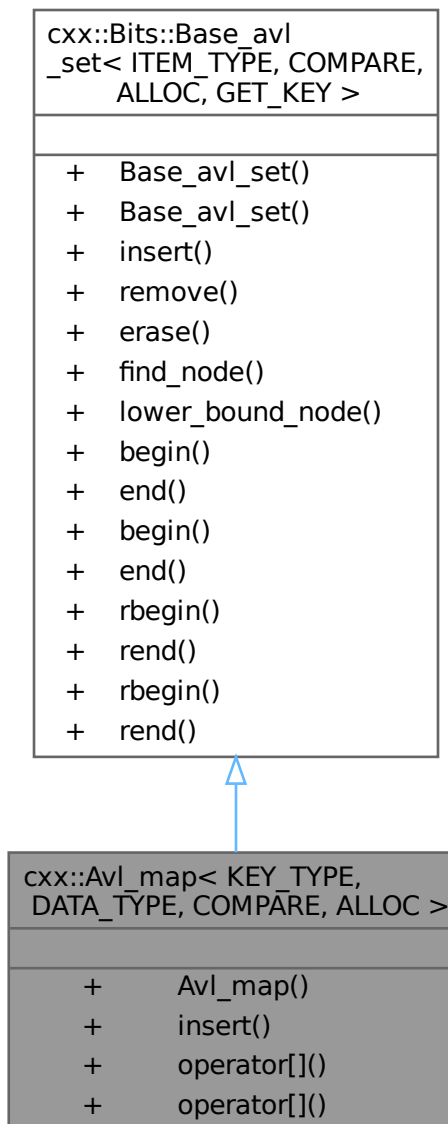
AVL tree based associative container.

```
#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.

15.17.1 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

<code>KEY_TYPE</code>	Type of the key values.
<code>DATA_TYPE</code>	Type of the data values.
<code>COMPARE</code>	Type comparison functor for the key values.
<code>ALLOC</code>	Type of the allocator used for the nodes.

Definition at line 56 of file [avl_map](#).

15.17.2 Constructor & Destructor Documentation

15.17.2.1 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

<code>alloc</code>	The node allocator.
--------------------	---------------------

Definition at line 91 of file `avl_map`.

15.17.3 Member Function Documentation

15.17.3.1 `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

<code>key</code>	The key value.
<code>data</code>	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:



15.17.3.2 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

key	The key value to use for lookup.
-----	----------------------------------

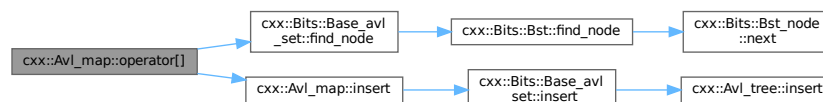
Returns

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 130 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:



15.17.3.3 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 118 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:

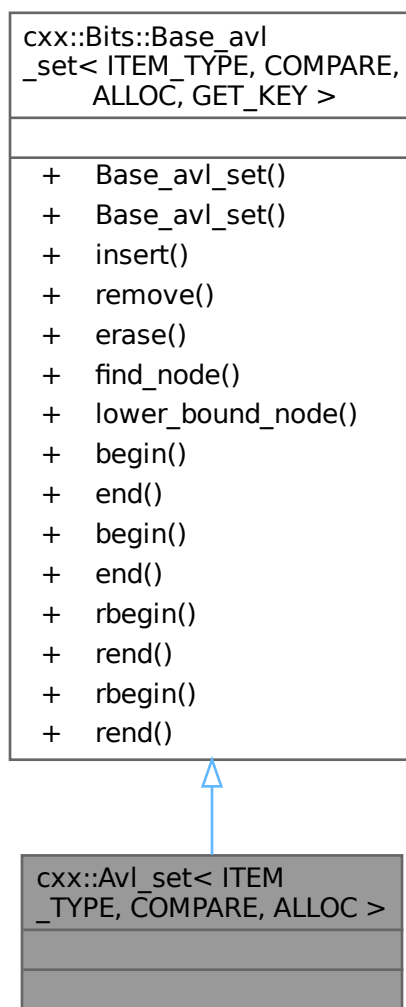
- [l4/cxx/avl_map](#)

15.18 `cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >` Class Template Reference

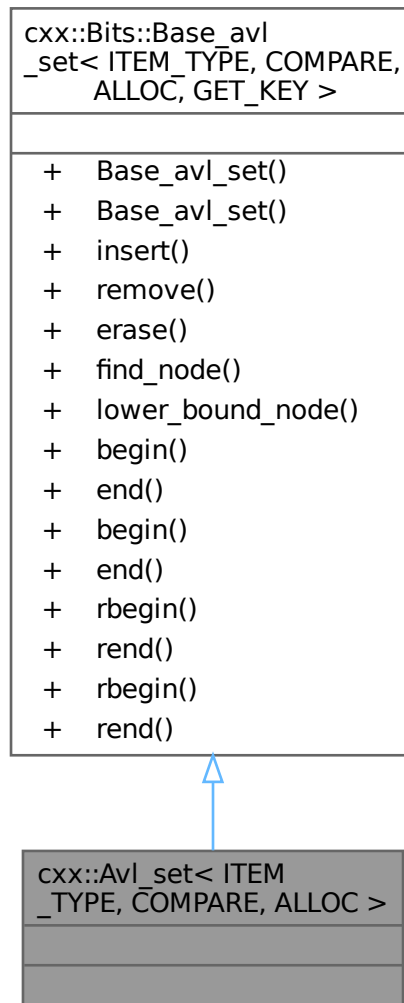
AVL set for simple compareable items.

```
#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.

15.18.1 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 444 of file [avl_set](#).

The documentation for this class was generated from the following file:

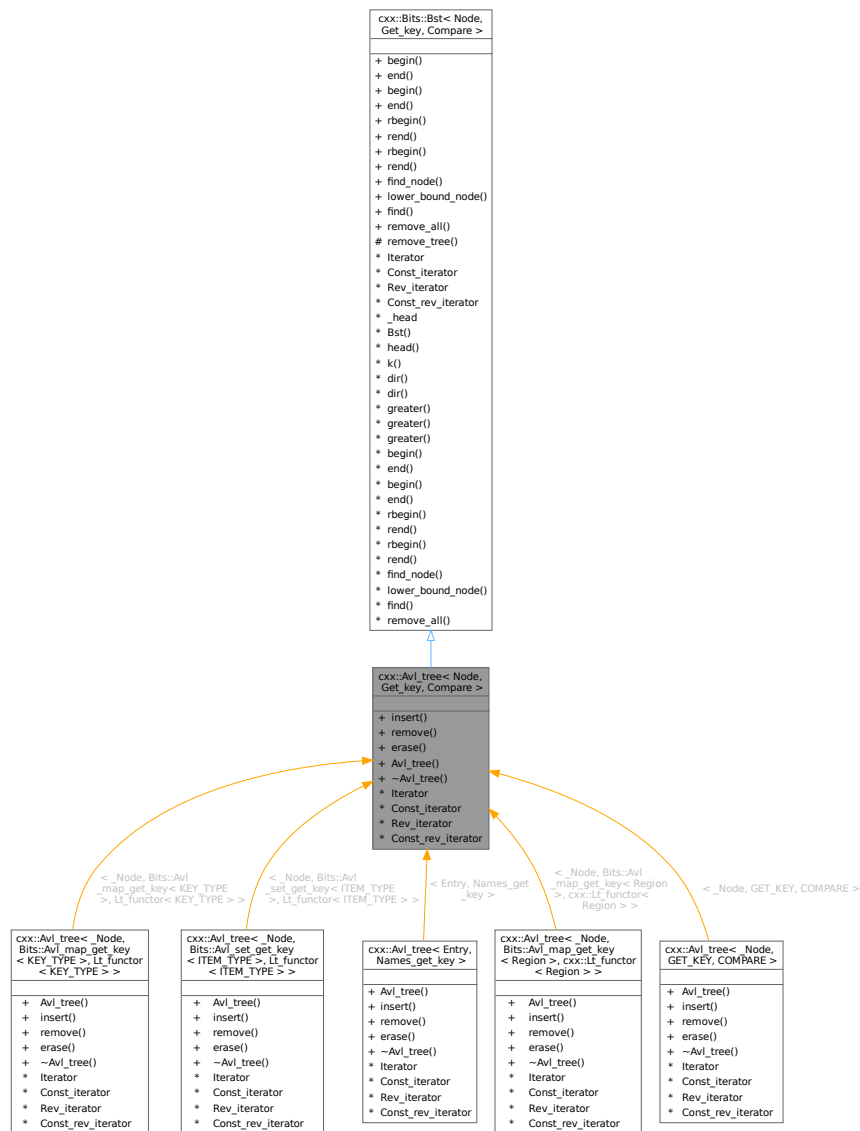
- [l4/cxx/avl_set](#)

15.19 `cxx::Avl_tree< Node, Get_key, Compare >` Class Template Reference

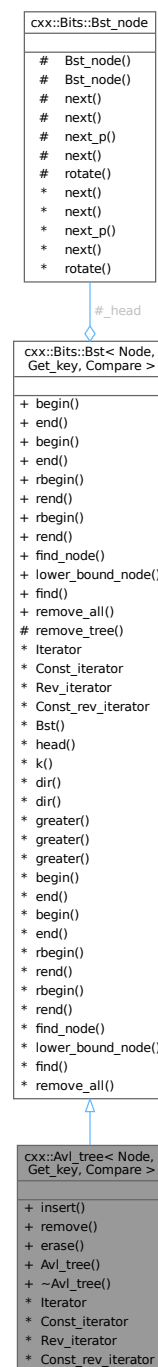
A generic AVL tree.

```
#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.

15.19.1 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 Avl_tree_node).
<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 107 of file [avl_tree](#).

15.19.2 Member Typedef Documentation

15.19.2.1 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 137 of file [avl_tree](#).

15.19.3 Member Function Documentation

15.19.3.1 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.
-----------------	----------------------------

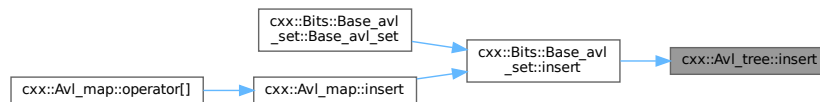
Returns

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 227 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:



15.19.3.2 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.
------------	--------------------------------

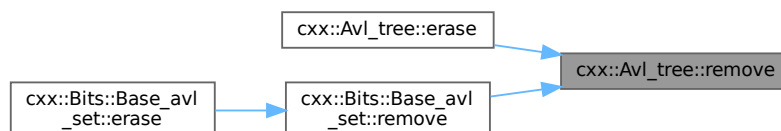
Returns

The pointer to the removed node on success, or 0 if no node with the *key* exists.

Definition at line 289 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:

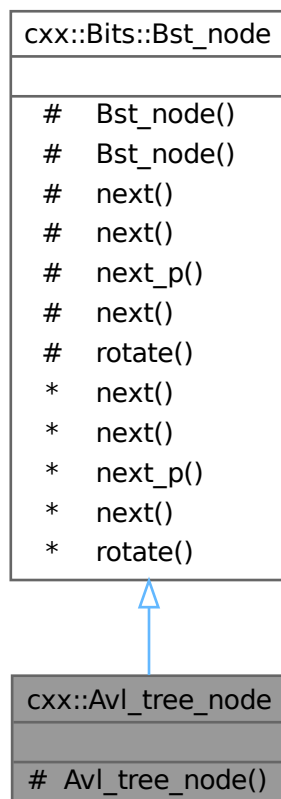
- [I4/cxx/avl_tree](#)

15.20 cxx::Avl_tree_node Class Reference

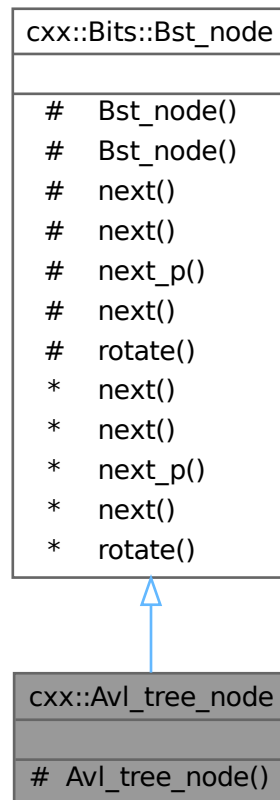
Node of an AVL tree.

```
#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.

15.20.1 Detailed Description

Node of an AVL tree.

Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 38 of file [avl_tree](#).

The documentation for this class was generated from the following file:

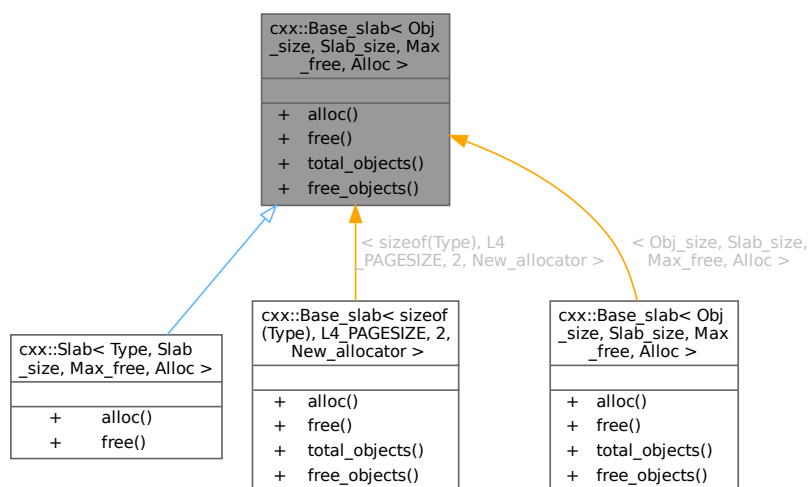
- [l4/cxx/avl_tree](#)

15.21 cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference

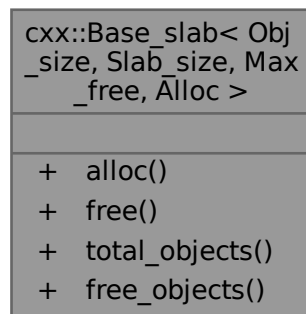
Basic slab allocator.

```
#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.

15.21.1 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](#).

15.21.2 Member Enumeration Documentation

15.21.2.1 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

object_size	Size of an object.
slab_size	Size of a slab.
objects_per_slab	Objects per slab.
max_free_slabs	Maximum number of free slabs.

Definition at line 76 of file [slab_alloc](#).

15.21.3 Member Function Documentation

15.21.3.1 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`.

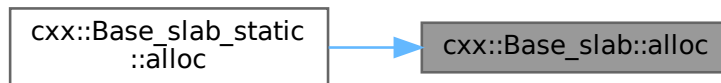
References `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::free()`, `cxx::H_list< T, POLICY >::push_front()`, and `cxx::H_list< T, POLICY >::remove()`.

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:

**15.21.3.2 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

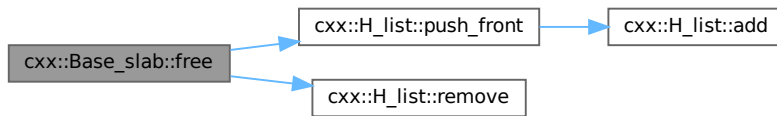
The object must have been allocated with this allocator.

Definition at line 257 of file `slab_alloc`.

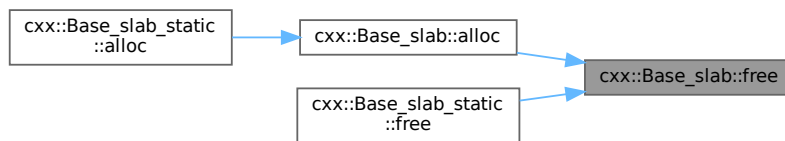
References `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::max_free_slabs`, `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::free()`, `cxx::H_list< T, POLICY >::push_front()`, `cxx::H_list< T, POLICY >::remove()`, and `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::alloc()`.

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:



15.21.3.3 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

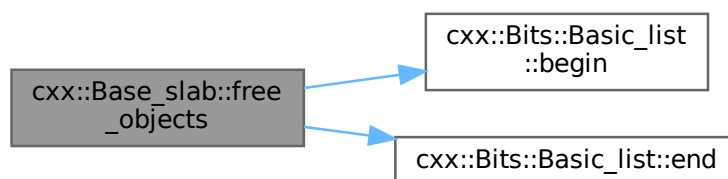
The number of free objects in the slab allocator.

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:



15.21.3.4 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

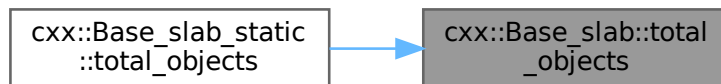
The number of objects managed by the allocator (including the free objects).

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](#)

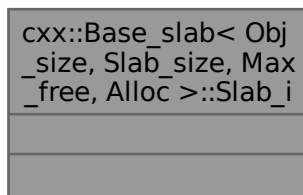
15.22 `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i` Struct Reference

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`:



15.22.1 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:

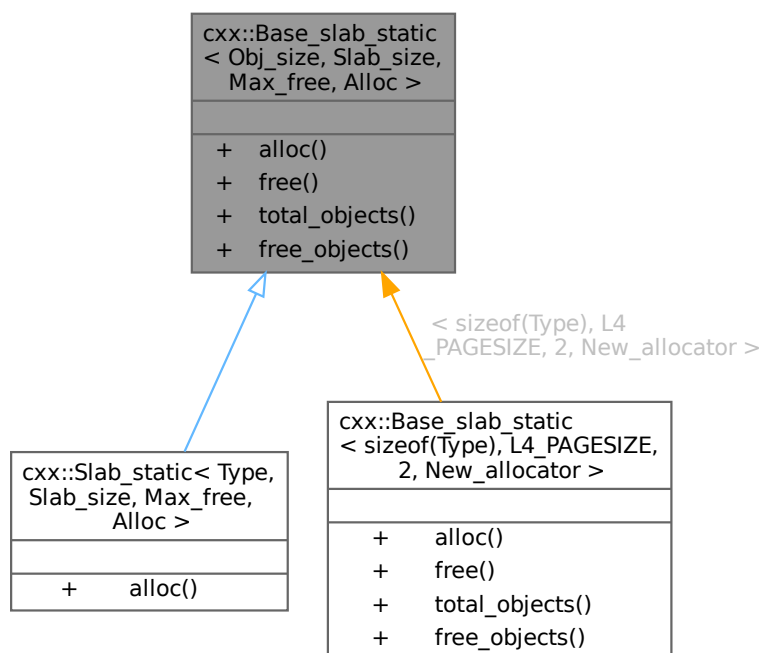
- `I4/cxx/slab_alloc`

15.23 `cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >` Class Template Reference

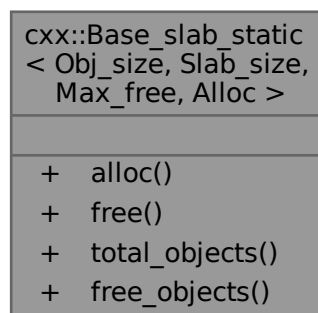
Merged slab allocator (allocators for objects of the same size are merged together).

```
#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.

15.23.1 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_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](#).

15.23.2 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>	Number of objects per slab.
<code>max_free_slabs</code>	Maximum number of free slabs.

Definition at line 406 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_static< Obj_size, Slab_size, Max_free, Alloc >::alloc ( ) [inline],
[noexcept]
```

Allocate an object.

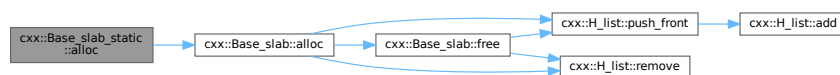
Note

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:



15.23.3.2 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_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:



15.23.3.3 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_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.

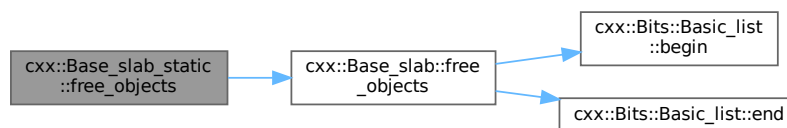
Note

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:



15.23.3.4 `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\_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:

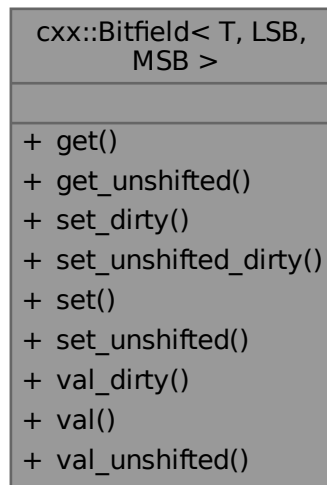
- `I4/cxx/slab_alloc`

15.24 `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` : `T` { `Low_mask` = `((T)~0ULL) >> (sizeof(T)*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< T & >` `Ref`
Reference type to access the bits inside a raw bit field.
- typedef `Value< T const >` `Val`
`Value` type to access the bits inside a raw bit field.
- typedef `Value_unshifted< T & >` `Ref_unshifted`
Reference type to access the bits inside a raw bit field (in place).
- typedef `Value_unshifted< T const >` `Val_unshifted`
`Value` type to access the bits inside a raw bit field (in place).

Static Public Member Functions

- static `Bits_type get (Shift_type val)`
Get the bits out of `val`.
- static `T get_unshifted (Shift_type val)`
Get the bits in place out of `val`.
- static `T set_dirty (T dest, Shift_type val)`
Set the bits corresponding to `val`.
- static `T set_unshifted_dirty (T dest, Shift_type val)`
Set the bits corresponding to `val`.
- static `T set (T dest, Bits_type val)`
Set the bits corresponding to `val`.
- static `T set_unshifted (T dest, Shift_type val)`
Set the bits corresponding to `val`.
- static `T val_dirty (Shift_type val)`
Get the shifted bits for `val`.
- static `T val (Bits_type val)`
Get the shifted bits for `val`.
- static `T val_unshifted (Shift_type val)`
Get the shifted bits for `val`.

15.24.1 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](#).

15.24.2 Member Typedef Documentation

15.24.2.1 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 83 of file [bitfield](#).

15.24.2.2 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 [91](#) of file [bitfield](#).

15.24.3 Member Enumeration Documentation

15.24.3.1 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 [61](#) of file [bitfield](#).

15.24.3.2 Masks

```
template<typename T , unsigned LSB, unsigned MSB>
enum cxx::Bitfield::Masks : T
```

Masks for bitwise operation on internal parts of a bitfield.

Enumerator

Low_mask	Mask value to get Bits bits.
Mask	Mask value to the bits out of a T.

Definition at line [69](#) of file [bitfield](#).

15.24.4 Member Function Documentation

15.24.4.1 get()

```
template<typename T , unsigned LSB, unsigned MSB>
static Bits_type cxx::Bitfield< T, LSB, MSB >::get (
    Shift_type val ) [inline], [static]
```

Get the bits out of val.

Parameters

<code>val</code>	The raw value of the whole bit field.
------------------	---------------------------------------

Returns

The bits form `Lsb` to `Msb` shifted to the right.

Definition at line 108 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:

15.24.4.2 `get_unshifted()`

```

template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::get_unshifted (
    Shift_type val ) [inline], [static]

```

Get the bits in place out of `val`.

Parameters

<code>val</code>	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 121 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:



15.24.4.3 set()

```

template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::set (
    T 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.

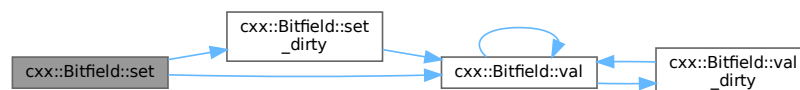
Returns

The new value of the whole bit field.

Definition at line 170 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:



15.24.4.4 set_dirty()

```

template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::set_dirty (
    T 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 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 136 of file `bitfield`.

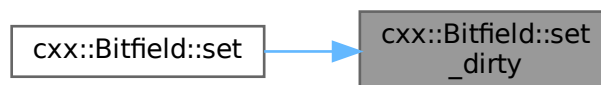
References `cxx::Bitfield< T, LSB, MSB >::Lsb`, `cxx::Bitfield< T, LSB, MSB >::Mask`, and `cxx::Bitfield< T, LSB, MSB >::val()`.

Referenced by `cxx::Bitfield< T, LSB, MSB >::set()`.

Here is the call graph for this function:



Here is the caller graph for this function:

15.24.4.5 `set_unshifted()`

```

template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::set_unshifted (
    T 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 Lsb bits to the left that shall be set into the bit field.

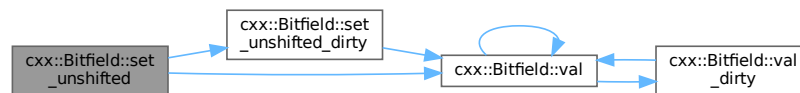
Returns

the new value of the whole bit field.

Definition at line [182](#) 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:

15.24.4.6 `set_unshifted_dirty()`

```

template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::set_unshifted_dirty (
    T 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 Lsb 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 156 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:

**15.24.4.7 val()**

```

template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::val (
    Bits_type val ) [inline], [static]
  
```

Get the shifted bits for `val`.

Parameters

<code>val</code>	The value to set into the bits.
------------------	---------------------------------

Returns

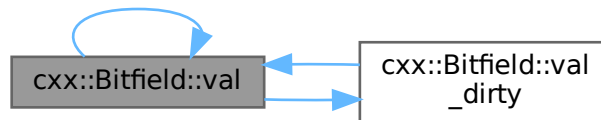
The raw bit field value.

Definition at line 205 of file `bitfield`.

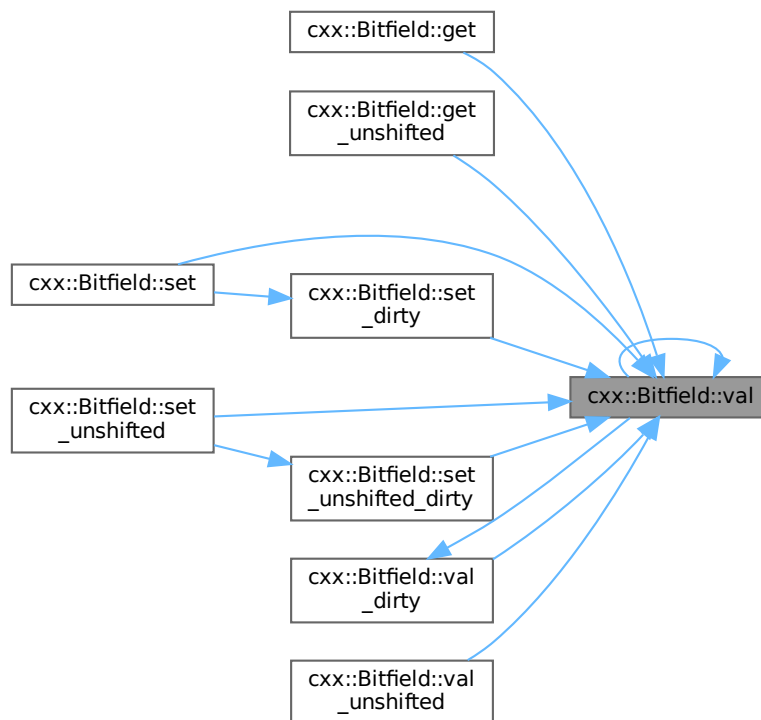
References `cxx::Bitfield< T, LSB, MSB >::Low_mask`, `cxx::Bitfield< T, LSB, MSB >::val()`, and `cxx::Bitfield< T, LSB, MSB >::val_dir`

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:



15.24.4.8 val_dirty()

```

template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::val_dirty (
    Shift_type val ) [inline], [static]
  
```

Get the shifted bits for `val`.

Parameters

<i>val</i>	The value to set into the bits.
------------	---------------------------------

Returns

The raw bit field value.

Precondition

val must not contain more than [Bits](#) bits.

Note

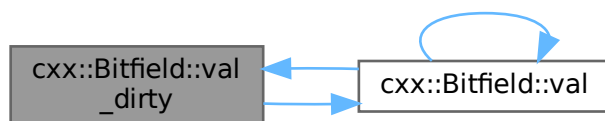
This function does not mask *val* to the right number of bits.

Definition at line [196](#) of file [bitfield](#).

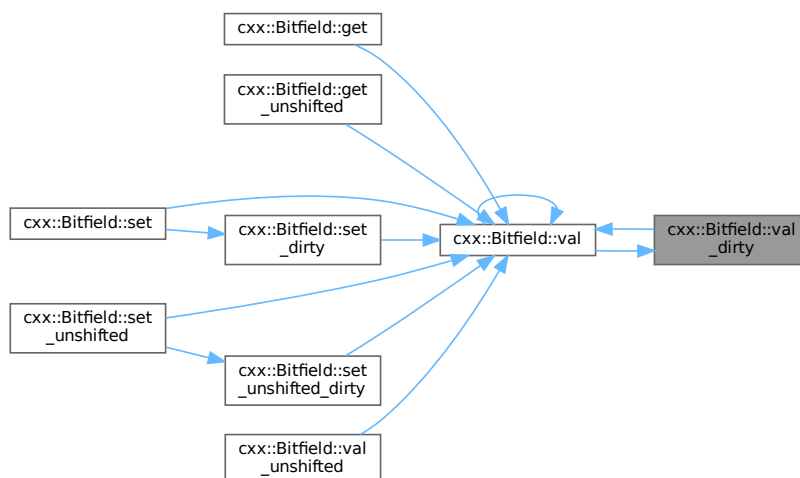
References [cxx::Bitfield< T, LSB, MSB >::Lsb](#), and [cxx::Bitfield< T, LSB, MSB >::val\(\)](#).

Referenced by [cxx::Bitfield< T, LSB, MSB >::val\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.24.4.9 `val_unshifted()`

```
template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::val_unshifted (
    Shift_type val ) [inline], [static]
```

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 215 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:

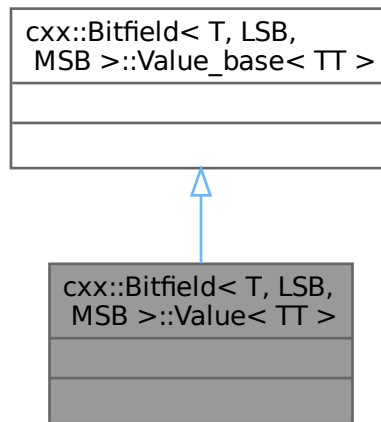
- `I4/cxx/bitfield`

15.25 `cxx::Bitfield< T, LSB, MSB >::Value< TT >` Class Template Reference

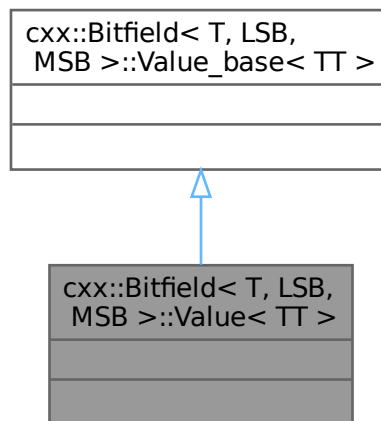
Internal helper type.

```
#include <bitfield>
```

Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >::Value< TT >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value< TT >`:



15.25.1 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 [237](#) of file [bitfield](#).

The documentation for this class was generated from the following file:

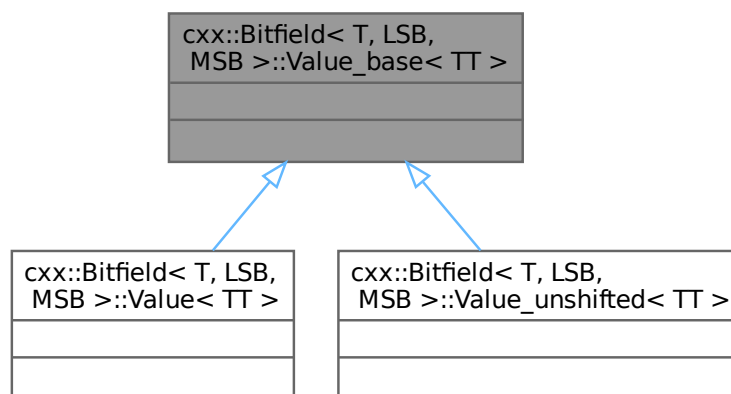
- `l4/cxx/bitfield`

15.26 `cxx::Bitfield< T, LSB, MSB >::Value_base< TT >` Class Template Reference

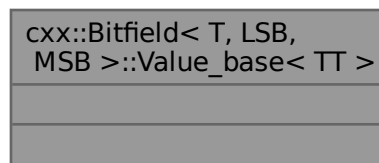
Internal helper type.

```
#include <bitfield>
```

Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >::Value_base< TT >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value_base< TT >`:



15.26.1 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 219 of file [bitfield](#).

The documentation for this class was generated from the following file:

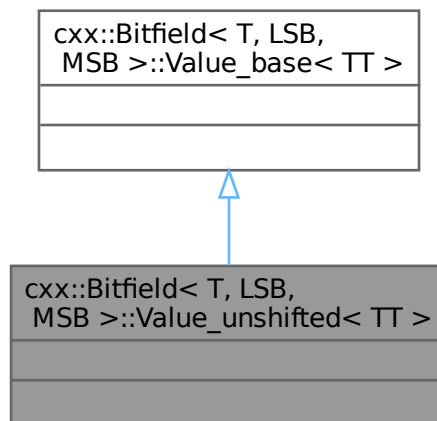
- I4/cxx/bitfield

15.27 `cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >` Class Template Reference

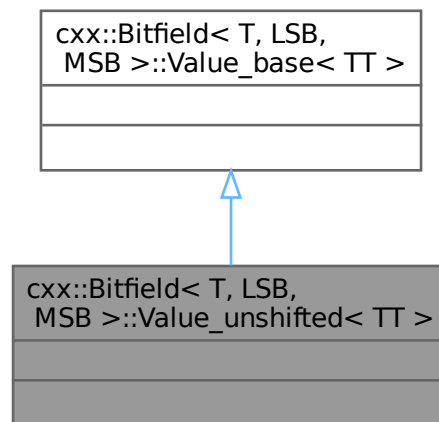
Internal helper type.

```
#include <bitfield>
```

Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >`:



15.27.1 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 251 of file [bitfield](#).

The documentation for this class was generated from the following file:

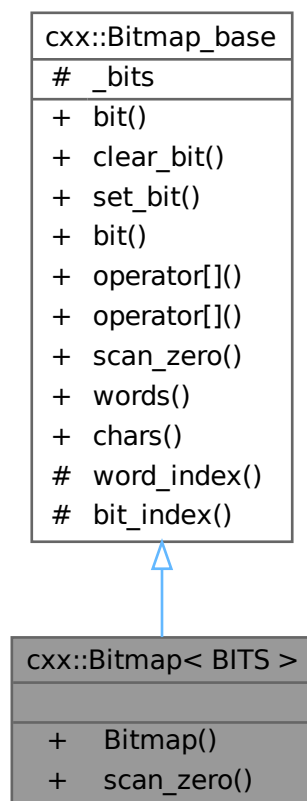
- `I4/cxx/bitfield`

15.28 `cxx::Bitmap< BITS >` Class Template Reference

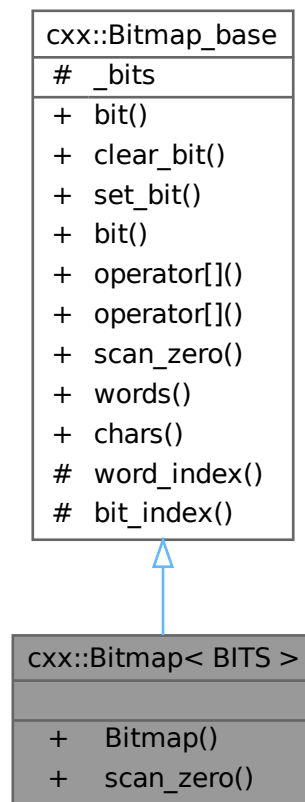
A static bit map.

```
#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 [set_bit](#) (long [bit](#)) noexcept
Set bit bit.
- [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) throw ()
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.

15.28.1 Detailed Description

```
template<int BITS>
class cxx::Bitmap< BITS >
```

A static bit map.

Parameters

<code>BITS</code>	the number of bits that shall be in the bitmap.
-------------------	---

Definition at line 180 of file `bitmap`.

15.28.2 Member Function Documentation

15.28.2.1 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

<i>>=</i>	0 Number of first zero bit found.
<i>-1</i>	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 285 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:

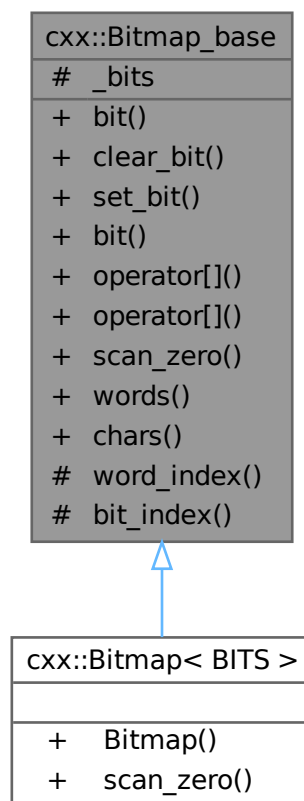
- [l4/cxx/bitmap](#)

15.29 cxx::Bitmap_base Class Reference

Basic bitmap abstraction.

```
#include <bitmap>
```

Inheritance diagram for cxx::Bitmap_base:



Collaboration diagram for cxx::Bitmap_base:

cxx::Bitmap_base
_bits
+ bit()
+ clear_bit()
+ set_bit()
+ bit()
+ operator[]()
+ operator[]()
+ scan_zero()
+ words()
+ chars()
word_index()
bit_index()

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 [set_bit](#) (long [bit](#)) noexcept
Set bit bit.
- [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) throw ()
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.

15.29.1 Detailed Description

Basic bitmap abstraction.

This abstraction keeps a pointer to a memory area that is used as bitmap.

Definition at line 30 of file [bitmap](#).

15.29.2 Member Enumeration Documentation

15.29.2.1 anonymous enum

```
anonymous enum [protected]
```

Enumerator

W_bits	number of bits in word_type
C_bits	number of bits in char

Definition at line 38 of file [bitmap](#).

15.29.3 Member Function Documentation

15.29.3.1 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.
------------	--------------------------------

Returns

0 if *bit* is not set, != 0 if *bit* is set.

Definition at line 238 of file [bitmap](#).

15.29.3.2 bit() [2/2]

```
void cxx::Bitmap_base::bit (
    long bit,
    bool on ) [inline], [noexcept]
```

Set the value of bit *bit* to *on*.

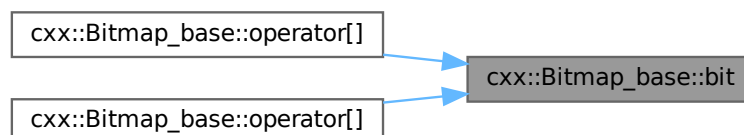
Parameters

<i>bit</i>	the number of the bit
<i>on</i>	the boolean value that shall be assigned to the bit.

Definition at line 211 of file [bitmap](#).

Referenced by [operator\[\]\(\)](#), and [operator\[\]\(\)](#).

Here is the caller graph for this function:



15.29.3.3 `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 61 of file [bitmap](#).

References [W_bits](#).

15.29.3.4 `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 220 of file [bitmap](#).

15.29.3.5 `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.
------------	--------------------------------

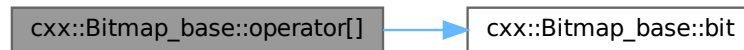
Returns

0 if *bit* is not set, != 0 if *bit* is set.

Definition at line 143 of file [bitmap](#).

References [bit\(\)](#).

Here is the call graph for this function:



15.29.3.6 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.
------------	-------------

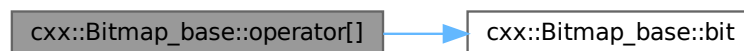
Returns

lvalue for *bit*

Definition at line 151 of file [bitmap](#).

References [bit\(\)](#).

Here is the call graph for this function:



15.29.3.7 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

<i>>=</i>	0 Number of first zero bit found.
<i>-1</i>	All bits between <i>start_bit</i> and <i>max_bit</i> are set.

Definition at line 259 of file [bitmap](#).

Referenced by [cxx::Bitmap< BITS >::scan_zero\(\)](#).

Here is the caller graph for this function:



15.29.3.8 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 229 of file [bitmap](#).

15.29.3.9 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 54 of file [bitmap](#).

References [W_bits](#).

The documentation for this class was generated from the following file:

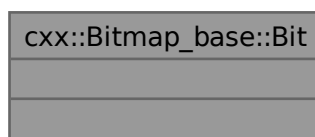
- I4/cxx/bitmap

15.30 cxx::Bitmap_base::Bit Class Reference

A writeable bit in a bitmap.

```
#include <bitmap>
```

Collaboration diagram for cxx::Bitmap_base::Bit:



15.30.1 Detailed Description

A writeable bit in a bitmap.

Definition at line 66 of file [bitmap](#).

The documentation for this class was generated from the following file:

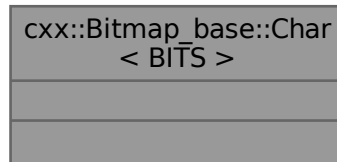
- I4/cxx/bitmap

15.31 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 >:



15.31.1 Detailed Description

```
template<long BITS>
class cxx::Bitmap_base::Char< BITS >
```

Helper abstraction for a byte contained in the bitmap.

Definition at line [103](#) of file [bitmap](#).

The documentation for this class was generated from the following file:

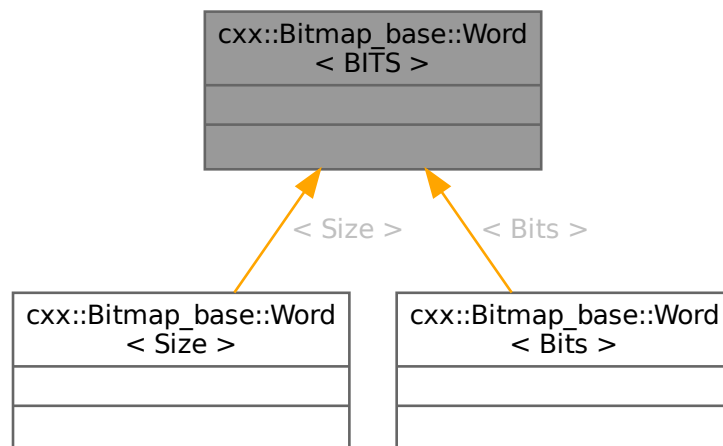
- [I4/cxx/bitmap](#)

15.32 cxx::Bitmap_base::Word< BITS > Class Template Reference

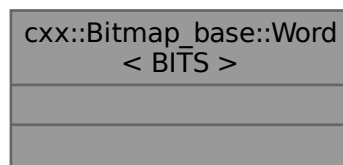
Helper abstraction for a word contained in the bitmap.

```
#include <bitmap>
```


Inheritance diagram for cxx::Bitmap_base::Word< BITS >:



Collaboration diagram for cxx::Bitmap_base::Word< BITS >:



15.32.1 Detailed Description

```

template<long BITS>
class cxx::Bitmap_base::Word< BITS >

```

Helper abstraction for a word contained in the bitmap.

Definition at line 87 of file [bitmap](#).

The documentation for this class was generated from the following file:

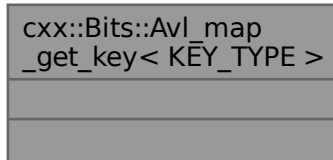
- I4/cxx/bitmap

15.33 `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 >`:



15.33.1 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:

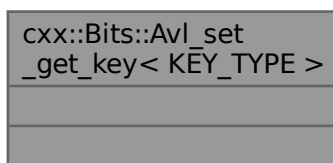
- [l4/cxx/avl_map](#)

15.34 `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 >`:



15.34.1 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 107 of file [avl_set](#).

The documentation for this struct was generated from the following file:

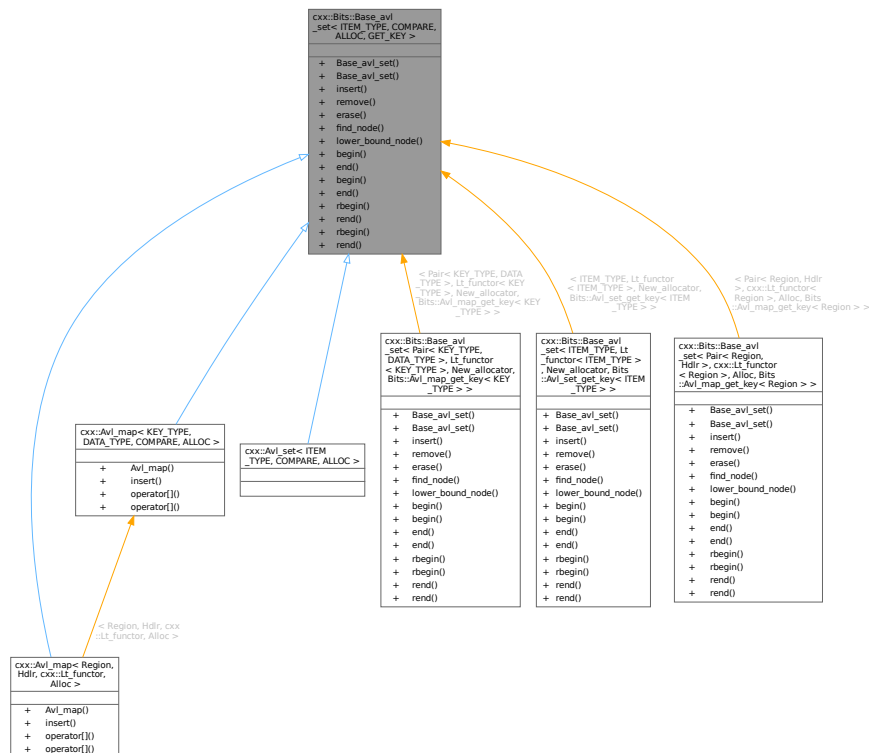
- [l4/cxx/avl_set](#)

15.35 cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > Class Template Reference

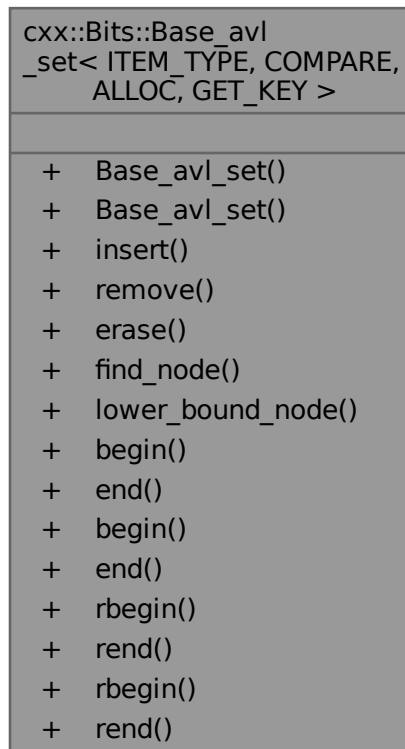
Internal: AVL set with internally managed nodes.

```
#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.

15.35.1 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 131 of file [avl_set](#).

15.35.2 Member Enumeration Documentation

15.35.2.1 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 140 of file [avl_set](#).

15.35.3 Constructor & Destructor Documentation

15.35.3.1 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>	Node allocator.
--------------	---------------------------------

Create an empty set (AVL-tree based).

Definition at line 246 of file [avl_set](#).

15.35.3.2 Base_avl_set() [2/2]

```
template<typename Item , class Compare , template< typename A > class Alloc, typename KEY_TYPE>
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

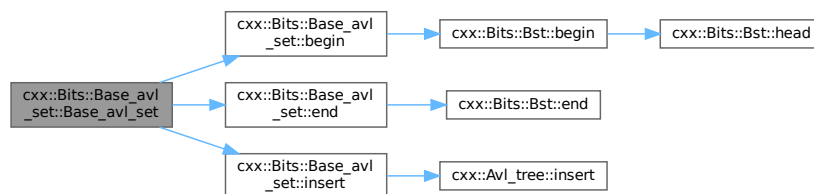
<i>o</i>	The set to copy.
----------	------------------

Creates a deep copy of the set with all its items.

Definition at line 402 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:



15.35.4 Member Function Documentation

15.35.4.1 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 356 of file [avl_set](#).

References [cxx::Bits::Bst< Node, Get_key, Compare >::begin\(\)](#).

Here is the call graph for this function:



15.35.4.2 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 345 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:



15.35.4.3 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 361 of file [avl_set](#).

References [cxx::Bits::Bst< Node, Get_key, Compare >::end\(\)](#).

Here is the call graph for this function:

**15.35.4.4 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 350 of file [avl_set](#).

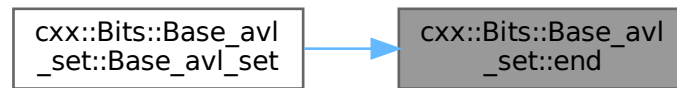
References [cxx::Bits::Bst< Node, Get_key, Compare >::end\(\)](#).

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:



15.35.4.5 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.

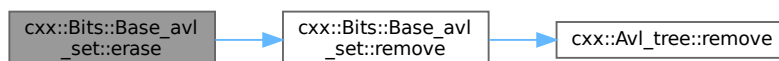
Parameters

<i>item</i>	The key of the item to remove.
-------------	--------------------------------

Definition at line 313 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:



15.35.4.6 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

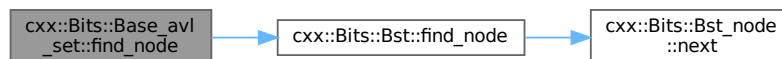
A smart pointer to the element found. If no element was found the smart pointer will be invalid.

Definition at line 324 of file `avl_set`.

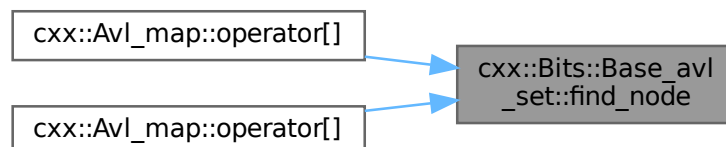
References `cxx::Bits::Bst< Node, Get_key, Compare >::find_node()`.

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:

15.35.4.7 `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.

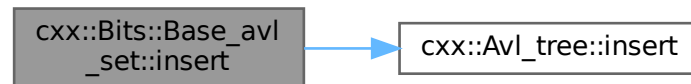
Insert a new item into the set, each item can only be once in the set.

Definition at line 412 of file [avl_set](#).

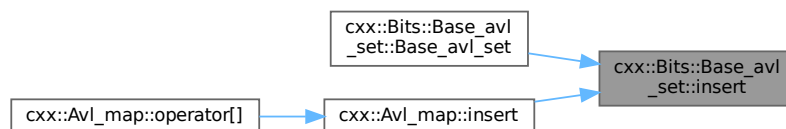
References [cxx::Pair< First, Second >::first](#), [cxx::Avl_tree< Node, Get_key, Compare >::insert\(\)](#), and [cxx::Pair< First, Second >::second](#).

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:

**15.35.4.8 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.
------------	--------------------------

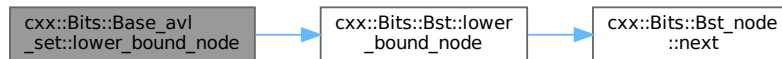
Returns

Smart pointer to the first node greater or equal to `key`. Will be invalid if no such element was found.

Definition at line 335 of file `avl_set`.

References `cxx::Bits::Bst< Node, Get_key, Compare >::lower_bound_node()`.

Here is the call graph for this function:



15.35.4.9 `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.

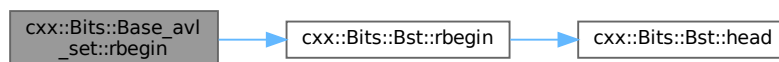
Returns

The mutable backward iterator for the last element of the set.

Definition at line 378 of file `avl_set`.

References `cxx::Bits::Bst< Node, Get_key, Compare >::rbegin()`.

Here is the call graph for this function:



15.35.4.10 `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.

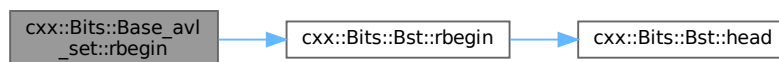
Returns

The constant backward iterator for the last element in the set.

Definition at line 367 of file [avl_set](#).

References [cxx::Bits::Bst< Node, Get_key, Compare >::rbegin\(\)](#).

Here is the call graph for this function:



15.35.4.11 `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

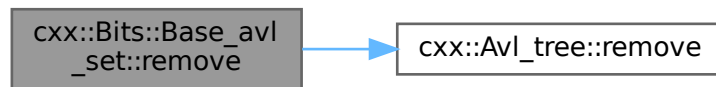
0	Success
-E_noent	Item does not exist

Definition at line 295 of file [avl_set](#).

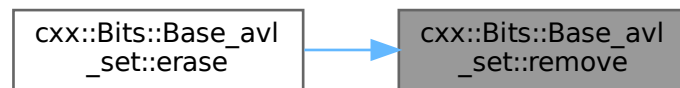
References [cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::E_noent](#), and [cxx::Avl_tree< Node, Get_key,](#)

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:



15.35.4.12 `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 383 of file [avl_set](#).

References [cxx::Bits::Bst< Node, Get_key, Compare >::rend\(\)](#).

Here is the call graph for this function:



15.35.4.13 `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 372 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:

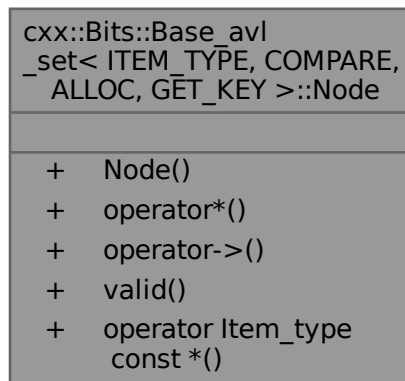
- [l4/cxx/avl_set](#)

15.36 `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.

15.36.1 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 175 of file `avl_set`.

15.36.2 Member Function Documentation

15.36.2.1 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 192 of file [avl_set](#).

15.36.2.2 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 198 of file [avl_set](#).

15.36.2.3 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 204 of file [avl_set](#).

The documentation for this class was generated from the following file:

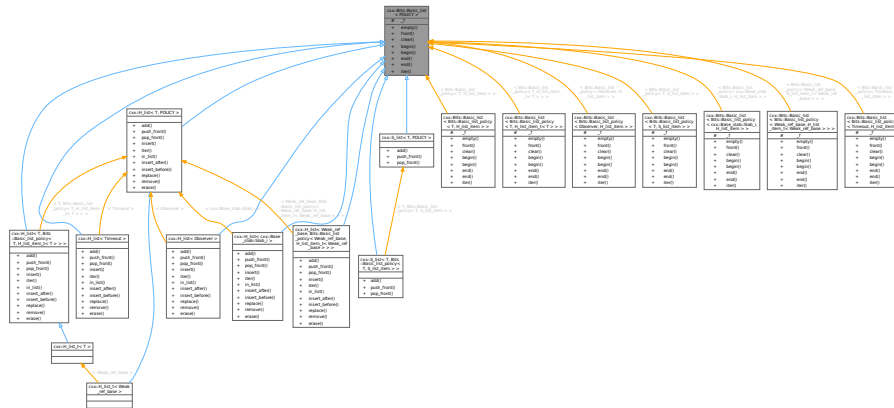
- [I4/cxx/avl_set](#)

15.37 cxx::Bits::Basic_list< POLICY > Class Template Reference

Internal: Common functions for all head-based list implementations.

```
#include <list_basics.h>
```

Inheritance diagram for cxx::Bits::Basic_list< POLICY >:



Collaboration diagram for cxx::Bits::Basic_list< POLICY >:

cxx::Bits::Basic_list< POLICY >	
#	_f
+	empty()
+	front()
+	clear()
+	begin()
+	begin()
+	end()
+	end()
+	iter()

Public Member Functions

- 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 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.

15.37.1 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](#).

15.37.2 Member Function Documentation

15.37.2.1 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](#).

15.37.2.2 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:

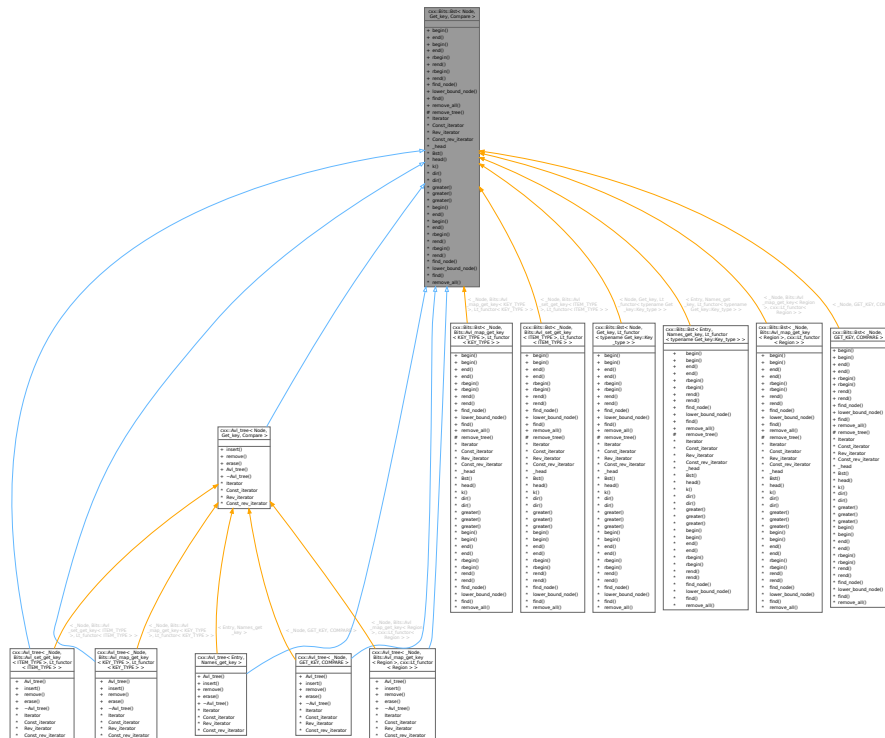
- `l4/cxx/bits/list_basics.h`

15.38 cxx::Bits::Bst< Node, Get_key, Compare > Class Template Reference

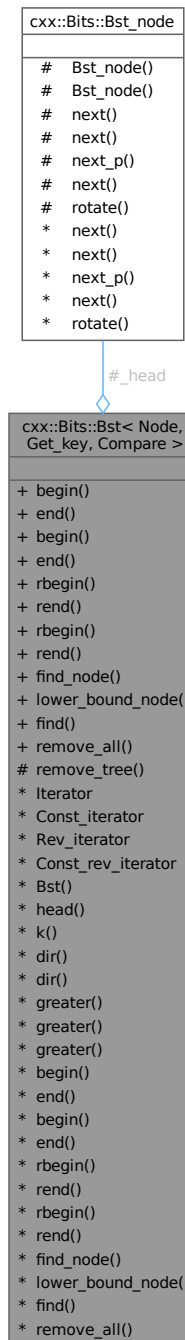
Basic binary search tree (BST).

```
#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.

15.38.1 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 40 of file [bst.h](#).

15.38.2 Member Function Documentation

15.38.2.1 `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 190 of file [bst.h](#).

References [cxx::Bits::Bst< Node, Get_key, Compare >::head\(\)](#).

Here is the call graph for this function:

**15.38.2.2 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 179 of file [bst.h](#).

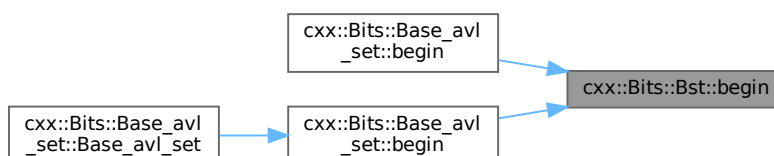
References [cxx::Bits::Bst< Node, Get_key, Compare >::head\(\)](#).

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:



15.38.2.3 `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.

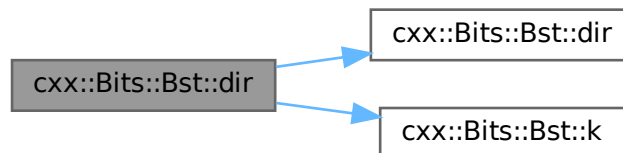
Return values

<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 135 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:

**15.38.2.4** `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

<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 118 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:

**15.38.2.5** `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 195 of file [bst.h](#).

15.38.2.6 `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.

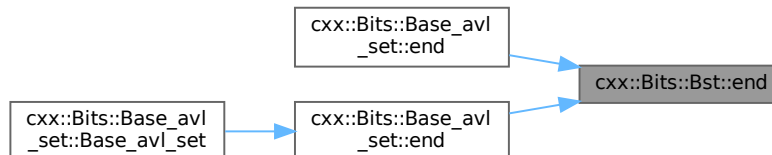
Returns

The end marker for the constant forward iterator.

Definition at line 184 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:

**15.38.2.7 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.
------------	---

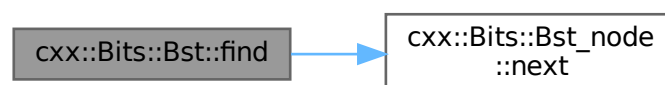
Returns

A valid iterator for the node with the given *key*, or an invalid iterator if *key* was not found.

Definition at line 312 of file [bst.h](#).

References [cxx::Bits::Bst_node::next\(\)](#).

Here is the call graph for this function:



15.38.2.8 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

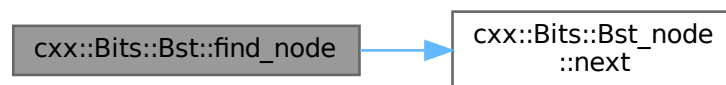
A pointer to the node with the given *key*, or NULL if *key* was not found.

Definition at line 276 of file [bst.h](#).

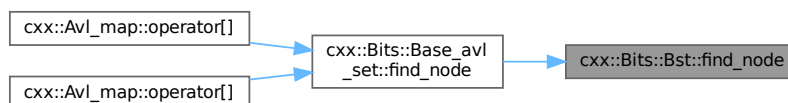
References [cxx::Bits::Bst_node::next\(\)](#).

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:



15.38.2.9 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 292 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:



15.38.2.10 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 212 of file `bst.h`.

References `cxx::Bits::Bst< Node, Get_key, Compare >::head()`.

Here is the call graph for this function:

**15.38.2.11 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 201 of file `bst.h`.

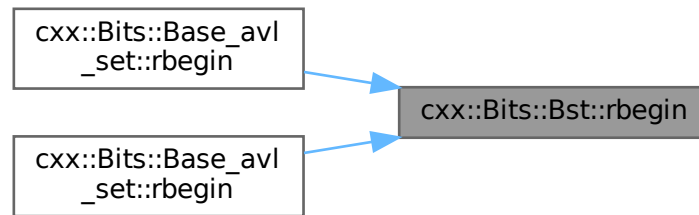
References `cxx::Bits::Bst< Node, Get_key, Compare >::head()`.

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:



15.38.2.12 `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.
-----------------	---

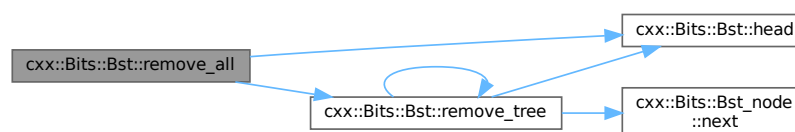
The callback may delete the elements. The function guarantees that the elements are no longer used after the callback has been called.

Definition at line 258 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:



15.38.2.13 `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

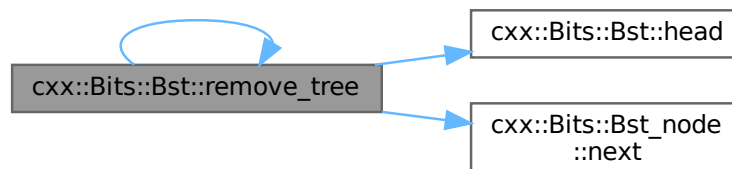
<i>head</i>	Head of the the subtree to remove
<i>callback</i>	Optional function called on each removed element.

Definition at line 158 of file `bst.h`.

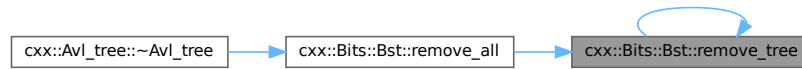
References `cxx::Bits::Bst< Node, Get_key, Compare >::head()`, `cxx::Bits::Direction::L`, `cxx::Bits::Bst_node::next()`, `cxx::Bits::Direction::R`, and `cxx::Bits::Bst< Node, Get_key, Compare >::remove_tree()`.

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:



15.38.2.14 `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 217 of file [bst.h](#).

15.38.2.15 `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.

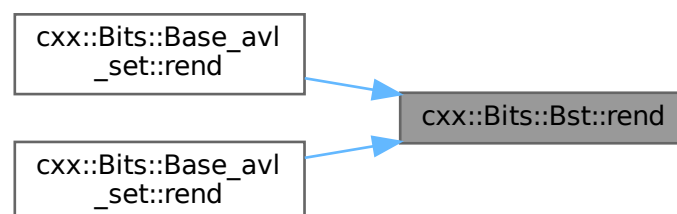
Returns

The end marker for the constant backward iterator.

Definition at line 206 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:

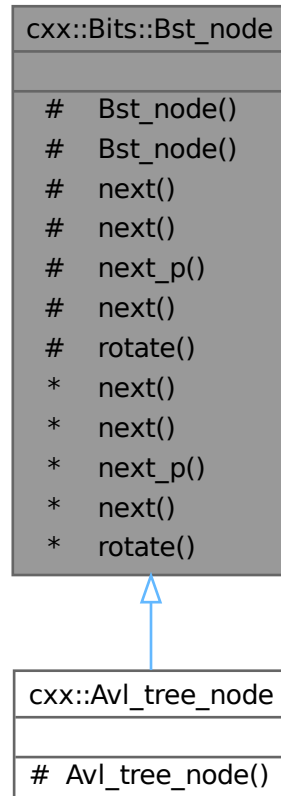
- [I4/cxx/bits/bst.h](#)

15.39 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.

15.39.1 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](#)

15.40 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.

15.40.1 Detailed Description

The direction to go in a binary search tree.

Definition at line 39 of file [bst_base.h](#).

15.40.2 Member Enumeration Documentation

15.40.2.1 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](#).

15.40.3 Member Function Documentation

15.40.3.1 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:

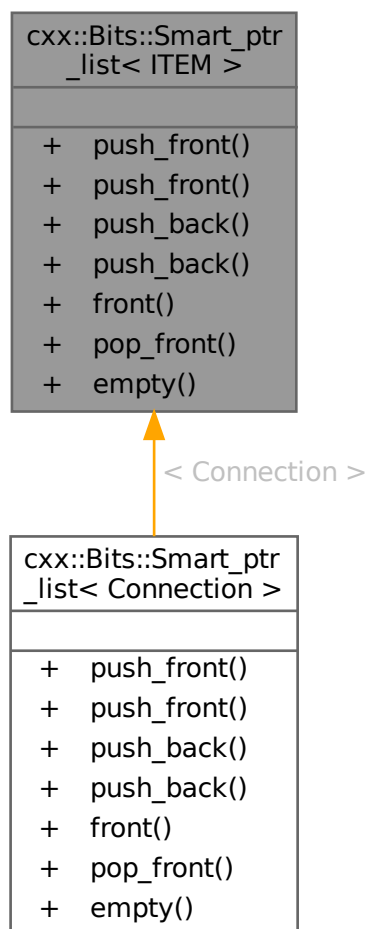
- [I4/cxx/bits/bst_base.h](#)

15.41 `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< ITEM ></code>
<ul style="list-style-type: none"> + <code>push_front()</code> + <code>push_front()</code> + <code>push_back()</code> + <code>push_back()</code> + <code>front()</code> + <code>pop_front()</code> + <code>empty()</code>

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.

15.41.1 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](#).

15.41.2 Member Function Documentation

15.41.2.1 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:

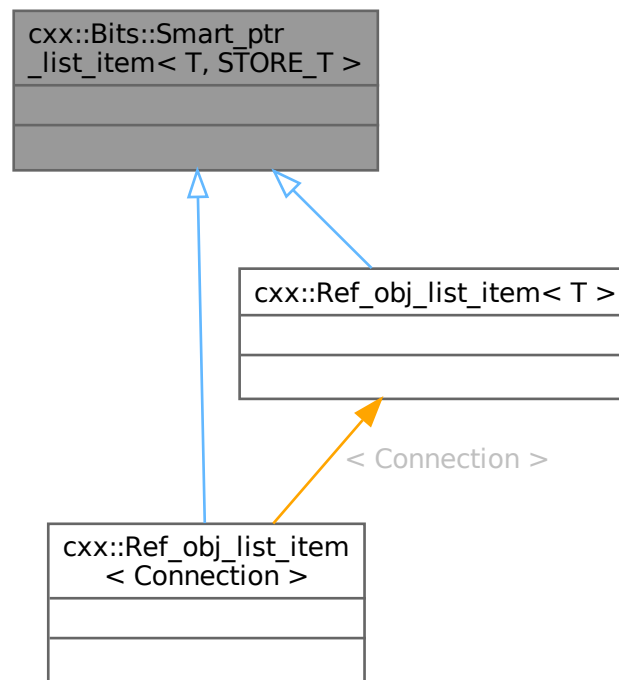
- I4/cxx/bits/[smart_ptr_list.h](#)

15.42 cxx::Bits::Smart_ptr_list_item< T, STORE_T > Class Template Reference

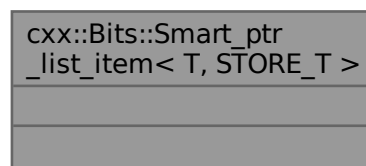
[List](#) item for an arbitrary item in a [Smart_ptr_list](#).

```
#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 >:



15.42.1 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 <code>get()</code> 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:

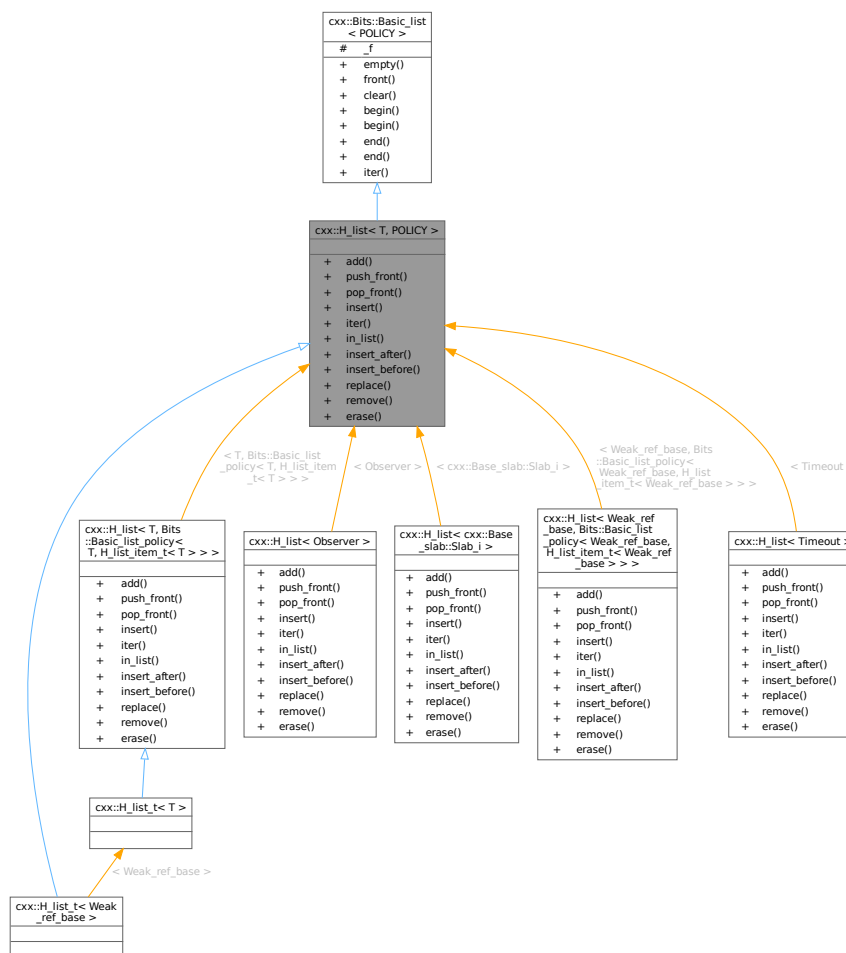
- [I4/cxx/bits/smart_ptr_list.h](#)

15.43 `cxx::H_list< T, POLICY >` Class Template Reference

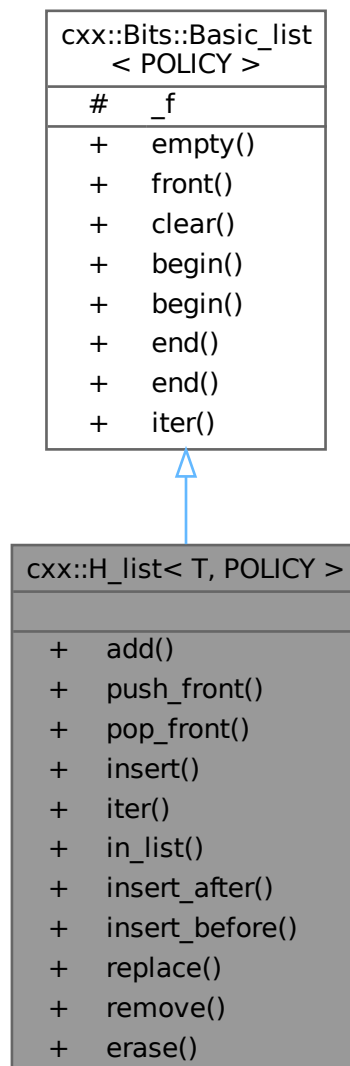
General double-linked list of unspecified `cxx::H_list_item` elements.

```
#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.

15.43.1 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](#).

15.43.2 Member Function Documentation

15.43.2.1 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 end() .
----------	---

Returns

New iterator pointing to the element after the removed one.

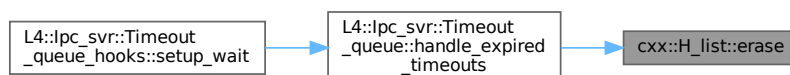
Note

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:



15.43.2.2 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 end() is given, the element will be inserted at the beginning of the queue.

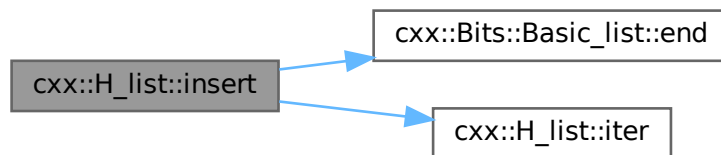
Returns

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:



15.43.2.3 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 end() .

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:

**15.43.2.4 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.

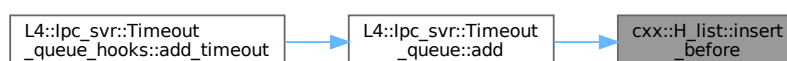
Parameters

<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 <code>end()</code> .

Definition at line 191 of file `hlist`.

Referenced by `L4::lpc_svr::Timeout_queue::add()`.

Here is the caller graph for this function:



15.43.2.5 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

c	List element to start the iteration.
----------	--------------------------------------

Returns

A mutable forward iterator.

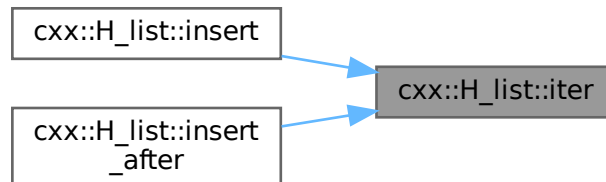
Precondition

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:



15.43.2.6 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.

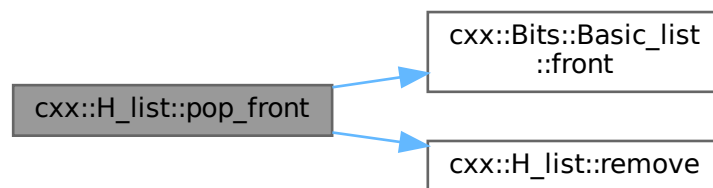
Precondition

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:

**15.43.2.7 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.

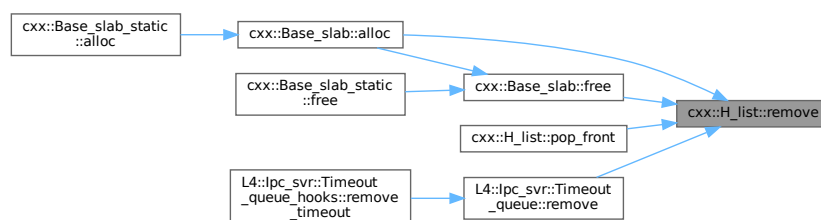
Parameters

<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:



15.43.2.8 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:

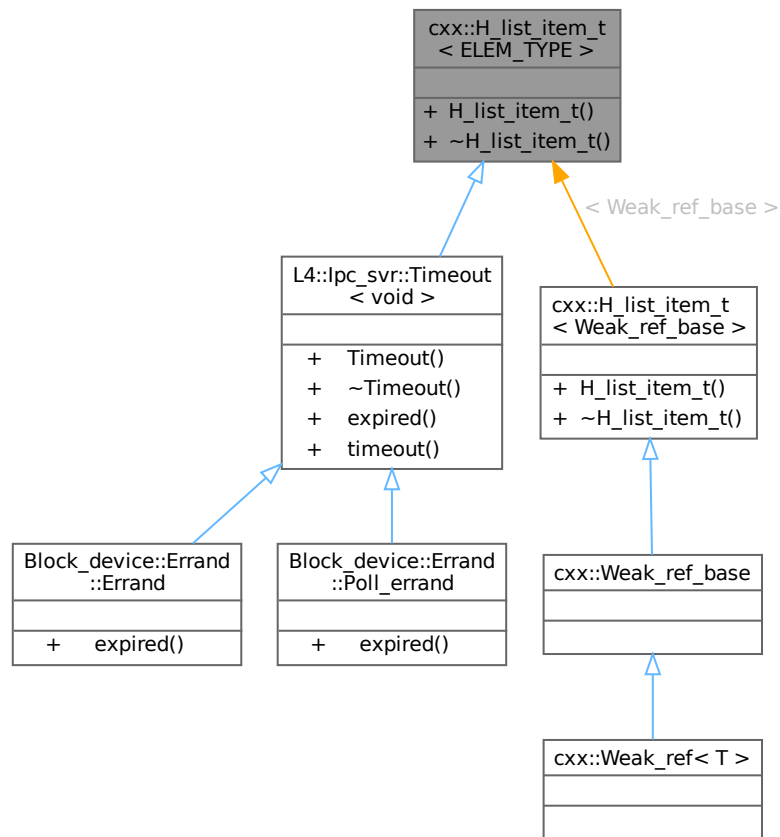
- l4/cxx/hlist

15.44 cxx::H_list_item_t< ELEM_TYPE > Class Template Reference

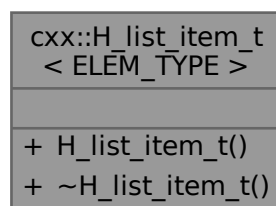
Basic element type for a double-linked [H_list](#).

```
#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.

15.44.1 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](#).

15.44.2 Constructor & Destructor Documentation

15.44.2.1 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](#).

15.44.2.2 ~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:

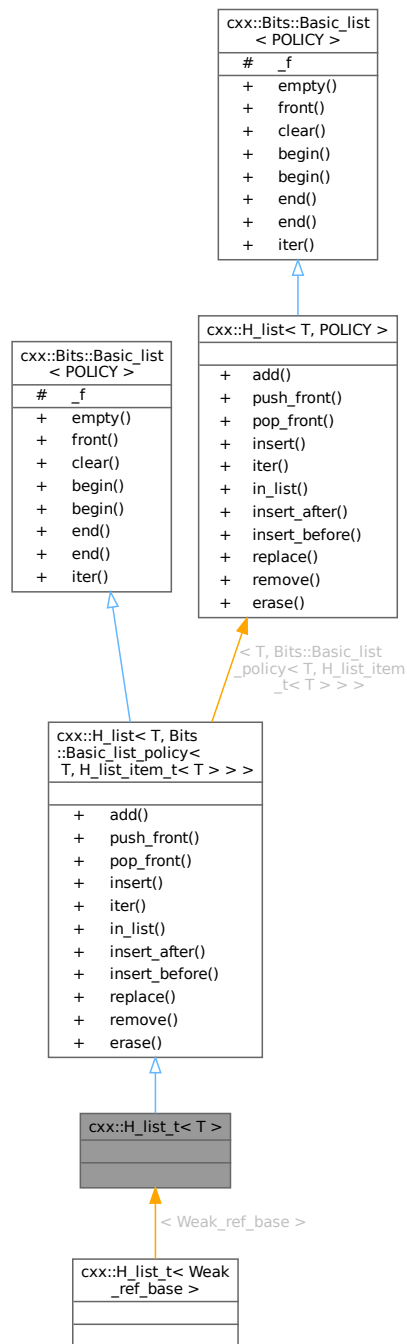
- I4/cxx/hlist

15.45 cxx::H_list_t< T > Struct Template Reference

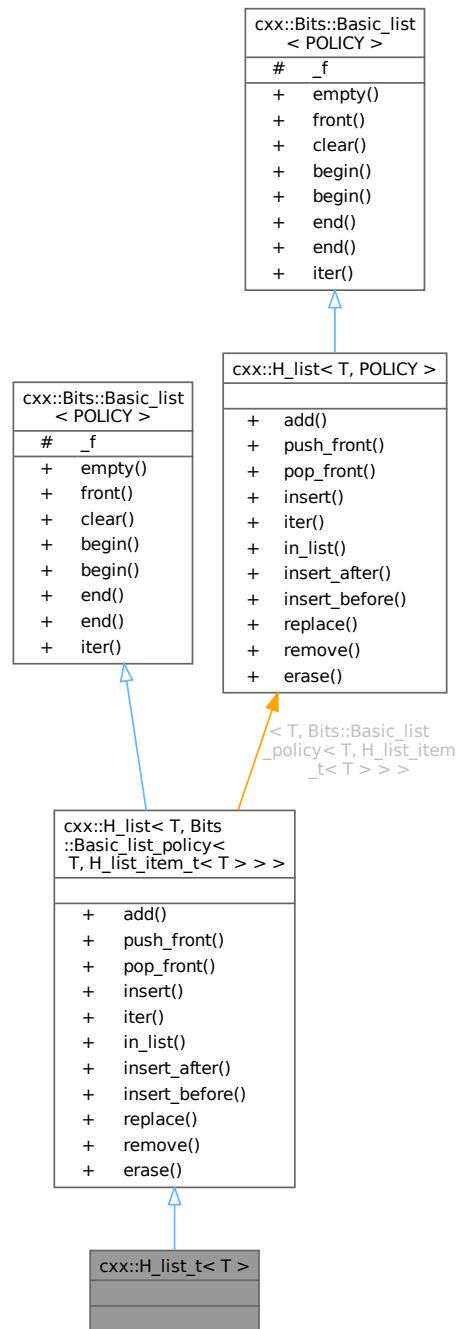
Double-linked list of typed [H_list_item_t](#) elements.

```
#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.

15.45.1 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:

- `I4/cxx/hlist`

15.46 `cxx::List< D, Alloc >` Class Template Reference

Doubly linked list, with internal allocation.

```
#include <list>
```

Collaboration diagram for `cxx::List< D, Alloc >`:

<code>cxx::List< D, Alloc ></code>
<ul style="list-style-type: none"> + <code>push_back()</code> + <code>push_front()</code> + <code>remove()</code> + <code>size()</code> + <code>operator[]()</code> + <code>operator[]()</code> + <code>items()</code>

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.

15.46.1 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](#).

15.46.2 Member Function Documentation

15.46.2.1 **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](#).

15.46.2.2 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:

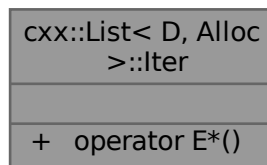
- l4/cxx/list

15.47 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)

15.47.1 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:

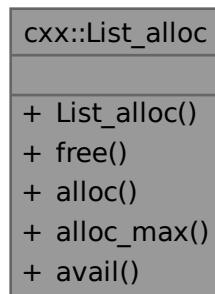
- l4/cxx/list

15.48 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)
Allocate a memory block.
- void * [alloc_max](#) (unsigned long min, unsigned long *max, unsigned long align, unsigned granularity)
Allocate a memory block of $min \leq size \leq max$.
- unsigned long [avail](#) ()
Get the amount of available memory.

15.48.1 Detailed Description

Standard list-based allocator.

Definition at line 31 of file [list_alloc](#).

15.48.2 Constructor & Destructor Documentation

15.48.2.1 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 56 of file [list_alloc](#).

15.48.3 Member Function Documentation

15.48.3.1 alloc()

```
void * cxx::List_alloc::alloc (
    unsigned long size,
    unsigned long align ) [inline]
```

Allocate a memory block.

Parameters

<i>size</i>	Size of the memory block.
<i>align</i>	Alignment constraint.

Returns

Pointer to memory block

Precondition

$0 < \text{size} \leq \sim 0\text{UL} - 32$.

Definition at line 363 of file [list_alloc](#).

15.48.3.2 alloc_max()

```
void * cxx::List_alloc::alloc_max (
    unsigned long min,
    unsigned long * max,
    unsigned long align,
    unsigned granularity ) [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).

Returns

Pointer to memory block

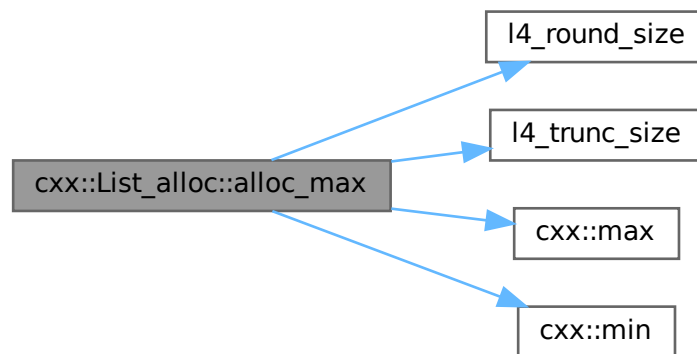
Precondition

$0 < \text{min} \leq \sim 0\text{UL} - 32$.
 $0 < \text{max}$.

Definition at line 266 of file [list_alloc](#).

References [l4_round_size\(\)](#), [l4_trunc_size\(\)](#), [cxx::max\(\)](#), and [cxx::min\(\)](#).

Here is the call graph for this function:

**15.48.3.3 avail()**

```
unsigned long cxx::List_alloc::avail ( ) [inline]
```

Get the amount of available memory.

Returns

Available memory in bytes

Definition at line 434 of file [list_alloc](#).

15.48.3.4 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 227 of file [list_alloc](#).

The documentation for this class was generated from the following file:

- `I4/cxx/list_alloc`

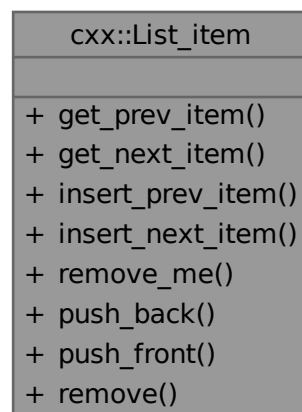
15.49 `cxx::List_item` Class Reference

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.

15.49.1 Detailed Description

Basic list item.

Basic item that can be member of a doubly linked, cyclic list.

Definition at line 37 of file [list](#).

15.49.2 Member Function Documentation

15.49.2.1 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:

**15.49.2.2 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.

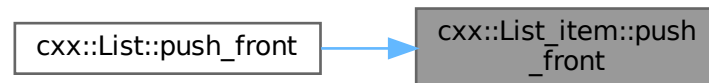
Returns

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:

**15.49.2.3 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:

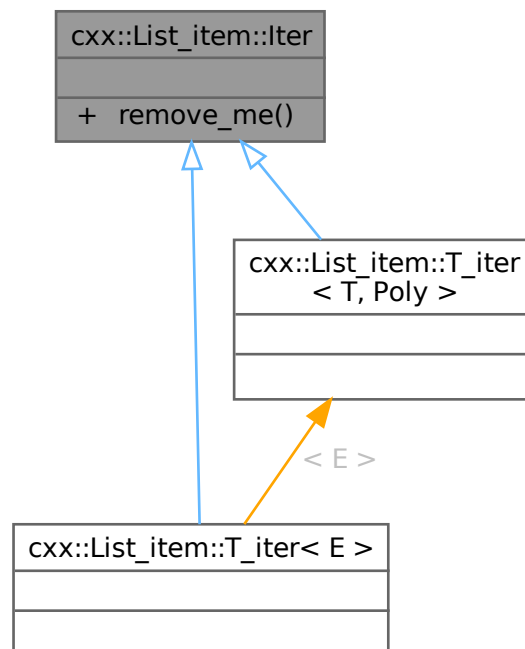
- l4/cxx/list

15.50 cxx::List_item::Iter Class Reference

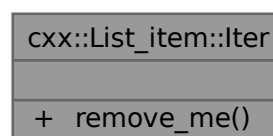
Iterator for a list of ListItem-s.

```
#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.

15.50.1 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:

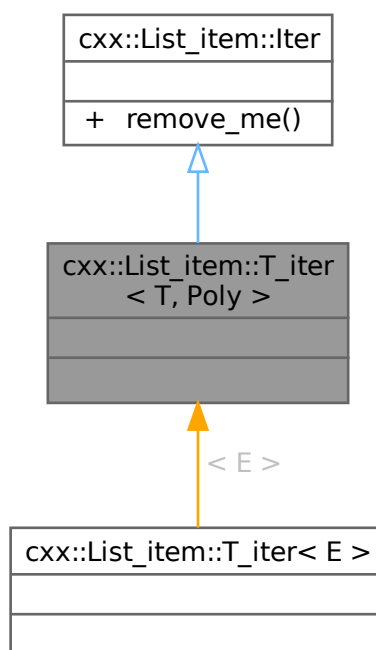
- l4/cxx/list

15.51 cxx::List_item::T_iter< T, Poly > Class Template Reference

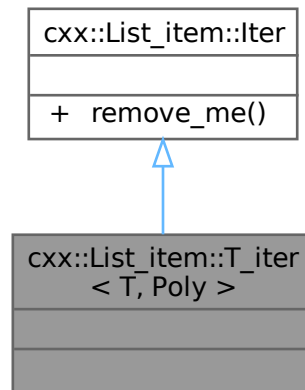
Iterator for derived classes from ListItem.

```
#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.

15.51.1 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:

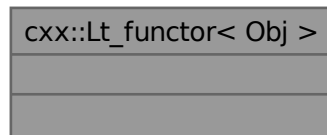
- `I4/cxx/list`

15.52 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 >:



15.52.1 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:

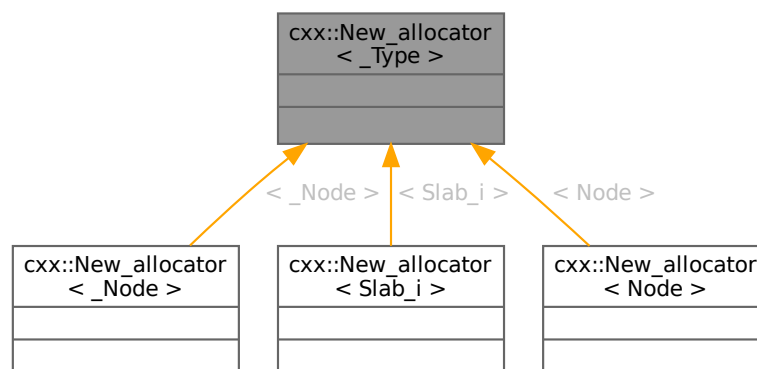
- I4/cxx/std_ops

15.53 cxx::New_allocator< _Type > Class Template Reference

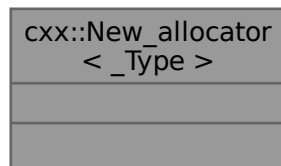
Standard allocator based on `operator new ()`.

```
#include <std_alloc>
```

Inheritance diagram for cxx::New_allocator< _Type >:



Collaboration diagram for `cxx::New_allocator<_Type>`:



15.53.1 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:

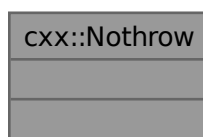
- `I4/cxx/std_alloc`

15.54 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`:



15.54.1 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:

- `l4/cxx/std_alloc`

15.55 `cxx::Pair< First, Second >` Struct Template Reference

`Pair` of two values.

```
#include <pair>
```

Collaboration diagram for `cxx::Pair< First, Second >`:

<code>cxx::Pair< First, Second ></code>	
+	<code>first</code>
+	<code>second</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.
- `Pair ()=default`
Default construction.

Data Fields

- First **first**
First value.
- Second **second**
Second value.

15.55.1 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 [36](#) of file [pair](#).

15.55.2 Constructor & Destructor Documentation

15.55.2.1 Pair()

```
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 [54](#) of file [pair](#).

The documentation for this struct was generated from the following file:

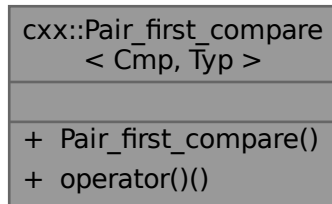
- [I4/cxx/pair](#)

15.56 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.

15.56.1 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 [75](#) of file [pair](#).

15.56.2 Constructor & Destructor Documentation

15.56.2.1 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 85 of file [pair](#).

15.56.3 Member Function Documentation

15.56.3.1 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 92 of file [pair](#).

The documentation for this class was generated from the following file:

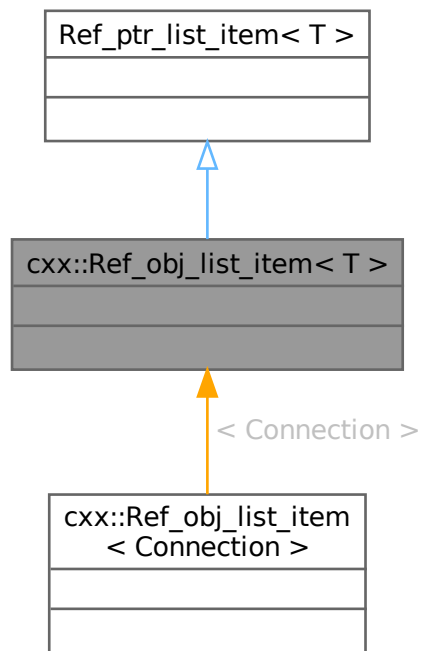
- [I4/cxx/pair](#)

15.57 cxx::Ref_obj_list_item< T > Struct Template Reference

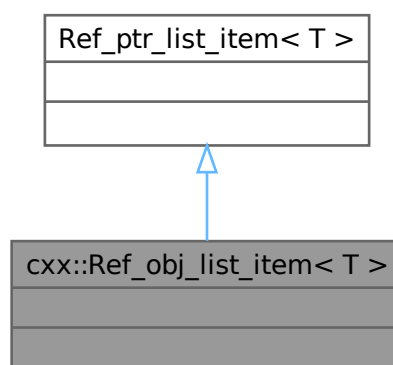
Item for list linked via [cxx::Ref_ptr](#) with default refence counting.

```
#include <ref_ptr_list>
```

Inheritance diagram for cxx::Ref_obj_list_item< T >:



Collaboration diagram for cxx::Ref_obj_list_item< T >:



15.57.1 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:

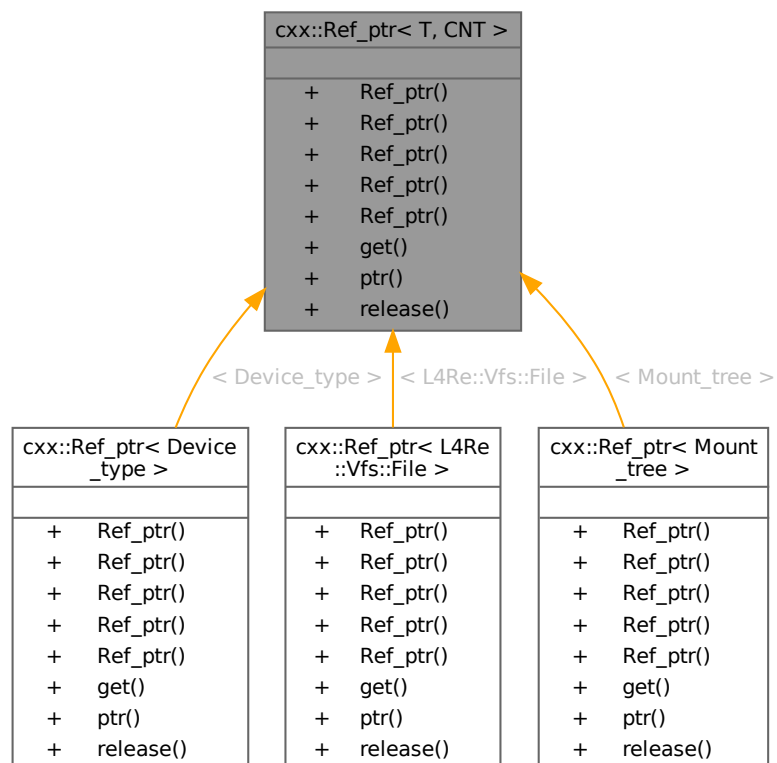
- [l4/cxx/ref_ptr_list](#)

15.58 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< T, CNT ></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.

15.58.1 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](#).

15.58.2 Constructor & Destructor Documentation

15.58.2.1 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 98 of file [ref_ptr](#).

15.58.2.2 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 111 of file [ref_ptr](#).

15.58.2.3 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 [124](#) of file [ref_ptr](#).

15.58.3 Member Function Documentation

15.58.3.1 `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 [131](#) of file [ref_ptr](#).

15.58.3.2 `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 [137](#) of file [ref_ptr](#).

15.58.3.3 `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 [148](#) of file [ref_ptr](#).

The documentation for this class was generated from the following file:

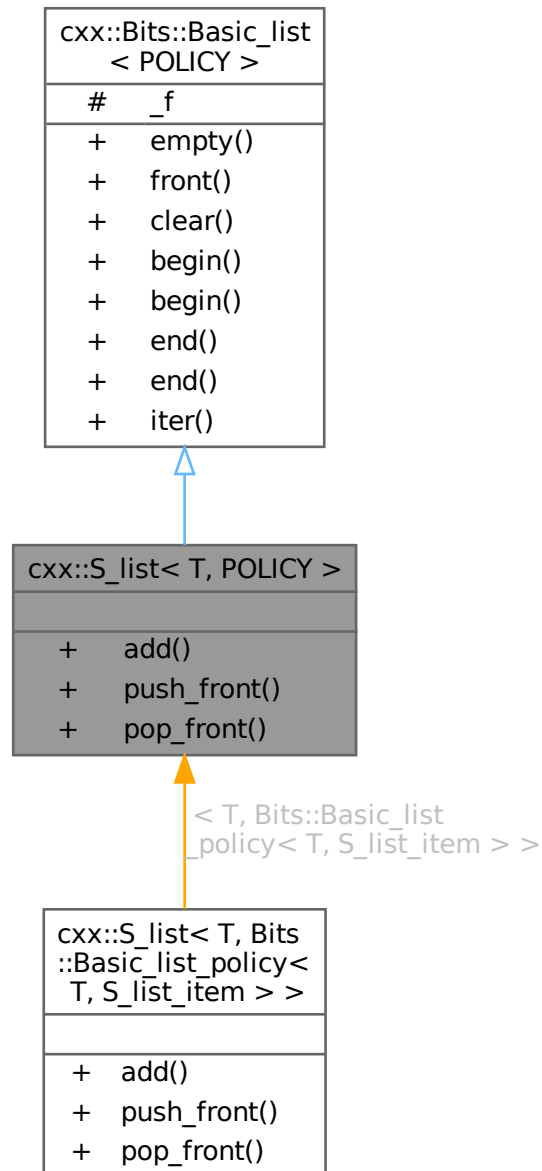
- `I4/cxx/ref_ptr`

15.59 cxx::S_list< T, POLICY > Class Template Reference

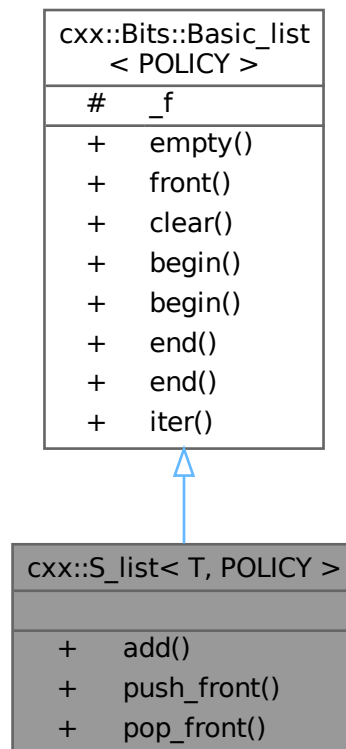
Simple single-linked list.

```
#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.

15.59.1 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

<code>T</code>	Type of elements saved in the list. Must inherit from <code>cxx::S_list_item</code>
----------------	---

Definition at line 50 of file `slist`.

15.59.2 Member Function Documentation

15.59.2.1 `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.

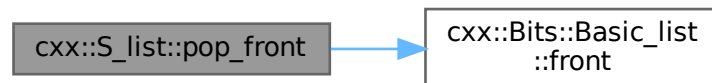
Precondition

The list must not be empty or the behaviour will be undefined.

Definition at line 99 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:

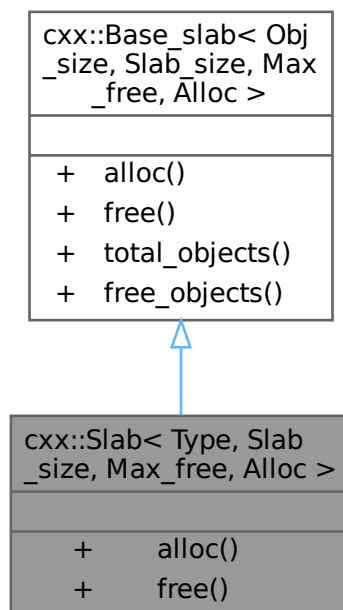
- `I4/cxx/slist`

15.60 `cxx::Slab< Type, Slab_size, Max_free, Alloc >` Class Template Reference

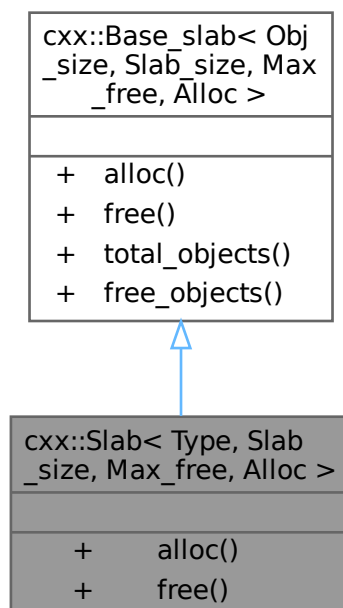
[Slab](#) allocator for object of type `Type`.

```
#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.

15.60.1 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`.

15.60.2 Member Function Documentation

15.60.2.1 `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](#).

15.60.2.2 `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](#).

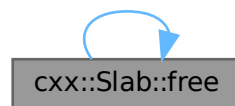
References [cxx::Slab](#)< `Type`, `Slab_size`, `Max_free`, `Alloc` >::free().

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:

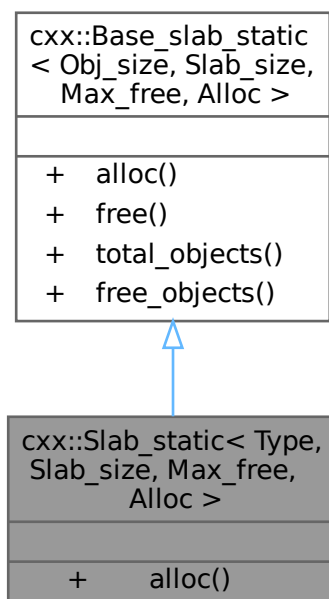
- `I4/cxx/slab_alloc`

15.61 `cxx::Slab_static< Type, Slab_size, Max_free, Alloc >` Class Template Reference

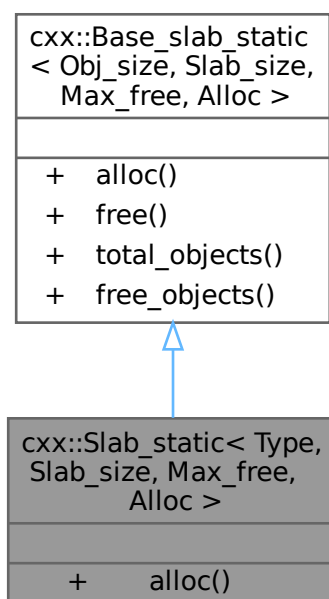
Merged slab allocator (allocators for objects of the same size are merged together).

```
#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` }

15.61.1 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`.

15.61.2 Member Function Documentation

15.61.2.1 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

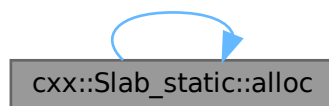
The object is not zeroed out by the slab allocator.

Definition at line [489](#) of file [slab_alloc](#).

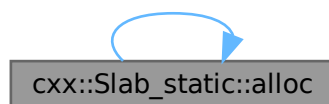
References [cxx::Slab_static](#)< Type, Slab_size, Max_free, Alloc >::alloc().

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:

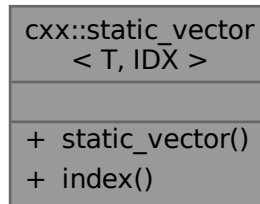
- [I4/cxx/slab_alloc](#)

15.62 `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.

15.62.1 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:

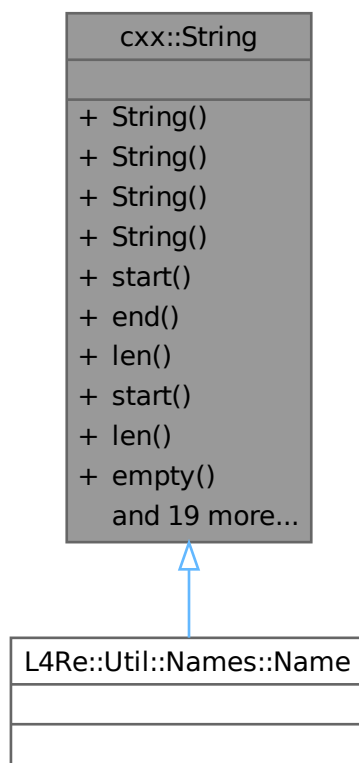
- `I4/cxx/static_vector`

15.63 cxx::String Class Reference

Allocation free string class with explicit length field.

```
#include <string>
```

Inheritance diagram for cxx::String:



Collaboration diagram for cxx::String:

cxx::String
<ul style="list-style-type: none"> + String() + String() + String() + String() + start() + end() + len() + start() + len() + empty() and 19 more...

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.

15.63.1 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](#).

15.63.2 Constructor & Destructor Documentation

15.63.2.1 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](#).

15.63.3 Member Function Documentation

15.63.3.1 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

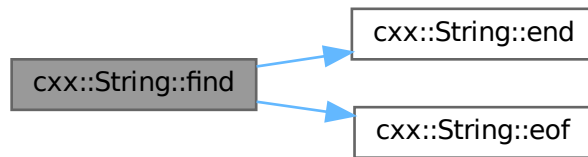
Return values

end()	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:



15.63.3.2 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
-----	---	-------------------

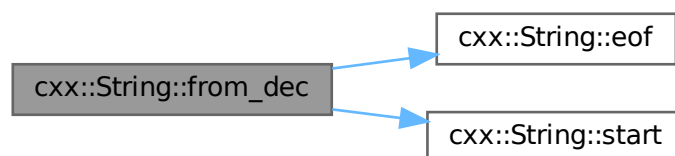
Returns

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:



15.63.3.3 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
-----	---	-------------------

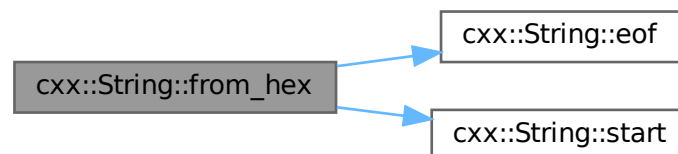
Return values

-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:

**15.63.3.4 starts_with()**

```
Index cxx::String::starts_with (
    cxx::String const & c ) const [inline]
```

Check if *c* is a prefix of string.

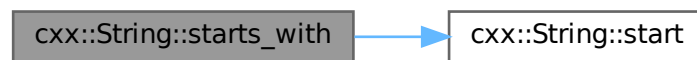
Returns

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:

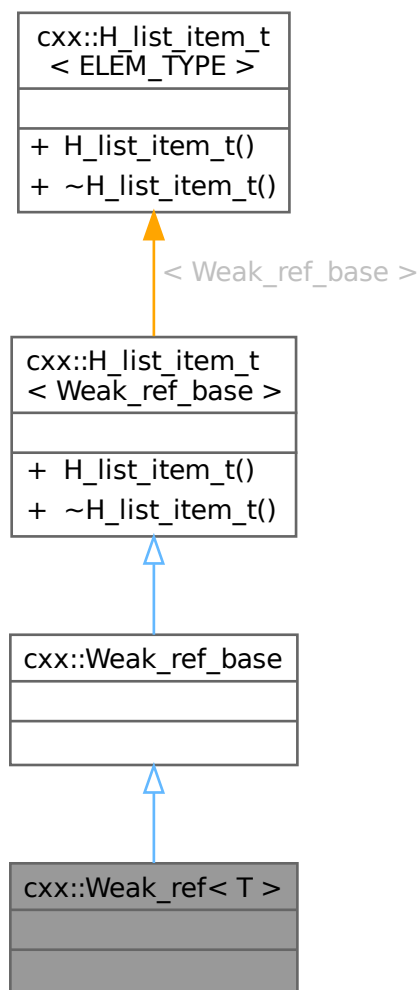
- `I4/cxx/string`

15.64 `cxx::Weak_ref< T >` Class Template Reference

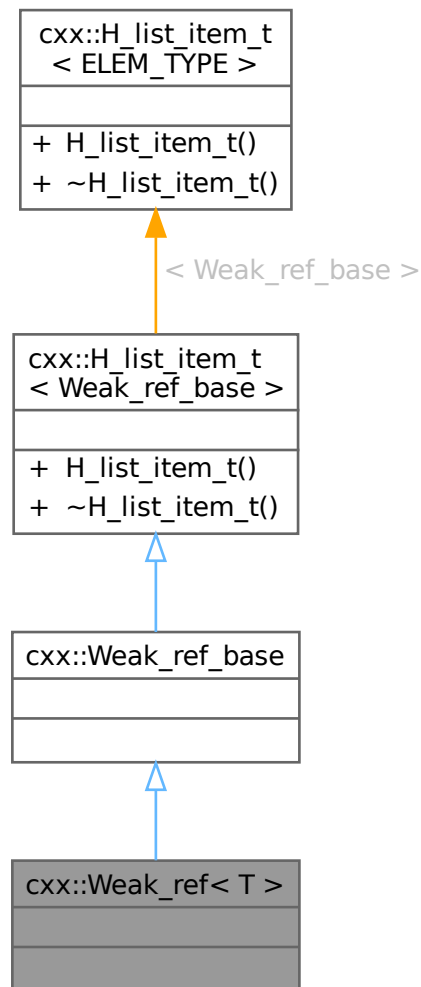
Typed weak reference to an object of type `T`.

```
#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.

15.64.1 Detailed Description

```
template<typename T>
class cxx::Weak_ref< T >
```

Typed weak reference to an object of type `T`.

Template Parameters

<i>T</i>	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 and all the weak references are iterated and reset by the `Weak_ref_base::List` destructor or `Weak_ref_base::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.

Definition at line 67 of file [weak_ref](#).

The documentation for this class was generated from the following file:

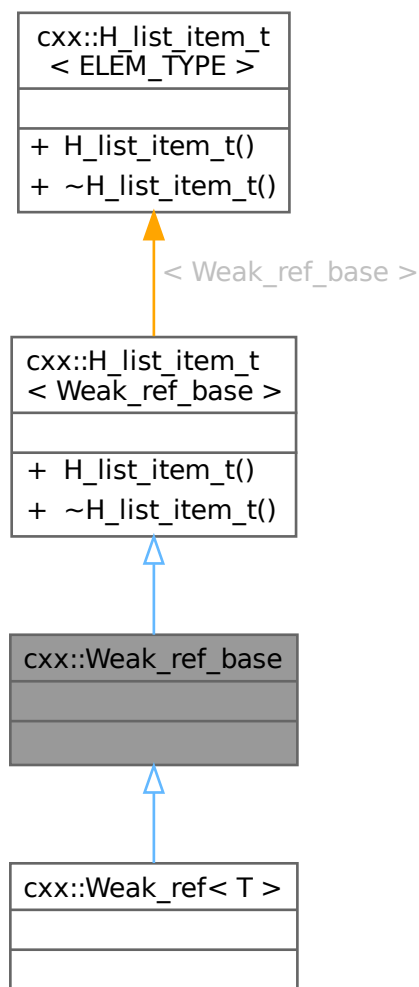
- `I4/cxx/weak_ref`

15.65 cxx::Weak_ref_base Class Reference

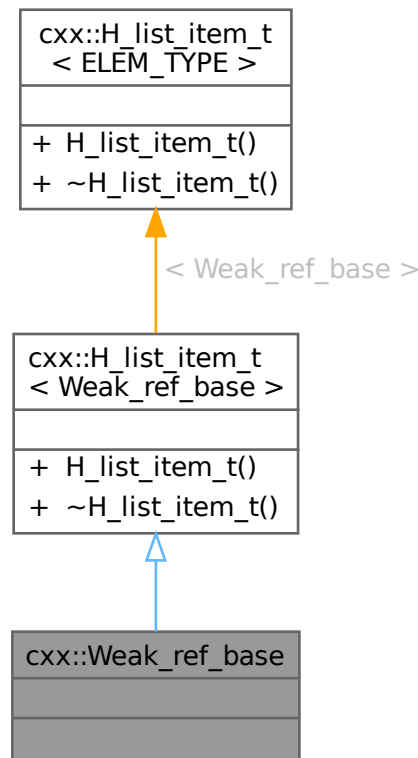
Generic (base) weak reference to some object.

```
#include <weak_ref>
```

Inheritance diagram for `cxx::Weak_ref_base`:



Collaboration diagram for cxx::Weak_ref_base:



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.

15.65.1 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:

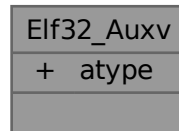
- I4/cxx/weak_ref

15.66 Elf32_Auxv Struct Reference

Auxiliary vector (32-bit).

```
#include <elf.h>
```

Collaboration diagram for Elf32_Auxv:



Data Fields

- [Elf32_Word atype](#)

15.66.1 Detailed Description

Auxiliary vector (32-bit).

Definition at line [958](#) of file [elf.h](#).

15.66.2 Field Documentation

15.66.2.1 atype

```
Elf32_Word Elf32_Auxv::atype
```

See also

[Elf_ATs](#)

Definition at line [960](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

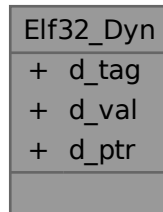
- [l4/util/elf.h](#)

15.67 Elf32_Dyn Struct Reference

ELF32 dynamic entry.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Dyn:



Data Fields

- [Elf32_Sword d_tag](#)

15.67.1 Detailed Description

ELF32 dynamic entry.

Definition at line [509](#) of file [elf.h](#).

15.67.2 Field Documentation

15.67.2.1 d_tag

```
Elf32_Sword Elf32_Dyn::d_tag
```

See also

[Elf_DTs](#)

Definition at line [511](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.68 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_Els
- [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

15.68.1 Detailed Description

ELF32 header.

Definition at line 121 of file [elf.h](#).

15.68.2 Field Documentation

15.68.2.1 e_flags

[Elf32_Word](#) Elf32_Ehdr::e_flags

processor-specific flags

See also

[Elf_EF_ARM_s](#)

Definition at line 130 of file [elf.h](#).

15.68.2.2 e_machine

[Elf32_Half](#) Elf32_Ehdr::e_machine

required architecture

See also

[Elf_EMs](#)

Definition at line 125 of file [elf.h](#).

15.68.2.3 e_type

`Elf32_Half Elf32_Ehdr::e_type`

type of ELF file

See also

[Elf_ETs](#)

Definition at line 124 of file [elf.h](#).

15.68.2.4 e_version

`Elf32_Word Elf32_Ehdr::e_version`

file version

See also

[Elf_EVs](#)

Definition at line 126 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.69 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

15.69.1 Detailed Description

ELF32 program header.

Definition at line 420 of file [elf.h](#).

15.69.2 Field Documentation

15.69.2.1 p_flags

[Elf32_Word](#) `Elf32_Phdr::p_flags`

flags

See also

[Elf_PFs](#)

Definition at line 428 of file [elf.h](#).

15.69.2.2 p_type

[Elf32_Word](#) `Elf32_Phdr::p_type`

type of program section

See also

[Elf_PTs](#)

Definition at line 422 of file [elf.h](#).

The documentation for this struct was generated from the following file:

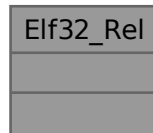
- [l4/util/elf.h](#)

15.70 Elf32_Rel Struct Reference

ELF32 relocation entry w/o addend.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Rel:



15.70.1 Detailed Description

ELF32 relocation entry w/o addend.

Definition at line [627](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

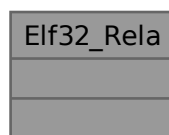
- [l4/util/elf.h](#)

15.71 Elf32_Rela Struct Reference

ELF32 relocation entry w/ addend.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Rela:



15.71.1 Detailed Description

ELF32 relocation entry w/ addend.

Definition at line 634 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.72 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

15.72.1 Detailed Description

ELF32 section header.

Definition at line 342 of file [elf.h](#).

15.72.2 Field Documentation

15.72.2.1 sh_flags

[Elf32_Word](#) `Elf32_Shdr::sh_flags`

section's flags

See also

[Elf_SHFs](#)

Definition at line 346 of file [elf.h](#).

15.72.2.2 sh_type

[Elf32_Word](#) `Elf32_Shdr::sh_type`

section's type

See also

[Elf_SHTs](#)

Definition at line 345 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.73 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

15.73.1 Detailed Description

ELF32 symbol table entry.

Definition at line 867 of file [elf.h](#).

The documentation for this struct was generated from the following file:

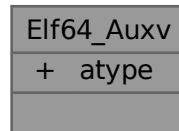
- [l4/util/elf.h](#)

15.74 Elf64_Auxv Struct Reference

Auxiliary vector (64-bit).

```
#include <elf.h>
```

Collaboration diagram for Elf64_Auxv:



Data Fields

- [Elf64_Word atype](#)

15.74.1 Detailed Description

Auxiliary vector (64-bit).

Definition at line [965](#) of file [elf.h](#).

15.74.2 Field Documentation

15.74.2.1 atype

```
Elf64_Word Elf64_Auxv::atype
```

See also

[Elf_ATs](#)

Definition at line [967](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

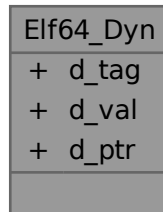
- [l4/util/elf.h](#)

15.75 Elf64_Dyn Struct Reference

ELF64 dynamic entry.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Dyn:



Data Fields

- [Elf64_Sxword d_tag](#)

15.75.1 Detailed Description

ELF64 dynamic entry.

Definition at line [520](#) of file [elf.h](#).

15.75.2 Field Documentation

15.75.2.1 d_tag

[Elf64_Sxword](#) Elf64_Dyn::d_tag

See also

[Elf_DTs](#)

Definition at line [522](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.76 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_Els
- [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

15.76.1 Detailed Description

Elf64 header.

Definition at line 142 of file [elf.h](#).

15.76.2 Field Documentation

15.76.2.1 e_flags

[Elf64_Word](#) Elf64_Ehdr::e_flags

processor-specific flags

See also

[Elf_EF_ARM_s](#)

Definition at line 151 of file [elf.h](#).

15.76.2.2 e_machine

[Elf64_Half](#) Elf64_Ehdr::e_machine

required architecture

See also

[Elf_EMs](#)

Definition at line 146 of file [elf.h](#).

15.76.2.3 e_type

`Elf64_Half Elf64_Ehdr::e_type`

type of ELF file

See also

[Elf_ETs](#)

Definition at line 145 of file [elf.h](#).

15.76.2.4 e_version

`Elf64_Word Elf64_Ehdr::e_version`

file version

See also

[Elf_EVs](#)

Definition at line 147 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.77 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

15.77.1 Detailed Description

ELF64 program header.

Definition at line 433 of file [elf.h](#).

15.77.2 Field Documentation

15.77.2.1 p_flags

[Elf64_Word](#) Elf64_Phdr::p_flags

flags

See also

[Elf_PFs](#)

Definition at line 436 of file [elf.h](#).

15.77.2.2 p_type

[Elf64_Word](#) Elf64_Phdr::p_type

type of program section

See also

[Elf_PTs](#)

Definition at line 435 of file [elf.h](#).

The documentation for this struct was generated from the following file:

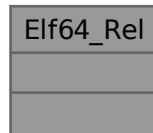
- [l4/util/elf.h](#)

15.78 Elf64_Rel Struct Reference

ELF64 relocation entry w/o addend.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Rel:



15.78.1 Detailed Description

ELF64 relocation entry w/o addend.

Definition at line [642](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

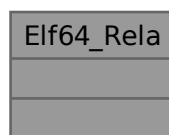
- [l4/util/elf.h](#)

15.79 Elf64_Rela Struct Reference

ELF64 relocation entry w/ addend.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Rela:



15.79.1 Detailed Description

ELF64 relocation entry w/ addend.

Definition at line 649 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.80 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

15.80.1 Detailed Description

ELF64 section header.

Definition at line 357 of file [elf.h](#).

15.80.2 Field Documentation

15.80.2.1 sh_flags

[Elf64_Xword](#) [Elf64_Shdr::sh_flags](#)

section's flags

See also

[Elf_SHFs](#)

Definition at line 361 of file [elf.h](#).

15.80.2.2 sh_type

[Elf64_Word](#) [Elf64_Shdr::sh_type](#)

section's type

See also

[Elf_SHTs](#)

Definition at line 360 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.81 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

15.81.1 Detailed Description

ELF64 symbol table entry.

Definition at line 878 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.82 gfxbitmap_offset Struct Reference

offsets in pmap[] and bmap[]

```
#include <bitmap.h>
```

Collaboration diagram for gfxbitmap_offset:

gfxbitmap_offset
+ preskip_x
+ preskip_y
+ endskip_x

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

15.82.1 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:

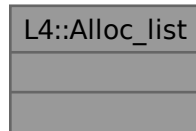
- [l4/libgfxbitmap/bitmap.h](#)

15.83 L4::Alloc_list Class Reference

A simple list-based allocator.

```
#include <alloc.h>
```

Collaboration diagram for L4::Alloc_list:



15.83.1 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](#)

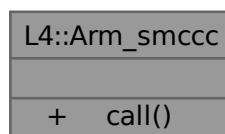
15.84 L4::Arm_smccc Class Reference

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.

15.84.1 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`.

15.84.2 Member Function Documentation

15.84.2.1 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"> • 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. • Bit 30 defines the calling convention: • Bit 30 == 1: 64-bit calling convention. • Bit 30 == 0: 32-bit calling convention. • Bits 24..29 determine the service call ID. Only service IDs >= 0x30000000 (<i>Trusted Application Calls</i> and <i>Trusted OS Calls</i>) are allowed.
in	<i>in0</i>	First input parameter.
in	<i>in1</i>	Second input parameter.

Parameters

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:

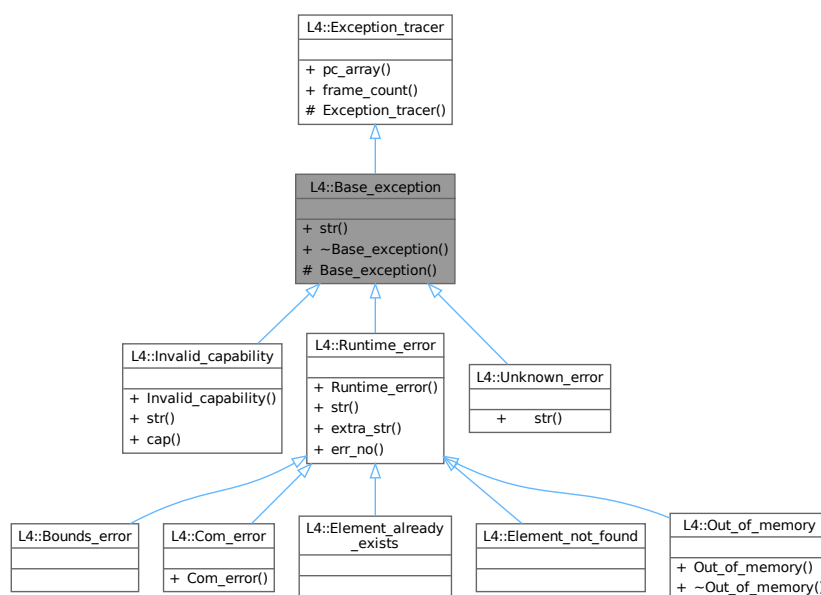
- l4/sys/arm_smccc

15.85 L4::Base_exception Class Reference

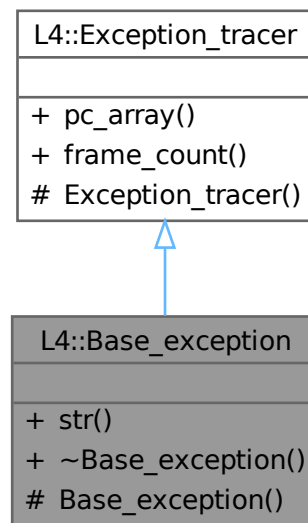
Base class for all exceptions, thrown by the L4Re framework.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Base_exception:



Collaboration diagram for L4::Base_exception:



Public Member Functions

- virtual char const * **str** () const throw () =0
Return a human readable string for the exception.
- virtual ~**Base_exception** () throw ()
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.

15.85.1 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:

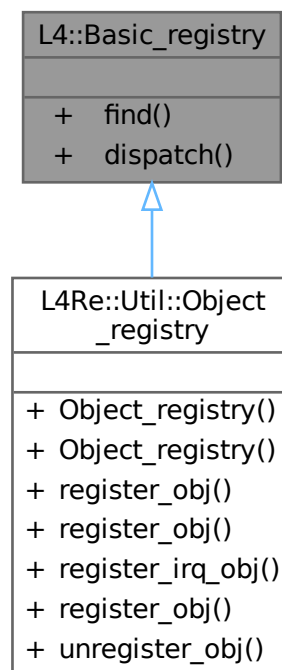
- [l4/cxx/exceptions](#)

15.86 L4::Basic_registry Class Reference

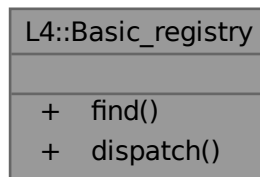
This registry returns the corresponding server object based on the label of an [lpc_gate](#).

```
#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.

15.86.1 Detailed Description

This registry returns the corresponding server object based on the label of an [lpc_gate](#).

Definition at line [539](#) of file [ipc_epiface](#).

15.86.2 Member Function Documentation

15.86.2.1 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.

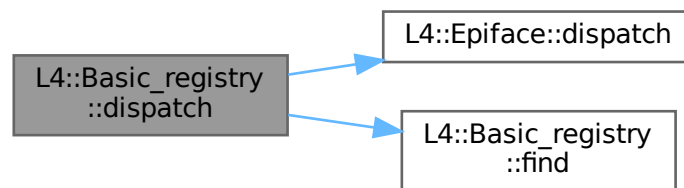
Returns

The return code from the object's dispatch function or -L4_ENOENT if the object does not exist.

Definition at line 564 of file [ipc_epiface](#).

References [L4::Epiface::dispatch\(\)](#), and [find\(\)](#).

Here is the call graph for this function:

**15.86.2.2 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 lpc_gate .
--------------	---

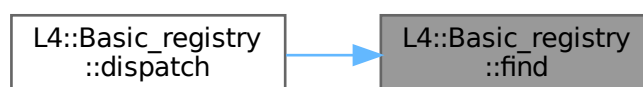
Returns

A pointer to the [Epiface](#) identified by the given label.

Definition at line 548 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:

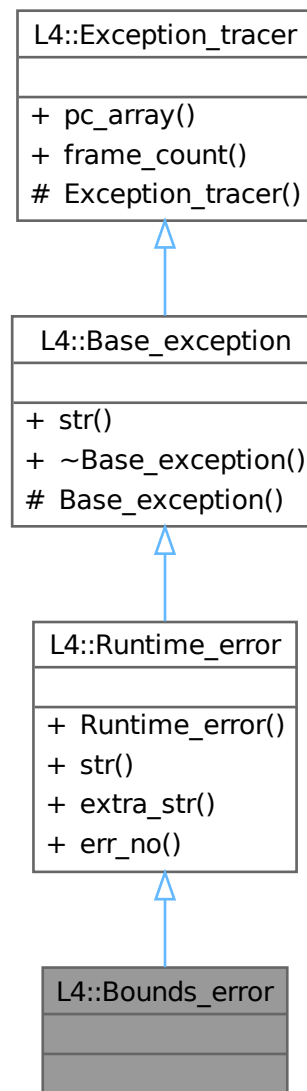
- l4/sys/cxx/ipc_epiface

15.87 L4::Bounds_error Class Reference

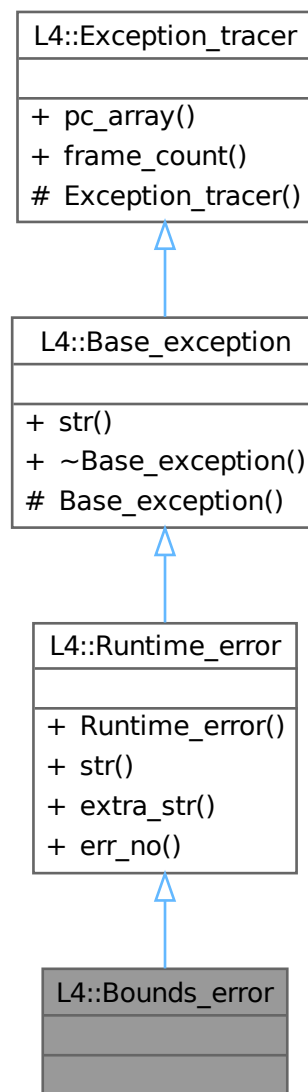
Access out of bounds.

```
#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) throw ()
Create a new [Runtime_error](#).
- char const * [str](#) () const override throw ()
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 () throw ()`

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.

15.87.1 Detailed Description

Access out of bounds.

Definition at line 289 of file [exceptions](#).

The documentation for this class was generated from the following file:

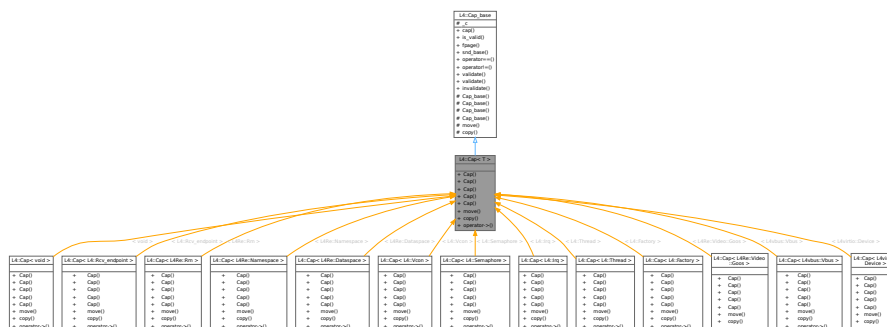
- [l4/cxx/exceptions](#)

15.88 L4::Cap< T > Class Template Reference

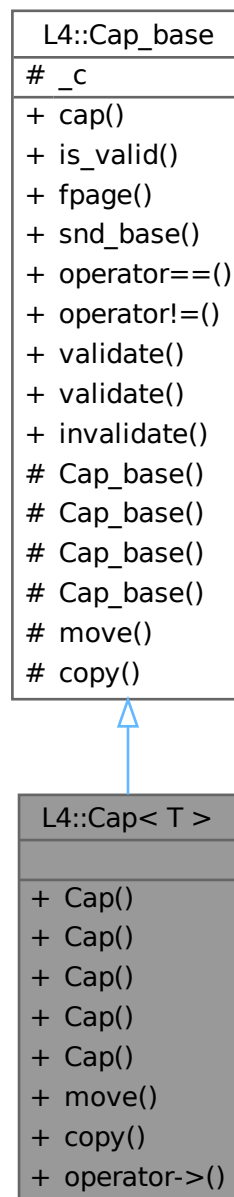
C++ interface for capabilities.

```
#include <capability.h>
```

Inheritance diagram for `L4::Cap< T >`:



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).

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.

15.88.1 Detailed Description

```
template<typename T>
class L4::Cap< T >
```

C++ interface for capabilities.

Template Parameters

<i>T</i>	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.cc](#)

Definition at line 221 of file [capability.h](#).

15.88.2 Constructor & Destructor Documentation

15.88.2.1 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 247 of file [capability.h](#).

15.88.2.2 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 254 of file [capability.h](#).

15.88.2.3 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

<code>cap</code>	Capability selector.
------------------	----------------------

Definition at line 260 of file [capability.h](#).

15.88.2.4 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 266 of file [capability.h](#).

15.88.3 Member Function Documentation

15.88.3.1 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 289 of file [capability.h](#).

15.88.3.2 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 279 of file [capability.h](#).

The documentation for this class was generated from the following file:

- [l4/sys/cxx/capability.h](#)

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.

15.89.1 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](#).

15.89.2 Member Enumeration Documentation

15.89.2.1 Cap_type

enum [L4::Cap_base::Cap_type](#)

Invalid capability type.

Enumerator

Invalid	Invalid capability selector.
---------	------------------------------

Definition at line [43](#) of file [capability.h](#).

15.89.2.2 No_init_type

enum [L4::Cap_base::No_init_type](#)

Special value for uninitialized capability objects.

Enumerator

No_init	Special value for constructing uninitialized Cap objects.
---------	---

Definition at line [32](#) of file [capability.h](#).

15.89.3 Constructor & Destructor Documentation

15.89.3.1 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 [147](#) of file [capability.h](#).

15.89.3.2 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 158 of file [capability.h](#).

15.89.4 Member Function Documentation

15.89.4.1 cap()

```
l4_cap_idx_t L4::Cap_base::cap ( ) const [inline], [noexcept]
```

Return capability selector.

Returns

Capability selector.

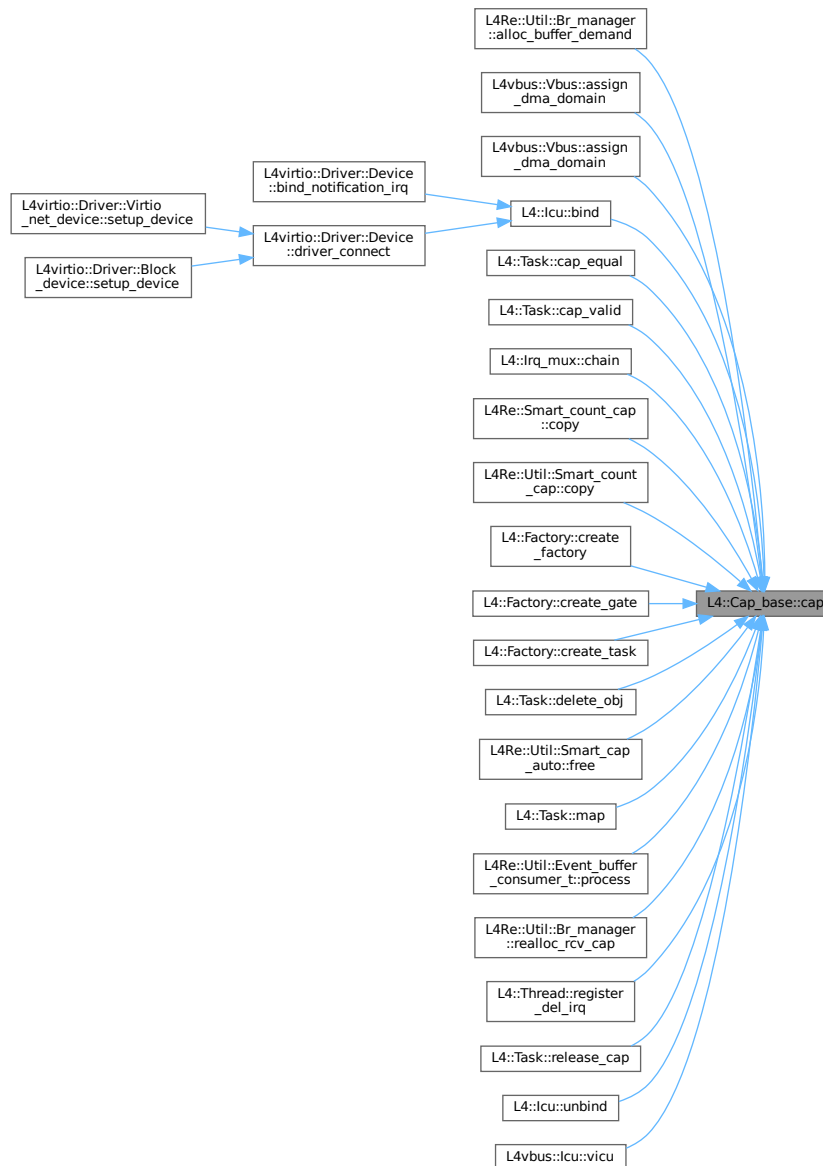
Examples

[examples/libs/l4re/streammap/client.cc](#).

Definition at line 52 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:



15.89.4.2 copy()

```
void L4::Cap_base::copy (
    Cap_base const & src ) const [inline], [protected]
```

Copy a capability.

Parameters

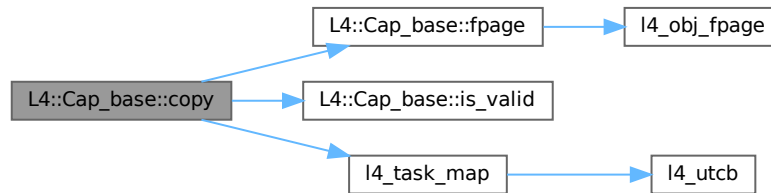
src	the source capability.
-----	------------------------

After this operation this capability refers to the same object as `src`.

Definition at line 190 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:



15.89.4.3 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

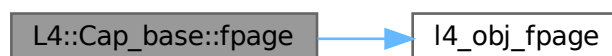
flex-page

Definition at line 72 of file [capability.h](#).

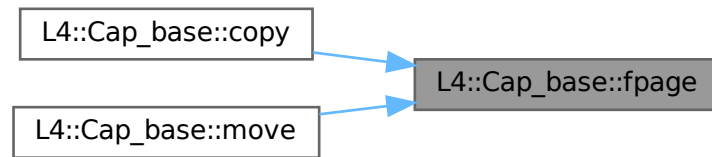
References [l4_obj_fpage\(\)](#).

Referenced by [copy\(\)](#), and [move\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.89.4.4 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

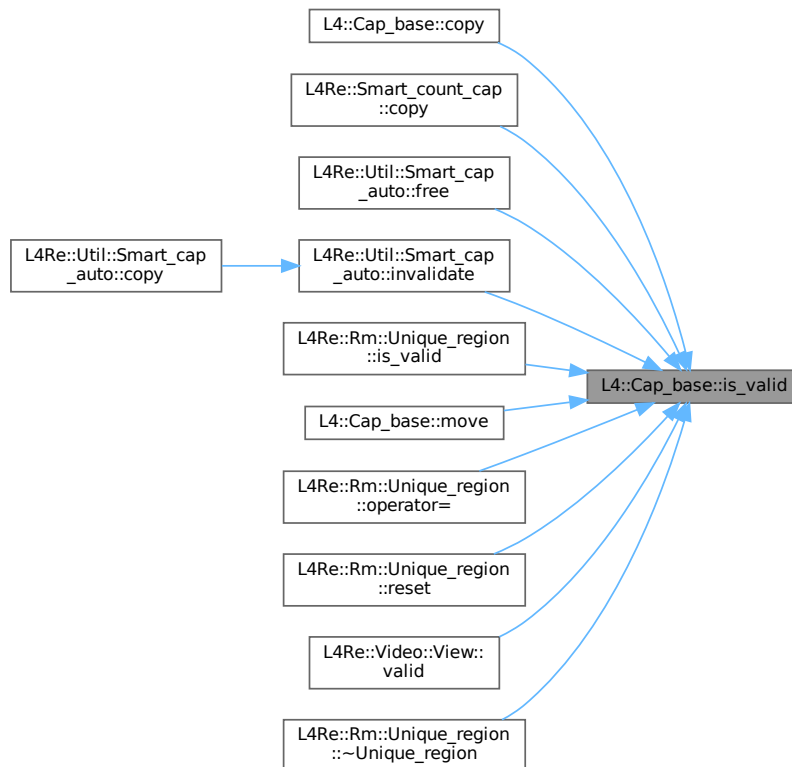
Examples

[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 60 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:



15.89.4.5 move()

```
void L4::Cap_base::move (
    Cap_base const & src ) const [inline], [protected]
```

Replace this capability with the contents of `src`.

Parameters

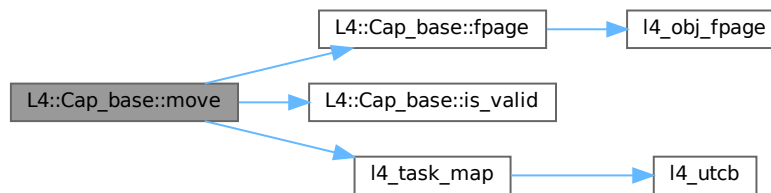
<code>src</code>	the source capability.
------------------	------------------------

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 174 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:



15.89.4.6 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: L4_MAP_ITEM_MAP , L4_MAP_ITEM_GRANT .
<i>base</i>	Base capability (first in a bundle of aligned capabilities)

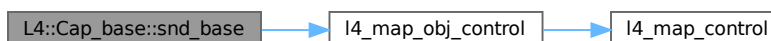
Returns

Map object.

Definition at line 84 of file [capability.h](#).

References [L4_INVALID_CAP](#), and [l4_map_obj_control\(\)](#).

Here is the call graph for this function:



15.89.4.7 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>	Task to check the capability in.
<i>u</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb .

Return values

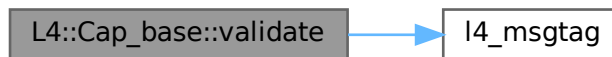
<i>l4_msgtag_t::label()</i> > 0	Capability is present (refers to an object).
<i>l4_msgtag_t::label()</i> == 0	No capability present (void object or invalid capability slot).

A capability is considered present when it refers to an existing kernel object.

Definition at line 83 of file [capability](#).

References [l4_msgtag\(\)](#).

Here is the call graph for this function:



15.89.4.8 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 l4_utcb .
----------	--

Return values

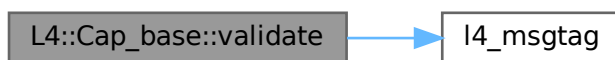
<i>l4_msgtag_t::label()</i> > 0	Capability is present (refers to an object).
<i>l4_msgtag_t::label()</i> == 0	No capability present (void object or invalid capability slot).

A capability is considered present when it refers to an existing kernel object.

Definition at line 90 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:

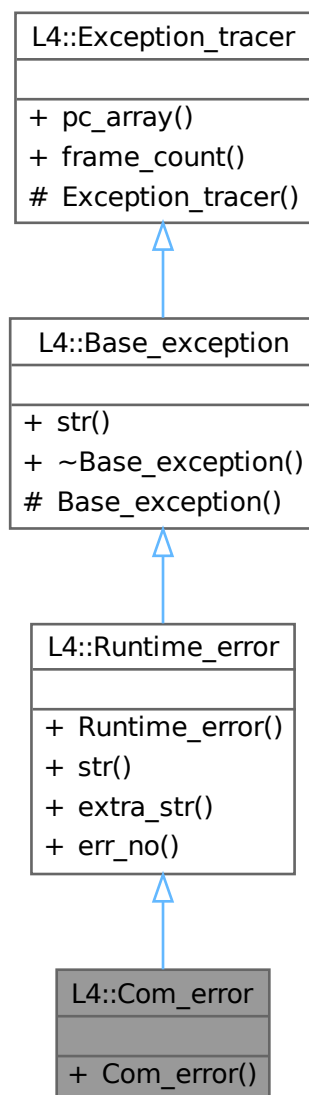
- `l4/sys/cxx/capability.h`
- `l4/sys/capability`

15.90 L4::Com_error Class Reference

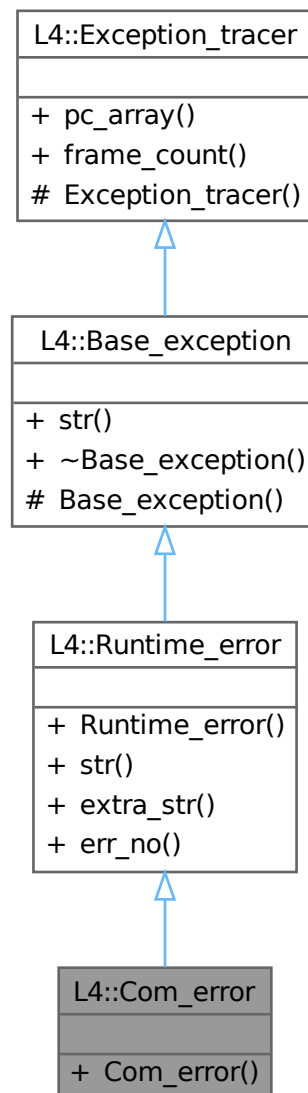
Error conditions during IPC.

```
#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) throw ()
Create a new [Runtime_error](#).
- char const * [str](#) () const override throw ()

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** () throw ()

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.

15.90.1 Detailed Description

Error conditions during IPC.

This exception encapsulates all IPC error conditions of [L4](#) IPC.

Definition at line [274](#) of file [exceptions](#).

15.90.2 Constructor & Destructor Documentation

15.90.2.1 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 L4 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:

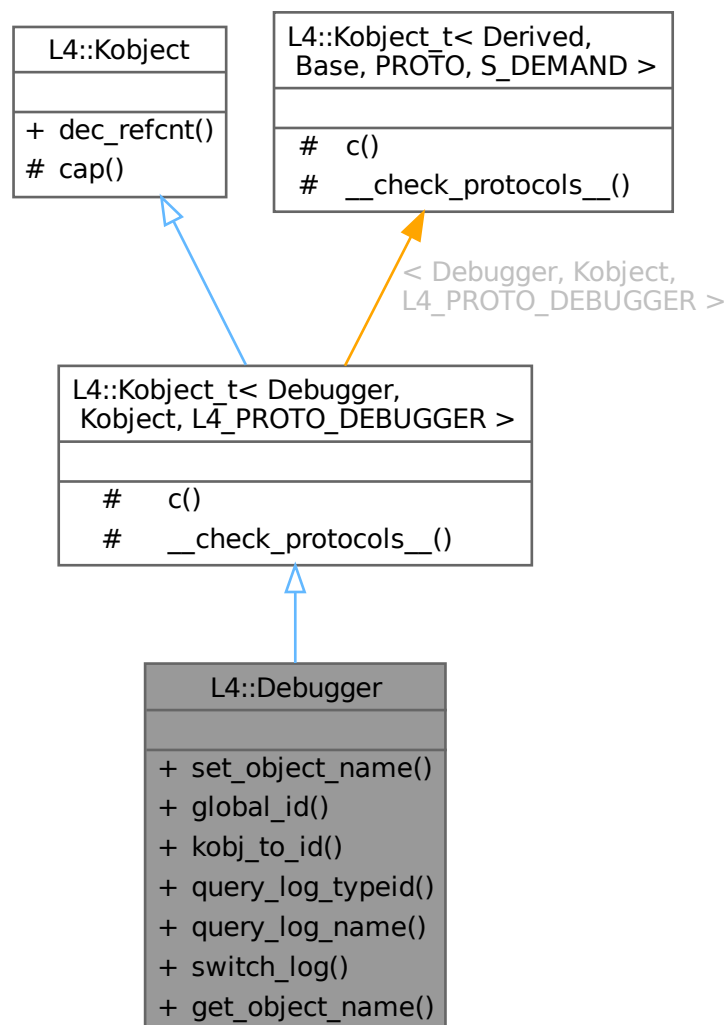
- [l4/cxx/exceptions](#)

15.91 L4::Debugger Class Reference

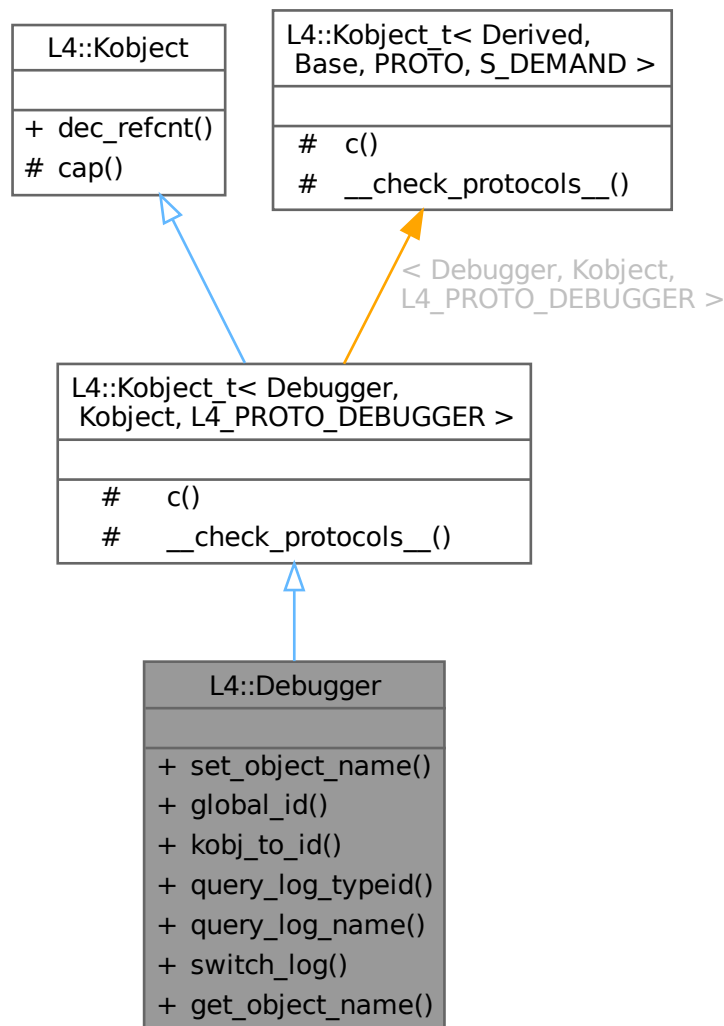
C++ kernel debugger API.

```
#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 id.

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](#) > **__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](#) < [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.

15.91.1 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](#).

15.91.2 Member Function Documentation

15.91.2.1 `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 l4_utcb .

Returns

Syscall return tag

Definition at line 159 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.91.2.2 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

utcb	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb .
----------------------	--

Return values

$\sim 0UL$	The capability is invalid.
≥ 0	The global debugger id.

Definition at line 82 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.91.2.3 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>	Kobject pointer
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb .

Return values

$\sim 0UL$	The capability or the Kobject pointer are invalid.
≥ 0	The globally unique id.

Definition at line 94 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.91.2.4 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 shortname to. The buffer must be allocated by the caller.
	<i>shortnamelen</i>	Buffer length of shortname.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb .

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:



15.91.2.5 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 l4_utcb .

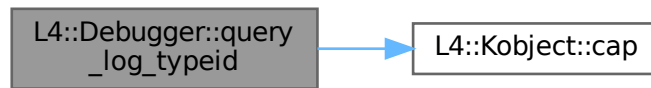
Return values

>=0	Id
<0	Error

Definition at line 108 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.91.2.6 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 l4_utcb .

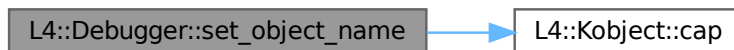
Returns

System call return tag.

Definition at line 70 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.91.2.7 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 l4_utcb .

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:

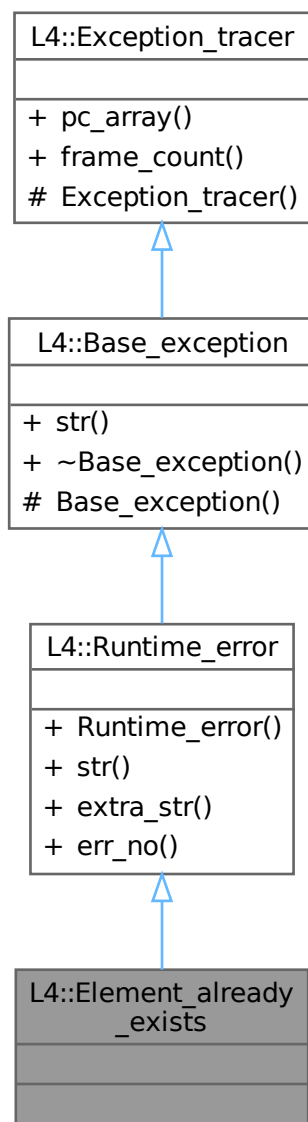
- [l4/sys/debugger](#)

15.92 L4::Element_already_exists Class Reference

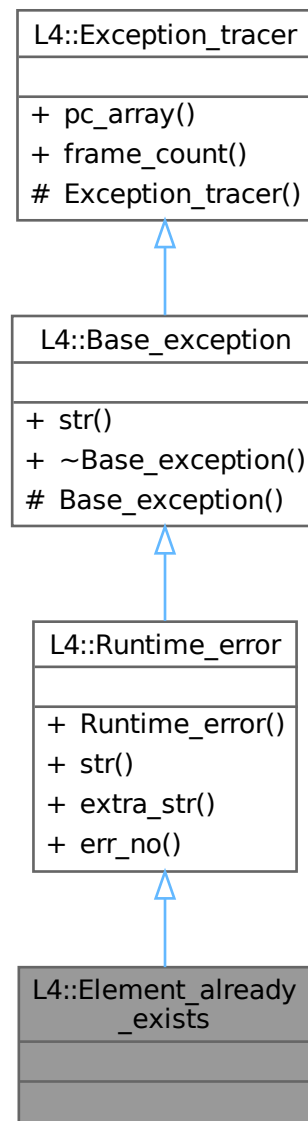
[Exception](#) for duplicate element insertions.

```
#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) throw ()
Create a new [Runtime_error](#).
- char const * [str](#) () const override throw ()
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** () throw ()
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.

15.92.1 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:

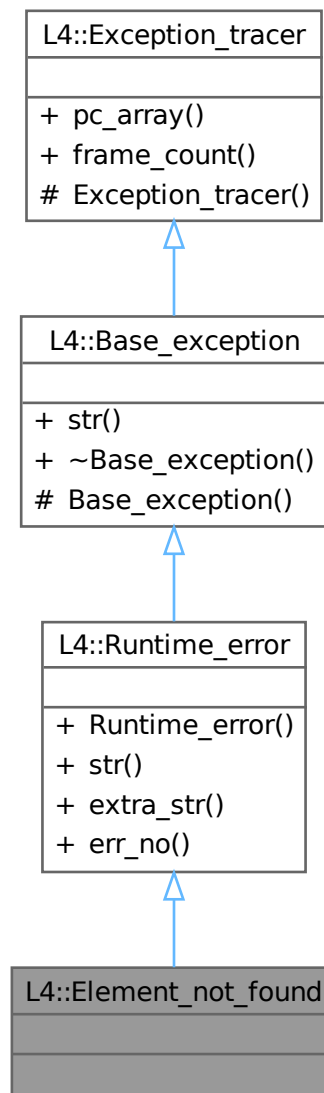
- l4/cxx/[exceptions](#)

15.93 L4::Element_not_found Class Reference

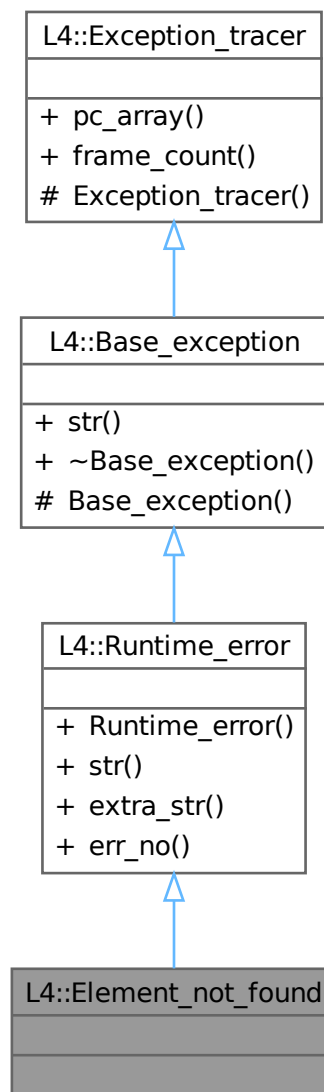
[Exception](#) for a failed lookup (element not found).

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Element_not_found:



Collaboration diagram for L4::Element_not_found:

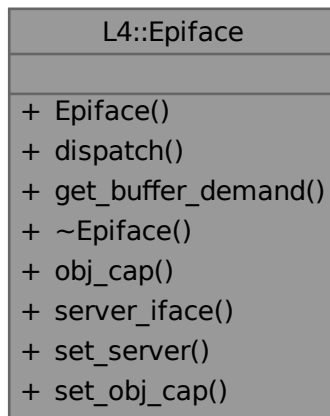


Additional Inherited Members

Public Member Functions inherited from [L4::Runtime_error](#)

- [Runtime_error](#) (long [err_no](#), char const *extra=0) throw ()
Create a new [Runtime_error](#).
- char const * [str](#) () const override throw ()
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::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.

15.94.1 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](#).

15.94.2 Member Function Documentation

15.94.2.1 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

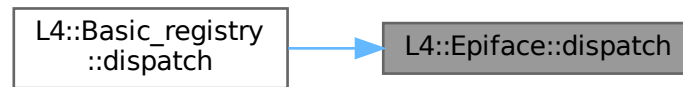
<code>-L4_ENOREPLY</code>	No reply message is send.
<code><0</code>	Error, reply with error code.
<code>>=0</code>	Success, reply with return value.

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< Irq_object >, and L4::lrqep_t< Derived, BASE, bool >.](#)

Referenced by [L4::Basic_registry::dispatch\(\)](#).

Here is the caller graph for this function:



15.94.2.2 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< L4virtio::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 >](#).

15.94.2.3 obj_cap()

```
Stored_cap L4::Epiface::obj_cap ( ) const [inline]
```

Get the capability to the kernel object belonging to this object.

Returns

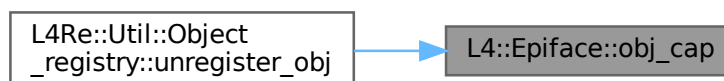
Capability for the kernel object behind the server.

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:



15.94.2.4 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](#).

15.94.2.5 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:

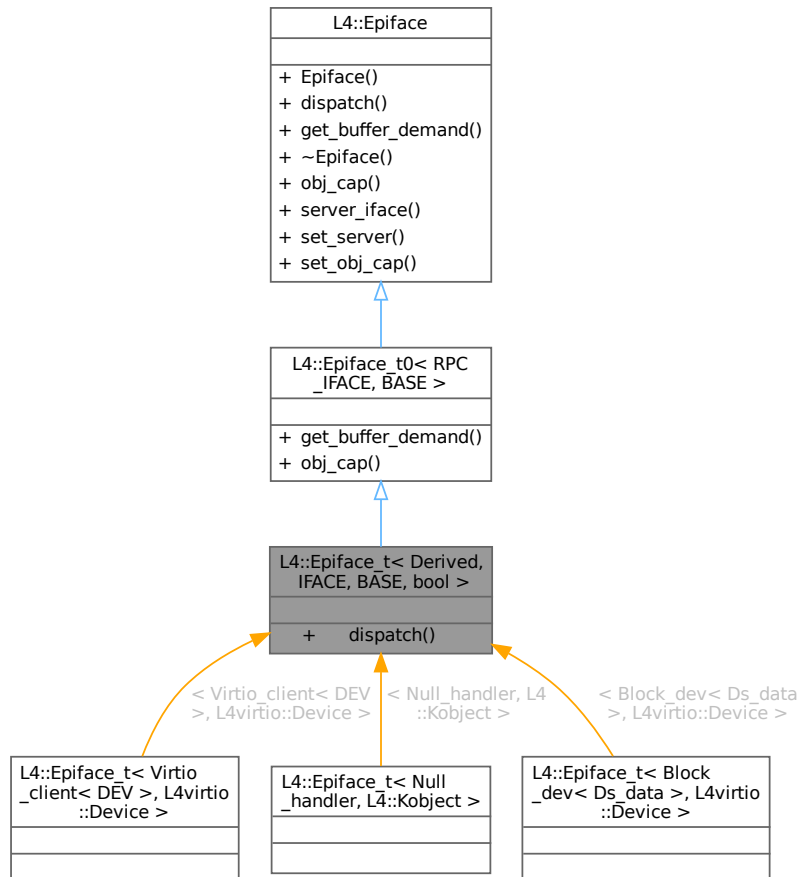
- [l4/sys/cxx/ipc_epiface](#)

15.95 L4::Epiface_t< Derived, IFACE, BASE, bool > Struct Template Reference

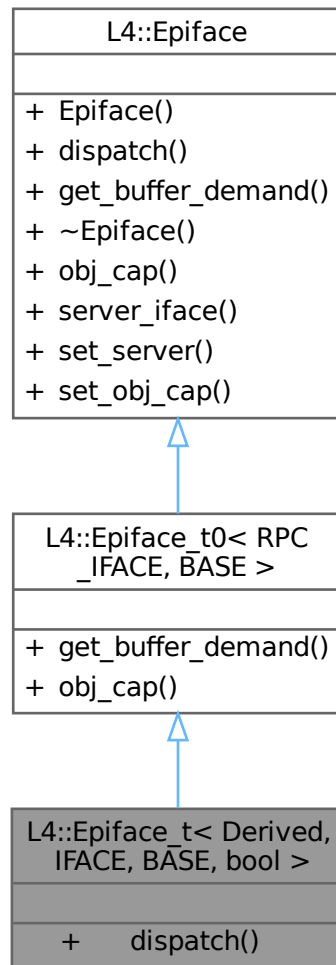
[Epiface](#) implementation for Kobject-based interface implementations.

```
#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.

15.95.1 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

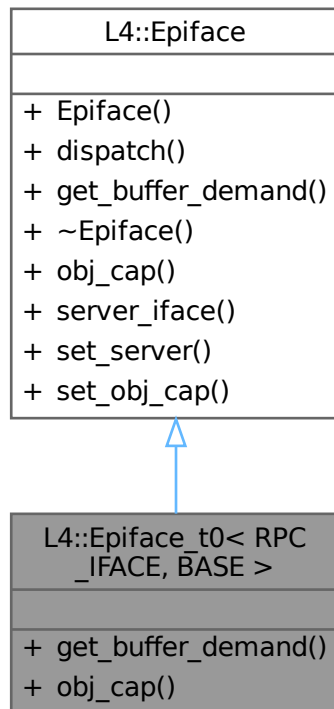
<i>Derived</i>	Class providing the interface implementations.
<i>BASE</i>	Epiface base class.

Examples

[examples/clntsrv/server.cc](#).

Definition at line 513 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.

15.96.1 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 Epiface class.

Definition at line 267 of file [ipc_epiface](#).

15.96.2 Member Function Documentation

15.96.2.1 [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](#)

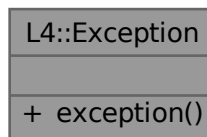
15.97 L4::Exception Class Reference

[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.

15.97.1 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](#).

15.97.2 Member Function Documentation

15.97.2.1 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:

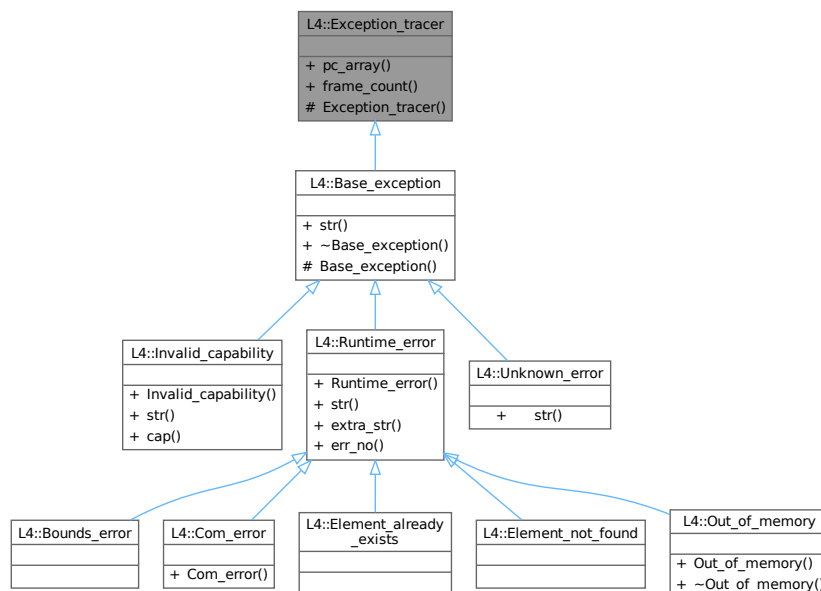
- [l4/sys/exception](#)

15.98 L4::Exception_tracer Class Reference

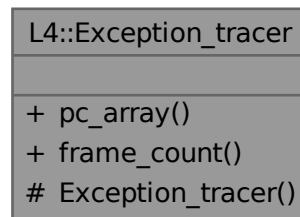
Back-trace support for exceptions.

```
#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.

15.98.1 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:

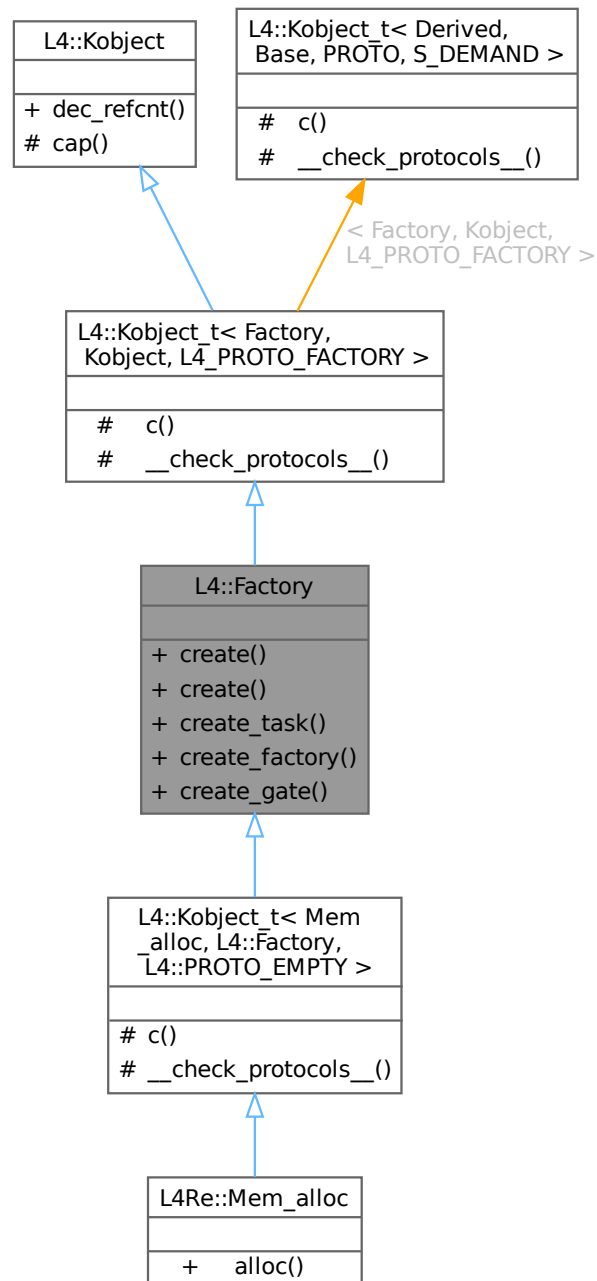
- [l4/cxx/exceptions](#)

15.99 L4::Factory Class Reference

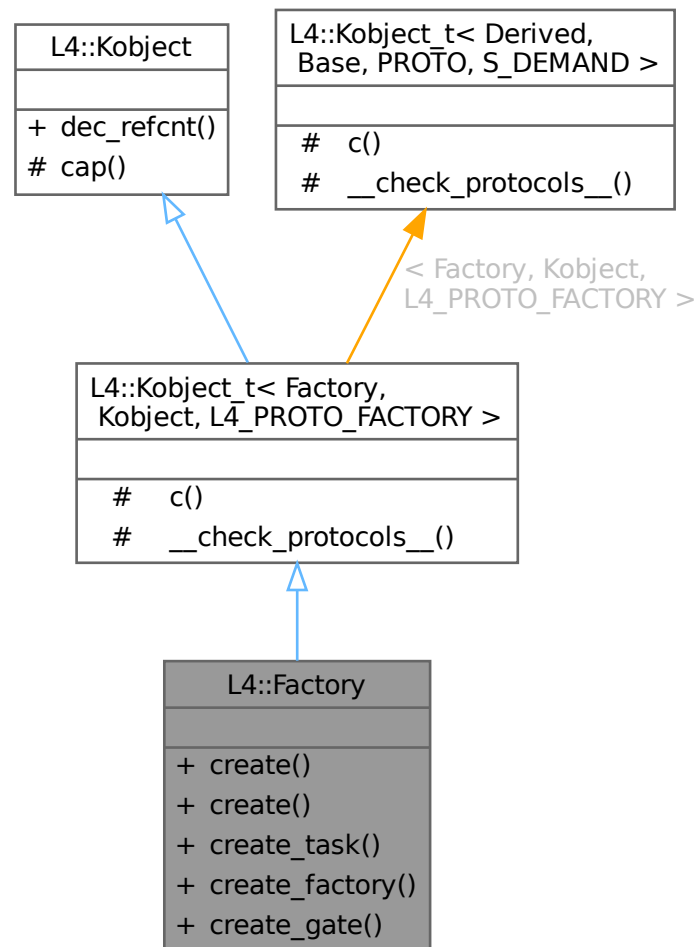
C++ Factory interface, see [Factory](#) for the C interface.

```
#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 const &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.

15.99.1 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](#).

15.99.2 Member Function Documentation

15.99.2.1 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 l4_utcb .

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:

**15.99.2.2 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 l4_utcb .

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:



15.99.2.3 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 l4_utcb .

Returns

Syscall return tag

Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	The factory instance requires L4_CAP_FPAGE_S rights on the invoked capability and L4_CAP_FPAGE_S is not present.
<i><0</i>	Error code.

Note

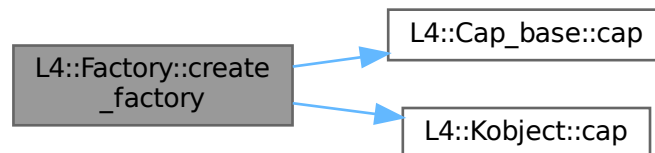
In addition to memory needed for internal data structures, the `limit` (quota) of the new factory is counted towards the quota of the creating factory. The `limit` must be within $1 \leq \text{limit} \leq 2^8 (\text{sizeof}(\text{l4_umword_t}) - 1) - 2$ otherwise the behavior is undefined.

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 377 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**15.99.2.4 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 L4_INVALID_CAP 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 l4_utcb .

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()`.

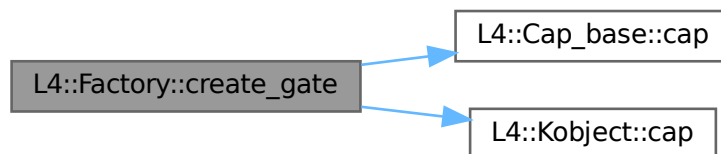
See also

[L4::lpc_gate](#)

Definition at line 410 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.99.2.5 create_task()

```

14_msgtag_t L4::Factory::create_task (
    Cap< Task > const & target_cap,
    14_fpage_t const & utcb_area,
    14_utcb_t * utcb = 14_utcb() ) [inline], [noexcept]
  
```

Create a new task.

Parameters

out	<i>target_cap</i>	The kernel stores the new task's capability into this slot.
	<i>utcb_area</i>	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.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to 14_utcb .

Returns

Syscall return tag

Return values

<code>L4_EOK</code>	No error occurred.
<code>-L4_EPERM</code>	The factory instance requires <code>L4_CAP_FPAGE_S</code> rights on the invoked capability and <code>L4_CAP_FPAGE_S</code> is not present.
<code><0</code>	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.

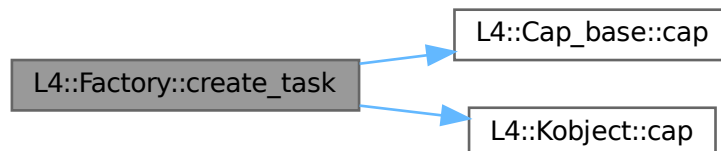
See also

[L4::Task](#)

Definition at line 344 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](#)

15.100 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.

15.100.1 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](#).

15.100.2 Constructor & Destructor Documentation

15.100.2.1 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:

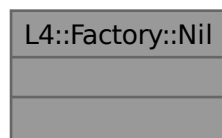
- [l4/sys/factory](#)

15.101 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:



15.101.1 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:

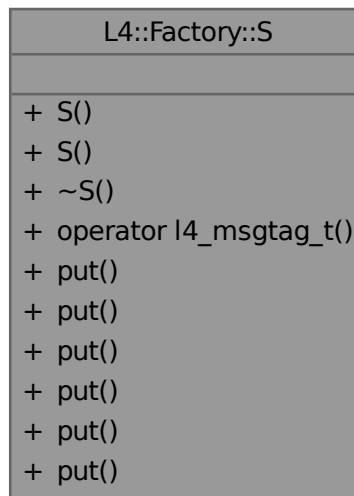
- [l4/sys/factory](#)

15.102 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.

15.102.1 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](#).

15.102.2 Constructor & Destructor Documentation

15.102.2.1 S() [1/2]

```
L4::Factory::S::S (
    S && o ) [inline], [noexcept]
```

Move ...

Parameters

<i>o</i>	Instance of S to move.
----------	--

Definition at line 108 of file [factory](#).

References [l4_msgtag_t::raw](#).

15.102.2.2 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 (L4::Factory).
	<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 l4_utcb .

Definition at line 132 of file [factory](#).

15.102.3 Member Function Documentation

15.102.3.1 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 L4_CAP_FPAGE_S rights on the invoked capability and L4_CAP_FPAGE_S is not present.
<i><0</i>	Error code.

Definition at line 158 of file [factory](#).

References [l4_msgtag_t::raw](#).

15.102.3.2 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](#).

15.102.3.3 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](#).

15.102.3.4 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](#).

15.102.3.5 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](#).

15.102.3.6 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:

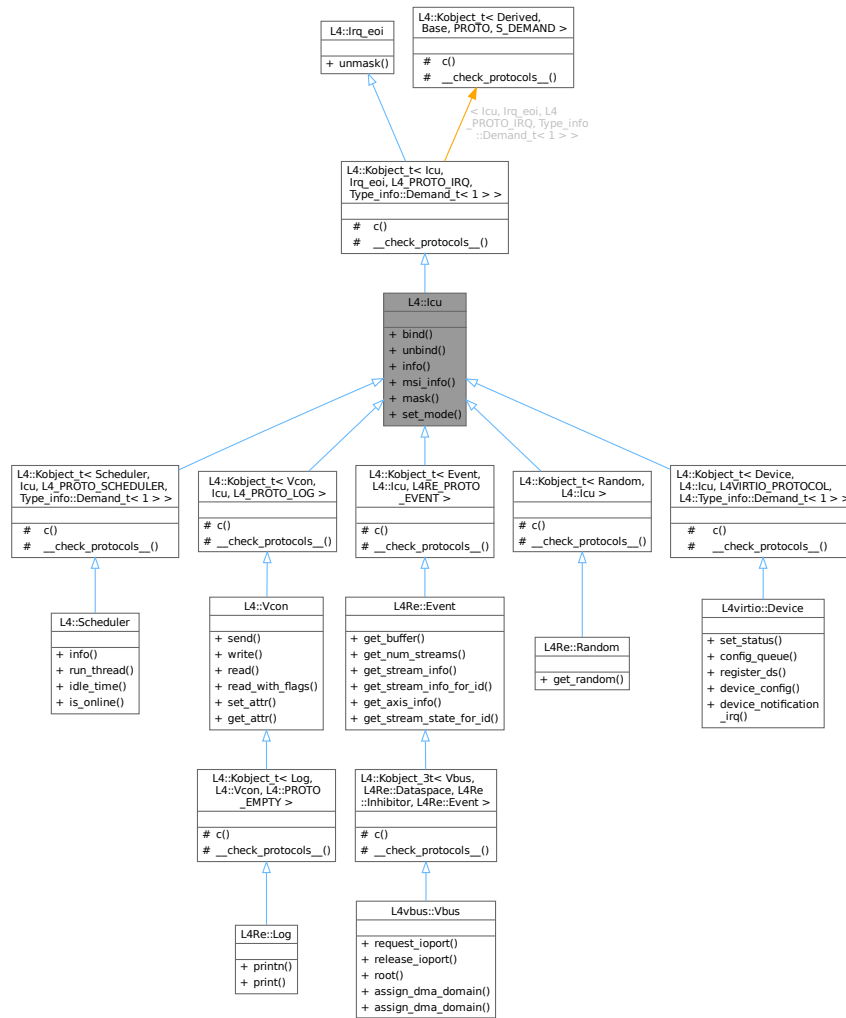
- [l4/sys/factory](#)

15.103 L4::lcu Class Reference

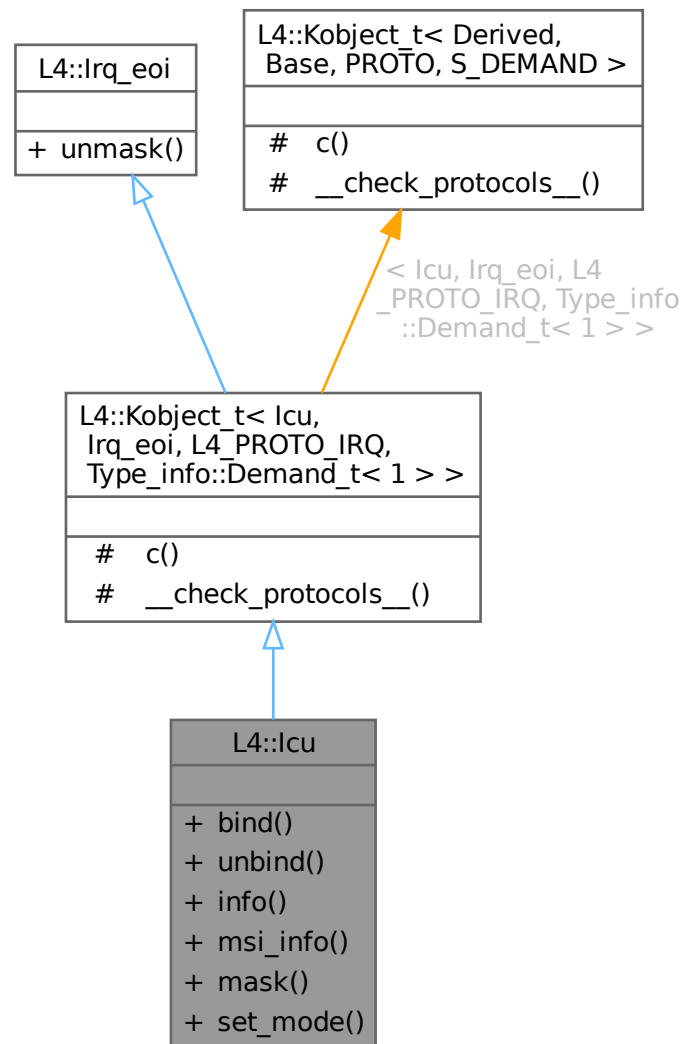
C++ [lcu](#) interface, see [Interrupt controller](#) for the C interface.

```
#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.

15.103.1 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 259 of file [irq](#).

15.103.2 Member Function Documentation

15.103.2.1 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 l4_utcb .

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 L4_CAP_FPAGE_W on <code>irq</code> and L4_CAP_FPAGE_W was not present.

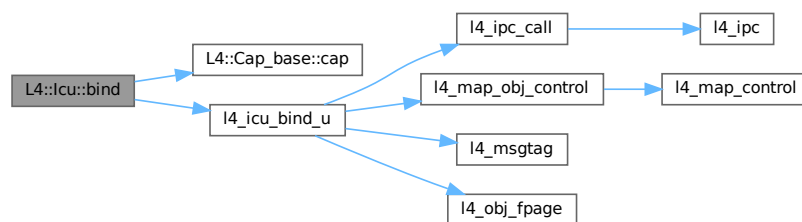
In case the `irq` is already bound to an interrupt source, it is unbound first. In case the `irq` is bound and the interrupt source is bound to a different [L4::Irq](#) object, only the unbinding happens. An [L4::Irq](#) object that is bound to an interrupt source will get unbound if the [L4::Irq](#) object is deleted.

Definition at line 318 of file [irq](#).

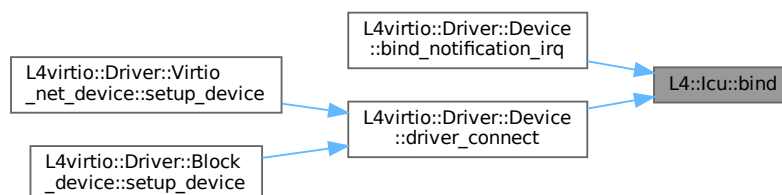
References [L4::Cap_base::cap\(\)](#), and [l4_icu_bind_u\(\)](#).

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:



15.103.2.2 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	Info structure to be filled with information.
	utcb	UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb .

Returns

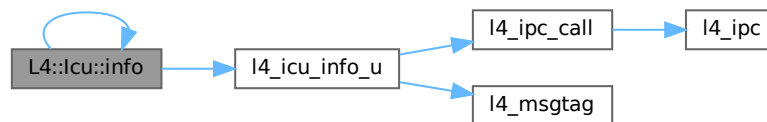
Syscall return tag

Definition at line 353 of file [irq](#).

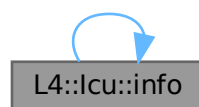
References [info\(\)](#), and [l4_icu_info_u\(\)](#).

Referenced by [info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.103.2.3 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 l4_utcb .

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 401 of file [irq](#).

References [l4_icu_mask_u\(\)](#).

Here is the call graph for this function:



15.103.2.4 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 l4_icu_msi_info_t structure receiving the address and the data value to trigger this MSI.

Returns

Syscall return tag

15.103.2.5 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 L4_irq_mode .
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb .

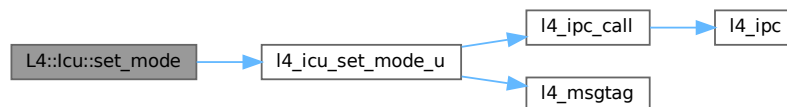
Returns

Syscall return tag

Definition at line 429 of file [irq](#).

References [l4_icu_set_mode_u\(\)](#).

Here is the call graph for this function:



15.103.2.6 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 l4_utcb .

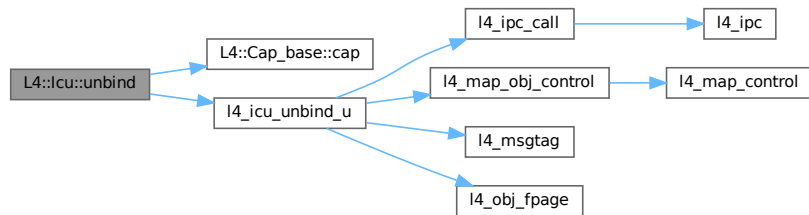
Returns

Syscall return tag

Definition at line 336 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:

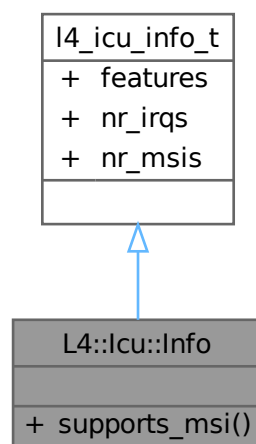
- [l4/sys/irq](#)

15.104 L4::lcu::Info Class Reference

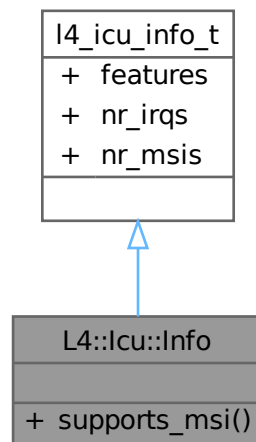
This class encapsulates information about an ICU.

```
#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,.

15.104.1 Detailed Description

This class encapsulates information about an ICU.

Definition at line [286](#) of file [irq](#).

The documentation for this class was generated from the following file:

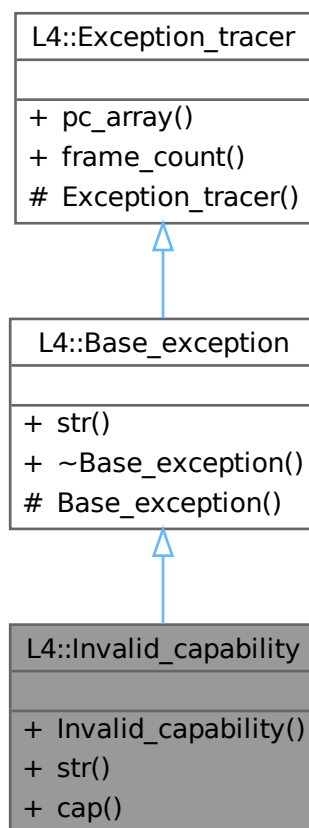
- [l4/sys/irq](#)

15.105 L4::Invalid_capability Class Reference

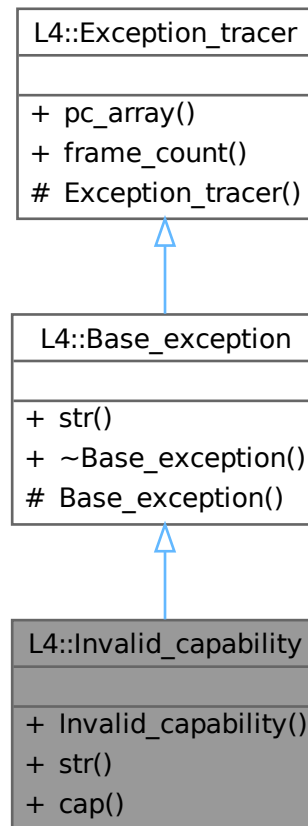
Indicates that an invalid object was invoked.

```
#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 () throw ()`
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.

15.105.1 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](#).

15.105.2 Constructor & Destructor Documentation

15.105.2.1 Invalid_capability()

```
L4::Invalid_capability::Invalid_capability (
    Cap< void > const & o ) [inline], [explicit], [noexcept]
```

Create an Invalid_object exception for the Object o.

Parameters

o	The object that caused the server side error.
----------	---

Definition at line [255](#) of file [exceptions](#).

15.105.3 Member Function Documentation

15.105.3.1 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:

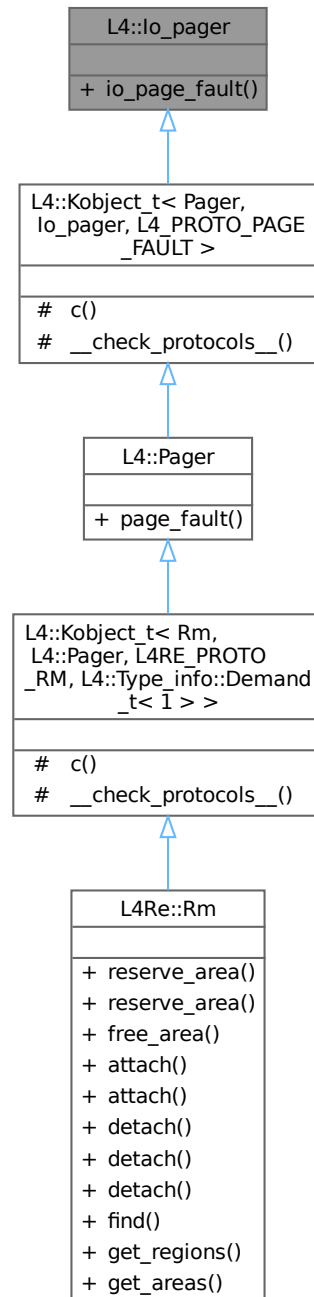
- [l4/cxx/exceptions](#)

15.106 L4::lo_pager Class Reference

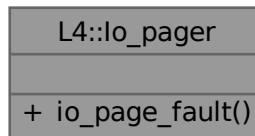
[lo_pager](#) interface.

```
#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.

15.106.1 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](#).

15.106.2 Member Function Documentation

15.106.2.1 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:

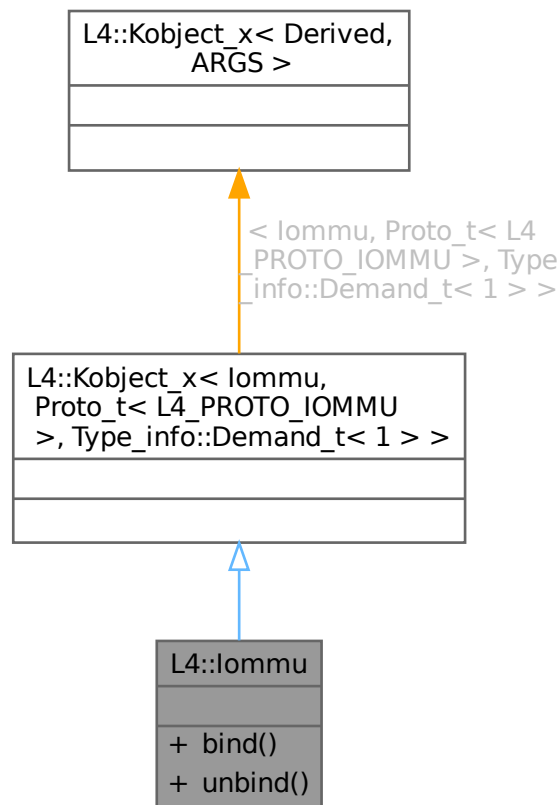
- [l4/sys/pager](#)

15.107 L4::lommu Class Reference

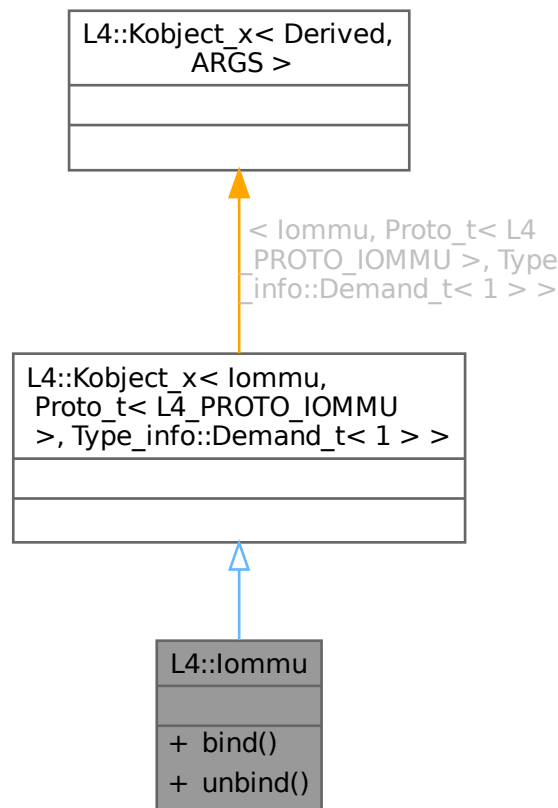
Interface for IO-MMUs used for DMA remapping.

```
#include <iommu>
```

Inheritance diagram for L4::lommu:



Collaboration diagram for L4::iommu:



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*.

15.107.1 Detailed Description

Interface for IO-MMUs used for DMA remapping.

Note

This interface is only present in the kernel if the kernel recognized 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](#).

15.107.2 Member Function Documentation

15.107.2.1 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 (L4::Task created with <code>L4_PROTO_DMA_SPACE</code>) providing the mappings that shall be used for the device(s).

15.107.2.2 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 bind() .

The documentation for this class was generated from the following file:

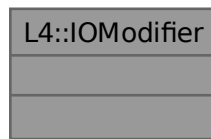
- `l4/sys/iommu`

15.108 L4::IOModifier Class Reference

Modifier class for the IO stream.

```
#include <basic_ostream>
```


Collaboration diagram for L4::IOModifier:



15.108.1 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:

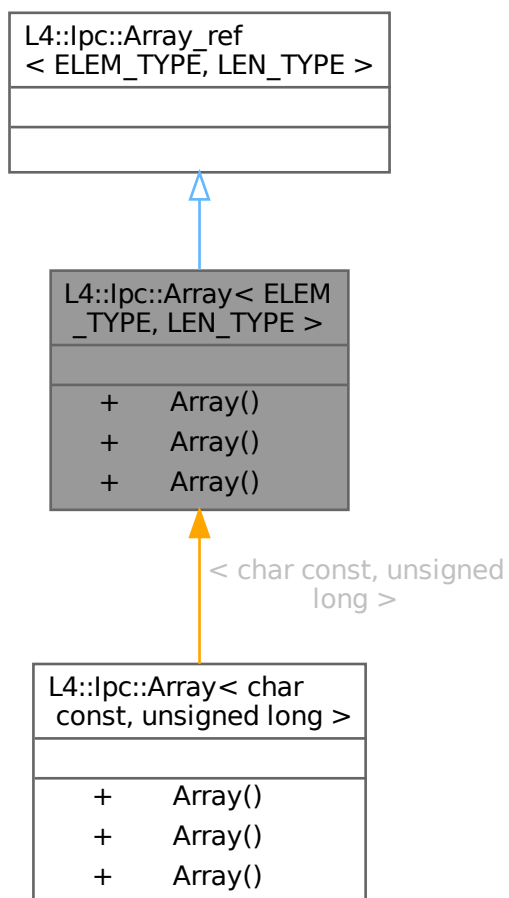
- [l4/cxx/basic_ostream](#)

15.109 L4::lpc::Array< ELEM_TYPE, LEN_TYPE > Struct Template Reference

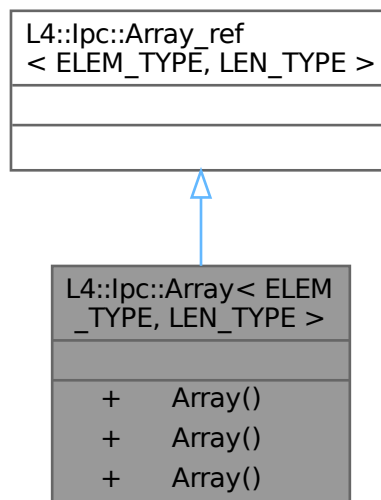
[Array](#) data type for dynamically sized arrays in RPCs.

```
#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.

15.109.1 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`

15.110 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.

15.110.1 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:

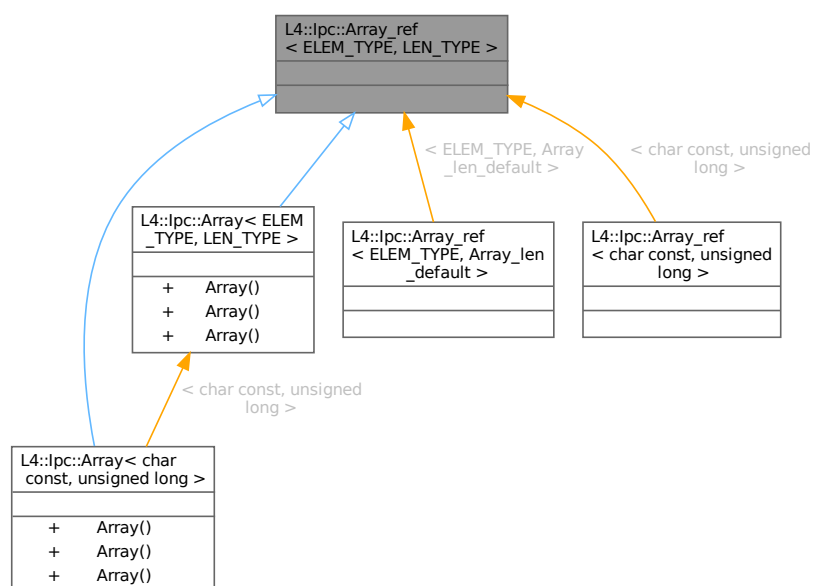
- `I4/sys/cxx/ipc_array`

15.111 L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE > Struct Template Reference

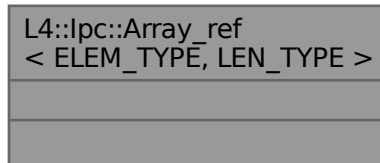
[Array](#) reference data type for arrays located in the message.

```
#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 >:



15.111.1 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:

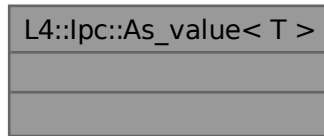
- `l4/sys/cxx/ipc_array`

15.112 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 >:



15.112.1 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:

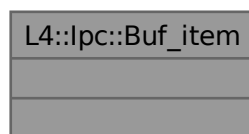
- [l4/sys/cxx/ipc_types](#)

15.113 L4::lpc::Buf_item Class Reference

RPC warpper for a receive item.

```
#include <ipc_types>
```

Collaboration diagram for L4::lpc::Buf_item:



15.113.1 Detailed Description

RPC warpper for a receive item.

Definition at line 305 of file [ipc_types](#).

The documentation for this class was generated from the following file:

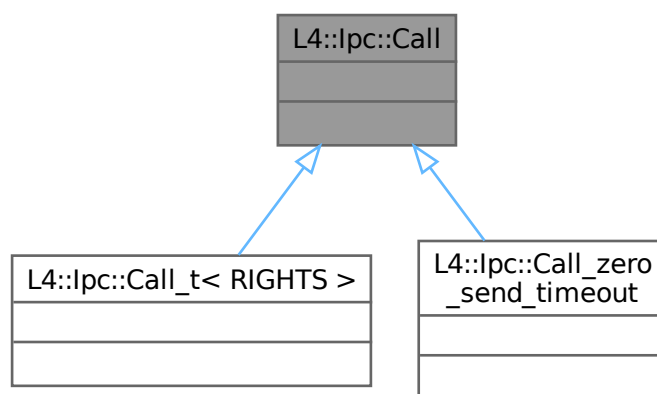
- [l4/sys/cxx/ipc_types](#)

15.114 L4::lpc::Call Struct Reference

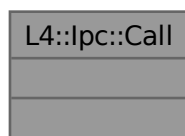
RPC attribute for a standard RPC call.

```
#include <ipc_iface>
```

Inheritance diagram for L4::lpc::Call:



Collaboration diagram for L4::lpc::Call:



15.114.1 Detailed Description

RPC attribute for a standard RPC call.

This is the default for the *FLAGS* parameter for L4::lpc::Msg::Rpc_call L4::lpc::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:

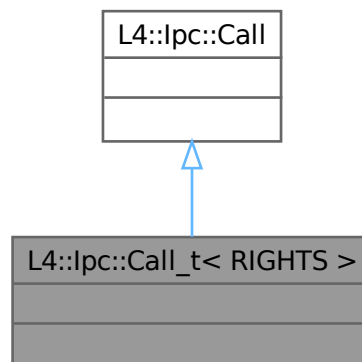
- [l4/sys/cxx/ipc_iface](#)

15.115 L4::lpc::Call_t< RIGHTS > Struct Template Reference

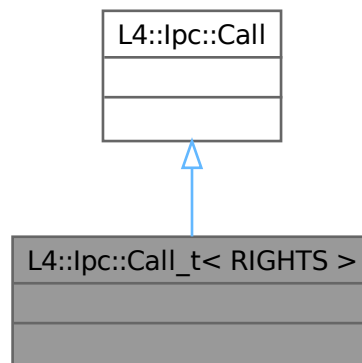
RPC attribute for an RPC call with required rights.

```
#include <ipc_iface>
```

Inheritance diagram for L4::lpc::Call_t< RIGHTS >:



Collaboration diagram for L4::ipc::Call_t< RIGHTS >:



15.115.1 Detailed Description

```
template<unsigned RIGHTS>
struct L4::ipc::Call_t< RIGHTS >
```

RPC attribute for an RPC call with required rights.

Template Parameters

<i>RIGHTS</i>	The capability rights required for this call. L4_CAP_FPAGE_W and L4_CAP_FPAGE_S are checked within the server (and -L4_EPERM shall be returned if the caller has insufficient rights). L4_CAP_FPAGE_R 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:

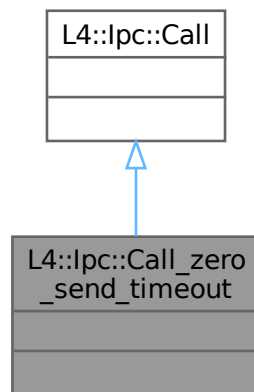
- [l4/sys/cxx/ipc_iface](#)

15.116 L4::ipc::Call_zero_send_timeout Struct Reference

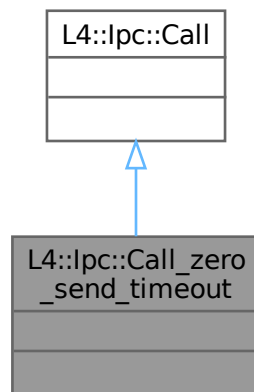
RPC attribute for an RPC call, with zero send timeout.

```
#include <ipc_iface>
```

Inheritance diagram for L4::lpc::Call_zero_send_timeout:



Collaboration diagram for L4::lpc::Call_zero_send_timeout:



15.116.1 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](#)

15.117 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"> + Cap() + Cap() + Cap() + Cap() + Cap() + cap() + rights() + fpage() + is_valid() + from_ci()

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.

15.117.1 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 545 of file [ipc_types](#).

15.117.2 Member Enumeration Documentation

15.117.2.1 anonymous enum

```
template<typename T >
anonymous enum
```

Enumerator

Rights_mask	Mask for rights bits stored internally. L4_FPAGE_RIGHTS_MASK L4_FPAGE_C_NO_REF_CNT L4_FPAGE_C_OBJ_RIGHTS).
Cap_mask	Mask for significant capability bits. (incl. the invalid bit to support invalid caps)

Definition at line 551 of file [ipc_types](#).

15.117.3 Constructor & Destructor Documentation

15.117.3.1 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 L4_fpage_rights and L4_obj_fpage_ctl .

Definition at line [593](#) of file [ipc_types](#).

15.117.4 Member Function Documentation

15.117.4.1 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 L4_fpage_rights).
----------	--

Definition at line [601](#) 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](#)

15.118 L4::lpc::Gen_fpage< T > Class Template Reference

Generic RPC wrapper for [L4](#) flex-pages.

```
#include <ipc_types>
```

Inherits `T`.

Collaboration diagram for L4::lpc::Gen_fpage< T >:

L4::lpc::Gen_fpage< T >
<ul style="list-style-type: none"> + is_valid() + cap_received() + id_received() + local_id_received() + is_compound() + data() + base_x()

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.

15.118.1 Detailed Description

template<typename T>
class L4::lpc::Gen_fpage< T >

Generic RPC wrapper for [L4](#) flex-pages.

Template Parameters

<i>T</i>	Underlying specific flexpage type.
----------	------------------------------------

Definition at line 321 of file [ipc_types](#).

15.118.2 Member Function Documentation

15.118.2.1 cap_received()

```
template<typename T >
bool L4::Ipc::Gen_fpage< T >::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 442 of file [ipc_types](#).

15.118.2.2 id_received()

```
template<typename T >
bool L4::Ipc::Gen_fpage< T >::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 454 of file [ipc_types](#).

15.118.2.3 is_compound()

```
template<typename T >
bool L4::Ipc::Gen_fpage< T >::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 472 of file [ipc_types](#).

15.118.2.4 local_id_received()

```
template<typename T >
bool L4::Ipc::Gen_fpage< T >::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 [464](#) of file [ipc_types](#).

The documentation for this class was generated from the following file:

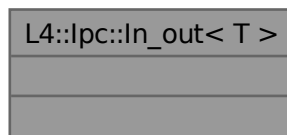
- [l4/sys/cxx/ipc_types](#)

15.119 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 >:

**15.119.1 Detailed Description**

```
template<typename T>
struct L4::lpc::ln_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:

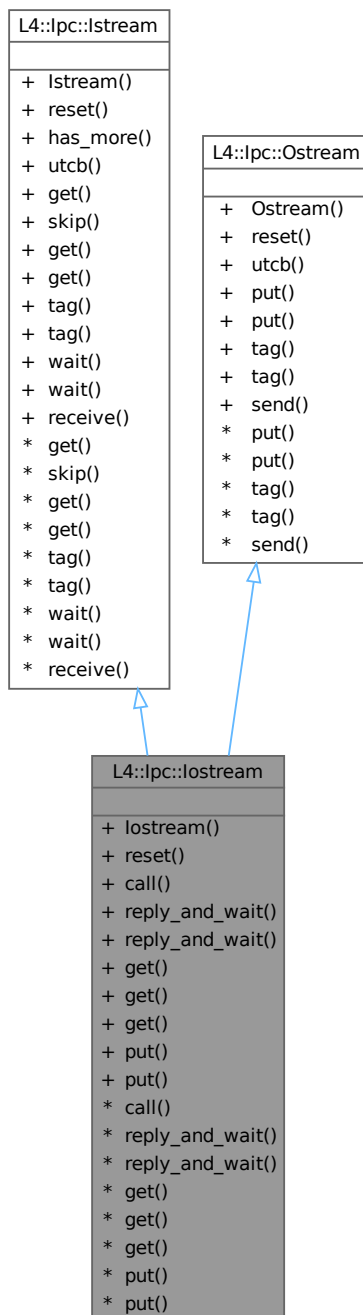
- [l4/sys/cxx/ipc_types](#)

15.120 L4::lpc::lostream Class Reference

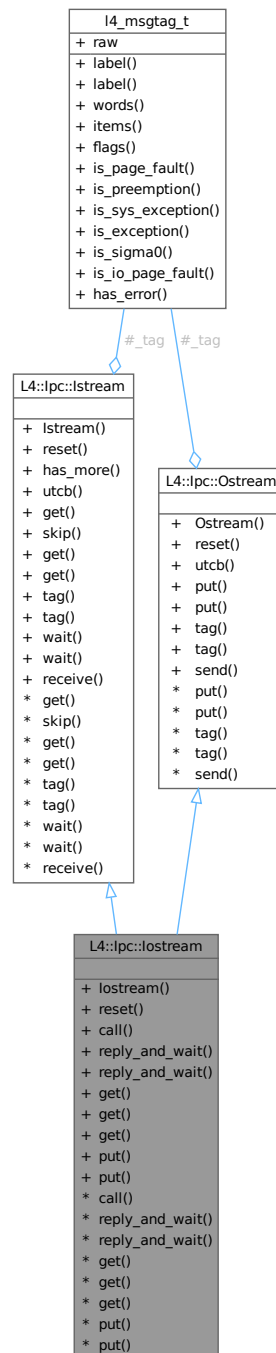
Input/Output stream for IPC [un]marshalling.

```
#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.

15.120.1 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`.

15.120.2 Constructor & Destructor Documentation

15.120.2.1 `lostream()`

```
L4::lpc::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](#).

15.120.3 Member Function Documentation

15.120.3.1 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.

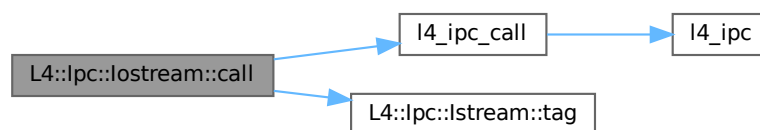
Examples

[examples/libs/l4re/streammap/client.cc](#).

Definition at line 971 of file [ipc_stream](#).

References [l4_ipc_call\(\)](#), and [L4::lpc::Iostream::tag\(\)](#).

Here is the call graph for this function:



15.120.3.2 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 Msg_ptr 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 `Istream::operator>>()`

Definition at line 450 of file [ipc_stream](#).

15.120.3.3 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 `Istream::operator>>()`

Definition at line 475 of file [ipc_stream](#).

15.120.3.4 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 `Istream::operator>>()`

Definition at line 405 of file [ipc_stream](#).

15.120.3.5 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 669 of file [ipc_stream](#).

15.120.3.6 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 687 of file [ipc_stream](#).

15.120.3.7 `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.

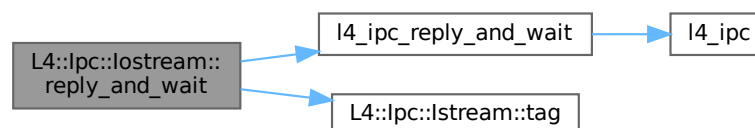
This is a combined IPC operation consisting of a send operation and an open wait for any message.

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 986 of file [ipc_stream](#).

References [l4_ipc_reply_and_wait\(\)](#), and [L4::Ipc::Iostream::tag\(\)](#).

Here is the call graph for this function:



15.120.3.8 `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.

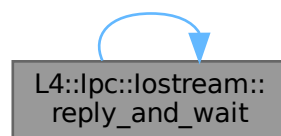
A reply and wait is usually used by servers that reply to a client and wait for the next request by any other client.

Definition at line 885 of file [ipc_stream](#).

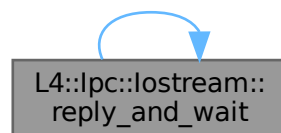
References [L4_IPC_SEND_TIMEOUT_0](#), and [reply_and_wait\(\)](#).

Referenced by [reply_and_wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.120.3.9 reset()

```
void L4::Ipc::Iostream::reset ( ) [inline]
```

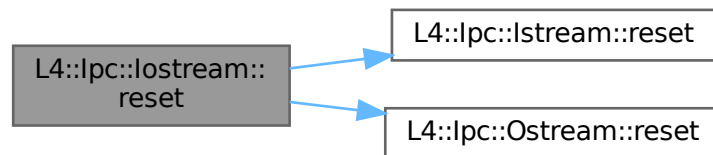
Reset the stream to its initial state.

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:

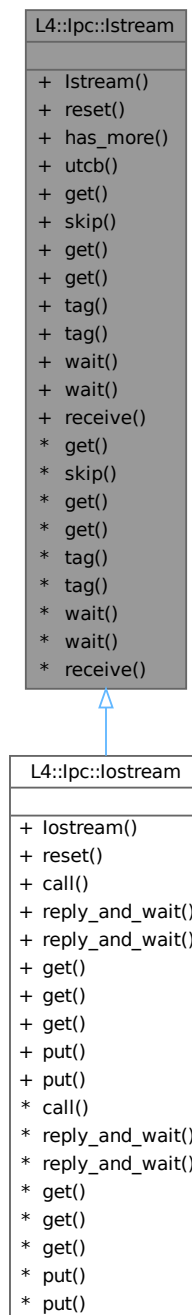
- [l4/cxx/ipc_stream](#)

15.121 L4::Ipc::Istream Class Reference

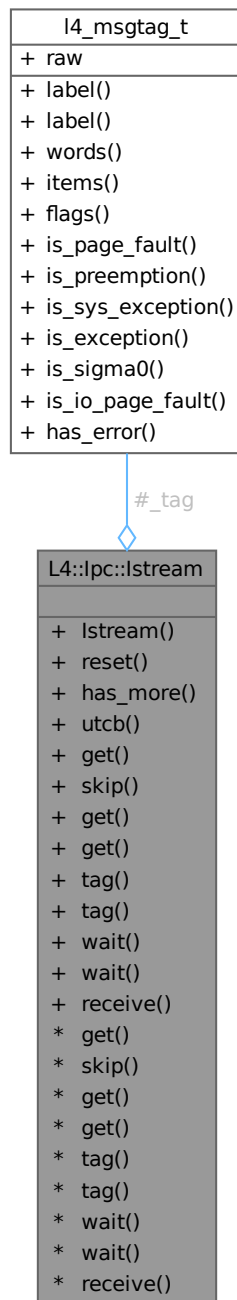
Input stream for IPC unmarshalling.

```
#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.

15.121.1 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](#).

15.121.2 Constructor & Destructor Documentation

15.121.2.1 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](#).

15.121.3 Member Function Documentation

15.121.3.1 `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 Msg_ptr 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 `Istream::operator>>()`

Definition at line 450 of file [ipc_stream](#).

References [L4_UNLIKELY](#).

15.121.3.2 `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 `Istream::operator>>()`

Definition at line 475 of file `ipc_stream`.

References [L4_UNLIKELY](#).

15.121.3.3 `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 <code>T</code> .
<i>elems</i>	Number of elements of type <code>T</code> to copy out.

Returns

The number of elements copied out.

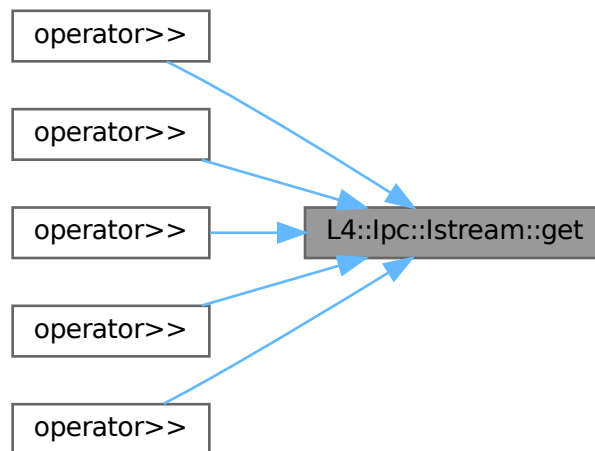
See `Istream::operator>>()`

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:



15.121.3.4 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:



15.121.3.5 reset()

```
void L4::Ipc::Istream::reset ( ) [inline]
```

Reset the stream to empty, and ready for [receive\(\)/wait\(\)](#).

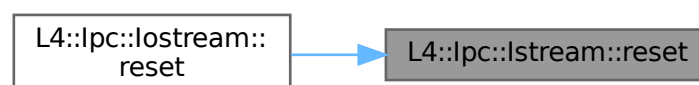
The stream is reset to the same state as on its creation.

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:



15.121.3.6 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](#).

15.121.3.7 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 [Istream::operator>>\(\)](#)

Definition at line 528 of file [ipc_stream](#).

15.121.3.8 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.

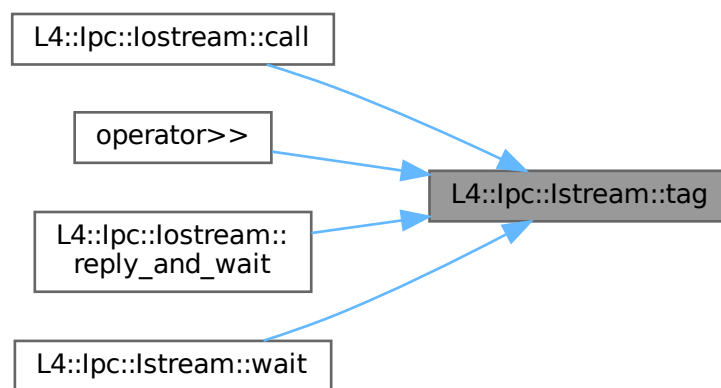
This is in particular useful for handling page faults or exceptions.

See `Istream::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:

**15.121.3.9 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:



15.121.3.10 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

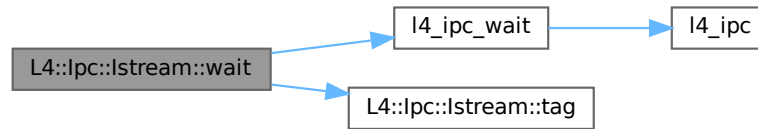
The IPC result tag ([l4_msgtag_t](#)).

This wait is actually known as 'open wait'.

Definition at line [1018](#) of file [ipc_stream](#).

References [l4_ipc_wait\(\)](#), and [tag\(\)](#).

Here is the call graph for this function:



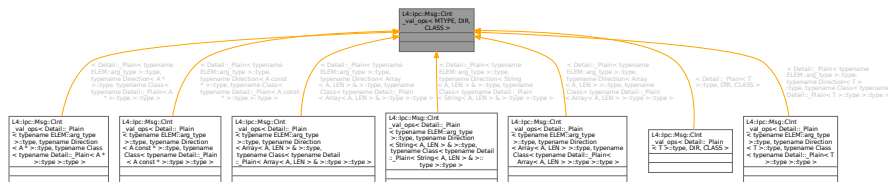
The documentation for this class was generated from the following file:

- [l4/cxx/ipc_stream](#)

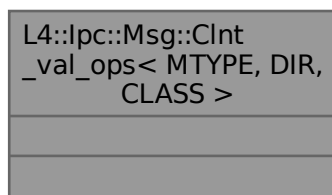
15.122 L4::ipc::Msg::Cnt_val_ops< MTYPE, DIR, CLASS > Struct Template Reference

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 >:



15.122.1 Detailed Description

```
template<typename MTYPE, typename DIR, typename CLASS>
struct L4::lpc::Msg::Clnt_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>	Dir_in (client -> server), or Dir_out (server -> client)
<i>CLASS</i>	Cls_data , Cls_item , or Cls_buffer

Definition at line [221](#) of file [ipc_basics](#).

The documentation for this struct was generated from the following file:

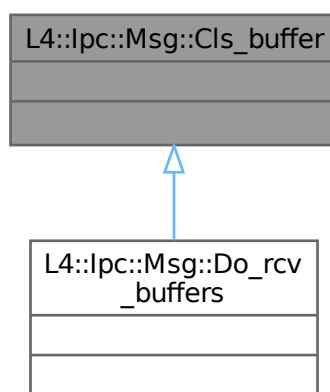
- [l4/sys/cxx/ipc_basics](#)

15.123 L4::lpc::Msg::Cls_buffer Struct Reference

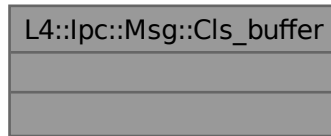
Marker type for receive buffer values.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Cls_buffer:



Collaboration diagram for L4::lpc::Msg::Cls_buffer:



15.123.1 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:

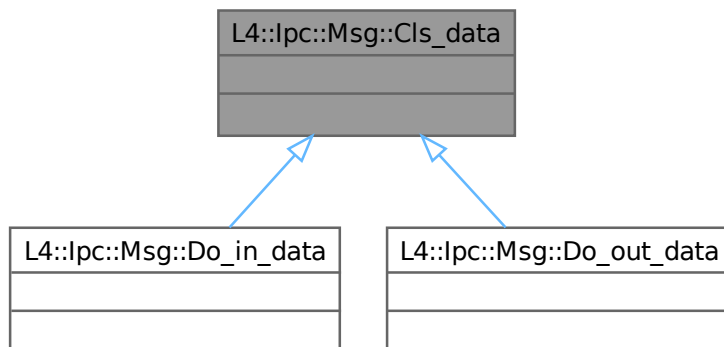
- `l4/sys/cxx/ipc_basics`

15.124 L4::lpc::Msg::Cls_data Struct Reference

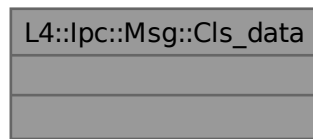
Marker type for data values.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Cls_data:



Collaboration diagram for L4::lpc::Msg::Cls_data:



15.124.1 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:

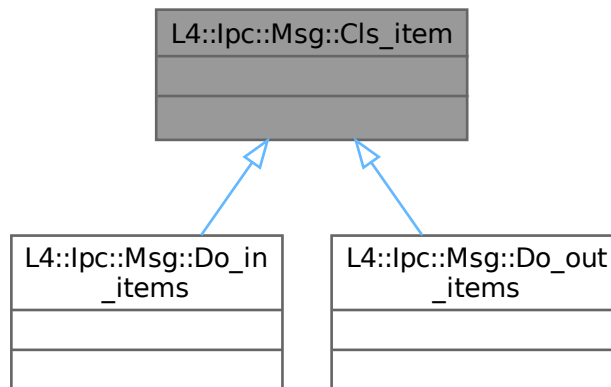
- l4/sys/cxx/ipc_basics

15.125 L4::lpc::Msg::Cls_item Struct Reference

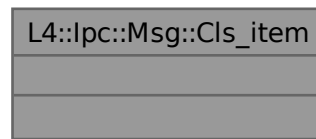
Marker type for item values.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Cls_item:



Collaboration diagram for L4::lpc::Msg::Cls_item:



15.125.1 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:

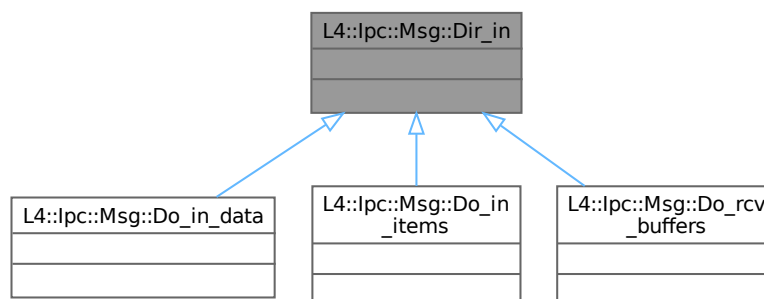
- `l4/sys/cxx/ipc_basics`

15.126 L4::lpc::Msg::Dir_in Struct Reference

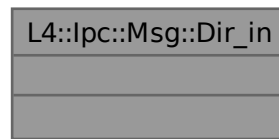
Marker type for input values.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Dir_in:



Collaboration diagram for L4::lpc::Msg::Dir_in:



15.126.1 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:

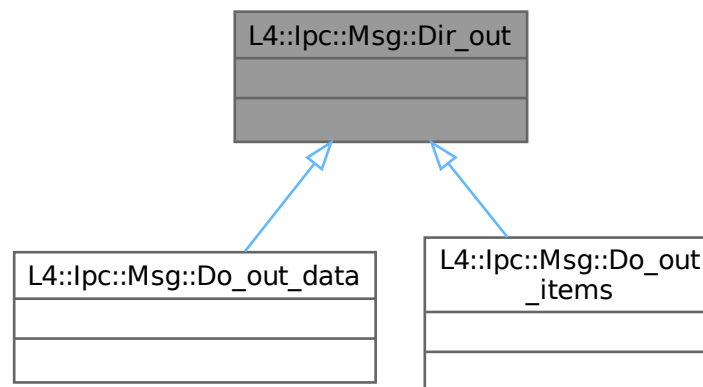
- l4/sys/cxx/ipc_basics

15.127 L4::lpc::Msg::Dir_out Struct Reference

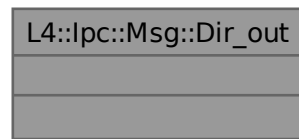
Marker type for output values.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Dir_out:



Collaboration diagram for L4::lpc::Msg::Dir_out:



15.127.1 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:

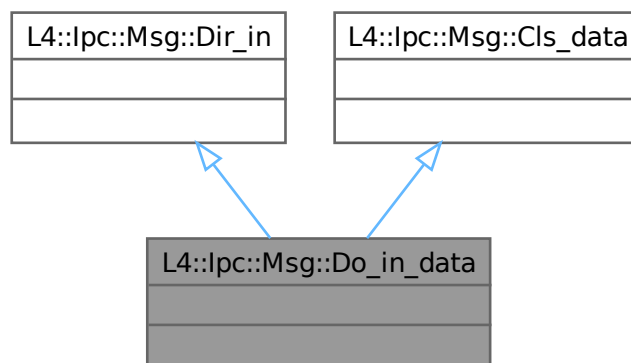
- l4/sys/cxx/ipc_basics

15.128 L4::lpc::Msg::Do_in_data Struct Reference

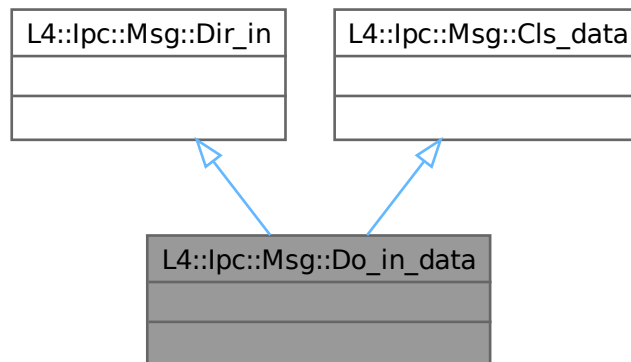
Marker for Input data.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do_in_data:



Collaboration diagram for L4::lpc::Msg::Do_in_data:



15.128.1 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:

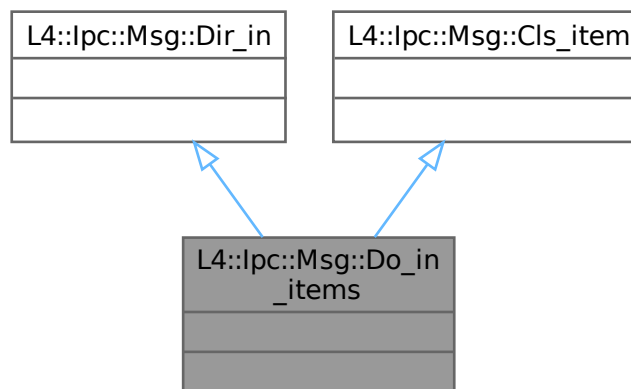
- `l4/sys/cxx/ipc_basics`

15.129 L4::lpc::Msg::Do_in_items Struct Reference

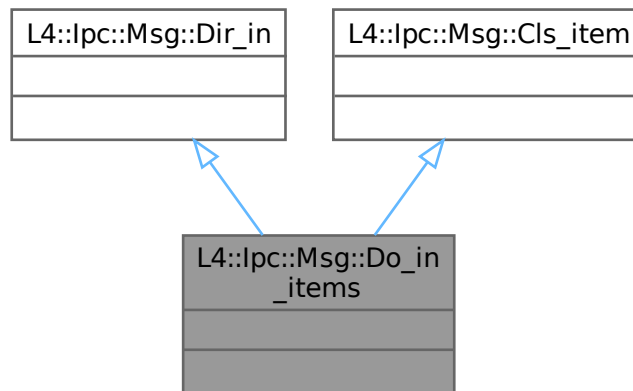
Marker for Input items.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do_in_items:



Collaboration diagram for L4::lpc::Msg::Do_in_items:



15.129.1 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:

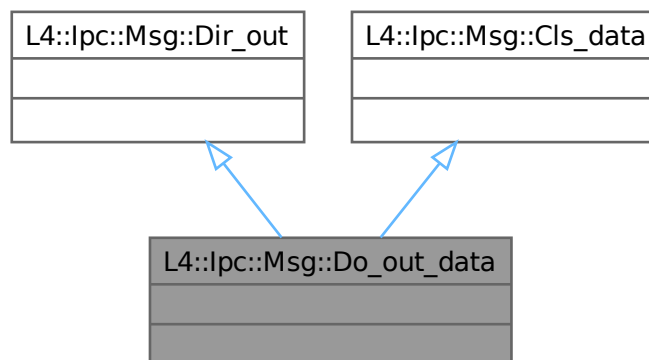
- `l4/sys/cxx/ipc_basics`

15.130 L4::lpc::Msg::Do_out_data Struct Reference

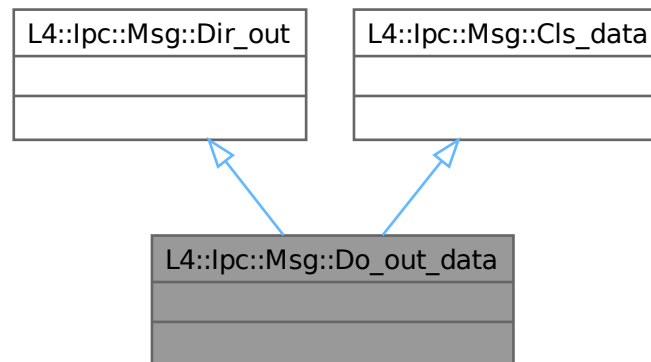
Marker for Output data.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do_out_data:



Collaboration diagram for L4::lpc::Msg::Do_out_data:



15.130.1 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:

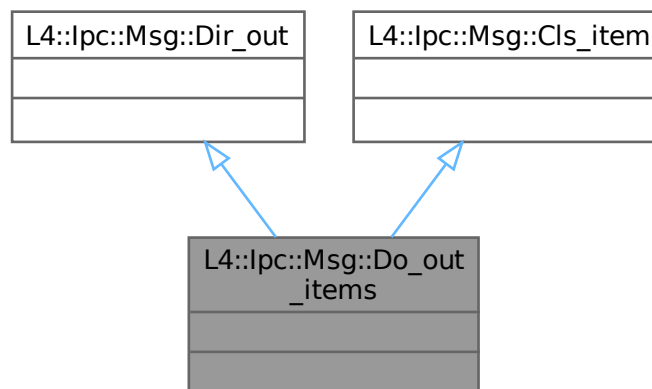
- `I4/sys/cxx/ipc_basics`

15.131 L4::lpc::Msg::Do_out_items Struct Reference

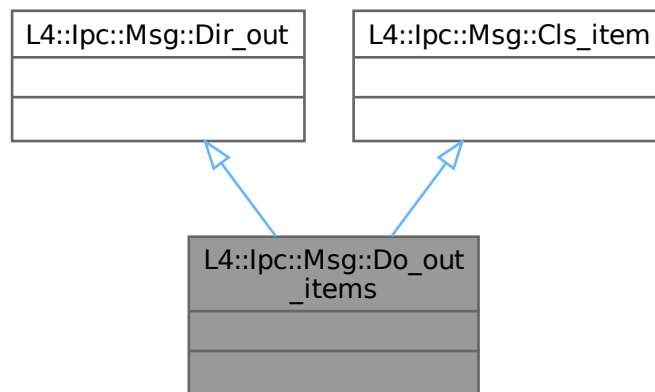
Marker for Output items.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do_out_items:



Collaboration diagram for L4::lpc::Msg::Do_out_items:



15.131.1 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:

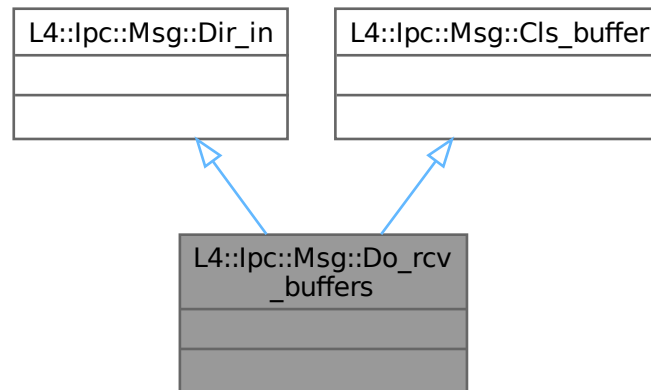
- l4/sys/cxx/ipc_basics

15.132 L4::lpc::Msg::Do_rcv_buffers Struct Reference

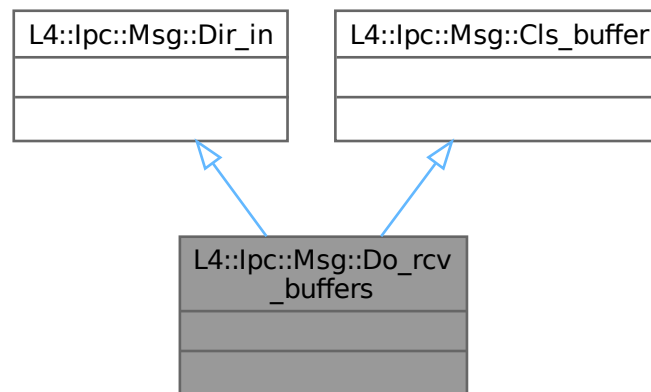
Marker for receive buffers.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do_rcv_buffers:



Collaboration diagram for L4::lpc::Msg::Do_rcv_buffers:



15.132.1 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:

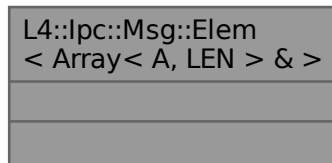
- l4/sys/cxx/ipc_basics

15.133 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.

15.133.1 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:

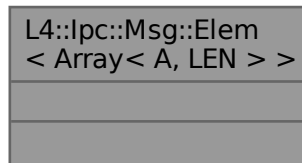
- `l4/sys/cxx/ipc_array`

15.134 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.

15.134.1 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:

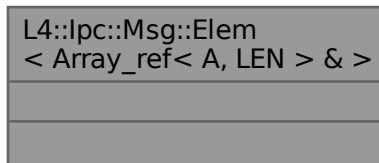
- `l4/sys/cxx/ipc_array`

15.135 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.

15.135.1 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:

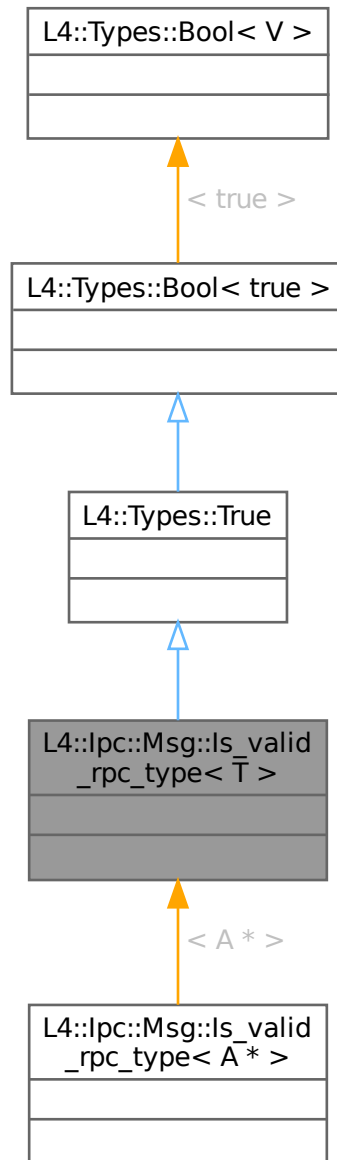
- l4/sys/cxx/ipc_array

15.136 L4::lpc::Msg::ls_valid_rpc_type< T > Struct Template Reference

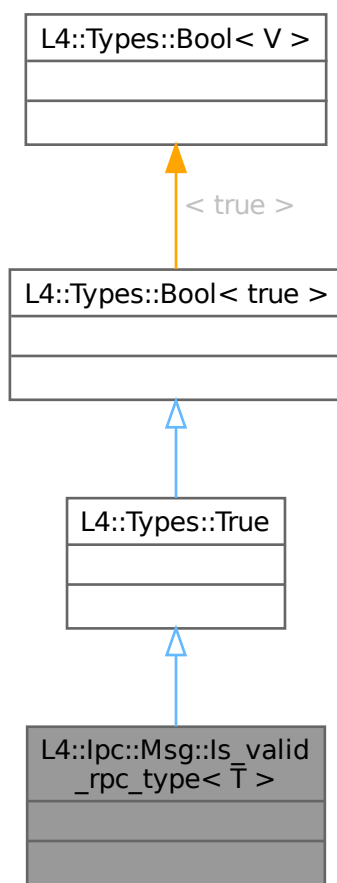
Type trait defining a valid RPC parameter type.

```
#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.

15.136.1 Detailed Description

```
template<typename T>
struct L4::lpc::Msg::ls_valid_rpc_type< T >
```

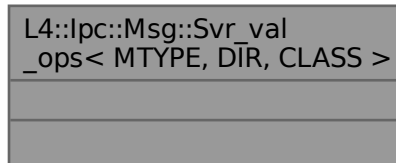
Type trait defining a valid RPC parameter type.

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 >:



15.138.1 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>	Dir_in (client -> server), or Dir_out (server -> client)
<i>CLASS</i>	Cls_data , Cls_item , or Cls_buffer

Definition at line 275 of file [ipc_basics](#).

The documentation for this struct was generated from the following file:

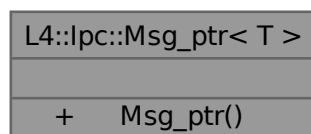
- l4/sys/cxx/ipc_basics

15.139 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.

15.139.1 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](#).

15.139.2 Constructor & Destructor Documentation

15.139.2.1 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](#)

15.140 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.

15.140.1 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:

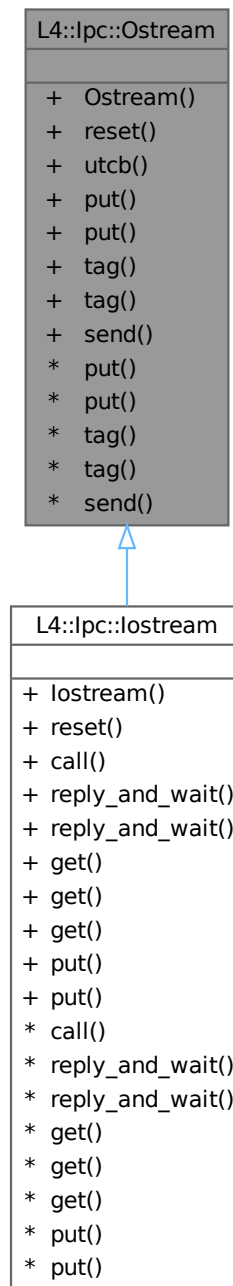
- [l4/sys/cxx/ipc_types](#)

15.141 L4::lpc::Ostream Class Reference

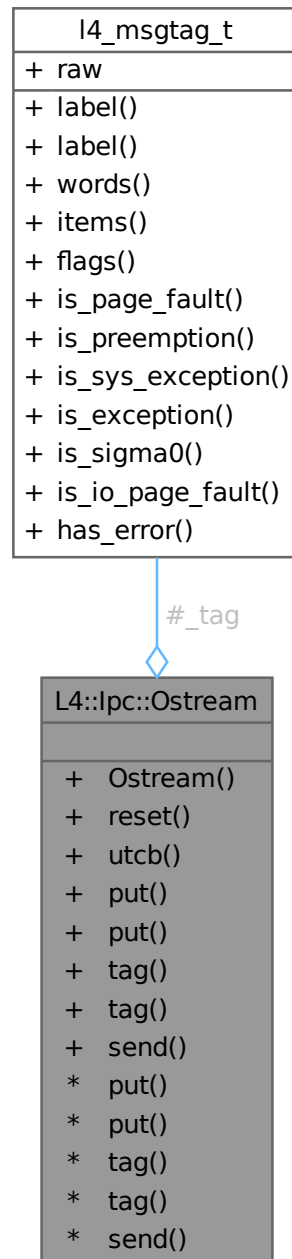
Output stream for IPC marshalling.

```
#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.

15.141.1 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 632 of file [ipc_stream](#).

15.141.2 Member Function Documentation

15.141.2.1 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.

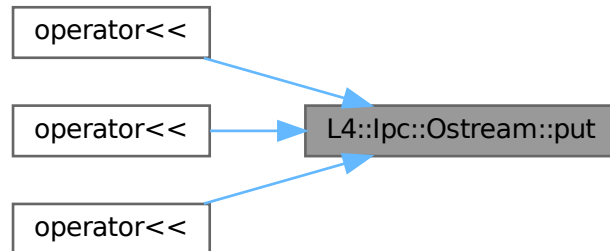
Parameters

<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 669 of file [ipc_stream](#).

Referenced by [operator<<\(\)](#), [operator<<\(\)](#), and [operator<<\(\)](#).

Here is the caller graph for this function:



15.141.2.2 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

<code>v</code>	The element to insert.
----------------	------------------------

Definition at line 687 of file [ipc_stream](#).

15.141.2.3 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

<code>dst</code>	The destination for the message.
<code>proto</code>	Protocol to use.
<code>flags</code>	Flags to use.

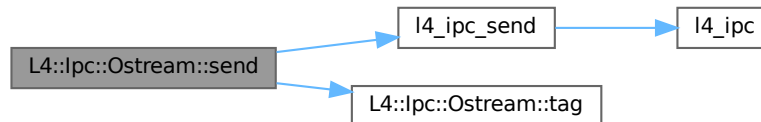
Returns

The syscall return tag.

Definition at line 964 of file [ipc_stream](#).

References [L4_IPC_NEVER](#), [l4_ipc_send\(\)](#), [L4_MSGTAG_FLAGS](#), and [tag\(\)](#).

Here is the call graph for this function:

**15.141.2.4 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 722 of file [ipc_stream](#).

15.141.2.5 tag() [2/2]

```
l4_msgtag_t L4::Ipc::Ostream::tag ( ) const [inline]
```

Extract the [L4](#) message tag from the stream.

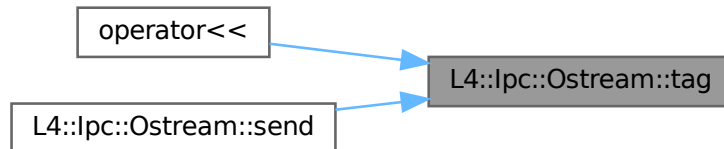
Returns

The extracted [L4](#) message tag.

Definition at line [715](#) of file [ipc_stream](#).

Referenced by [operator<<\(\)](#), and [send\(\)](#).

Here is the caller graph for this function:



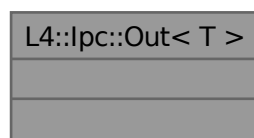
The documentation for this class was generated from the following file:

- [l4/cxx/ipc_stream](#)

15.142 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 >`:



15.142.1 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:

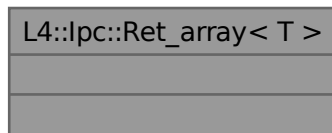
- [l4/sys/cxx/ipc_types](#)

15.143 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 >:



15.143.1 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:

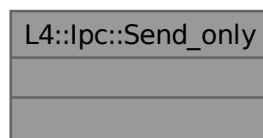
- [l4/sys/cxx/ipc_ret_array](#)

15.144 L4::lpc::Send_only Struct Reference

RPC attribute for a send-only RPC.

```
#include <ipc_iface>
```

Collaboration diagram for L4::lpc::Send_only:



15.144.1 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](#)

15.145 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:

L4::lpc::Small_buf
<ul style="list-style-type: none"> + Small_buf() + 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.

15.145.1 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](#).

15.145.2 Constructor & Destructor Documentation

15.145.2.1 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 l4_msg_item_consts_t . L4_RCV_ITEM_SINGLE_CAP will always be set.

Definition at line [278](#) of file [ipc_types](#).

15.145.2.2 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 l4_msg_item_consts_t . 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:

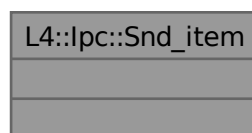
- [l4/sys/cxx/ipc_types](#)

15.146 L4::lpc::Snd_item Class Reference

RPC wrapper for a send item.

```
#include <ipc_types>
```

Collaboration diagram for L4::lpc::Snd_item:



15.146.1 Detailed Description

RPC wrapper for a send item.

Definition at line 294 of file [ipc_types](#).

The documentation for this class was generated from the following file:

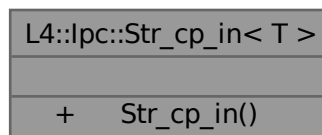
- [l4/sys/cxx/ipc_types](#)

15.147 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](#).

15.147.1 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](#).

15.147.2 Constructor & Destructor Documentation

15.147.2.1 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](#)

15.148 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"> + type() + length() + tag() + tag() + data() + data() + Varg() + Varg() + value() + is_of() and 8 more...

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.

15.148.1 Detailed Description

Variably sized RPC argument.

Definition at line 96 of file [ipc_varg](#).

15.148.2 Member Function Documentation

15.148.2.1 data()

```
char const * L4::Ipc::Varg::data ( ) const [inline]
```

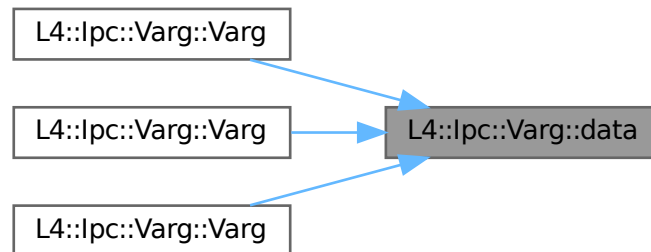
Returns

pointer to the data, also safe for direct data

Definition at line 123 of file [ipc_varg](#).

Referenced by [Varg\(\)](#), [Varg\(\)](#), and [Varg\(\)](#).

Here is the caller graph for this function:



15.148.2.2 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 Varg .
----------	---

Parameters

<i>v</i>	Pointer to store the value
----------	----------------------------

Returns

true when the [Varg](#) is of type T, false if not

Definition at line 184 of file [ipc_varg](#).

15.148.2.3 is_nil()

```
bool L4::Ipc::Varg::is_nil ( ) const [inline]
```

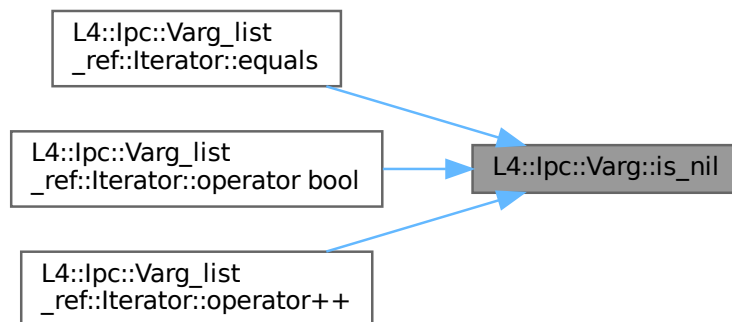
Returns

true if the [Varg](#) is of nil type.

Definition at line 171 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:

**15.148.2.4 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 168 of file [ipc_varg](#).

References [type\(\)](#).

Here is the call graph for this function:

**15.148.2.5 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 174 of file [ipc_varg](#).

References [type\(\)](#).

Here is the call graph for this function:

**15.148.2.6 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 114 of file [ipc_varg](#).

15.148.2.7 tag()

```
Tag L4::Ipc::Varg::tag ( ) const [inline]
```

Returns

the tag value (the Direct_data bit masked)

Definition at line 116 of file [ipc_varg](#).

15.148.2.8 type()

```
L4_varg_type L4::Ipc::Varg::type ( ) const [inline]
```

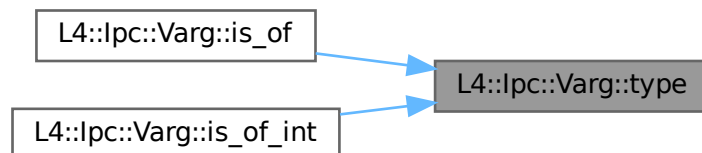
Returns

the type field of the tag

Definition at line 109 of file [ipc_varg](#).

Referenced by [is_of\(\)](#), and [is_of_int\(\)](#).

Here is the caller graph for this function:

**15.148.2.9 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 154 of file [ipc_varg](#).

The documentation for this class was generated from the following file:

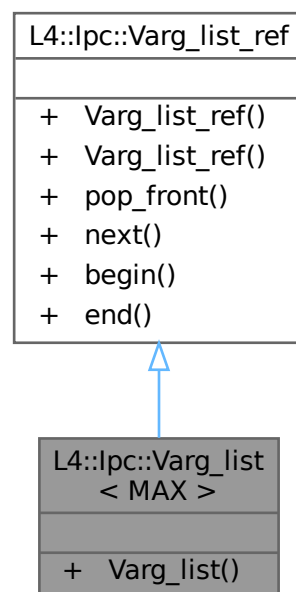
- l4/sys/cxx/ipc_varg

15.149 L4::lpc::Varg_list< MAX > Class Template Reference

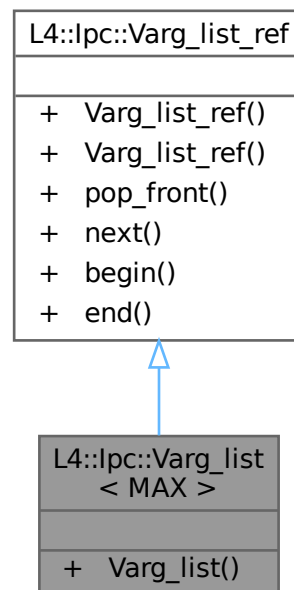
Self-contained list of variable-sized RPC parameters.

```
#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.

15.149.1 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 410 of file [ipc_varg](#).

The documentation for this class was generated from the following file:

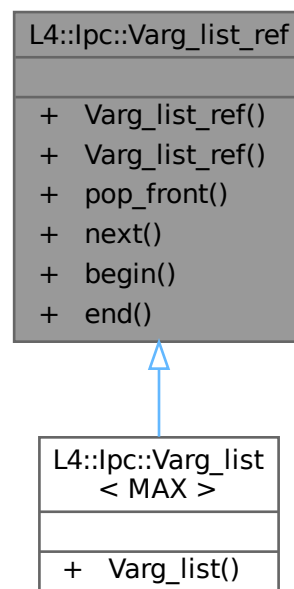
- [l4/sys/cxx/ipc_varg](#)

15.150 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"> + Varg_list_ref() + Varg_list_ref() + pop_front() + next() + begin() + end()

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.

15.150.1 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 252 of file [ipc_varg](#).

15.150.2 Constructor & Destructor Documentation

15.150.2.1 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 [331](#) of file [ipc_varg](#).

The documentation for this class was generated from the following file:

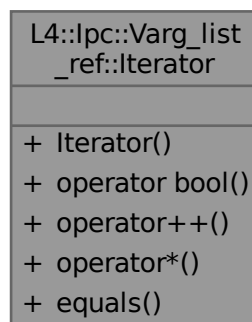
- [l4/sys/cxx/ipc_varg](#)

15.151 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

15.151.1 Detailed Description

[Iterator](#) for Valists.

Definition at line 337 of file [ipc_varg](#).

The documentation for this class was generated from the following file:

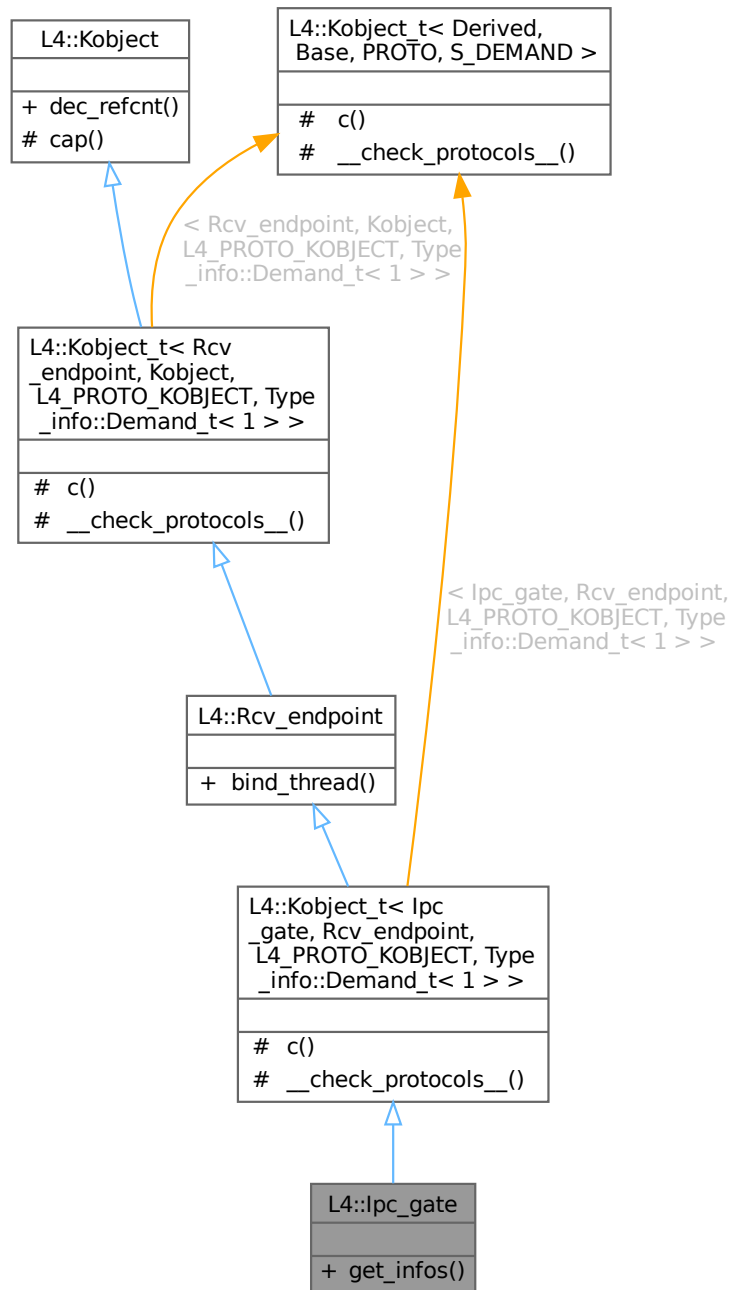
- l4/sys/cxx/ipc_varg

15.152 L4::lpc_gate Class Reference

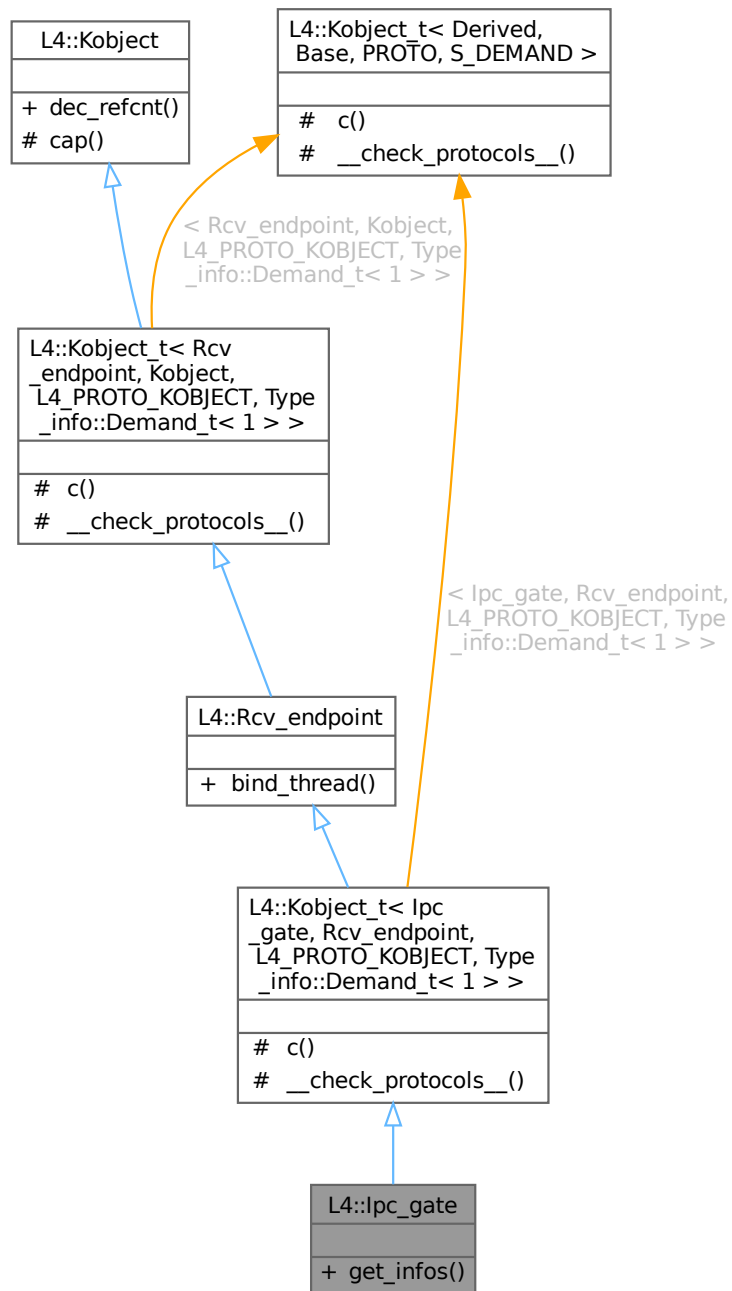
The C++ IPC gate interface, see [IPC-Gate API](#) for the C interface.

```
#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< lpc_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.

15.152.1 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::lpc_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::lpc_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](#).

15.152.2 Member Function Documentation

15.152.2.1 `get_infos()`

```
l4_msgtag_t L4::Ipc_gate::get_infos (
    l4_umword_t * label )
```

Get information about the IPC-gate.

Parameters

<code>out</code>	<code>label</code>	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:

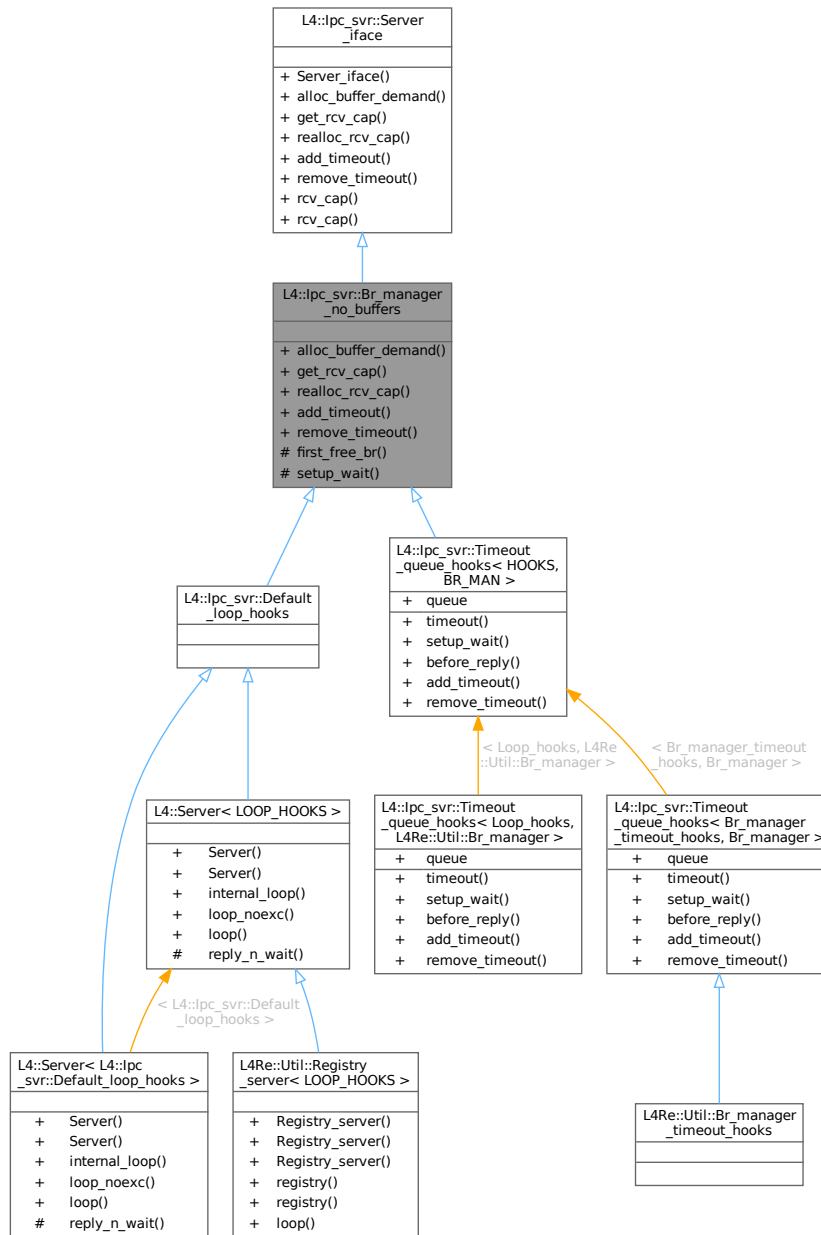
- `l4/sys/ipc_gate`

15.153 `L4::lpc_svr::Br_manager_no_buffers` Class Reference

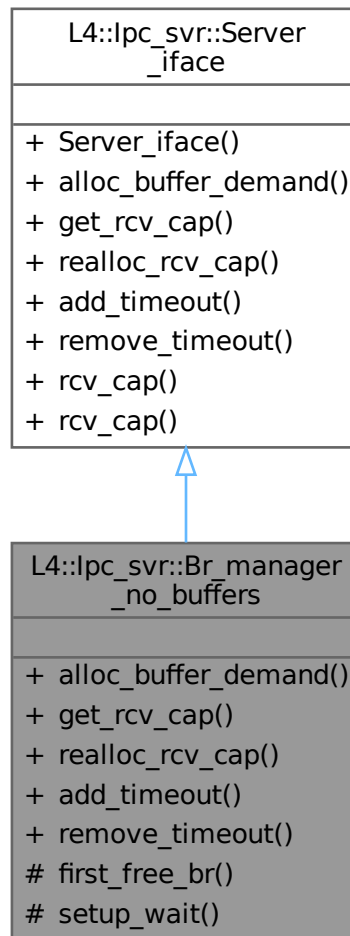
Empty implementation of [Server_iface](#).

```
#include <ipc_server_loop>
```


Inheritance diagram for L4::lpc_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.

15.153.1 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](#).

15.153.2 Member Function Documentation

15.153.2.1 alloc_buffer_demand()

```
int L4::lpc_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 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.

Returns

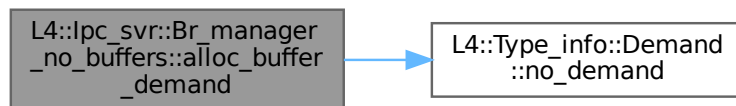
success (0) if demand is empty, -L4_ENOMEM else.

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:

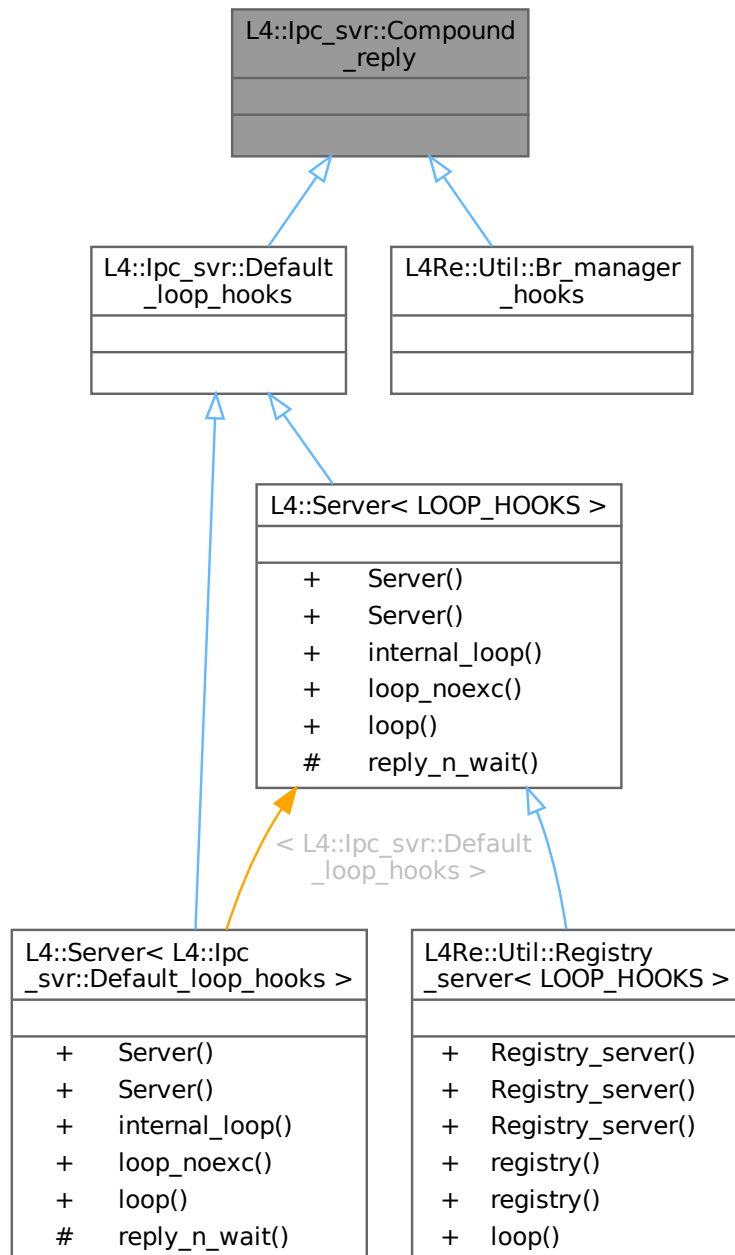
- [l4/sys/cxx/ipc_server_loop](#)

15.154 L4::lpc_svr::Compound_reply Struct Reference

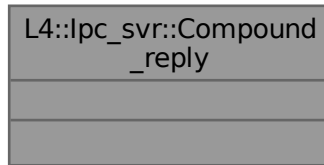
Mix in for LOOP_HOOKS to always use compound reply and wait.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc_svr::Compound_reply:



Collaboration diagram for L4::lpc_svr::Compound_reply:



15.154.1 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:

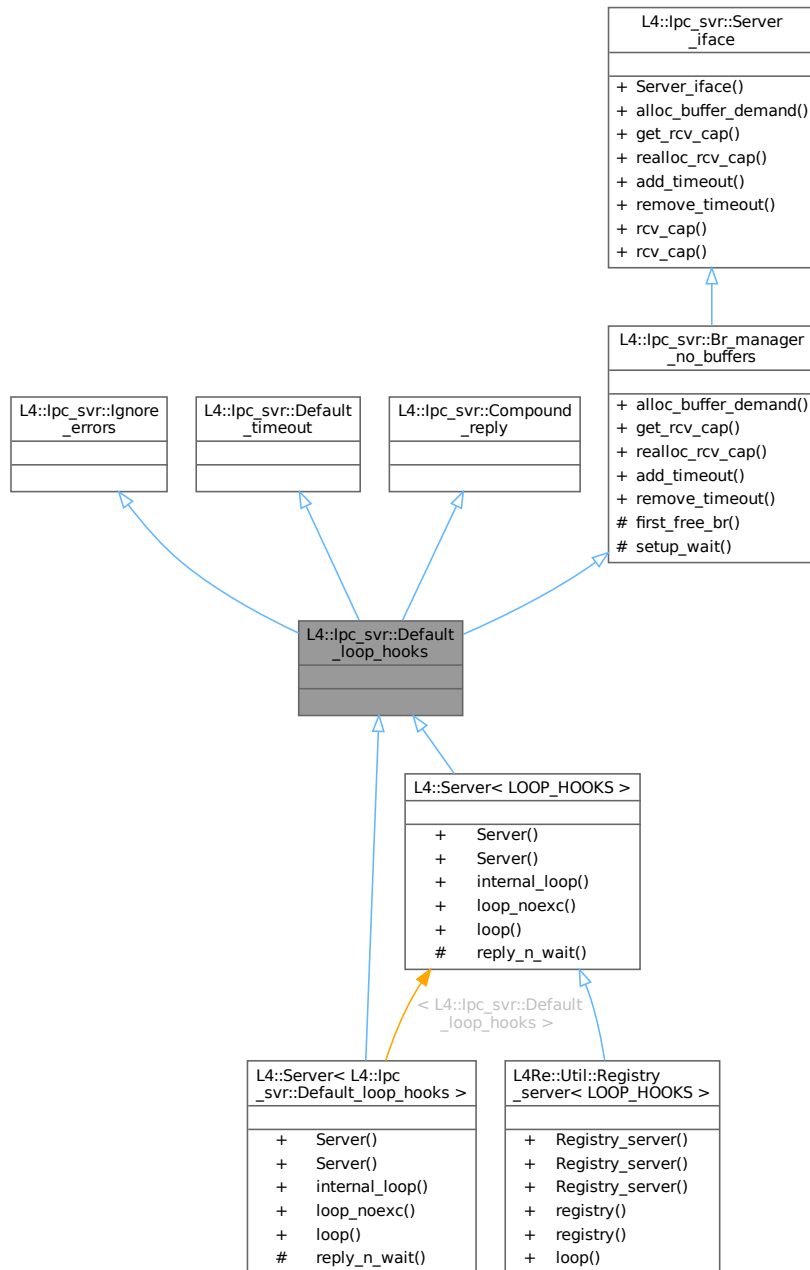
- `l4/sys/cxx/ipc_server_loop`

15.155 L4::lpc_svr::Default_loop_hooks Struct Reference

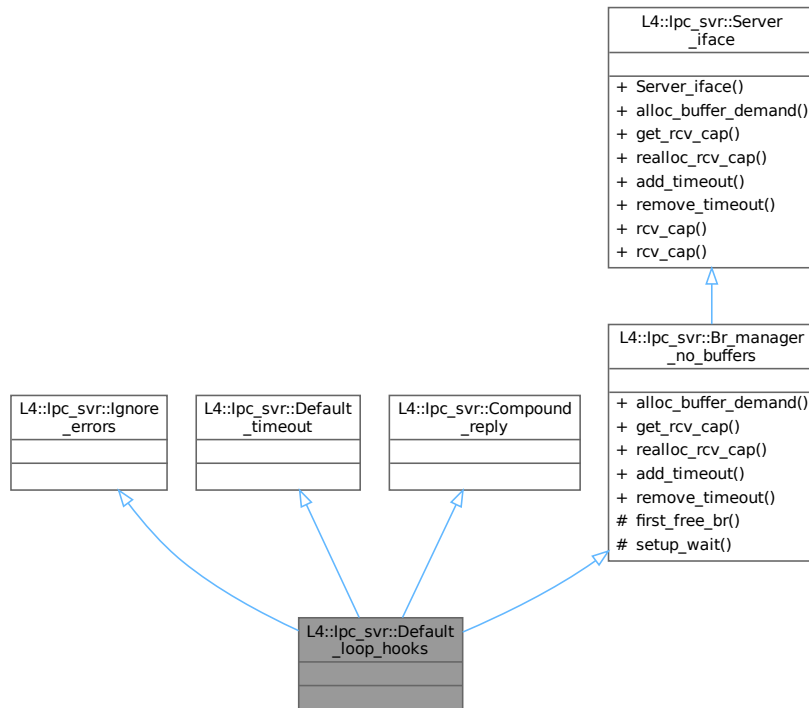
Default `LOOP_HOOKS`.

```
#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<>).

15.155.1 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:

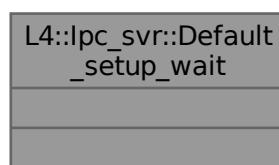
- l4/sys/cxx/ipc_server_loop

15.156 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:



15.156.1 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:

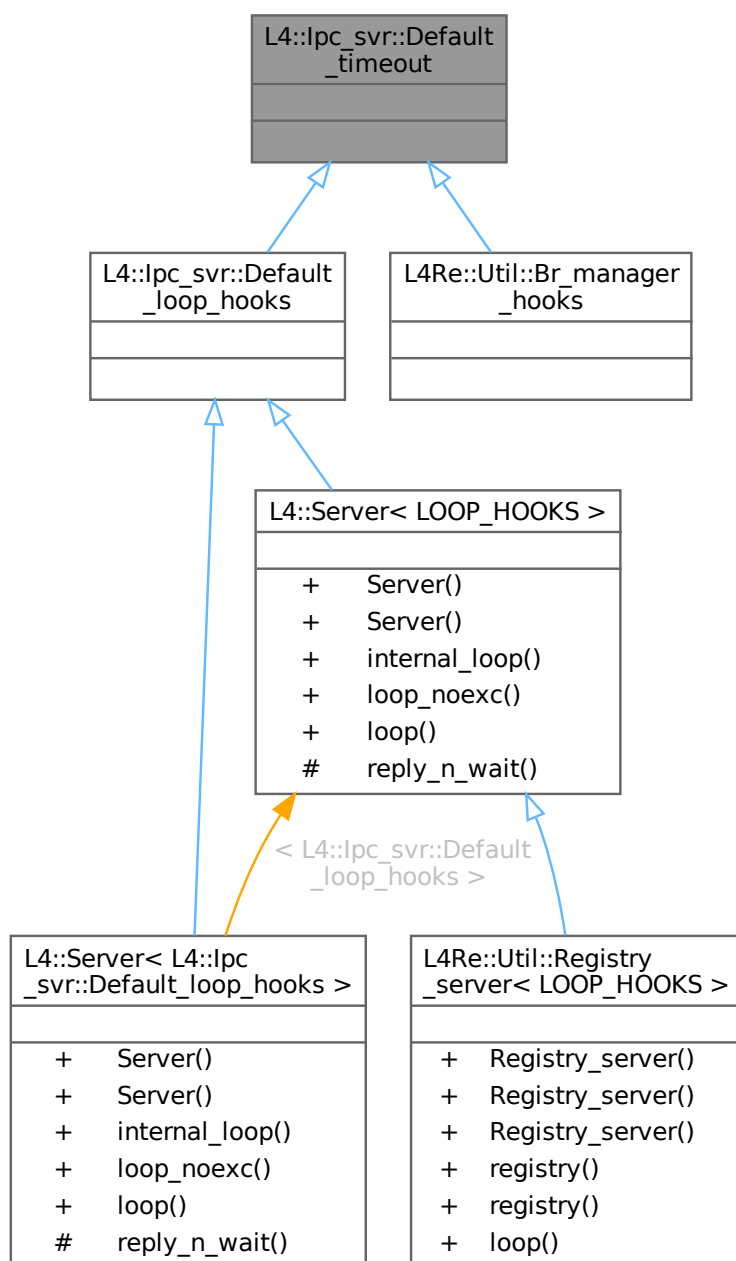
- l4/sys/cxx/ipc_server_loop

15.157 L4::lpc_svr::Default_timeout Struct Reference

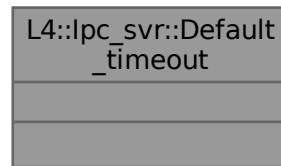
Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc_svr::Default_timeout:



Collaboration diagram for L4::lpc_svr::Default_timeout:



15.157.1 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:

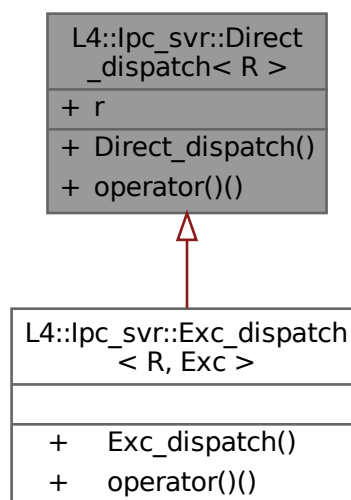
- l4/sys/cxx/ipc_server_loop

15.158 L4::lpc_svr::Direct_dispatch< R > Struct Template Reference

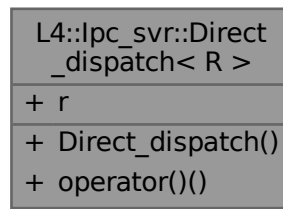
Direct dispatch helper, for forwarding dispatch calls to a registry *R*.

```
#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

15.158.1 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:

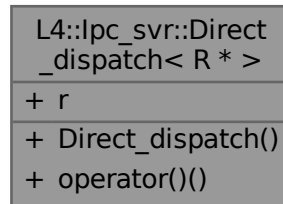
- l4/sys/cxx/ipc_server_loop

15.159 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

15.159.1 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:

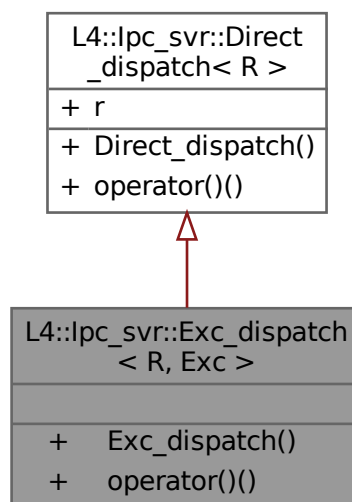
- l4/sys/cxx/ipc_server_loop

15.160 L4::lpc_svr::Exc_dispatch< R, Exc > Struct Template Reference

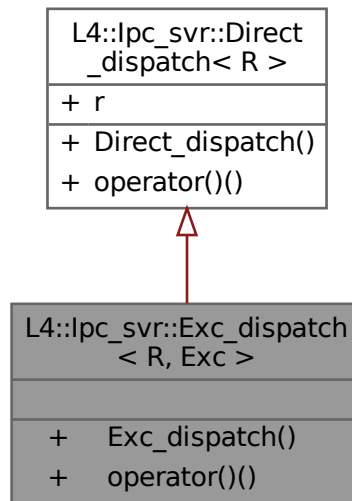
Dispatch helper wrapping try {} catch {} around the dispatch call.

```
#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.

15.160.1 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:

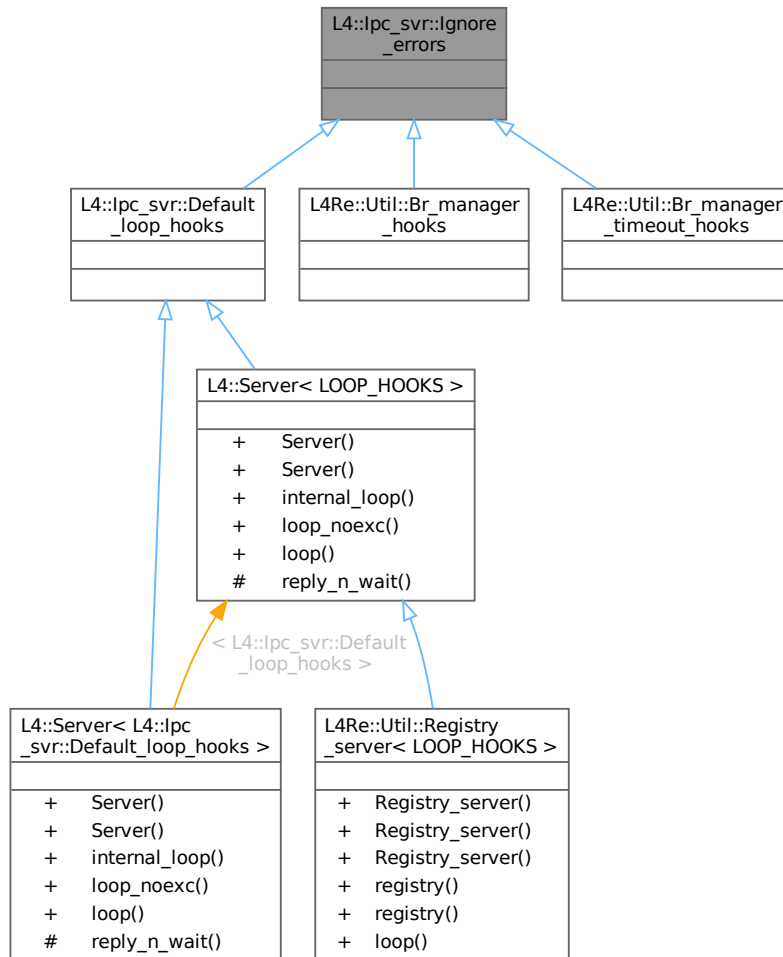
- l4/sys/cxx/ipc_server_loop

15.161 L4::lpc_svr::Ignore_errors Struct Reference

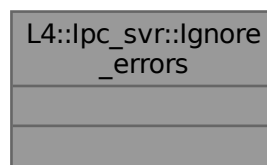
Mix in for LOOP_HOOKS to ignore IPC errors.

```
#include <ipc_server_loop>
```

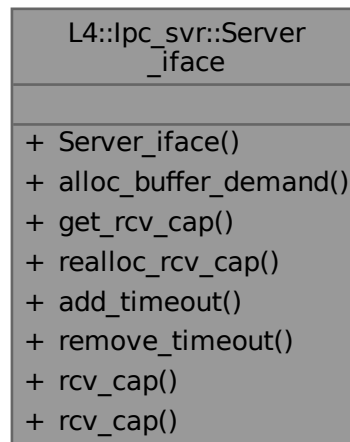
Inheritance diagram for L4::lpc_svr::Ignore_errors:



Collaboration diagram for L4::lpc_svr::Ignore_errors:



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.

15.162.1 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](#).

15.162.2 Member Function Documentation

15.162.2.1 `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](#)

15.162.2.2 `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](#).

15.162.2.3 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 alloc_buffer_demand()).
--------------	---

Precondition

$0 \leq \text{index} < \text{caps}$ registered with [alloc_buffer_demand\(\)](#)

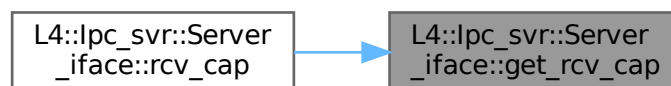
Returns

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:



15.162.2.4 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 alloc_buffer_demand() .)
--------------	--

Precondition

$0 \leq \text{index} < \text{caps}$ registered with [alloc_buffer_demand\(\)](#)

Returns

Capability slot currently allocated to the given receive buffer.

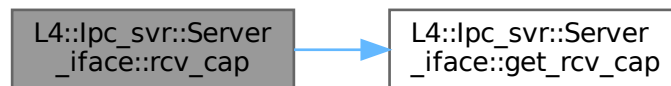
Note

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:

**15.162.2.5 rcv_cap() [2/2]**

```

L4::Cap< void > L4::Ipc_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 alloc_buffer_demand() .)
--------------	--

Returns

Capability slot currently allocated to the given capability buffer.

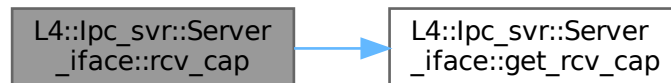
Note

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:

**15.162.2.6 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 alloc_buffer_demand()).
--------------	---

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](#).

15.162.2.7 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:

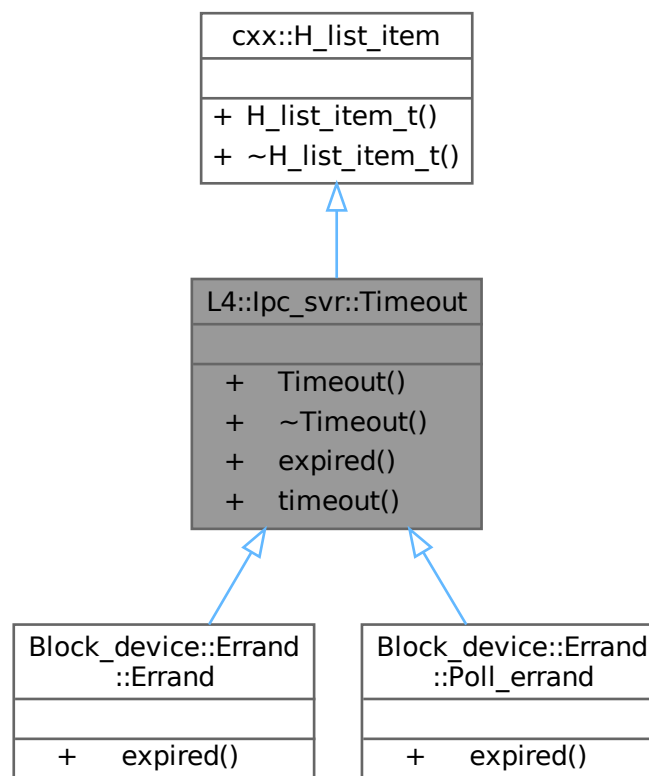
- [l4/sys/cxx/ipc_epiface](#)

15.163 L4::lpc_svr::Timeout Class Reference

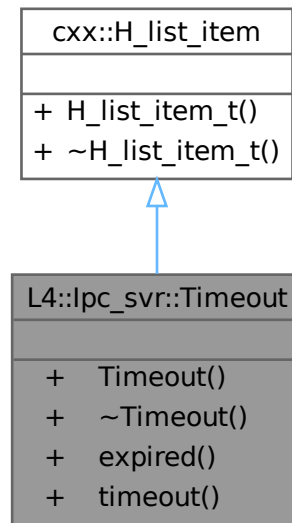
Callback interface for [Timeout_queue](#).

```
#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.

15.163.1 Detailed Description

Callback interface for `Timeout_queue`.

Definition at line 28 of file `ipc_timeout_queue`.

15.163.2 Member Function Documentation

15.163.2.1 expired()

```
virtual void L4::Ipc_svr::Timeout::expired ( ) [pure virtual]
```

callback function to be called when timeout happened

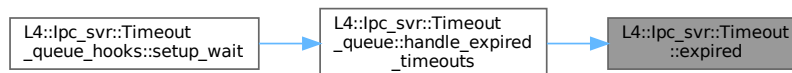
Note

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::ipc_svr::Timeout_queue::handle_expired_timeouts\(\)](#).

Here is the caller graph for this function:



15.163.2.2 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:

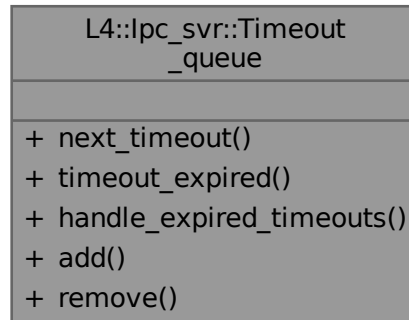
- [l4/cxx/ipc_timeout_queue](#)

15.164 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.

15.164.1 Detailed Description

[Timeout](#) queue to be used in l4re server loop.

Definition at line 65 of file [ipc_timeout_queue](#).

15.164.2 Member Function Documentation

15.164.2.1 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

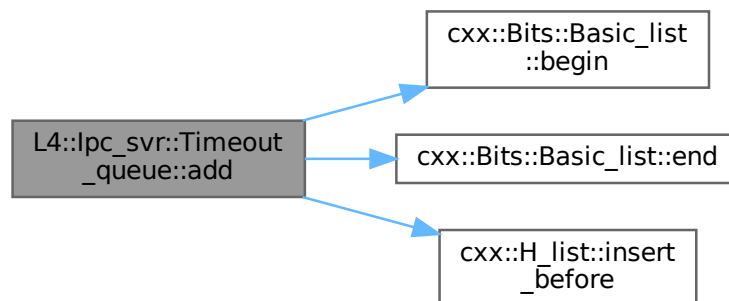
timeout must not be in any queue already

Definition at line 121 of file [ipc_timeout_queue](#).

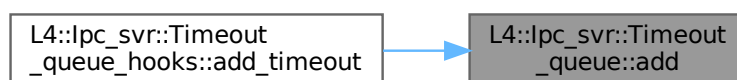
References [cxx::Bits::Basic_list< POLICY >::begin\(\)](#), [cxx::Bits::Basic_list< POLICY >::end\(\)](#), and [cxx::H_list< T, POLICY >::insert_](#)

Referenced by [L4::ipc_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:



15.164.2.2 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

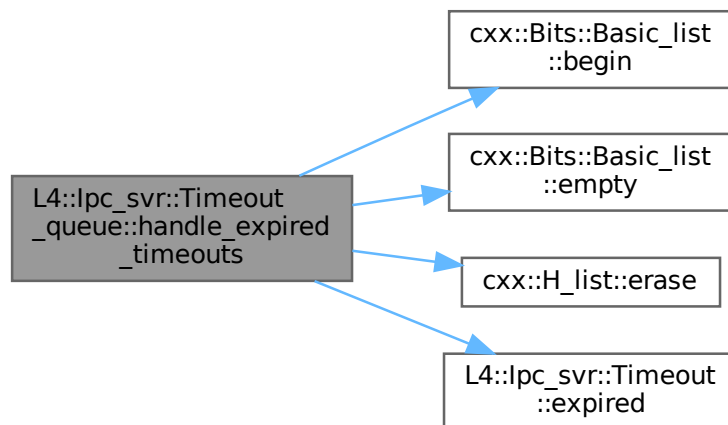
<i>now</i>	the current time.
------------	-------------------

Definition at line 101 of file [ipc_timeout_queue](#).

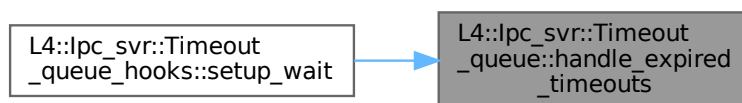
References [cxx::Bits::Basic_list< POLICY >::begin\(\)](#), [cxx::Bits::Basic_list< POLICY >::empty\(\)](#), [cxx::H_list< T, POLICY >::erase\(\)](#), and [L4::lpc_svr::Timeout::expired\(\)](#).

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:



15.164.2.3 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](#).

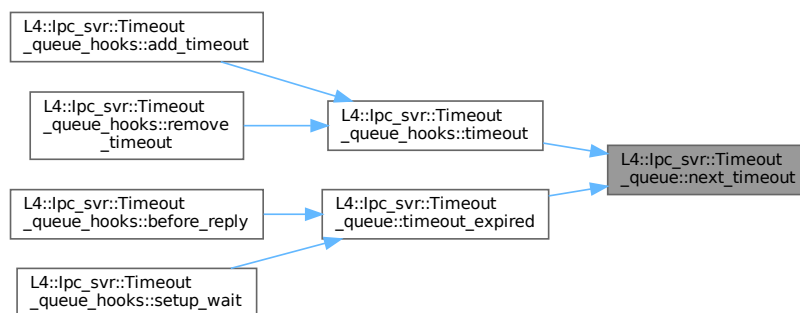
References [cxx::Bits::Basic_list< POLICY >::front\(\)](#).

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:



15.164.2.4 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:



15.164.2.5 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

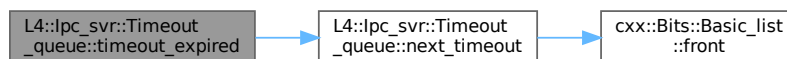
<code>true</code>	There is at least one expired timeout in the queue.
<code>false</code>	No expired timeout in the queue.

Definition at line 91 of file [ipc_timeout_queue](#).

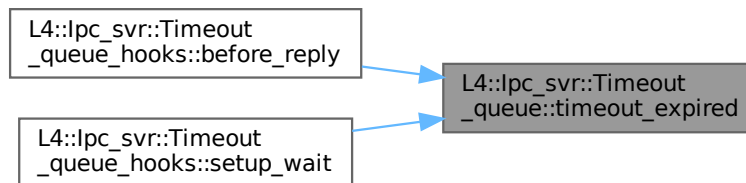
References [next_timeout\(\)](#).

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:

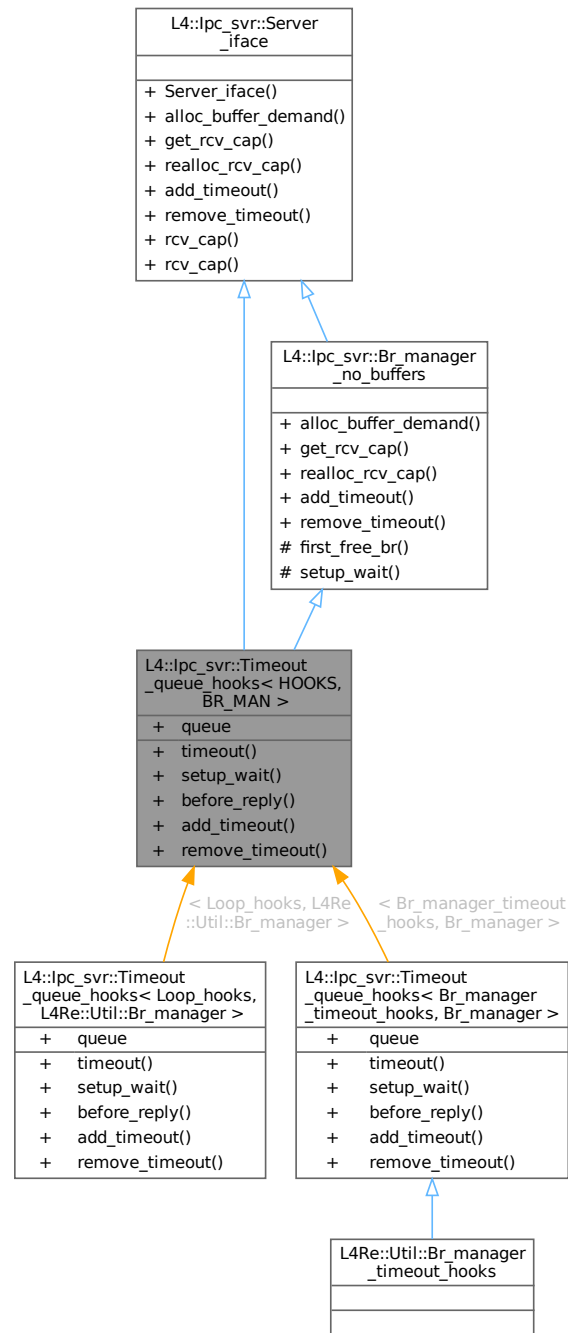
- `l4/cxx/ipc_timeout_queue`

15.165 L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN > Class Template Reference

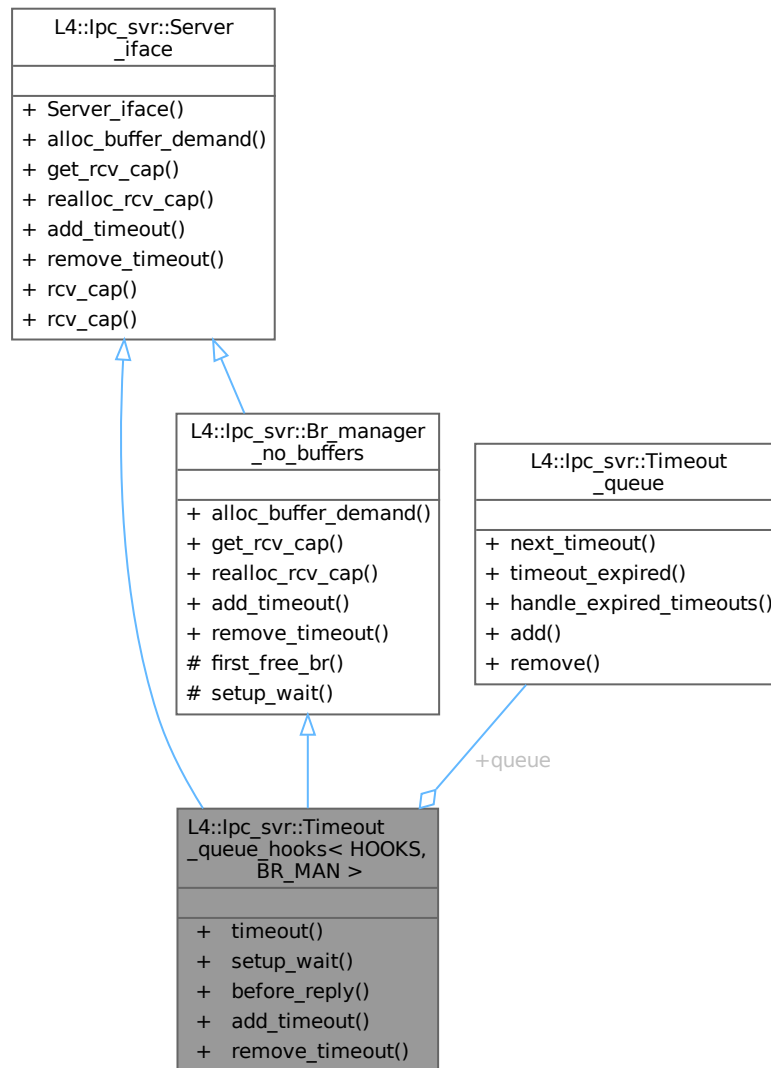
Loop hooks mixin for integrating a timeout queue into the server loop.

```
#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<>).

15.165.1 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 <code>Timeout_queue_hooks<></code> and provide the functions <code>now()</code> 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 L4Re::Util::Br_manager or L4::lpc_svr::Br_manager_no_buffers .

Definition at line 160 of file [ipc_timeout_queue](#).

15.165.2 Member Function Documentation

15.165.2.1 `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

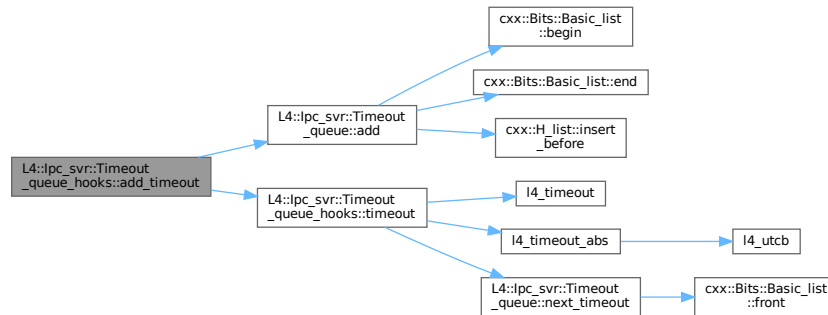
The timeout is automatically dequeued before the [Timeout::expired\(\)](#) function is called

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:



15.165.2.2 remove_timeout()

```

template<typename HOOKS , typename BR_MAN = Br_manager_no_buffers>
int L4::lpc_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.

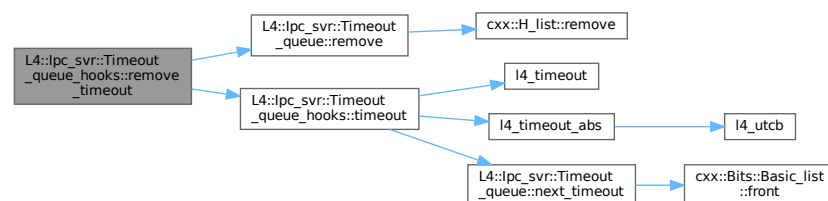
in [Timeout::expired\(\)](#) the timeout is already dequeued!

Implements [L4::lpc_svr::Server_iface](#).

Definition at line 225 of file [ipc_timeout_queue](#).

References [L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::queue](#), [L4::lpc_svr::Timeout_queue::remove\(\)](#), and [L4::lpc_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:

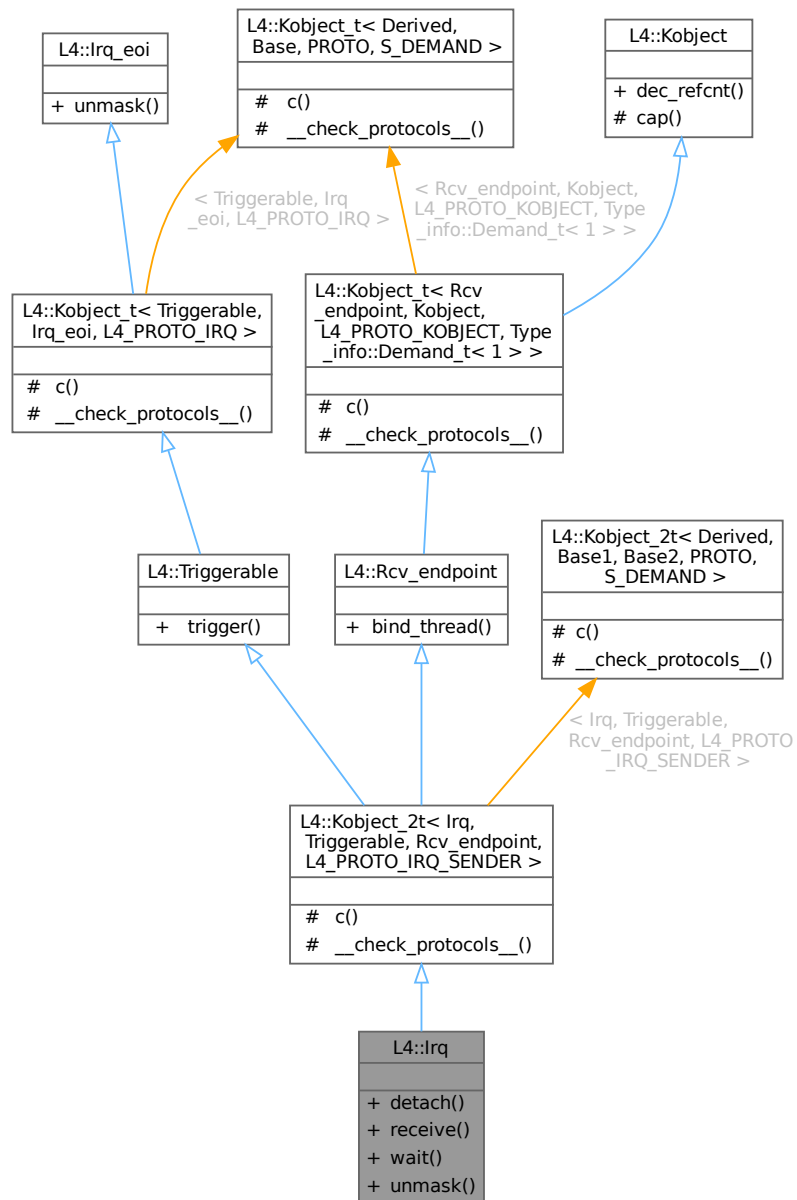
- [I4/cxx/ipc_timeout_queue](#)

15.166 L4::Irq Class Reference

C++ [Irq](#) interface, see [IRQs](#) for the C interface.

```
#include <irq>
```

Inheritance diagram for L4::Irq:





- Generated for L4Re by Doxygen

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.

15.166.1 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](#).

15.166.2 Member Function Documentation

15.166.2.1 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 l4_utcb .
-------------	--

Returns

Syscall return tag

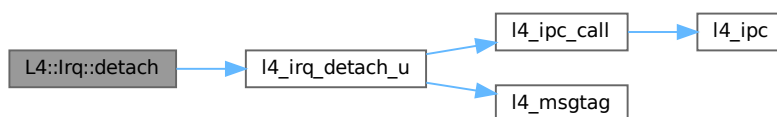
Return values

<i>-L4_EPERM</i>	No L4_CAP_FPAGE_S rights on the capability used to invoke this operation.
<i>-L4_EBUSY</i>	A detach operation on this IRQ object by another thread was in progress.

Definition at line 148 of file [irq](#).

References [l4_irq_detach_u\(\)](#).

Here is the call graph for this function:



15.166.2.2 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 l4_utcb .

Returns

Syscall return tag

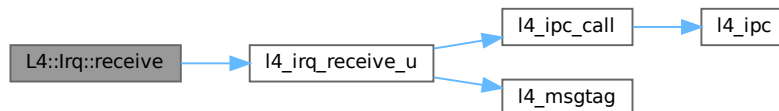
Note

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:

**15.166.2.3 unmask()**

```
l4_msgtag_t L4::Irq::unmask (
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

Unmask IRQ.

Parameters

<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb .
-------------	--

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.

Deprecated Use [L4::Irq_eoi::unmask\(\)](#)

Definition at line 195 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:



15.166.2.4 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 l4_utcb .

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:

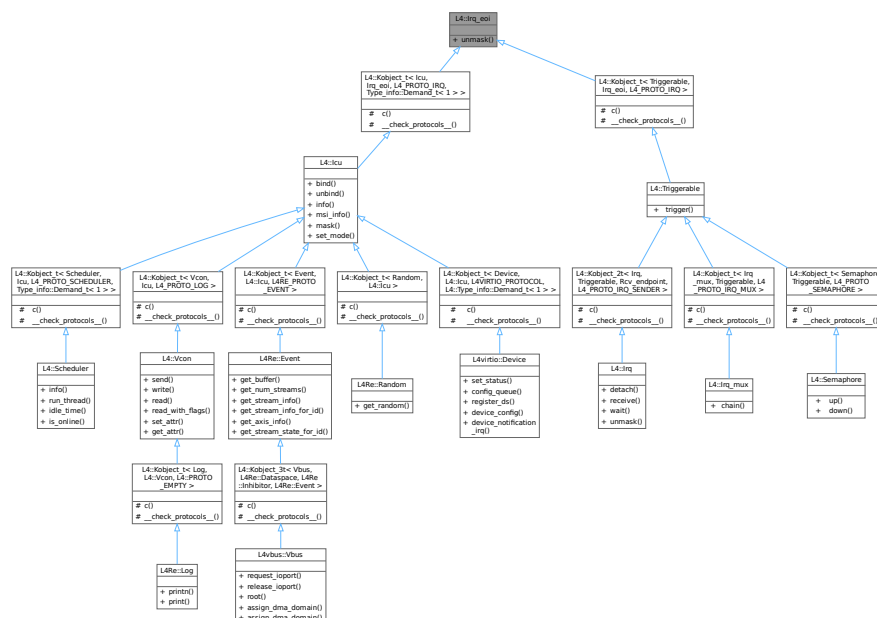
- [l4/sys/irq](#)

15.167 L4::Irq_eoi Class Reference

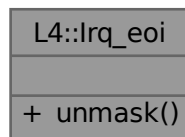
Interface for sending an unmask message to an object.

```
#include <irq>
```

Inheritance diagram for L4::Irq_eoi:



Collaboration diagram for L4::Irq_eoi:



Public Member Functions

- [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.

15.167.1 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](#).

15.167.2 Member Function Documentation

15.167.2.1 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 l4_utcb .

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:

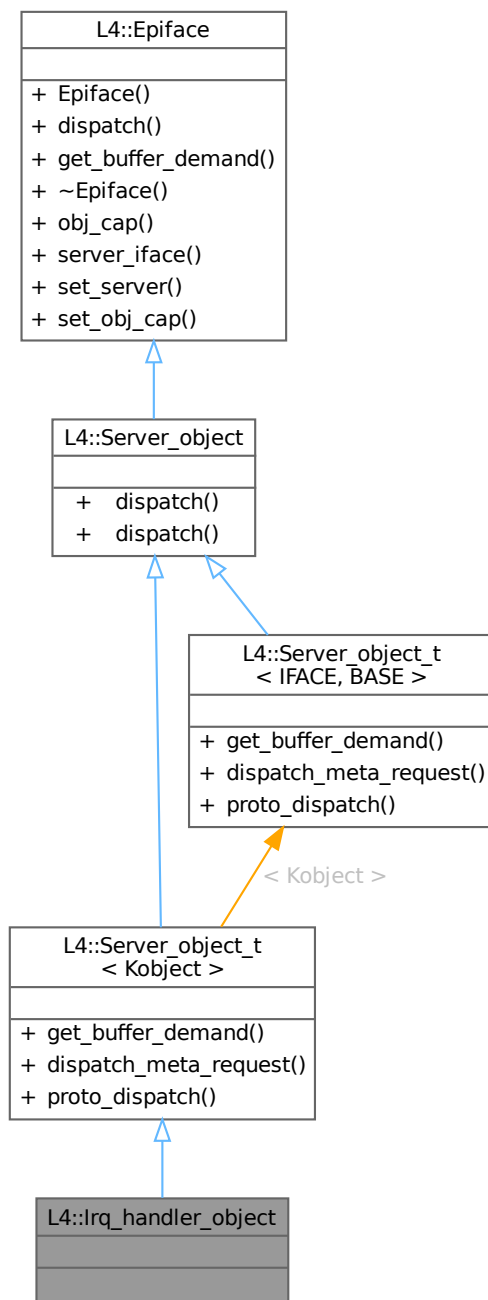
- [l4/sys/irq](#)

15.168 L4::lirq_handler_object Struct Reference

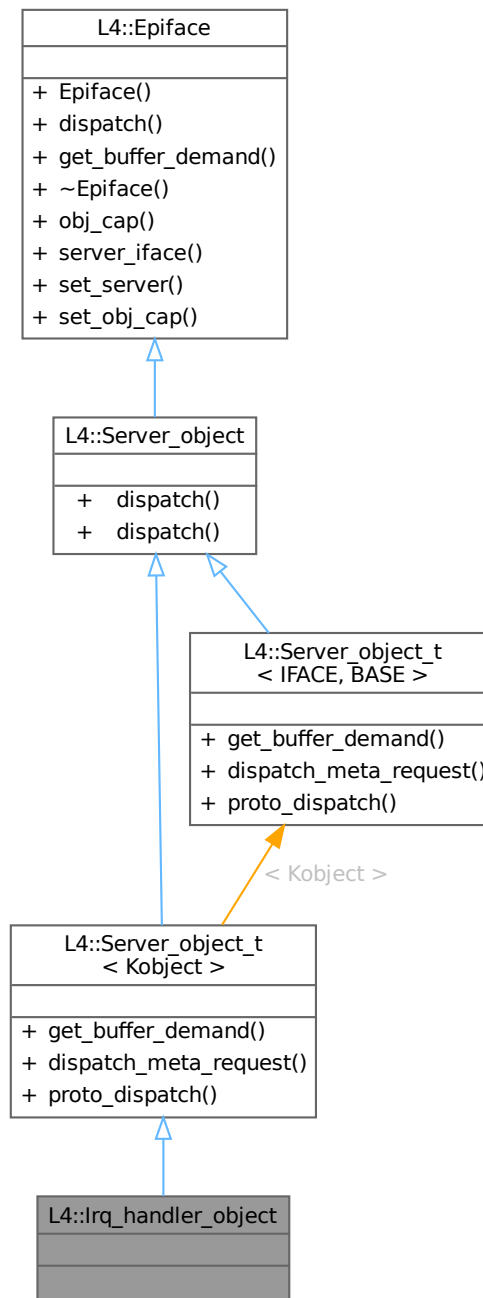
[Server](#) object base class for handling IRQ messages.

```
#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.

15.168.1 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:

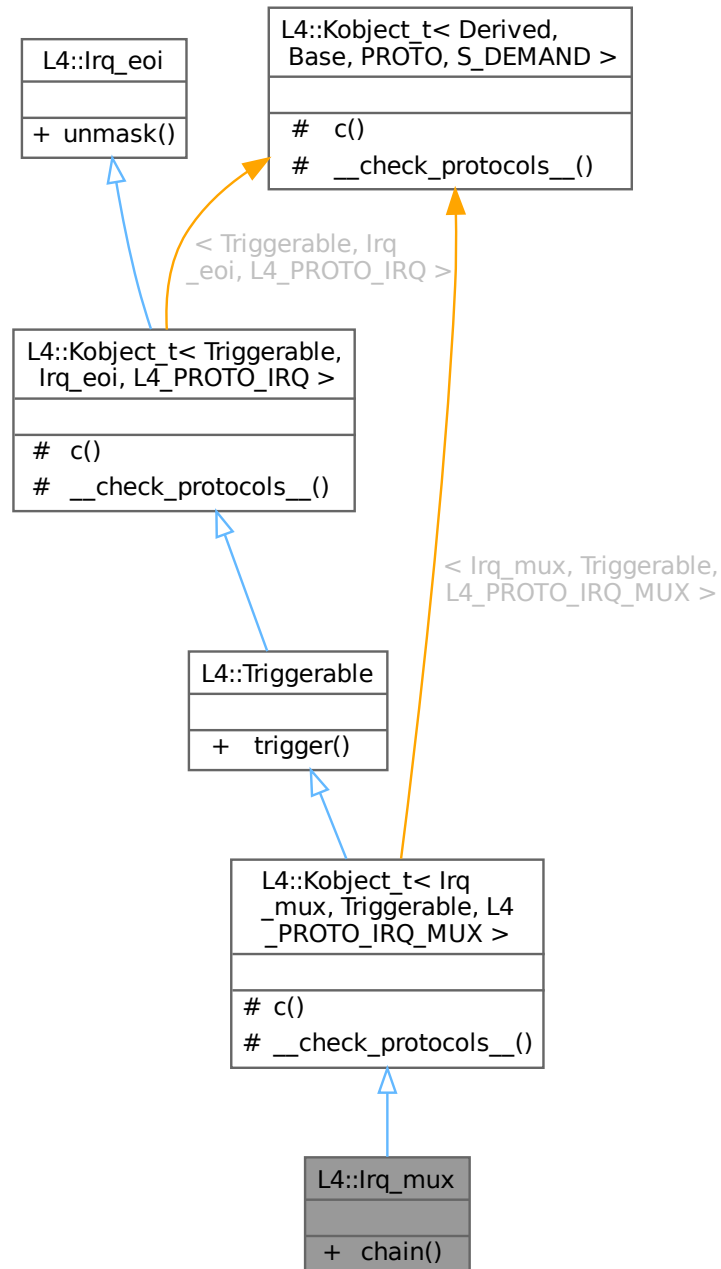
- [l4/cxx/ipc_server](#)

15.169 L4::Irq_mux Struct Reference

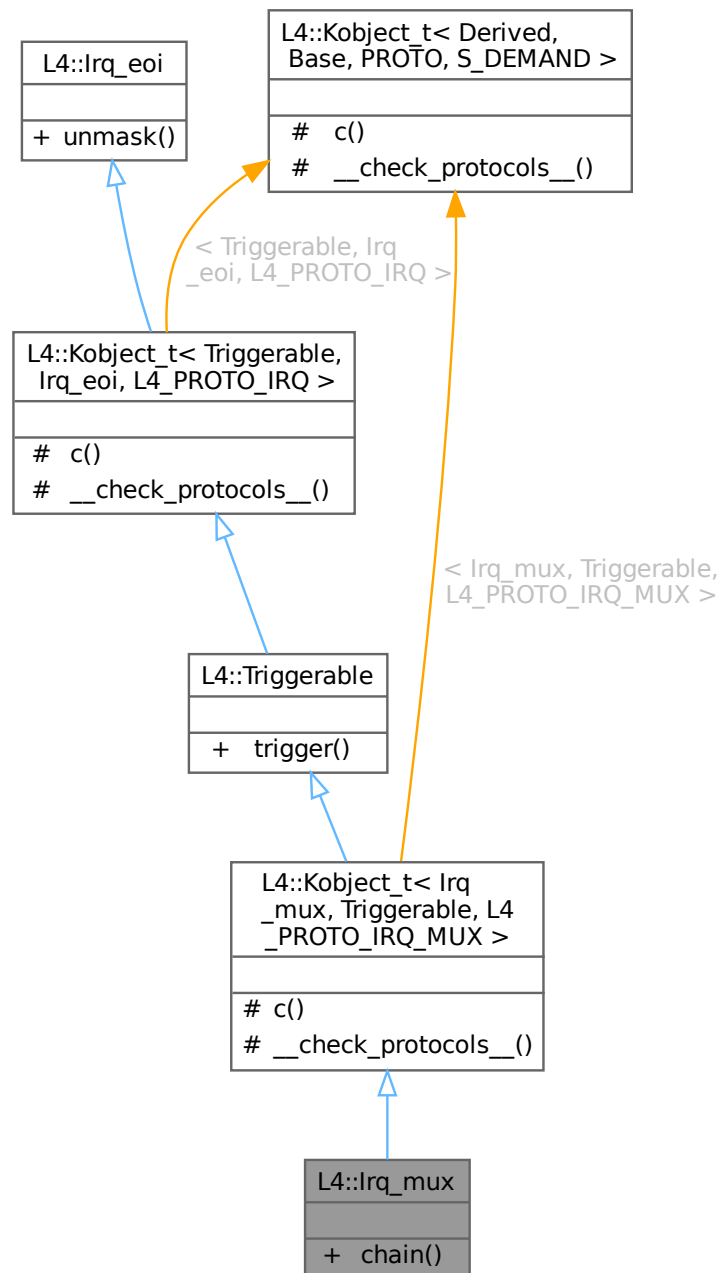
IRQ multiplexer for shared IRQs.

```
#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.

15.169.1 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 [214](#) of file [irq](#).

15.169.2 Member Function Documentation

15.169.2.1 `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 l4_utcb .

Returns

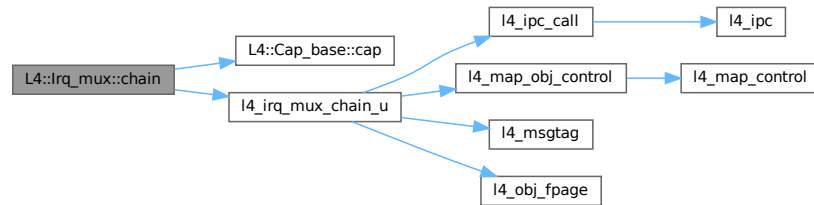
Syscall return tag

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 [229](#) 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:

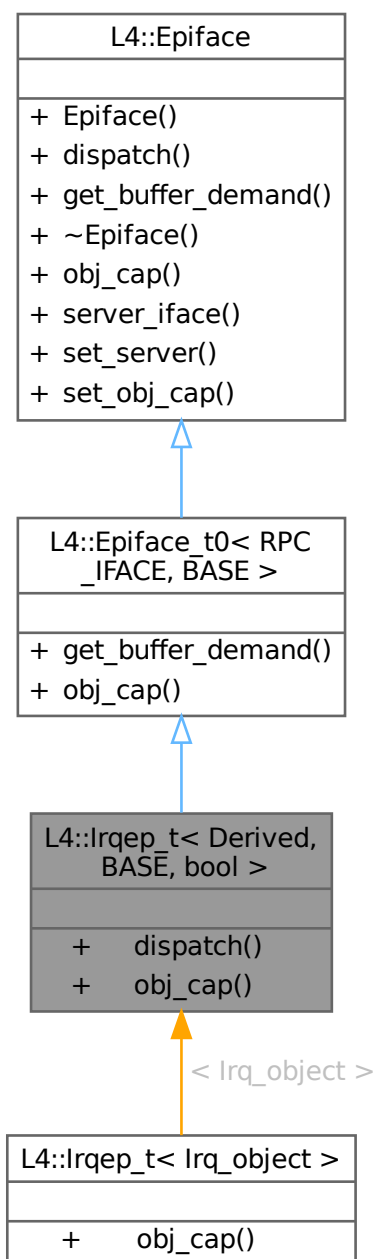
- [l4/sys/irq](#)

15.170 L4::lrqep_t< Derived, BASE, bool > Struct Template Reference

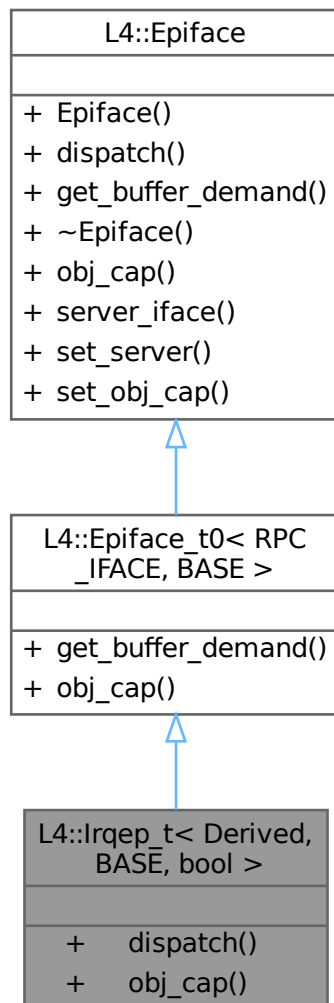
[Epiface](#) implementation for interrupt handlers.

```
#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.

15.170.1 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>	Irq handler implementation class. The class must provide a single function <code>handle_irq()</code> .
<i>BASE</i>	Base Epiface class.

Definition at line 293 of file `ipc_epiface`.

15.170.2 Member Function Documentation

15.170.2.1 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><0</code>	Error, reply with error code.
<code>>=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:



15.170.2.2 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](#)

15.171 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"> + Mem_desc() + start() + end() + size() + type() + sub_type() + is_virtual() + set() + first() + count() + count() + all() + all()

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) 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.
- void [set](#) (unsigned long [start](#), unsigned long [end](#), [Mem_type](#) t, unsigned char st=0, bool virt=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.

15.171.1 Detailed Description

Memory descriptors stored in the kernel interface page.

Include File

```
#include <l4/sys/kip>
```

Definition at line 53 of file [kip](#).

15.171.2 Member Enumeration Documentation

15.171.2.1 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](#).

15.171.2.2 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](#).

15.171.2.3 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](#).

15.171.3 Constructor & Destructor Documentation

15.171.3.1 Mem_desc()

```
L4::Kip::Mem_desc::Mem_desc (
    unsigned long start,
    unsigned long end,
    Mem_type t,
```

```
    unsigned char st = 0,  
    bool virt = false )    [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

Definition at line 175 of file [kip](#).

15.171.4 Member Function Documentation

15.171.4.1 [all\(\)](#) [1/2]

```
static cxx::static\_vector< Mem\_desc > L4::Kip::Mem_desc::all (  
    void * kip )    [inline], [static]
```

Return enumerable list of memory descriptors.

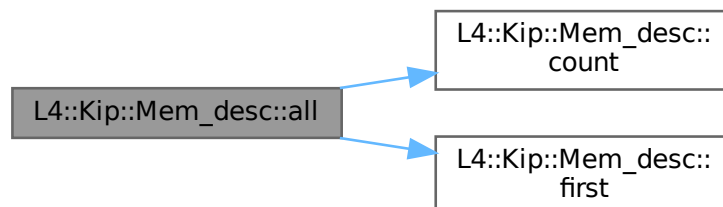
Parameters

<i>kip</i>	Pointer to the kernel info page.
------------	----------------------------------

Definition at line 159 of file [kip](#).

References [count\(\)](#), and [first\(\)](#).

Here is the call graph for this function:



15.171.4.2 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.

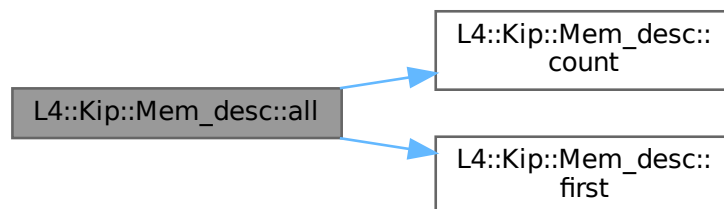
Parameters

<i>kip</i>	Pointer to the kernel info page.
------------	----------------------------------

Definition at line 148 of file [kip](#).

References [count\(\)](#), and [first\(\)](#).

Here is the call graph for this function:



15.171.4.3 count() [1/2]

```
static void L4::Kip::Mem_desc::count (
    void * kip,
    unsigned count ) [inline], [static], [noexcept]
```

Set number of memory descriptors.

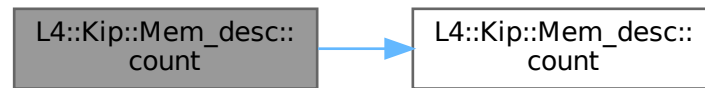
Parameters

<i>kip</i>	Pointer to the kernel info page
<i>count</i>	Number of memory descriptors

Definition at line 137 of file [kip](#).

References [count\(\)](#).

Here is the call graph for this function:



15.171.4.4 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
------------	---------------------------------

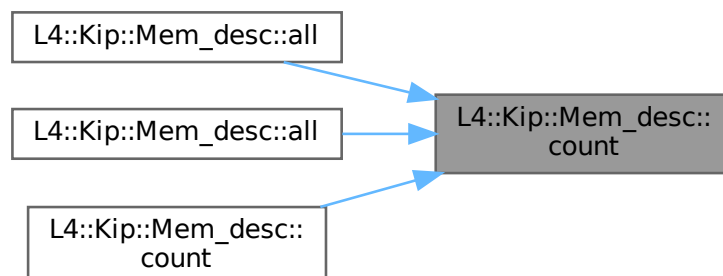
Returns

Number of memory descriptors in the kernel info page.

Definition at line 125 of file [kip](#).

Referenced by [all\(\)](#), [all\(\)](#), and [count\(\)](#).

Here is the caller graph for this function:



15.171.4.5 end()

```
unsigned long L4::Kip::Mem_desc::end ( ) const [inline], [noexcept]
```

Return end address of memory descriptor.

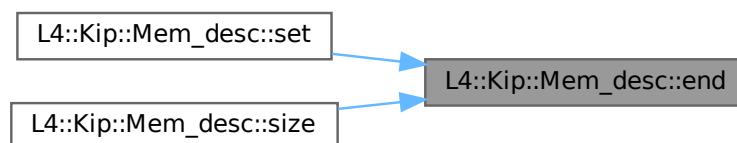
Returns

End address of memory descriptor

Definition at line 193 of file [kip](#).

Referenced by [set\(\)](#), and [size\(\)](#).

Here is the caller graph for this function:



15.171.4.6 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
------------	---------------------------------

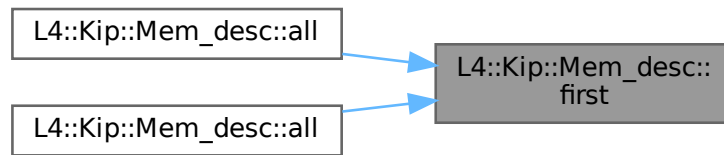
Returns

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:



15.171.4.7 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 222 of file [kip](#).

15.171.4.8 set()

```
void L4::Kip::Mem_desc::set (
    unsigned long start,
    unsigned long end,
    Mem_type t,
    unsigned char st = 0,
    bool virt = false ) [inline], [noexcept]
```

Set values of a memory descriptor.

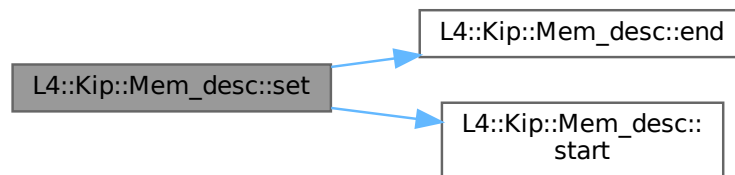
Parameters

<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

Definition at line 233 of file [kip](#).

References [end\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



15.171.4.9 size()

```
unsigned long L4::Kip::Mem_desc::size ( ) const [inline], [noexcept]
```

Return size of region described by the memory descriptor.

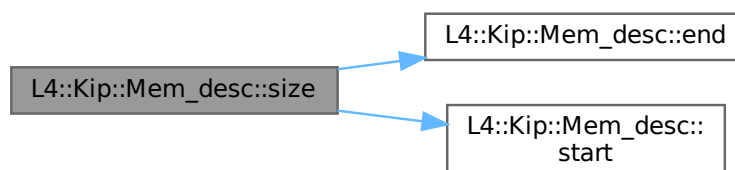
Returns

Size of the region described by the memory descriptor

Definition at line 200 of file [kip](#).

References [end\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



15.171.4.10 start()

```
unsigned long L4::Kip::Mem_desc::start ( ) const [inline], [noexcept]
```

Return start address of memory descriptor.

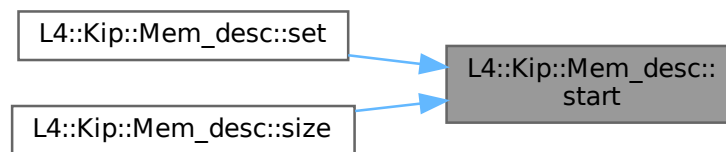
Returns

Start address of memory descriptor

Definition at line 186 of file [kip](#).

Referenced by [set\(\)](#), and [size\(\)](#).

Here is the caller graph for this function:

**15.171.4.11 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 214 of file [kip](#).

15.171.4.12 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 207 of file [kip](#).

The documentation for this class was generated from the following file:

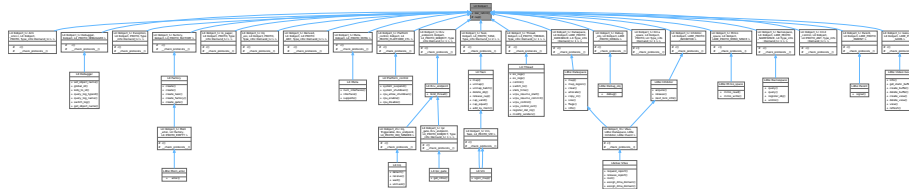
- [l4/sys/kip](#)

15.172 L4::Kobject Class Reference

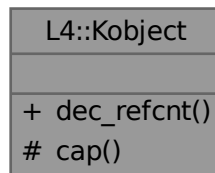
Base class for all kinds of kernel objects and remote objects, referenced by capabilities.

```
#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.

15.172.1 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](#).

15.172.2 Member Function Documentation

15.172.2.1 cap()

```
l4_cap_idx_t L4::Kobject::cap ( ) const [inline], [protected], [noexcept]
```

Return capability selector.

Returns

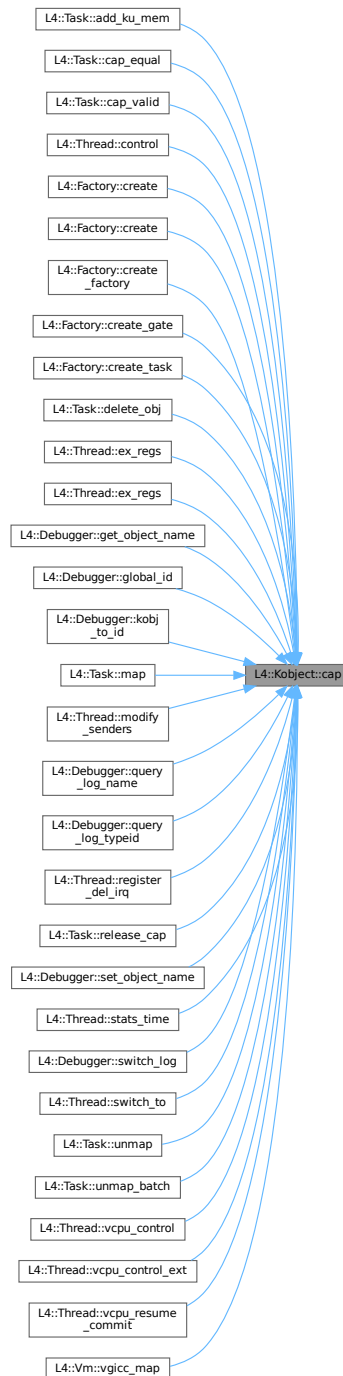
Capability selector.

This method is for derived classes to gain access to the actual capability selector.

Definition at line 79 of file [kobject](#).

Referenced by [L4::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:



15.172.2.2 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 l4_utcb .

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:

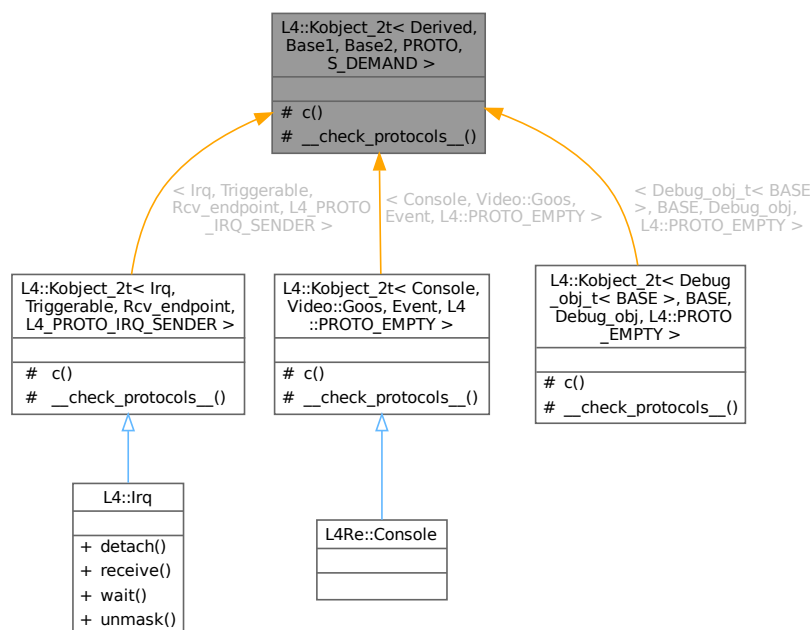
- [l4/sys/kobject](#)

15.173 L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND > Class Template Reference

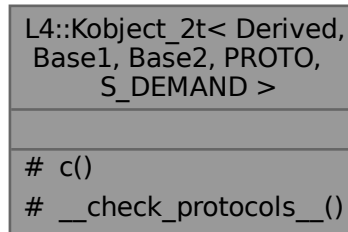
Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject_t](#)).

```
#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.

15.173.1 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 L4::Type_info::Demand_t . 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 837 of file `__typeinfo.h`.

15.173.2 Member Typedef Documentation

15.173.2.1 `__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 843 of file `__typeinfo.h`.

15.173.2.2 `__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 851 of file `__typeinfo.h`.

15.173.2.3 `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 841 of file `__typeinfo.h`.

15.173.3 Member Function Documentation

15.173.3.1 `__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 854 of file `__typeinfo.h`.

15.173.3.2 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 873 of file [__typeinfo.h](#).

The documentation for this class was generated from the following file:

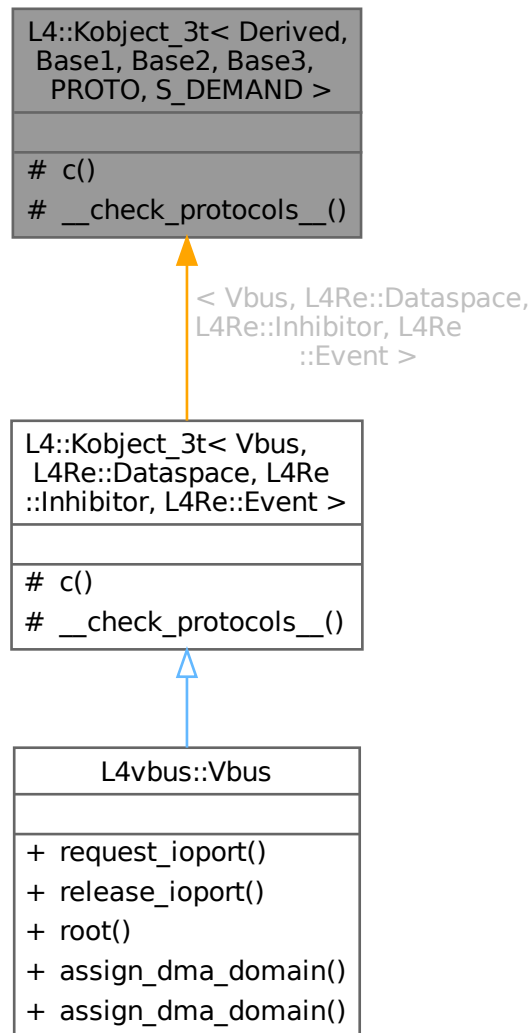
- [l4/sys/__typeinfo.h](#)

15.174 L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND > Struct Template Reference

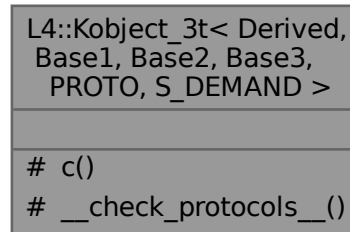
Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject_t](#)).

```
#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.

15.174.1 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 L4::Type_info::Demand_t . 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 940 of file [__typeinfo.h](#).

15.174.2 Member Typedef Documentation

15.174.2.1 __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 946 of file [__typeinfo.h](#).

15.174.2.2 __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 957 of file [__typeinfo.h](#).

15.174.2.3 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 944 of file [__typeinfo.h](#).

15.174.3 Member Function Documentation

15.174.3.1 __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 960 of file [__typeinfo.h](#).

15.174.3.2 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 988 of file [__typeinfo.h](#).

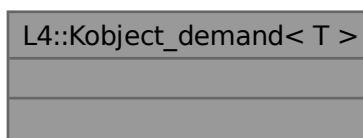
The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

15.175 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 >:

**15.175.1 Detailed Description**

```
template<typename ... T>
struct L4::Kobject_demand< T >
```

Get the combined server-side resource requirements for all type T...

Template Parameters

T	List of IPC interface types for which the combined server-side resource requirements shall be calculated.
----------	---

Definition at line 1041 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

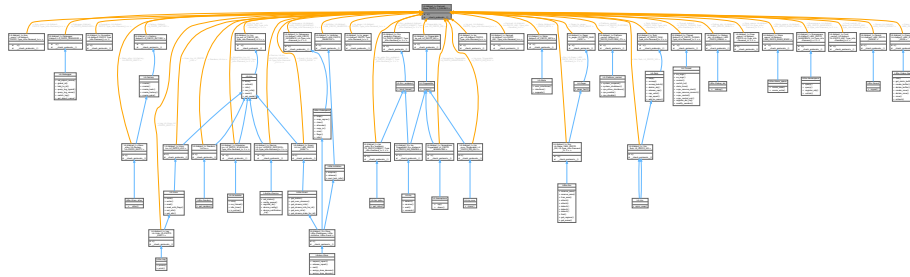
- [l4/sys/__typeinfo.h](#)

15.176 L4::Kobject_t< Derived, Base, PROTO, S_DEMAND > Class Template Reference

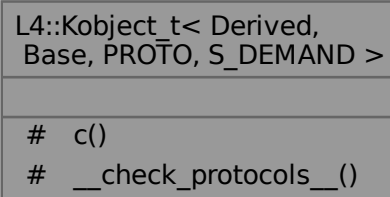
Helper class to create an [L4Re](#) interface class that is derived from a single base class.

```
#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.

15.176.1 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 L4::Type_info::Demand_t . 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 759 of file [__typeinfo.h](#).

The documentation for this class was generated from the following file:

- [l4/sys/__typeinfo.h](#)

15.177 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 >:

L4::Kobject_typeid< T >
<ul style="list-style-type: none"> + <code>id()</code> + <code>demand()</code> + <code>proto_dispatch()</code>

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.

15.177.1 Detailed Description

```
template<typename T>
struct L4::Kobject_typeid< T >
```

[Meta](#) object for handling access to type information of Kobjects.

Template Parameters

T	The data type derived from Kobject , usually using Kobject_t .
-------------------	--

Definition at line 620 of file [__typeinfo.h](#).

15.177.2 Member Typedef Documentation

15.177.2.1 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 632 of file [__typeinfo.h](#).

15.177.3 Member Function Documentation

15.177.3.1 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 649 of file [__typeinfo.h](#).

15.177.3.2 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 640 of file [__typeinfo.h](#).

Referenced by [L4::kobject_typeid\(\)](#).

Here is the caller graph for this function:



15.177.3.3 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

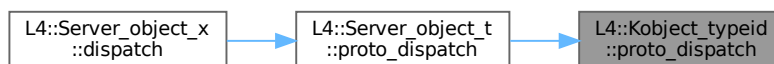
<i>self</i>	The pointer to the server object
<i>proto</i>	The protocol number used by the caller
<i>a1</i>	The second argument passed to self->p_dispatch()
<i>a2</i>	The third argument passed to self->p_dispatch()

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 670 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:

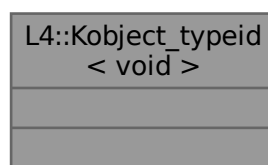
- [l4/sys/__typeinfo.h](#)

15.178 L4::Kobject_typeid< void > Struct Reference

Minimalistic ID for void interface.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Kobject_typeid< void >:



15.178.1 Detailed Description

Minimalistic ID for `void` interface.

Definition at line 677 of file [__typeinfo.h](#).

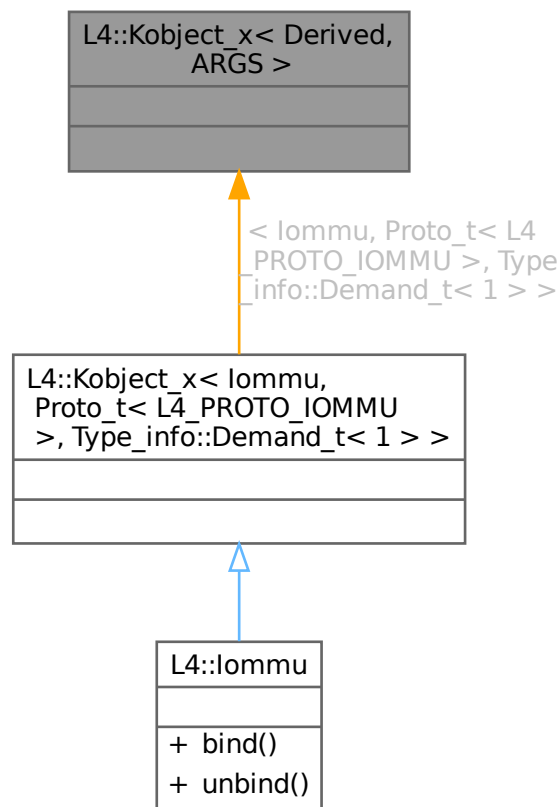
The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

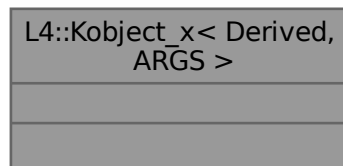
15.179 L4::Kobject_x< Derived, ARGS > Struct Template Reference

Generic [Kobject](#) inheritance template.

Inheritance diagram for L4::Kobject_x< Derived, ARGS >:



Collaboration diagram for L4::Kobject_x< Derived, ARGS >:



15.179.1 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 Kobject_x .
<i>ARGS</i>	An optional protocol number via L4::Proto_t , followed by an optional server-side requirement passed as L4::Type_info::Demand_t , followed by the list of base classes.

Definition at line 1206 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

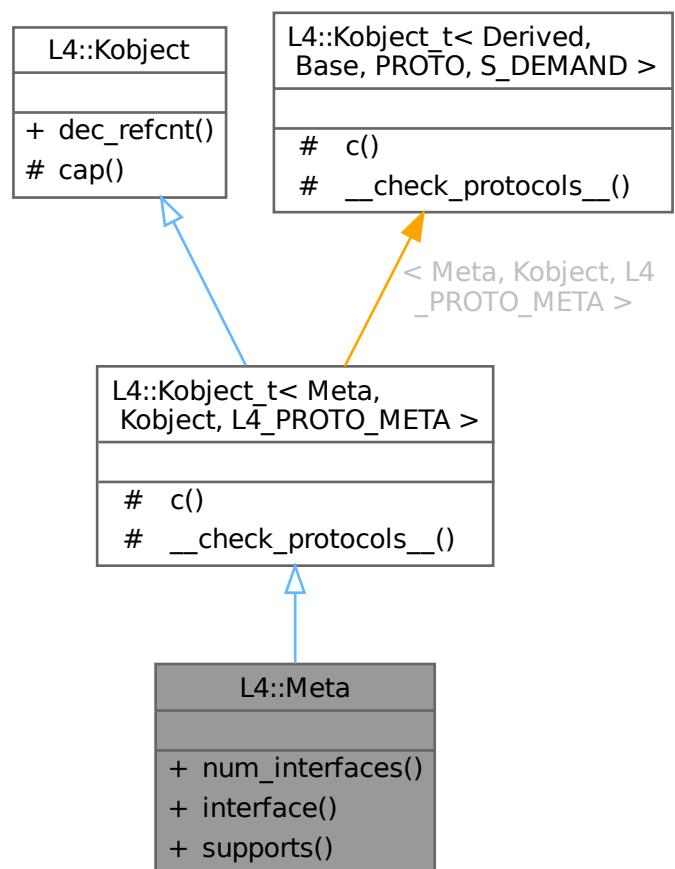
- [l4/sys/__typeinfo.h](#)

15.180 L4::Meta Class Reference

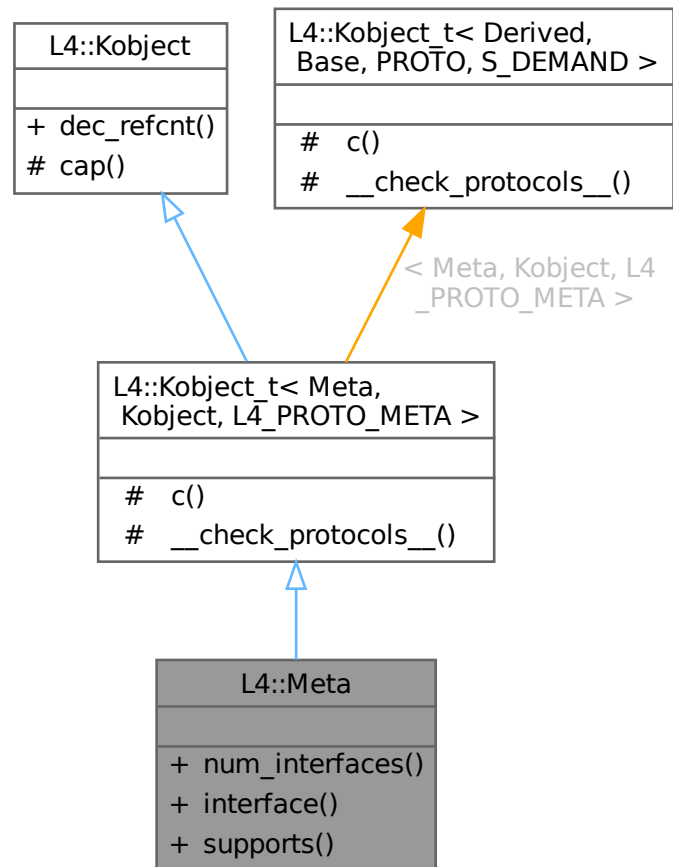
[Meta](#) interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.

```
#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.

15.180.1 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](#).

15.180.2 Member Function Documentation

15.180.2.1 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 ≥ 0 and $<$ the return value of num_interfaces() .
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.

15.180.2.2 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> ≥ 0	The number of supported interfaces.
<i>l4_msgtag_t::label()</i> < 0	Error code of the occurred error.

15.180.2.3 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:

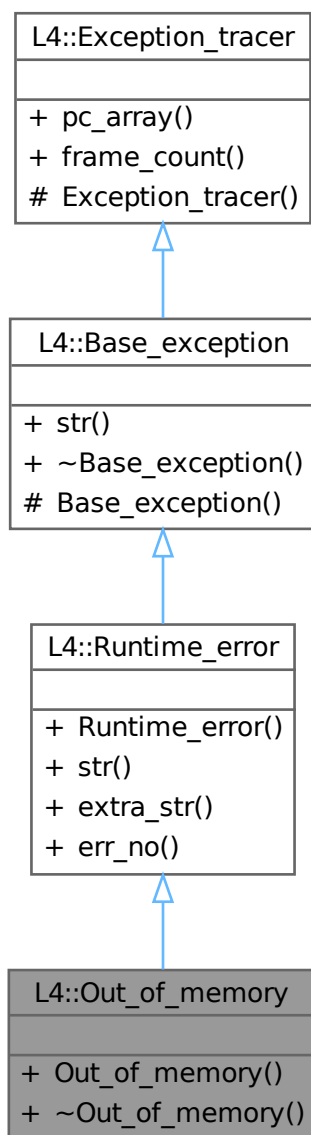
- [l4/sys/meta](#)

15.181 L4::Out_of_memory Class Reference

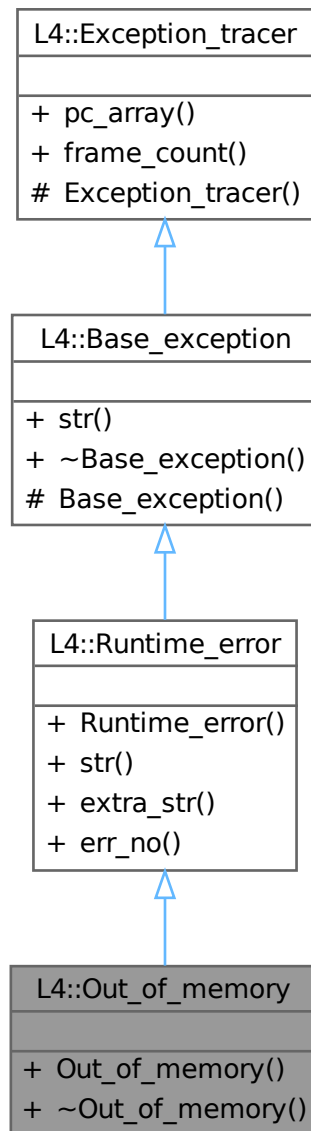
[Exception](#) signalling insufficient memory.

```
#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) throw ()

Create a new [Runtime_error](#).

- char const * **str** () const override throw ()
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** () throw ()
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.

15.181.1 Detailed Description

[Exception](#) signalling insufficient memory.

Definition at line 188 of file [exceptions](#).

The documentation for this class was generated from the following file:

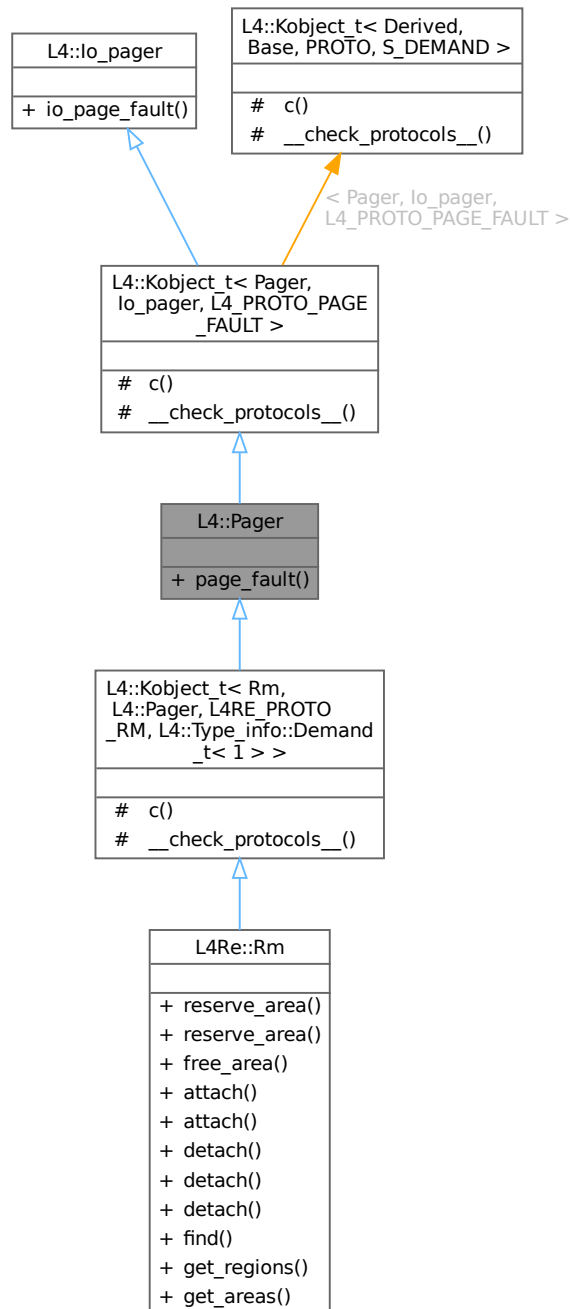
- [l4/cxx/exceptions](#)

15.182 L4::Pager Class Reference

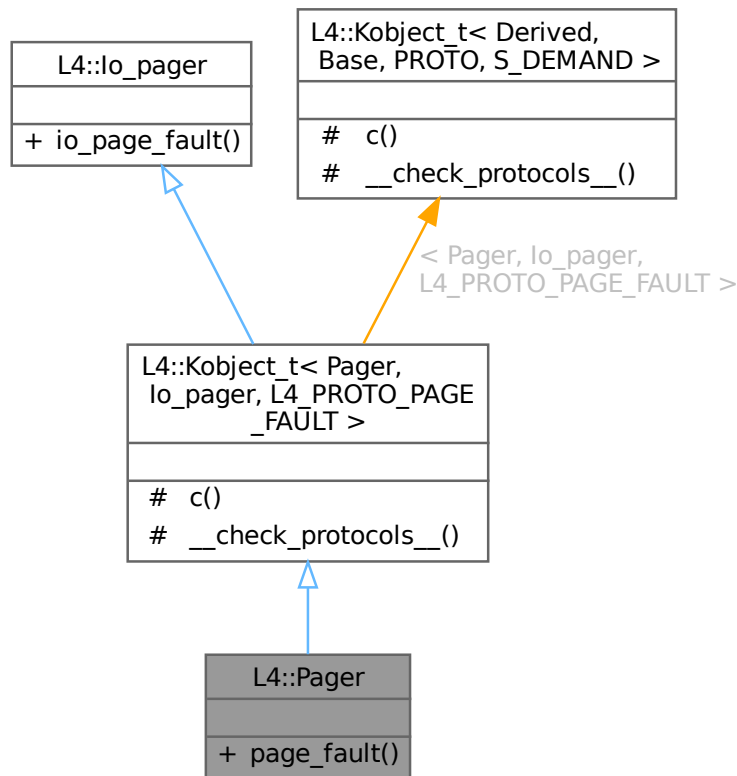
[Pager](#) interface including the [lo_pager](#) interface.

```
#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.

15.182.1 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](#).

15.182.2 Member Function Documentation

15.182.2.1 [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:

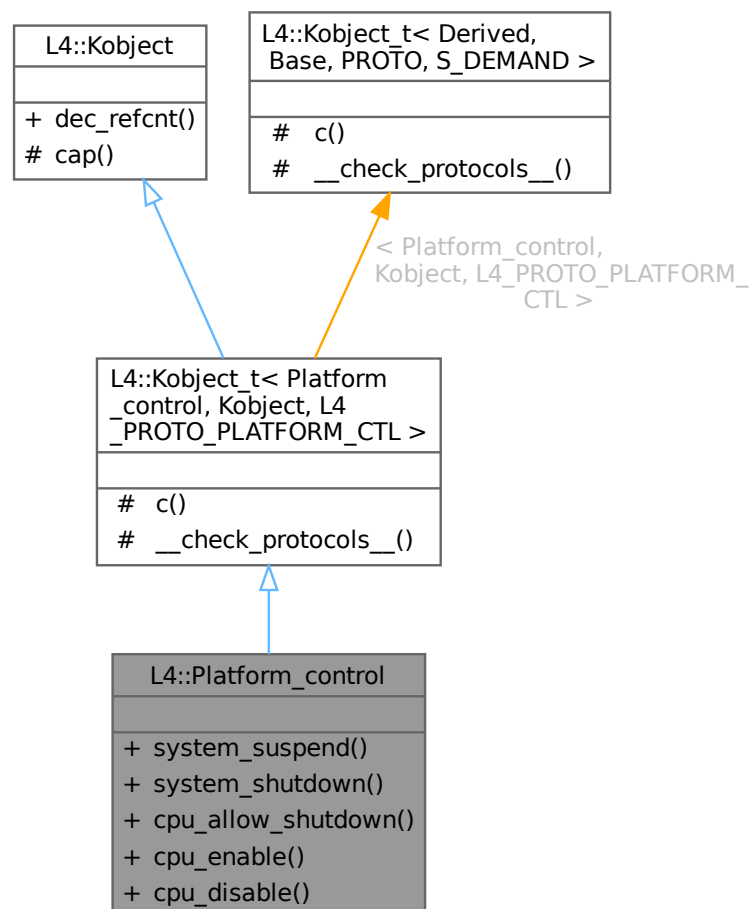
- `l4/sys/pager`

15.183 L4::Platform_control Class Reference

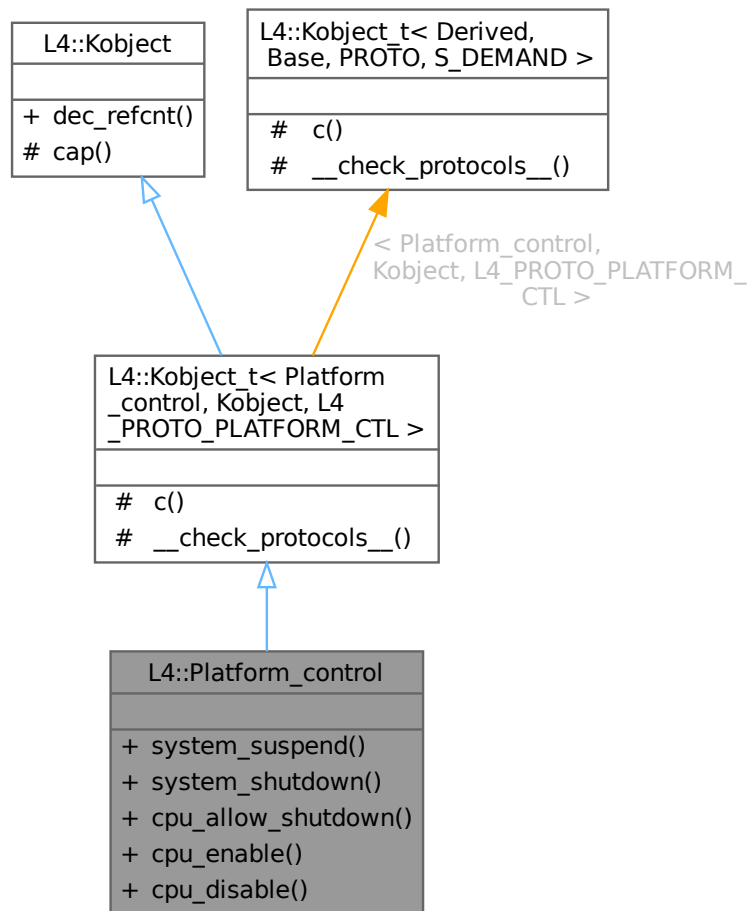
[L4](#) C++ interface for controlling platform-wide properties, see [Platform Control C API](#) for the C interface.

```
#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.

15.183.1 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](#).

15.183.2 Member Function Documentation

15.183.2.1 [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.

15.183.2.2 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.

15.183.2.3 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.

15.183.2.4 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
---------------	-------------------------------

15.183.2.5 system_suspend()

```
l4_msgtag_t L4::Platform_control::system_suspend (
    l4_umword_t extras )
```

Enter suspend to RAM.

Parameters

<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.
---------------	---

The documentation for this class was generated from the following file:

- [l4/sys/platform_control](#)

15.184 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
+ Poll_timeout_counter() + set() + test() + timed_out()

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.

15.184.1 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](#).

15.184.2 Constructor & Destructor Documentation

15.184.2.1 Poll_timeout_counter()

```
L4::Poll_timeout_counter::Poll_timeout_counter (
    unsigned counter_val ) [inline]
```

Constructor.

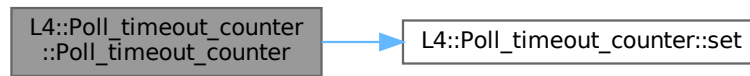
Parameters

<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:



15.184.3 Member Function Documentation

15.184.3.1 set()

```
void L4::Poll_timeout_counter::set (
    unsigned counter_val ) [inline]
```

Set the counter to a certain value.

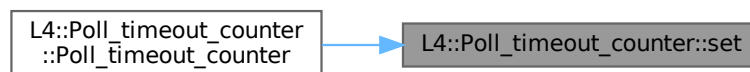
Parameters

<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:



15.184.3.2 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:

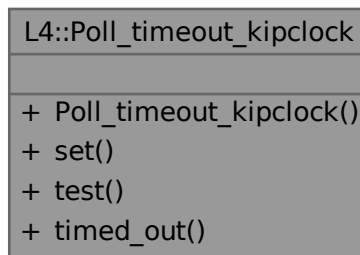
- [pkg/drivers-frst/include/poll_timeout_counter.h](#)

15.185 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.

15.185.1 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](#).

15.185.2 Constructor & Destructor Documentation

15.185.2.1 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:



15.185.3 Member Function Documentation

15.185.3.1 set()

```
void L4::Poll_timeout_kipclock::set (
    unsigned poll_time_us ) [inline]
```

(Re-)Set relative timeout in microseconds

Parameters

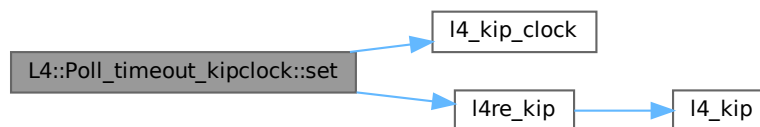
<i>poll_time_us</i>	Polling timeout in microseconds.
---------------------	----------------------------------

Definition at line 54 of file [poll_timeout_kipclock](#).

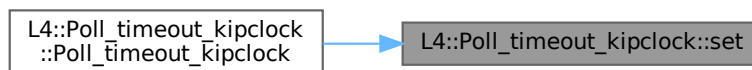
References [l4_kip_clock\(\)](#), and [l4re_kip\(\)](#).

Referenced by [Poll_timeout_kipclock\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.185.3.2 test()

```
bool L4::Poll_timeout_kipclock::test (
    bool expression = true ) [inline]
```

Test whether timeout has expired.

Parameters

<i>expression</i>	Optional expression.
-------------------	----------------------

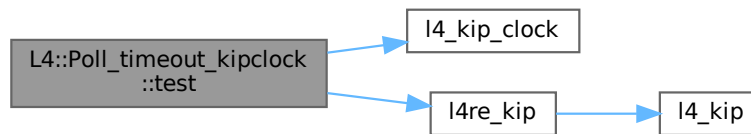
Return values

<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:



15.185.3.3 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:

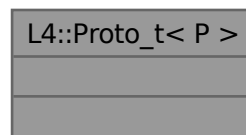
- `l4/re/util/poll_timeout_kipclock`

15.186 L4::Proto_t< P > Struct Template Reference

Data type for defining protocol numbers.

```
#include <__typeinfo.h>
```

Collaboration diagram for `L4::Proto_t< P >`:



15.186.1 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 1190 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

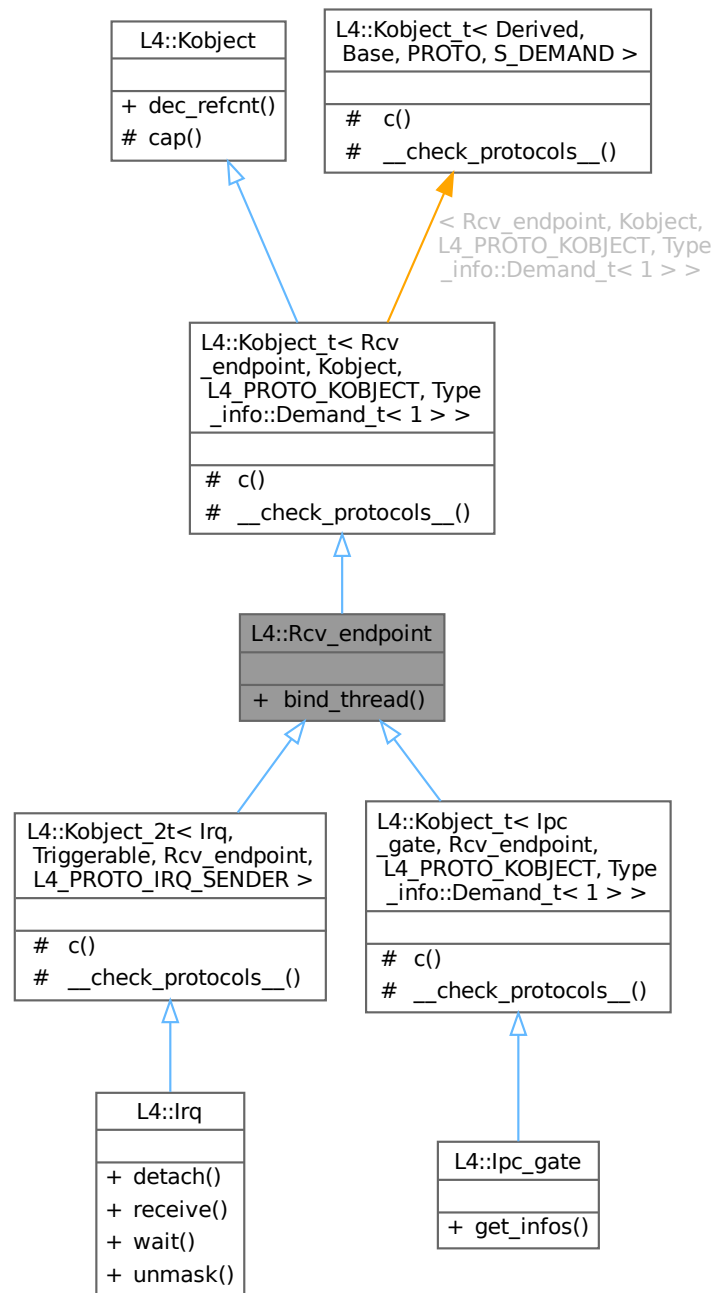
- [l4/sys/__typeinfo.h](#)

15.187 L4::Rcv_endpoint Class Reference

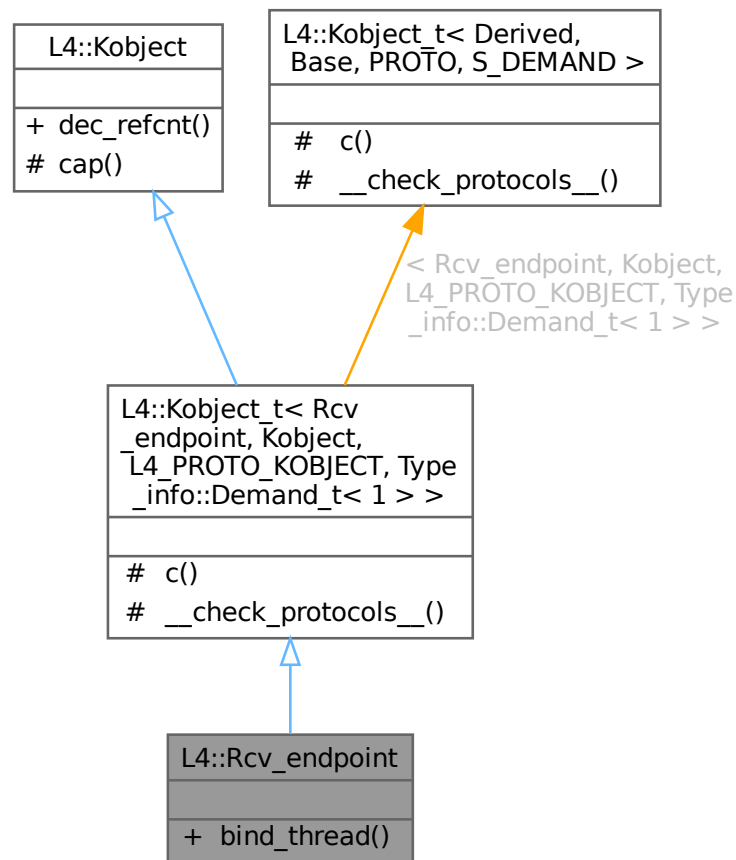
Interface for kernel objects that allow to receive IPC from them.

```
#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.

15.187.1 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](#).

15.187.2 Member Function Documentation

15.187.2.1 [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>	Thread 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 L4_CAP_FPAGE_S 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:

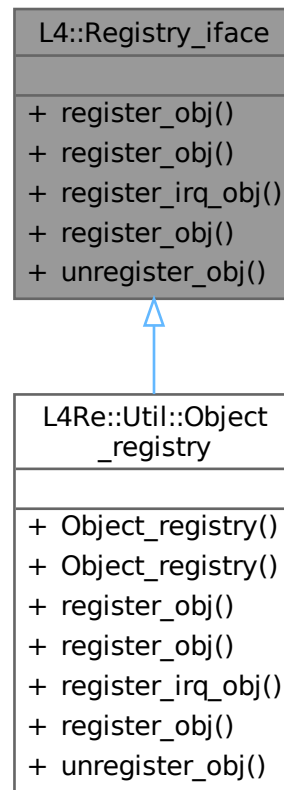
- [l4/sys/rcv_endpoint](#)

15.188 L4::Registry_iface Class Reference

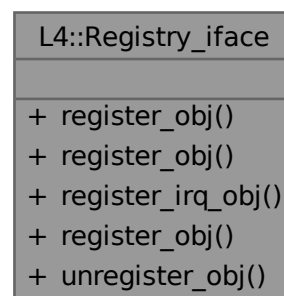
Abstract interface for object registries.

```
#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.

15.188.1 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](#).

15.188.2 Member Function Documentation

15.188.2.1 [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 Epiface object that shall be registered as server-side object for IRQs.
----------	---

Return values

L4::Cap < L4::Irq >	Capability to a new IRQ object on success.
L4::Cap < L4::Irq >::Invalid	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](#).

15.188.2.2 [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 Epiface object that shall be registered as server-side object for RPC.
----------	--

Return values

<i>L4::Cap<void></i>	A valid capability to a new IPC gate.
<i>L4::Cap<void>::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](#).

15.188.2.3 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 Epiface 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<void></i>	The capability known as <code>service</code> on success.
<i>L4::Cap<void>::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](#).

15.188.2.4 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 Epiface 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<L4::Rcv_endpoint></i>	Capability <i>ep</i> on success.
<i>L4::Cap<L4::Rcv_endpoint>::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](#).

15.188.2.5 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 Epiface object that shall be unregistered. The object must have been registered with any of the register methods if Registry_iface .
<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:

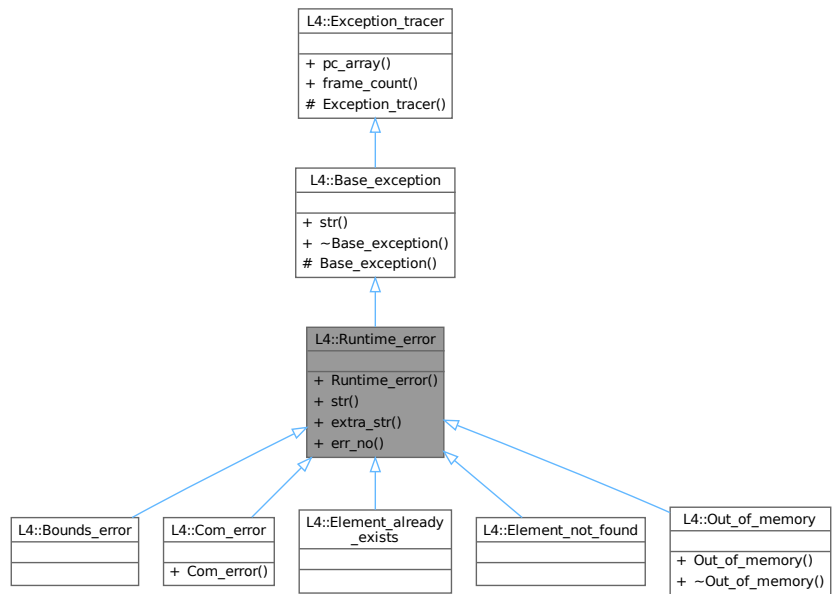
- `l4/sys/cxx/ipc_epiface`

15.189 L4::Runtime_error Class Reference

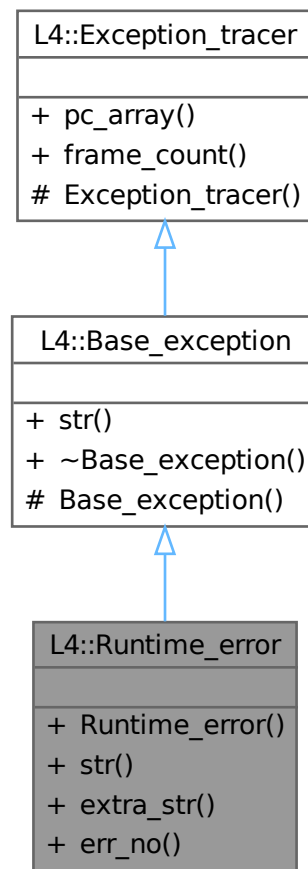
[Exception](#) for an abstract runtime error.

```
#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) throw ()
Create a new [Runtime_error](#).
- char const * `str` () const override throw ()
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` () throw ()
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.

15.189.1 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](#).

15.189.2 Constructor & Destructor Documentation

15.189.2.1 Runtime_error()

```
L4::Runtime_error::Runtime_error (
    long err_no,
    char const * extra = 0 ) throw ( )    [inline], [explicit]
```

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](#).

15.189.3 Member Function Documentation

15.189.3.1 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](#).

15.189.3.2 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:

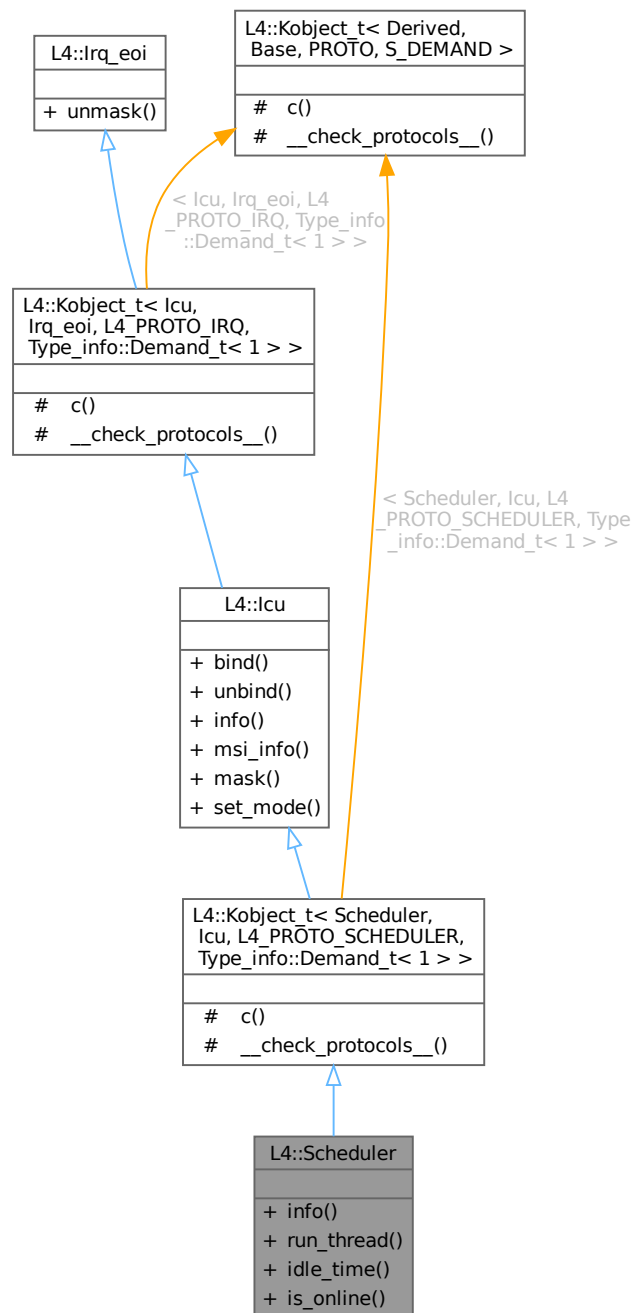
- [l4/cxx/exceptions](#)

15.190 L4::Scheduler Class Reference

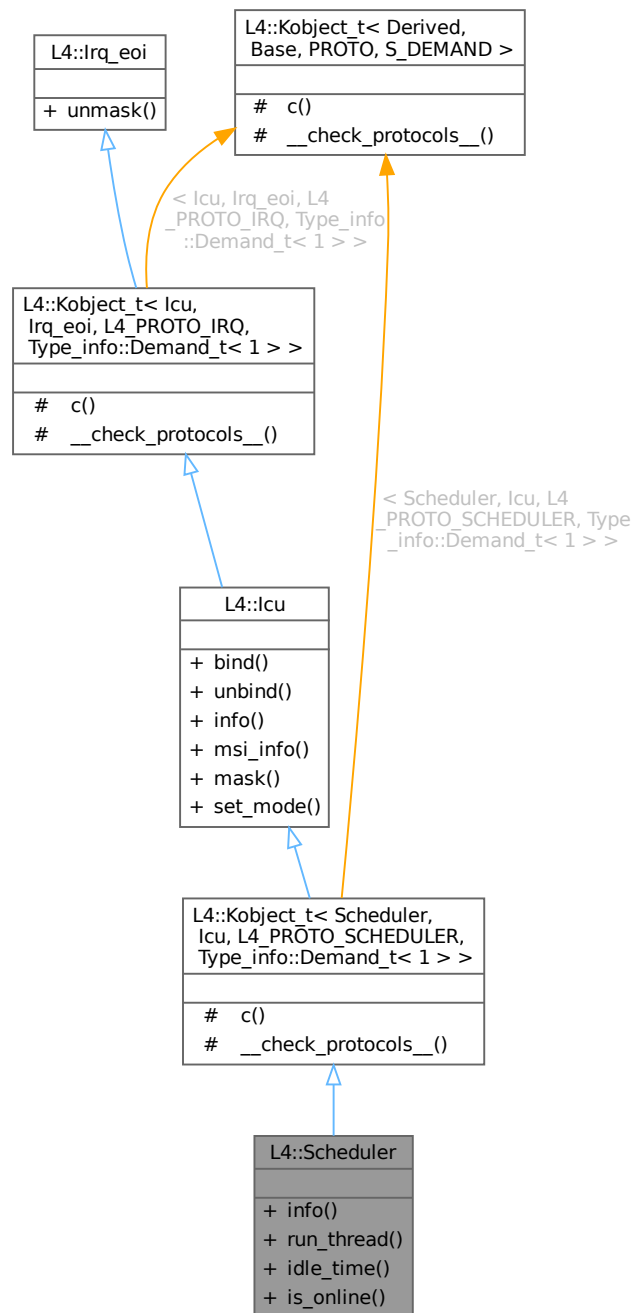
C++ interface of the [Scheduler](#) kernel object, see [Scheduler](#) for the C interface.

```
#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 μ 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::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](#)< [Scheduler](#), [Icu](#), [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](#)< [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< 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.***15.190.1 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](#).

15.190.2 Member Function Documentation**15.190.2.1 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 μ s.

Return values

0	Success.
-L4_EINVAL	Invalid CPU requested in cpu set.

This function retrieves the idle time in μ 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.

15.190.2.2 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 l4_sched_cpu_set_t). <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 l4_utcb .

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](#).

15.190.2.3 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 l4_utcb .

Return values

<i>true</i>	The CPU is online.
<i>false</i>	The CPU is offline

Definition at line 163 of file [scheduler](#).

15.190.2.4 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:

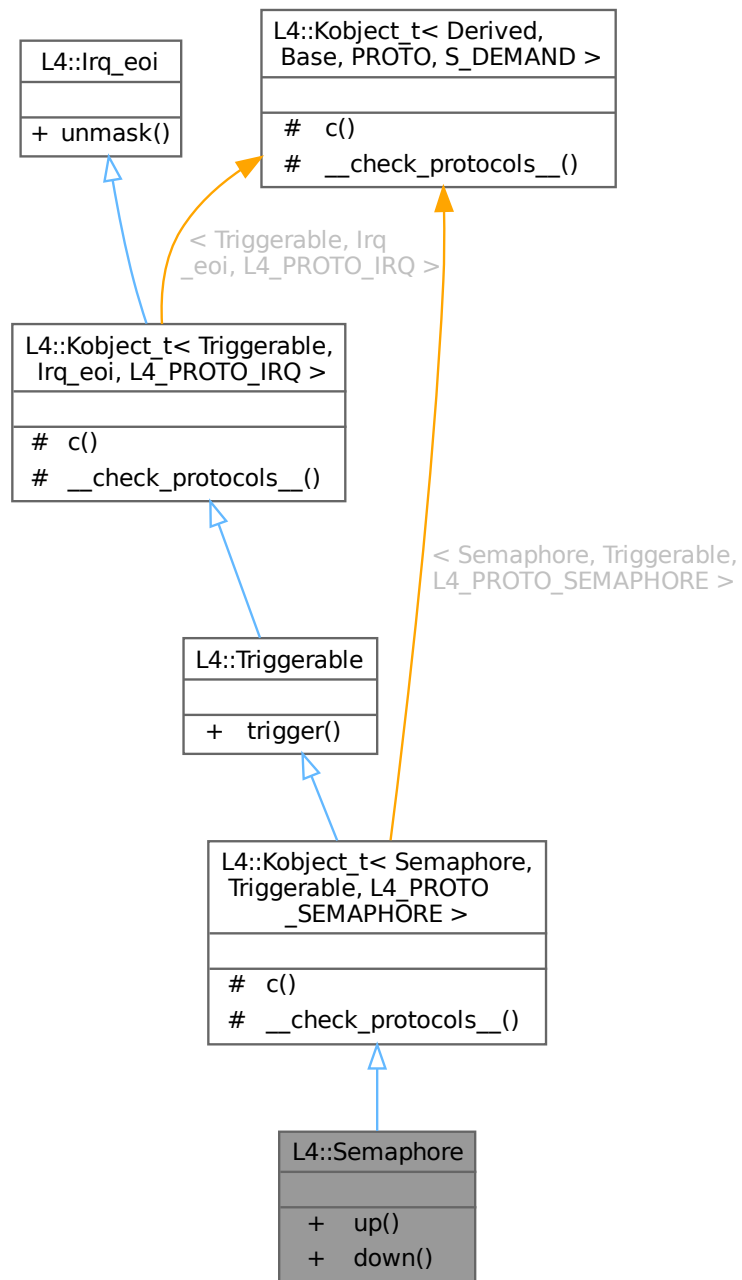
- [l4/sys/scheduler](#)

15.191 L4::Semaphore Struct Reference

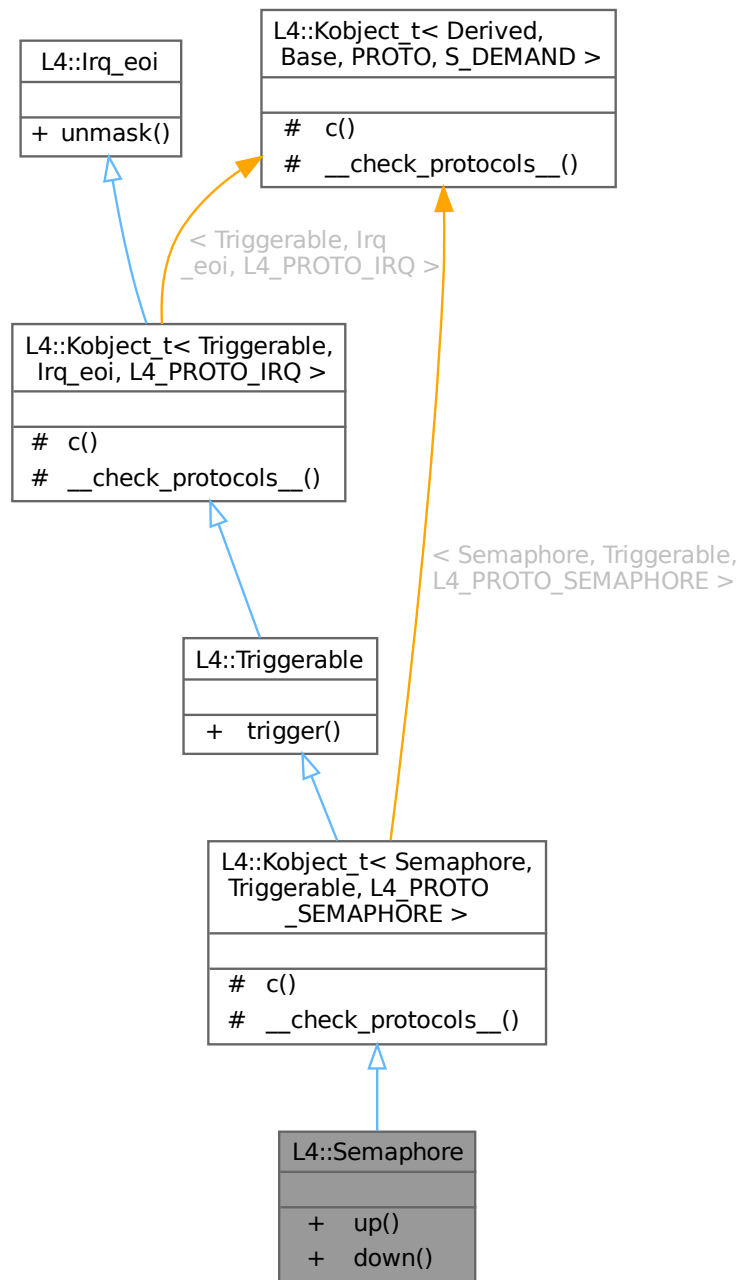
C++ Kernel-provided semaphore interface, see [Kernel-provided semaphore](#) for the C interface.

```
#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.

15.191.1 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.

The main reason for using a semaphore instead of an [L4::lrq](#) 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 52 of file [semaphore](#).

15.191.2 Member Function Documentation

15.191.2.1 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 l4_utcb .

Returns

Syscall return tag. Use [l4_error\(\)](#) to check for errors.

Return values

<code>-L4_EPERM</code>	No L4_CAP_FPAGE_S 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 88 of file [semaphore](#).

15.191.2.2 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 l4_utcb .
-------------------	--

Returns

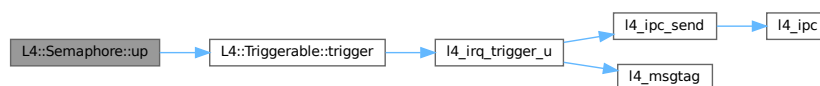
Syscall return tag for a send-only operation, this means there is no return value except [L4_MSGTAG_ERROR](#) indicating success or failure of the send operation. Use [l4_ipc_error\(\)](#) to check for errors and **do not** use [l4_error\(\)](#).

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 68 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:

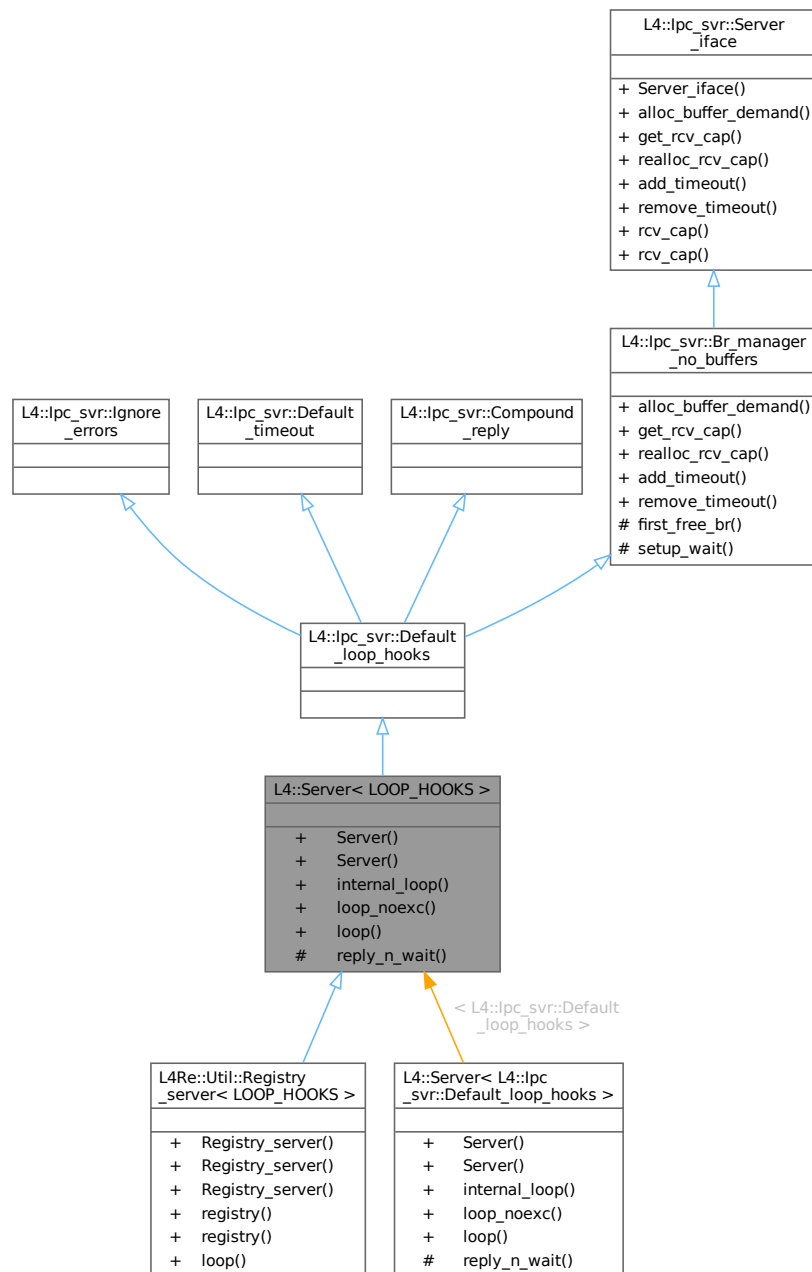
- [l4/sys/semaphore](#)

15.192 L4::Server< LOOP_HOOKS > Class Template Reference

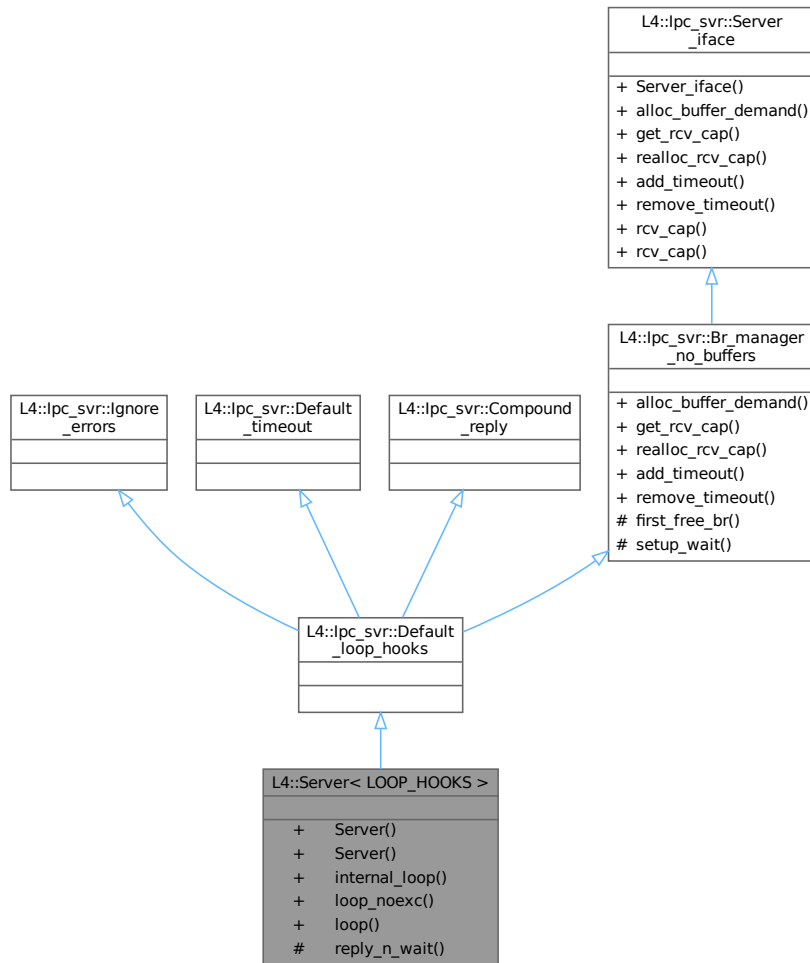
Basic server loop for handling client requests.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::Server< LOOP_HOOKS >:



Collaboration diagram for L4::Server< LOOP_HOOKS >:



Public Member Functions

- **Server** (*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.

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.

15.192.1 Detailed Description

```
template<typename LOOP_HOOKS = lpc_svr::Default_loop_hooks>
class L4::Server< LOOP_HOOKS >
```

Basic server loop for handling client requests.

Parameters

<i>LOOP_HOOKS</i>	the server inherits from LOOP_HOOKS and calls the hooks defined in LOOP_HOOKS in the server loop. See lpc_svr::Default_loop_hooks , lpc_svr::Ignore_errors , lpc_svr::Default_timeout , lpc_svr::Compound_reply , and lpc_svr::Br_manager_no_buffers .
-------------------	--

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](#).

15.192.2 Constructor & Destructor Documentation

15.192.2.1 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](#).

15.192.3 Member Function Documentation

15.192.3.1 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.

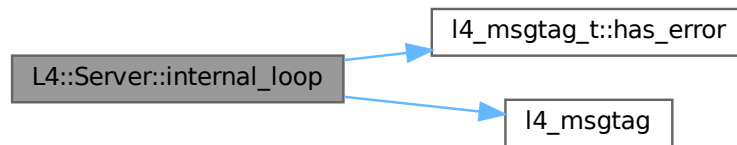
This function usually never returns, it waits for incoming messages calls the dispatcher, sends a reply and waits again.

Definition at line 344 of file [ipc_server_loop](#).

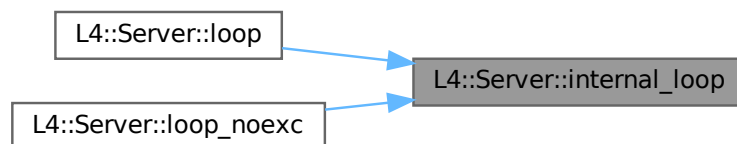
References [l4_msgtag_t::has_error\(\)](#), [L4_ENOREPLY](#), and [l4_msgtag\(\)](#).

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:



15.192.3.2 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]
  
```

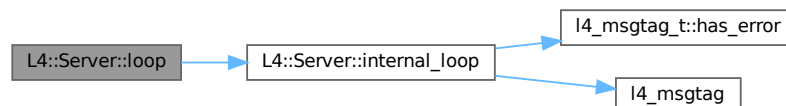
[Server](#) loop with internal exception handling.

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:

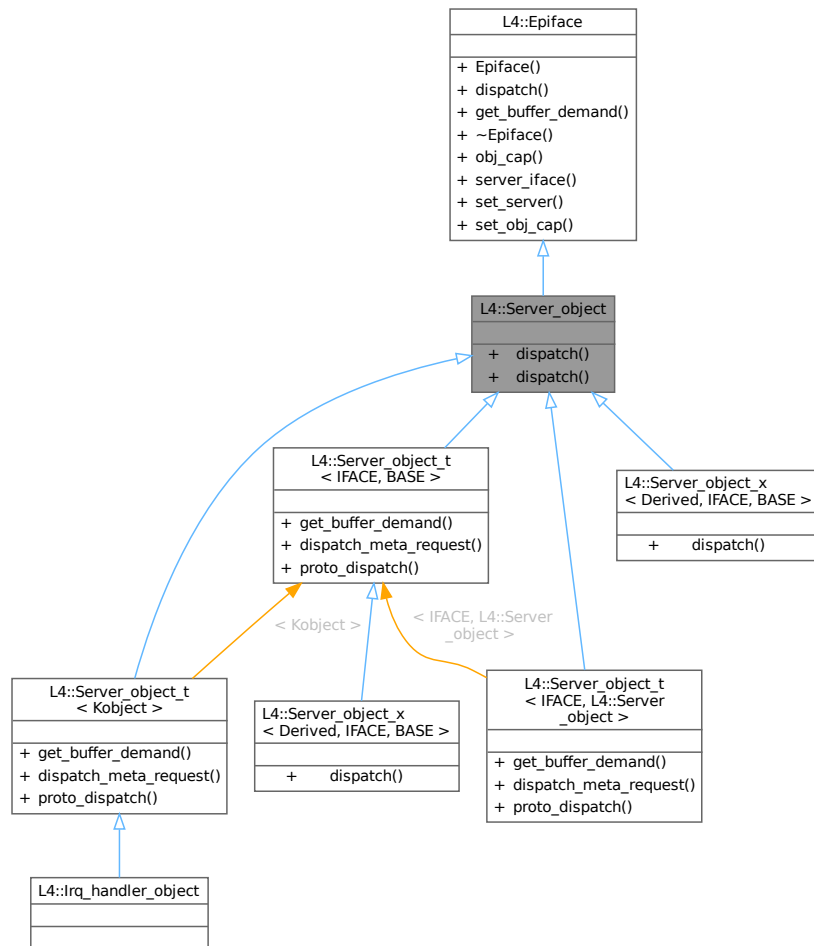
- `l4/sys/cxx/ipc_server_loop`

15.193 L4::Server_object Class Reference

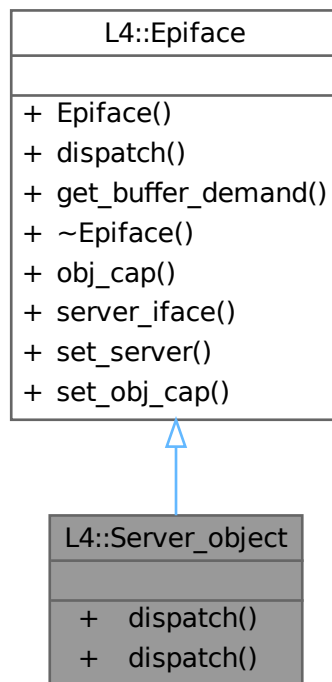
Abstract server object to be used with [L4::Server](#) and [L4::Basic_registry](#).

```
#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.

15.193.1 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](#).

15.193.2 Member Function Documentation

15.193.2.1 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><0</code>	Error, reply with error code.
<code>>=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:



15.193.2.2 `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 ipc::iostream for reading the request and writing the reply.

Return values

<code>-L4_ENOREPLY</code>	Instructs the server loop to not send a reply.
<code>< 0</code>	Error, reply with error code.
<code>>= 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.

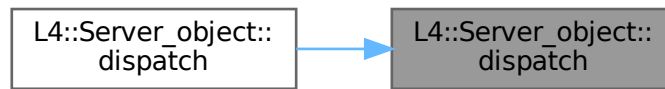
Implemented in [L4::Server_object_x< Derived, IFACE, BASE >](#).

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:

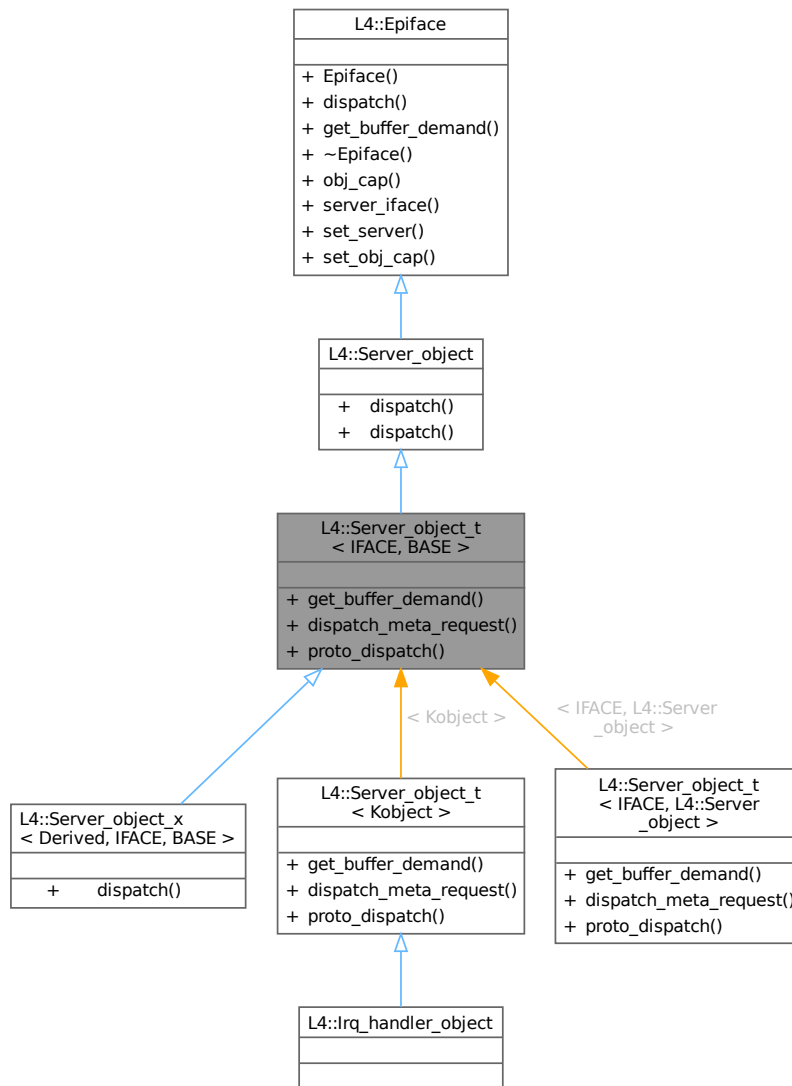
- [l4/cxx/ipc_server](#)

15.194 L4::Server_object_t< IFACE, BASE > Struct Template Reference

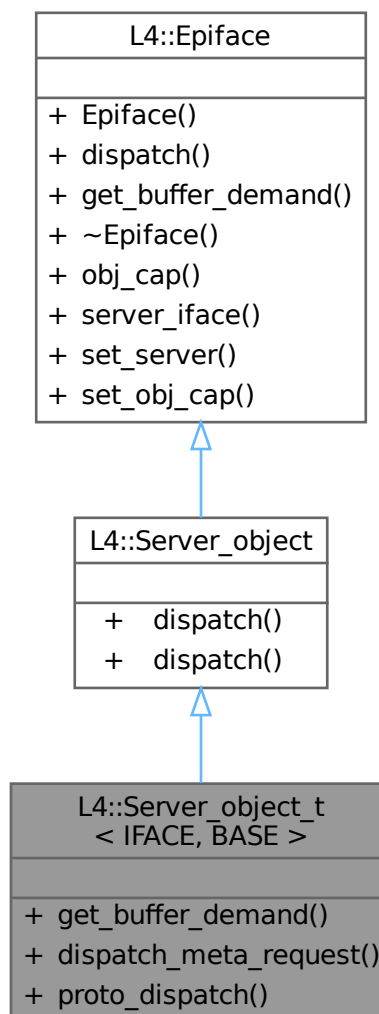
Base class (template) for server implementing server objects.

```
#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.

15.194.1 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](#).

15.194.2 Member Function Documentation

15.194.2.1 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](#).

15.194.2.2 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](#).

15.194.2.3 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 Server_object_t).
<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.

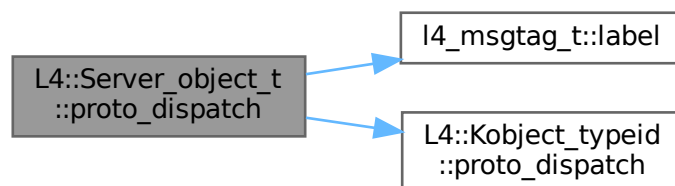
[Server](#) objects may call this function from their [dispatch\(\)](#) function. This function reads the protocol ID from the message tag and uses the `p_dispatch` code to dispatch to overloaded `p_dispatch` functions of self.

Definition at line 125 of file [ipc_server](#).

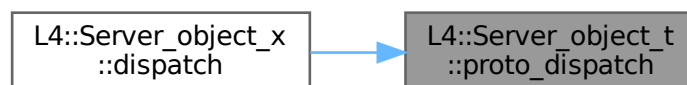
References [l4_msgtag_t::label\(\)](#), and [L4::Kobject_typeid< T >::proto_dispatch\(\)](#).

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:

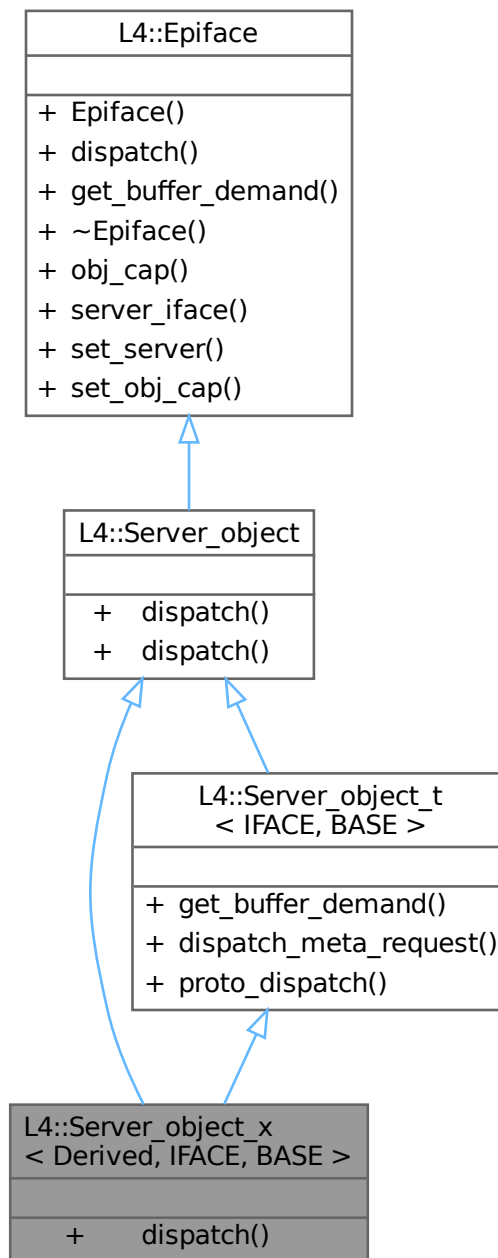
- [l4/cxx/ipc_server](#)

15.195 L4::Server_object_x< Derived, IFACE, BASE > Struct Template Reference

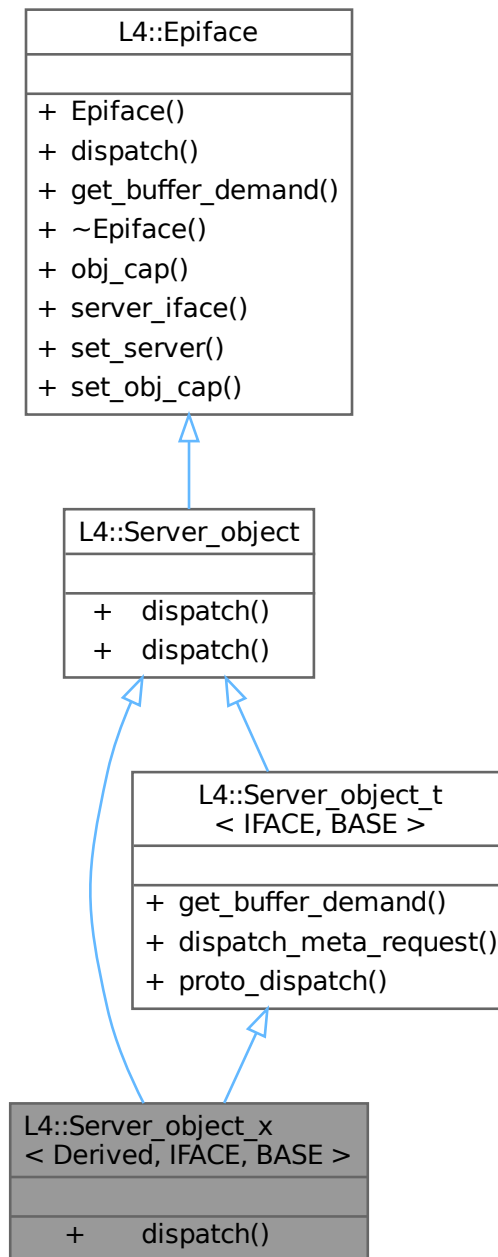
Helper class to implement p_dispatch based server objects.

```
#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.

15.195.1 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 L4::Server_object)

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:

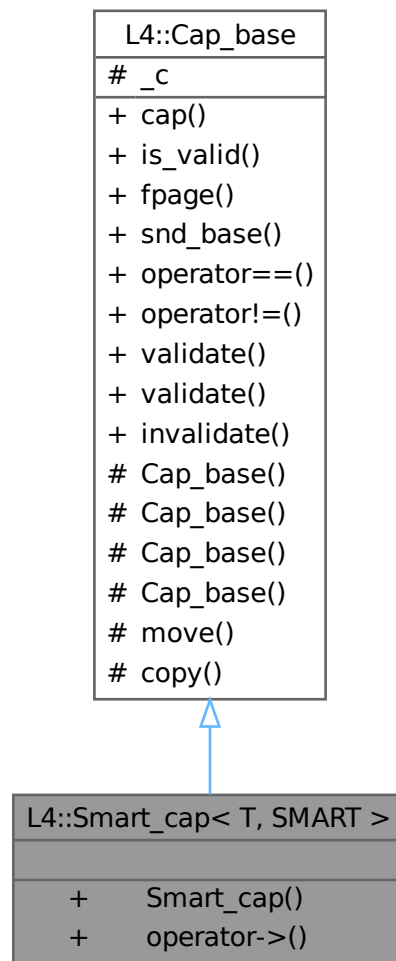
- [l4/cxx/ipc_server](#)

15.196 L4::Smart_cap< T, SMART > Class Template Reference

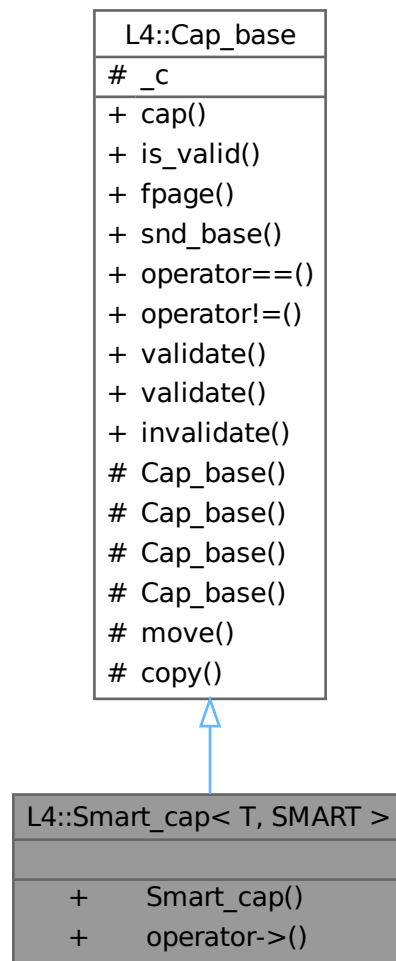
Smart capability class.

```
#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.

15.196.1 Detailed Description

```
template<typename T, typename SMART>
class L4::Smart_cap< T, SMART >
```

Smart capability class.

Definition at line 36 of file [smart_capability](#).

15.196.2 Constructor & Destructor Documentation

15.196.2.1 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 Kobject or derived object
----------	--

Definition at line 73 of file [smart_capability](#).

The documentation for this class was generated from the following file:

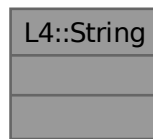
- [l4/sys/smart_capability](#)

15.197 L4::String Class Reference

A null-terminated string container class.

```
#include <string.h>
```

Collaboration diagram for L4::String:



15.197.1 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:

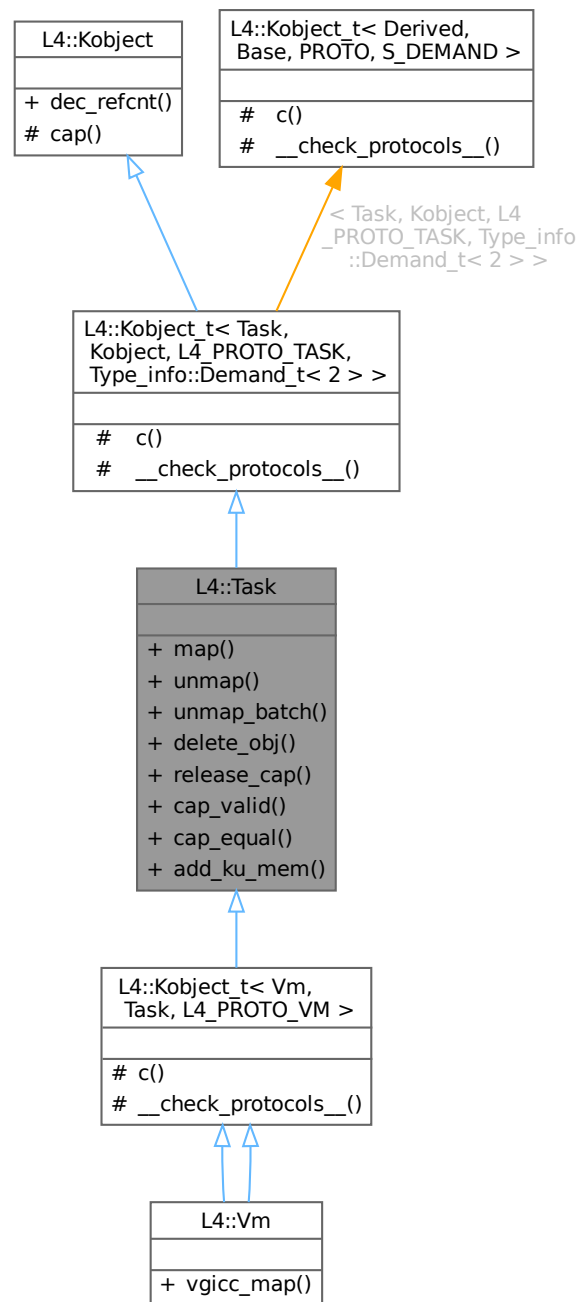
- [l4/cxx/string.h](#)

15.198 L4::Task Class Reference

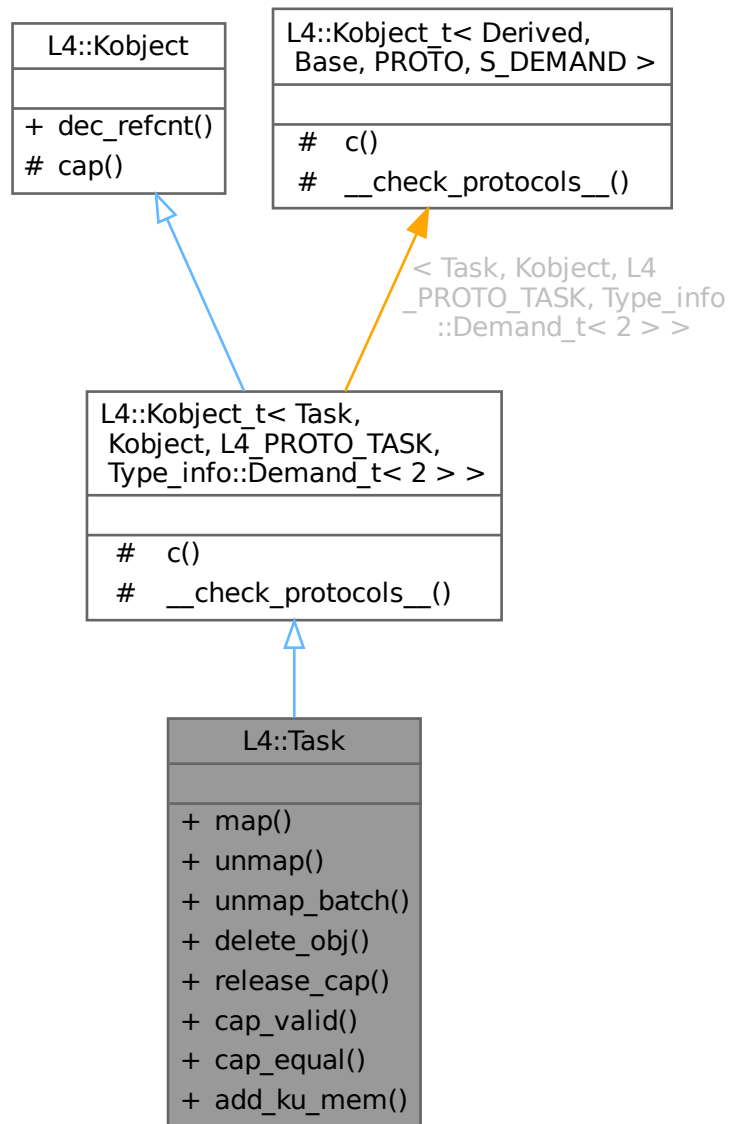
C++ interface of the [Task](#) kernel object, see [Task](#) for the C interface.

```
#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 &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 const &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.

15.198.1 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](#).

15.198.2 Member Function Documentation

15.198.2.1 add_ku_mem()

```
l4_msgtag_t L4::Task::add_ku_mem (
    l4_fpage_t const & fpage,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

Add kernel-user memory.

Parameters

<i>fpage</i>	Flexpage describing the virtual area the memory goes to.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb .

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

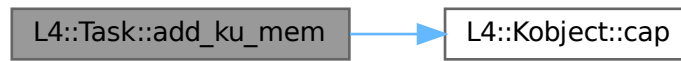
The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page ([L4_PAGE_SIZE](#)). A portable implementation should not depend on allocations greater than 16KiB to succeed.

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 253 of file [task](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.198.2.2 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</i> _↔ <i>_a</i>	First capability selector to compare.
<i>cap</i> _↔ <i>_b</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 l4_utcb .

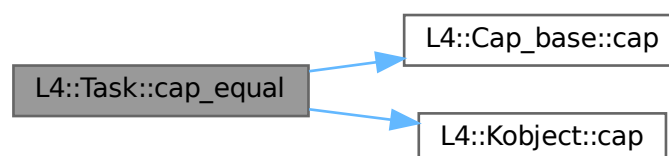
Return values

<i>l4_msgtag_t::label()</i> = 1	<i>cap_a</i> and <i>cap_b</i> point to the same object.
<i>l4_msgtag_t::label()</i> = 0	The two caps do not 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:



15.198.2.3 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 l4_utcb .

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.

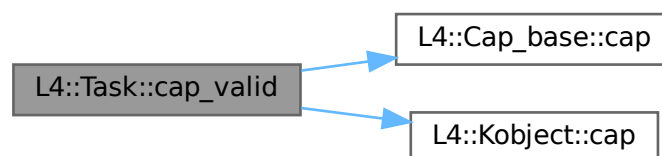
Precondition

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:



15.198.2.4 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 l4_utcb .

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.

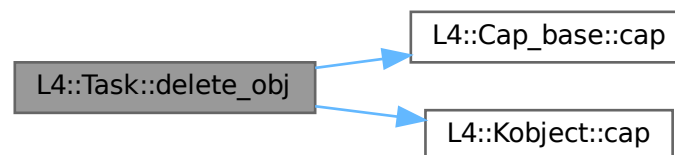
If *obj* does not have the delete permission, no error will be reported and only the capability *obj* is removed. (Note that, depending on the object's reference counter, this might still imply initiation of deletion.)

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:



15.198.2.5 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 l4_fpage_cacheability_opt_t for memory mappings, L4_obj_fpage_ctl for object mappings, and L4_MAP_ITEM_GRANT ; also see l4_map_control() and l4_map_obj_control()).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb .

Returns

Syscall return tag. The function [l4_error\(\)](#) shall be used to test if the map operation was successful.

Return values

L4_EOK	Operation successful (but see notes below).
-L4_EPERM	No L4_CAP_FPAGE_W right on capability used to invoke this operation.
-L4_EINVAL	Invalid source task capability.
-L4_IPC_SEMAPFAILED	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](#).

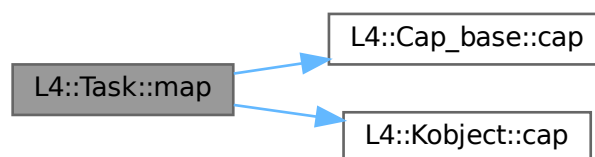
Note

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:

15.198.2.6 `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 l4_utcb .

Returns

Syscall return tag.

This operation unmaps the capability from `this` task.

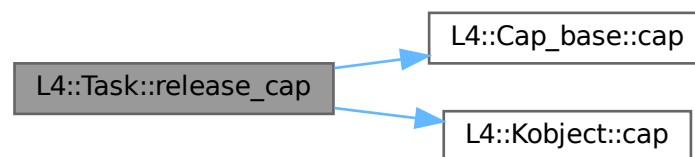
Note

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:



15.198.2.7 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 l4_unmap_flags_t
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb .

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:



15.198.2.8 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 <code>fpages</code> array.
<i>map_mask</i>	Unmap mask, see <code>l4_unmap_flags_t</code> .
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <code>l4_utcb</code> .

Revoke rights for an array of flexpages, see `unmap` for details.

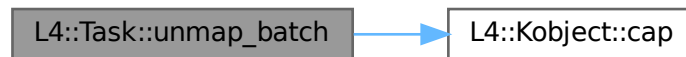
Precondition

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:

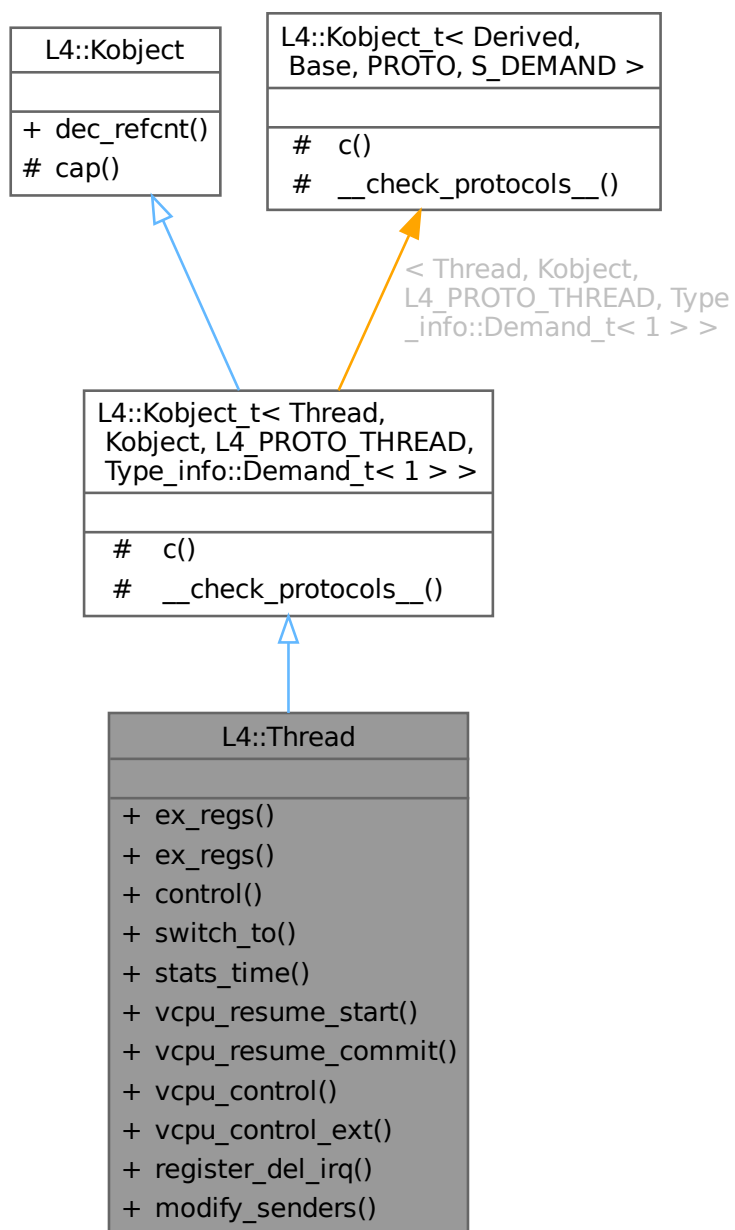
- [l4/sys/task](#)

15.199 L4::Thread Class Reference

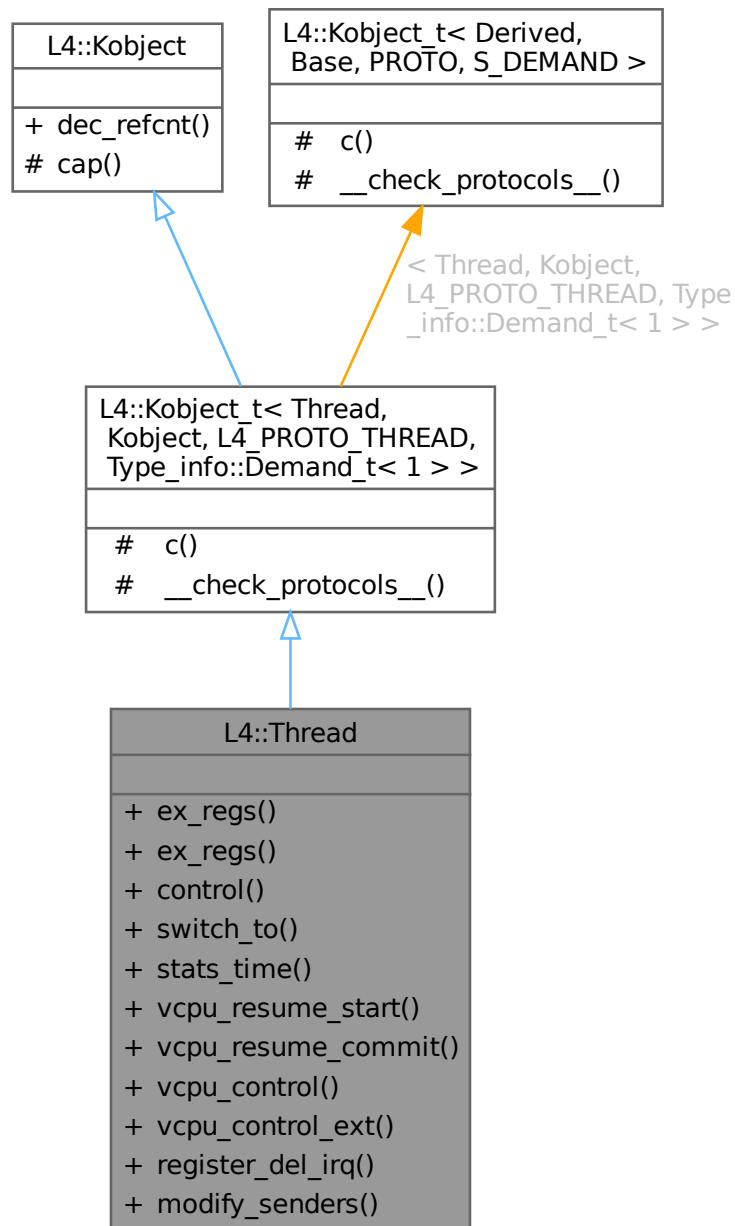
C++ [L4](#) kernel thread interface, see [Thread](#) for the C interface.

```
#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](#) < [lirq](#) > 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.

15.199.1 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](#).

15.199.2 Member Function Documentation

15.199.2.1 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 L4_CAP_FPAGE_S right on the capability used to invoke this operation or the task capability set in Attr::bind() .
<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:

15.199.2.2 `ex_regs()` [1/2]

```

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 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 L4_thread_ex_regs_flags , 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 l4_utcb .

Returns

System call return tag. [out] parameters are only valid if the function returns successfully. Use [l4_error\(\)](#) to check.

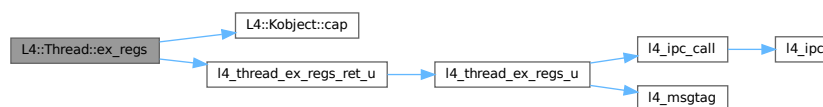
This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`). If the thread 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:



15.199.2.3 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 L4_thread_ex_regs_flags .
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb .

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

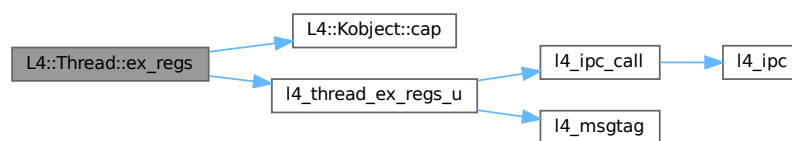
to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if `L4_THREAD_EX_REGS_TRIGGER_EXCEPTION` forces an IPC then changes in IP and SP take effect directly after returning from this IPC.

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:



15.199.2.4 modify_senders()

```
l4_msgtag_t L4::Thread::modify_senders (
    Modify_senders const & todo ) [inline], [noexcept]
```

Apply sender modification rules.

Parameters

<code>todo</code>	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.

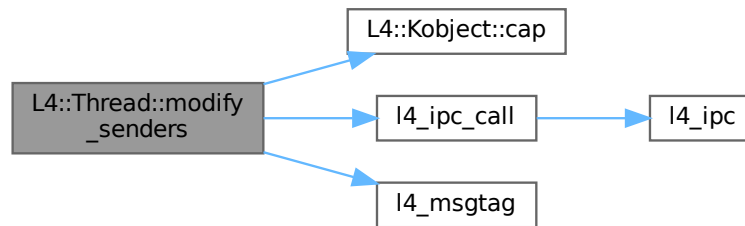
See also

`l4_thread_modify_sender_commit()`

Definition at line 498 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:



15.199.2.5 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 l4_utcb .

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>	L4_CAP_FPAGE_W 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.

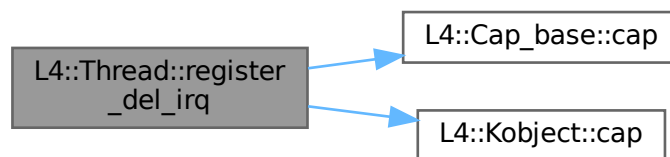
See also

[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:



15.199.2.6 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 μ s.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb .

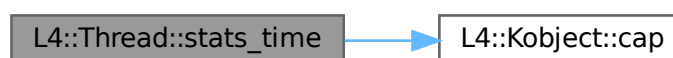
Returns

Syscall return tag.

Definition at line 270 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.199.2.7 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 l4_utcb .
-------------	--

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:



15.199.2.8 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 l4_vcpu_state_t . It must be a valid kernel-user-memory address (see L4::Task::add_ku_mem()).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb .

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](#).

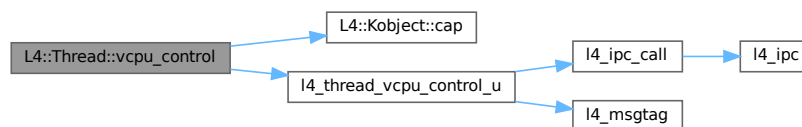
Note

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:



15.199.2.9 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 L4::Task::add_ku_mem()).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb .

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.

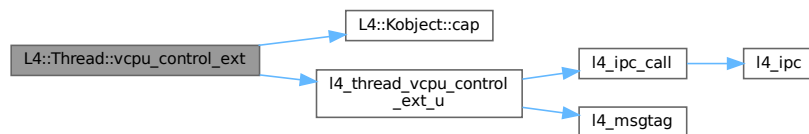
Disabling of the extended vCPU feature is currently not supported.

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:



15.199.2.10 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 l4_thread_vcpu_resume_start() .
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb .

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 L4_CAP_FPAGE_S 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](#).

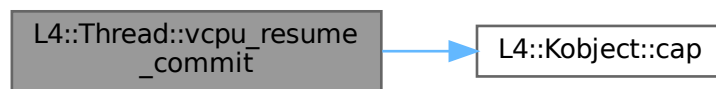
See also

[l4_thread_vcpu_resume_commit](#)

Definition at line 330 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.199.2.11 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 l4_utcb .
-------------	--

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:

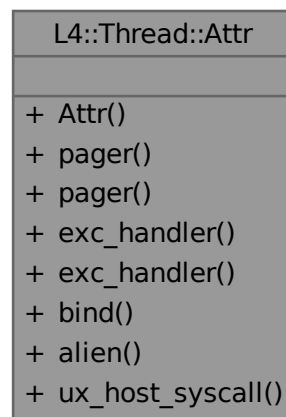
- [l4/sys/thread](#)

15.200 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**

15.200.1 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](#).

15.200.2 Constructor & Destructor Documentation

15.200.2.1 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 L4::Thread::control() function. Usually this is the UTCB of the calling thread. See l4_utcb() .
-------------	---

Definition at line [155](#) of file [thread](#).

15.200.3 Member Function Documentation

15.200.3.1 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](#).

15.200.3.2 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 <code>L4_UTCB_OFFSET</code> 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](#).

15.200.3.3 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:

**15.200.3.4 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:



15.200.3.5 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:



15.200.3.6 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:



15.200.3.7 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:

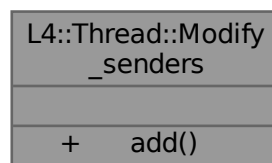
- [l4/sys/thread](#)

15.201 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.

15.201.1 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`.

15.201.2 Member Function Documentation

15.201.2.1 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

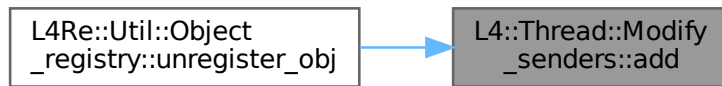
[l4_thread_modify_sender_add\(\)](#)

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:

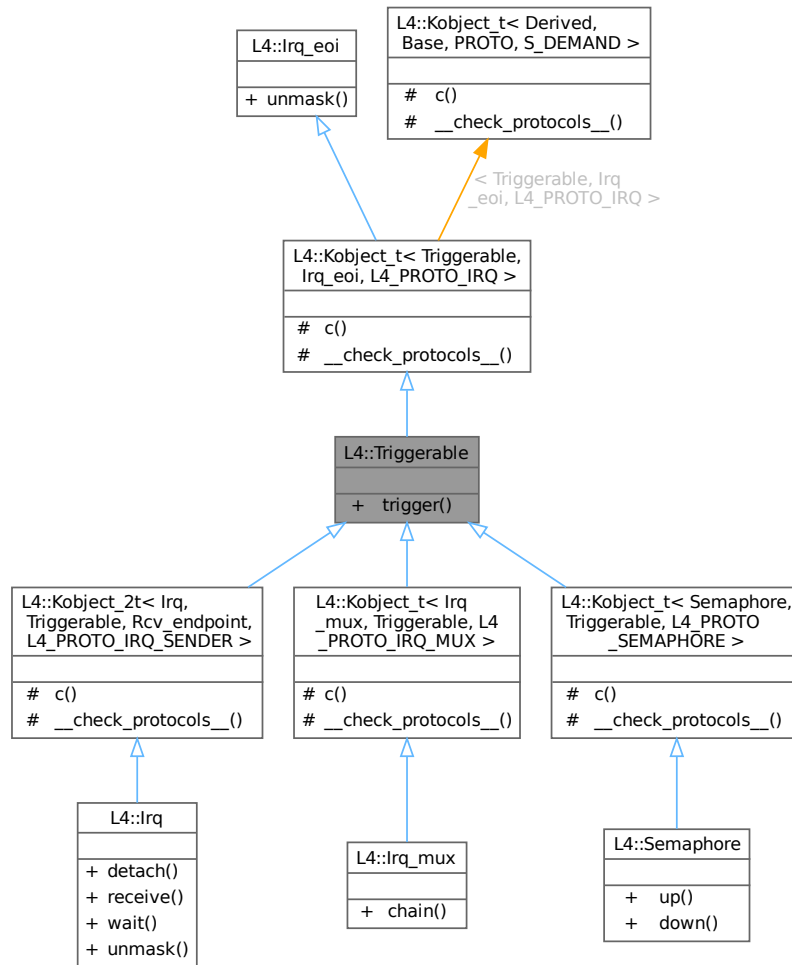
- [l4/sys/thread](#)

15.202 L4::Triggerable Struct Reference

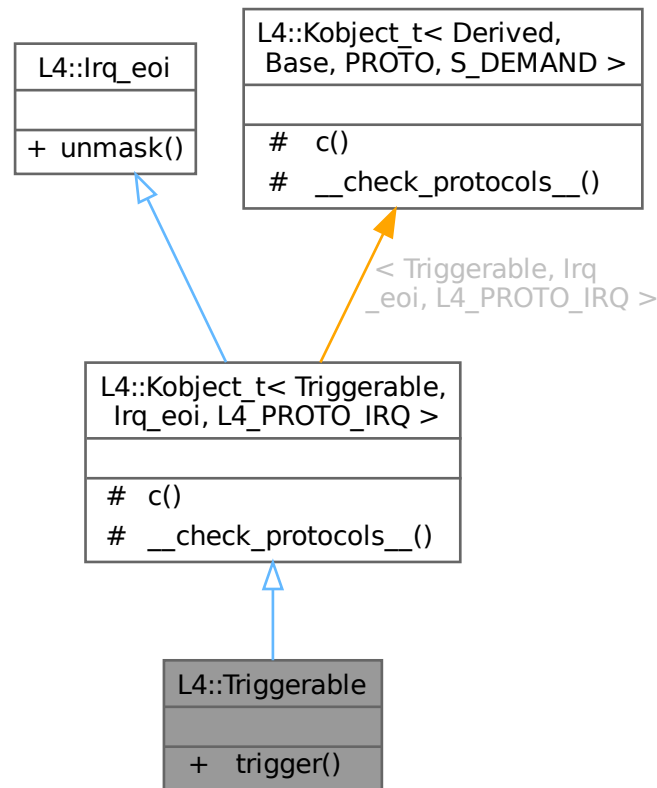
Interface that allows an object to be triggered by some source.

```
#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.

15.202.1 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](#).

15.202.2 Member Function Documentation

15.202.2.1 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 l4_utcb .
-------------	--

Returns

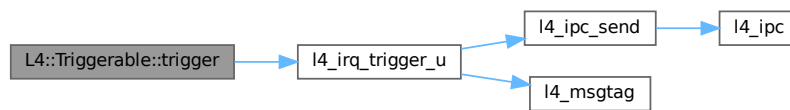
Syscall return tag for a send-only operation, this means there is no return value except [L4_MSGTAG_ERROR](#) indicating success or failure of the send operation. Use [l4_ipc_error\(\)](#) to check for errors and **do not** use [l4_error\(\)](#).

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:

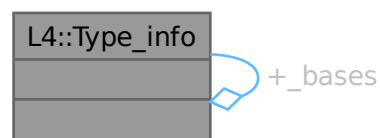
- [l4/sys/irq](#)

15.203 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.

15.203.1 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 [509](#) of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

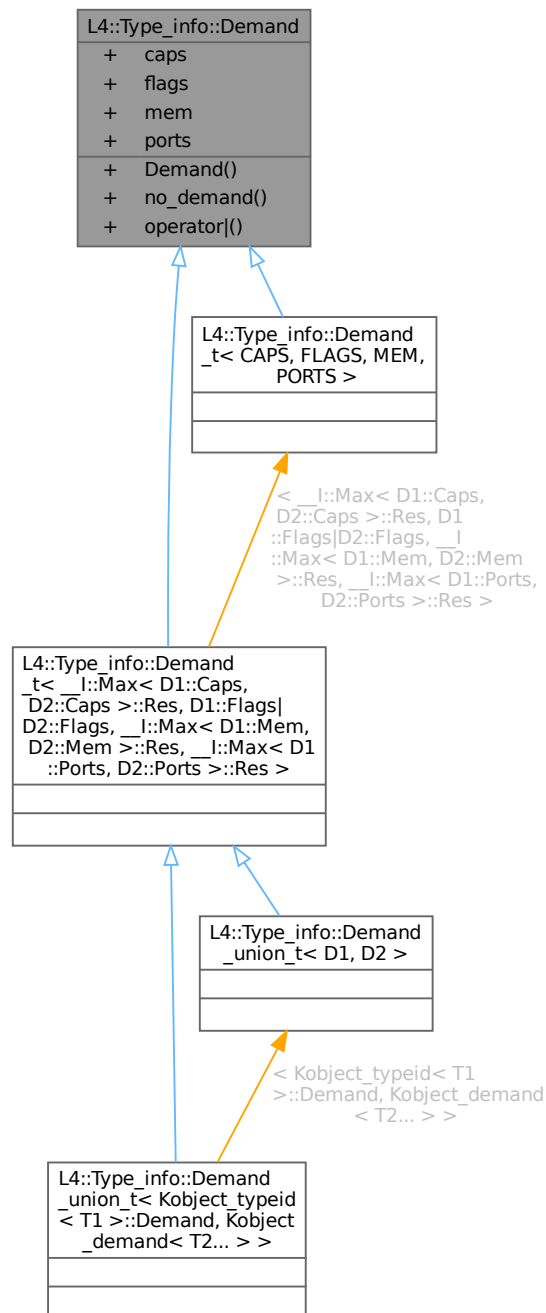
- [l4/sys/__typeinfo.h](#)

15.204 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.

15.204.1 Detailed Description

Data type for expressing the needed receive buffers at the server-side of an interface.

Definition at line 516 of file [__typeinfo.h](#).

15.204.2 Constructor & Destructor Documentation

15.204.2.1 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 537 of file [__typeinfo.h](#).

15.204.3 Member Function Documentation

15.204.3.1 no_demand()

```
bool L4::Type_info::Demand::no_demand ( ) const [inline], [noexcept]
```

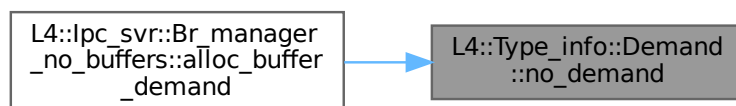
Returns

true if there is no demand at all

Definition at line 542 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:

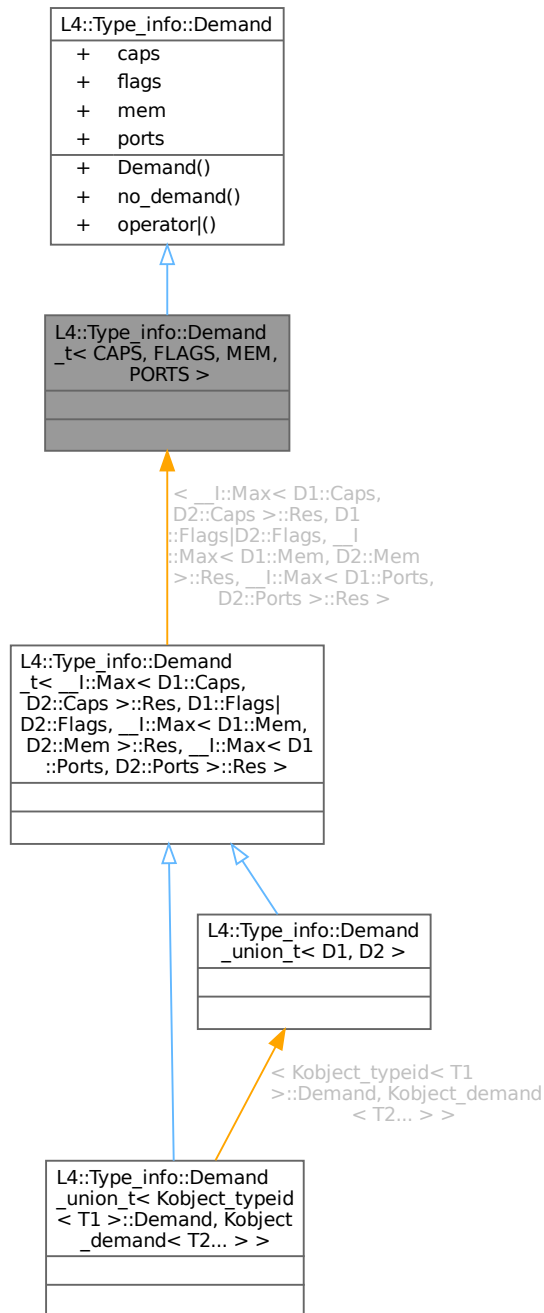
- [l4/sys/__typeinfo.h](#)

15.205 L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS > Struct Template Reference

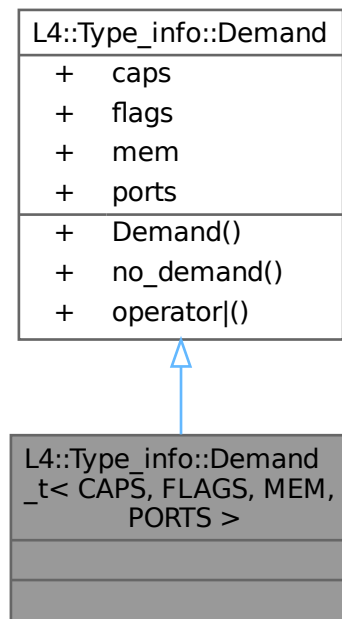
Template type statically describing demand of receive buffers.

```
#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.

15.205.1 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 563 of file [__typeinfo.h](#).

15.205.2 Member Enumeration Documentation

15.205.2.1 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 565 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

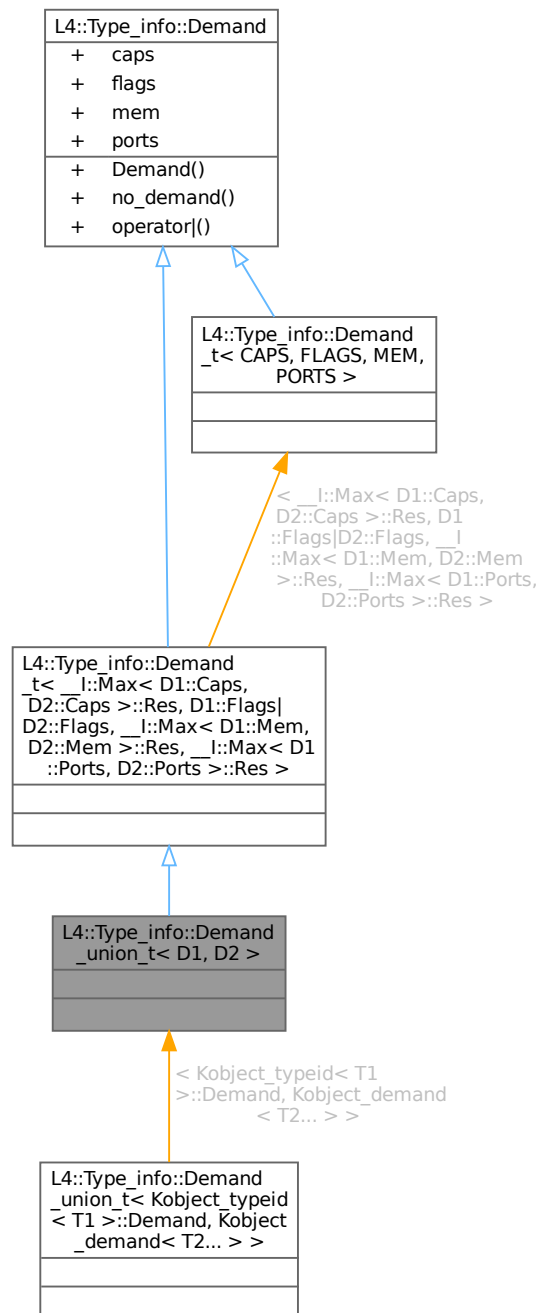
- [l4/sys/__typeinfo.h](#)

15.206 L4::Type_info::Demand_union_t< D1, D2 > Struct Template Reference

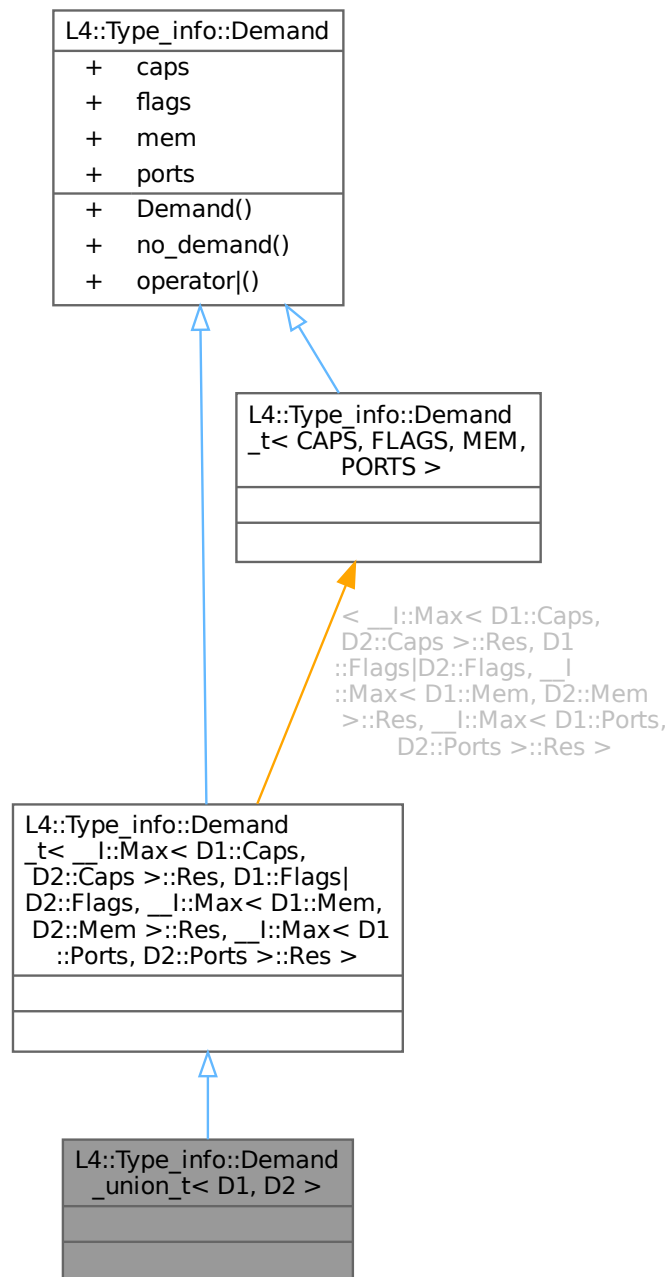
Template type statically describing the combination of two [Demand](#) object.

```
#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.

15.206.1 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 [583](#) of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

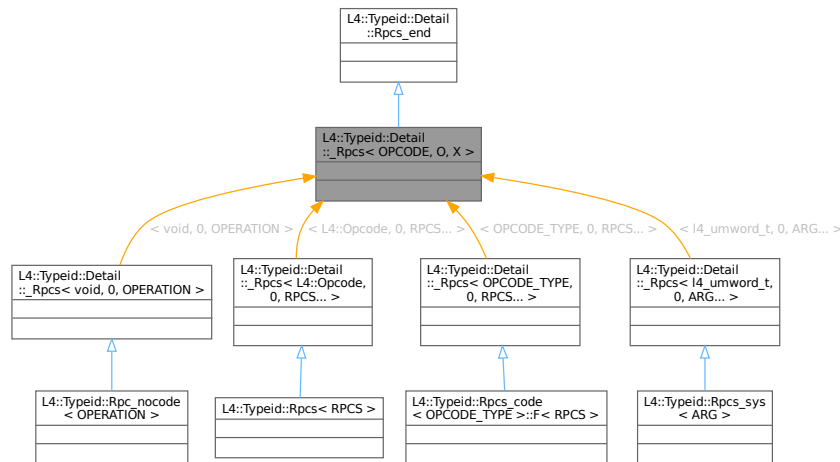
- [l4/sys/__typeinfo.h](#)

15.207 L4::Typeid::Detail::_Rpc< OPCODE, O, X > Struct Template Reference

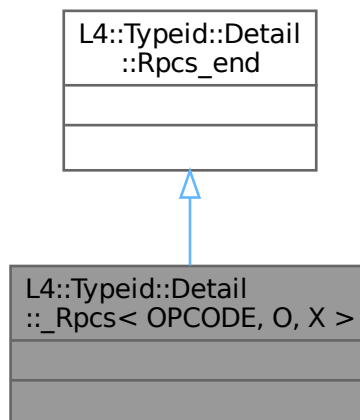
Empty list of RPCs.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Detail::_Rpc< OPCODE, O, X >:



Collaboration diagram for L4::Typeid::Detail::_Rpc< OPCODE, O, X >:



15.207.1 Detailed Description

```
template<typename OPCODE, unsigned O, typename ... X>
struct L4::Typeid::Detail::_Rpc< OPCODE, O, X >
```

Empty list of RPCs.

Definition at line 375 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

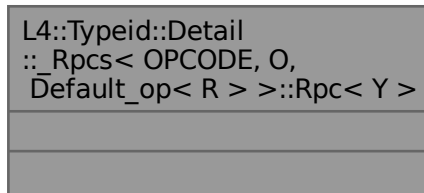
15.208 L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y > Struct Template Reference

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 >:



15.208.1 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 409 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

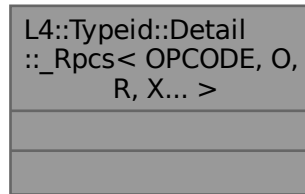
- [l4/sys/__typeinfo.h](#)

15.209 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.

15.209.1 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 379 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

- l4/sys/[__typeinfo.h](#)

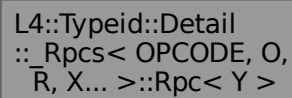
15.210 L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y > Struct Template Reference

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 >:



```
classDiagram
    class L4_Typeid_Detail_Rpc {
        < OPCODE, O, R, X... >::Rpc< Y >
    }
```

15.210.1 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 392 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

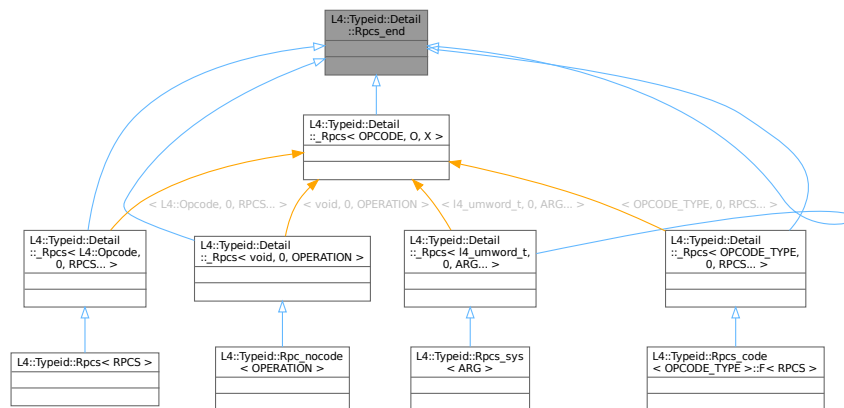
- [l4/sys/__typeinfo.h](#)

15.211 L4::Typeid::Detail::_Rpc_end Struct Reference

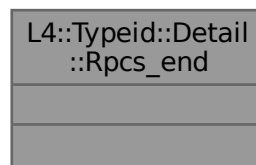
Internal end-of-list marker.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Detail::Rpc_end:



Collaboration diagram for L4::Typeid::Detail::Rpc_end:



15.211.1 Detailed Description

Internal end-of-list marker.

Definition at line 327 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

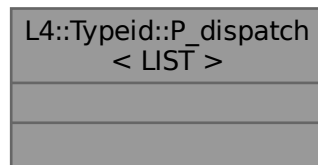
15.212 L4::Typeid::P_dispatch< LIST > Struct Template Reference

Use for protocol based dispatch stage.

```
#include <__typeinfo.h>
```

Inherits L4::Typeid::P_dispatch< LIST >.

Collaboration diagram for L4::Typeid::P_dispatch< LIST >:



15.212.1 Detailed Description

```
template<typename LIST>
struct L4::Typeid::P_dispatch< LIST >
```

Use for protocol based dispatch stage.

Definition at line 318 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

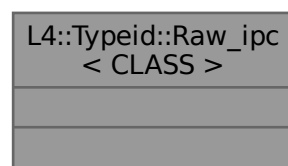
- [l4/sys/__typeinfo.h](#)

15.213 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 >:



15.213.1 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., L4::lcu)
--------------	--

This template allows to have fully handcrafted IPC protocols.

Definition at line 422 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

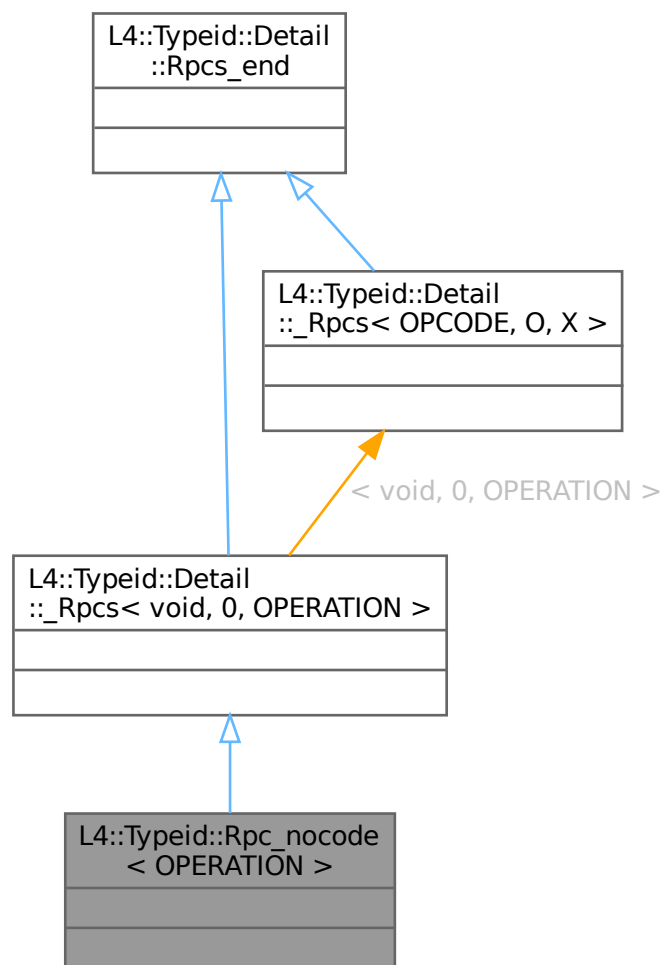
- [l4/sys/__typeinfo.h](#)

15.214 L4::Typeid::Rpc_nocode< OPERATION > Struct Template Reference

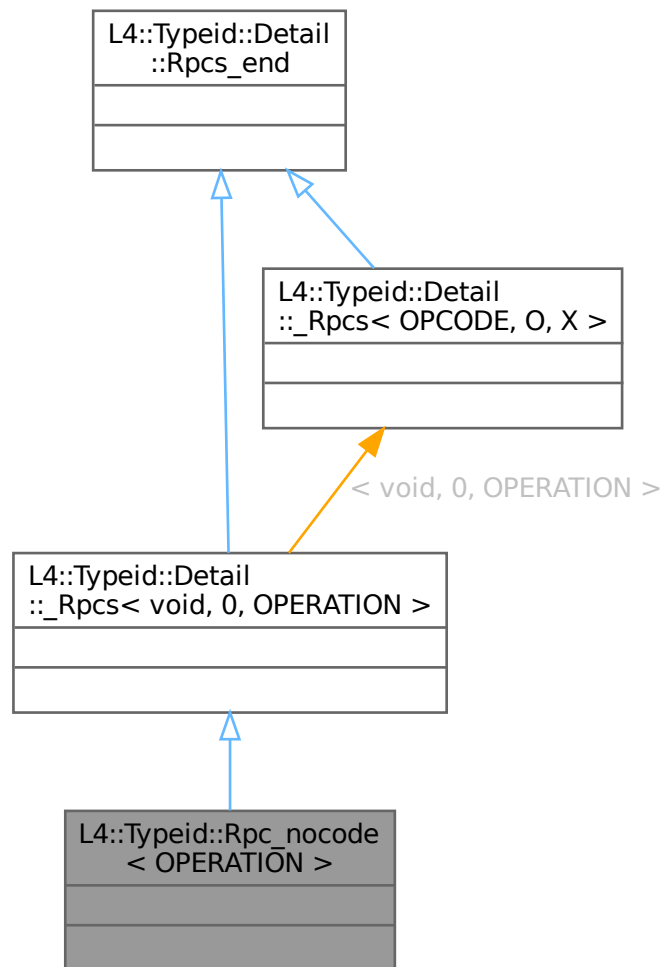
List of RPCs of an interface using a single operation without an opcode.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpc_nocode< OPERATION >:



Collaboration diagram for L4::Typeid::Rpc_nocode< OPERATION >:



15.214.1 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 464 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

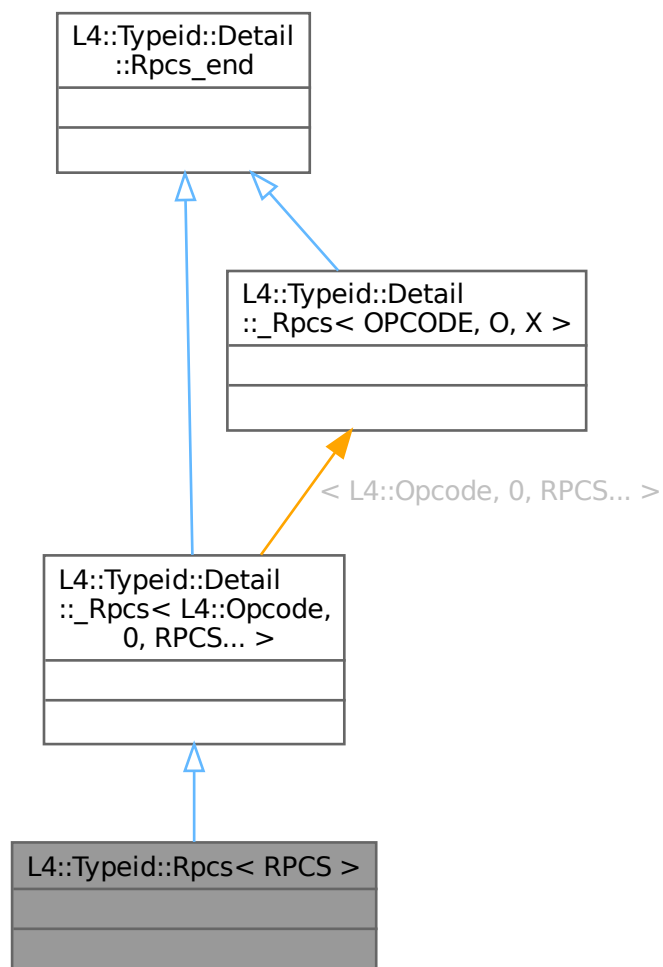
- [l4/sys/__typeinfo.h](#)

15.215 L4::Typeid::Rpc< RPCS > Struct Template Reference

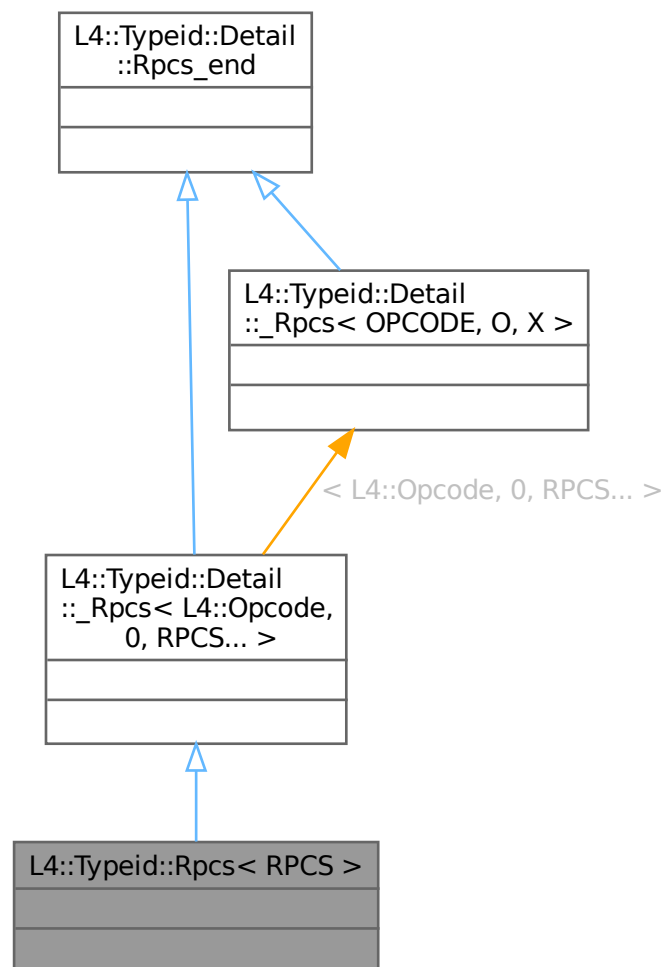
Standard list of RPCs of an interface.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpc< RPCS >:



Collaboration diagram for L4::Typeid::Rpc< RPCS >:



15.215.1 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 438 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

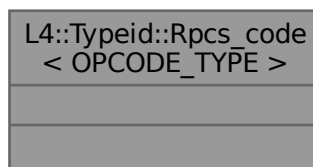
- [l4/sys/__typeinfo.h](#)

15.216 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](#)

15.216.1 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 449 of file [__typeinfo.h](#).

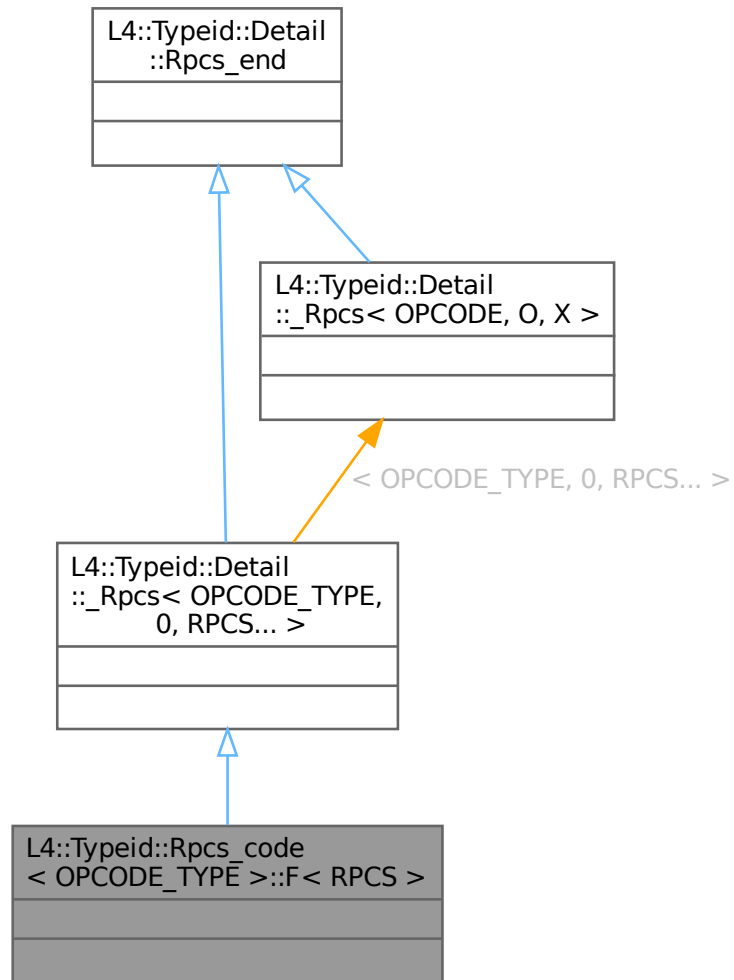
The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

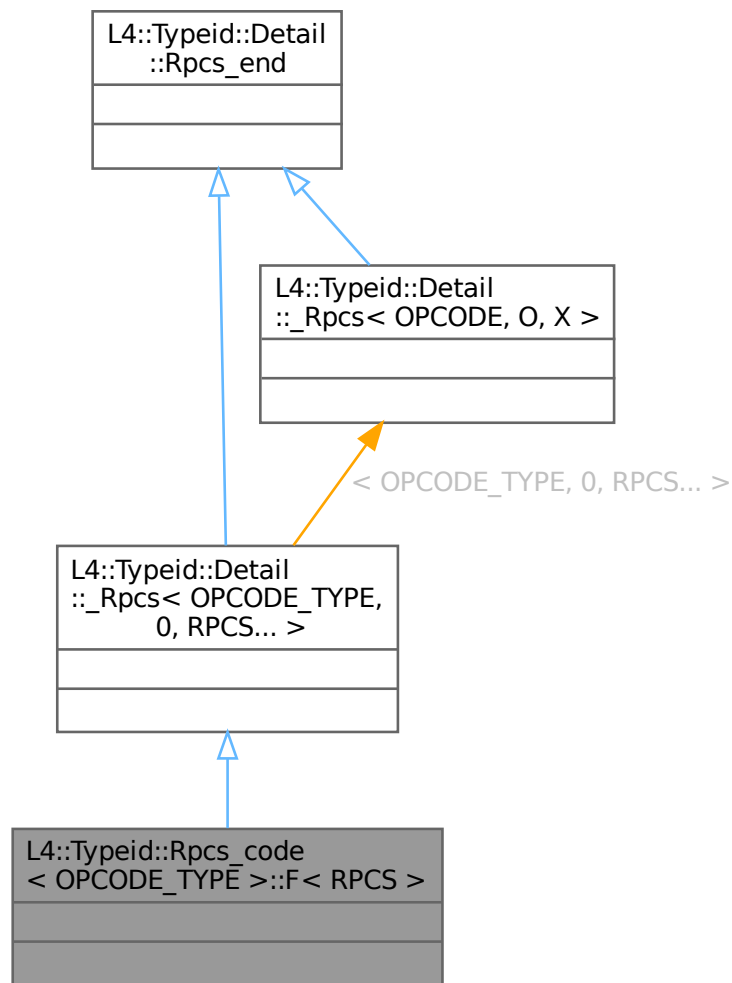
15.217 L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS > Struct Template Reference

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >:



Collaboration diagram for L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >:



15.217.1 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 455 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

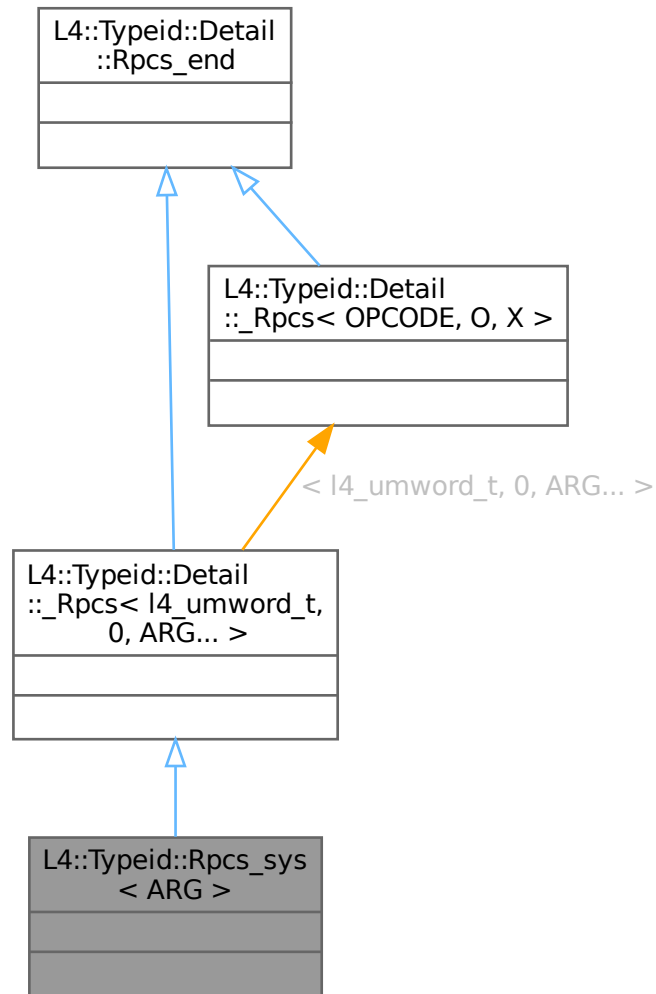
- [l4/sys/__typeinfo.h](#)

15.218 L4::Typeid::Rpcsys< ARG > Struct Template Reference

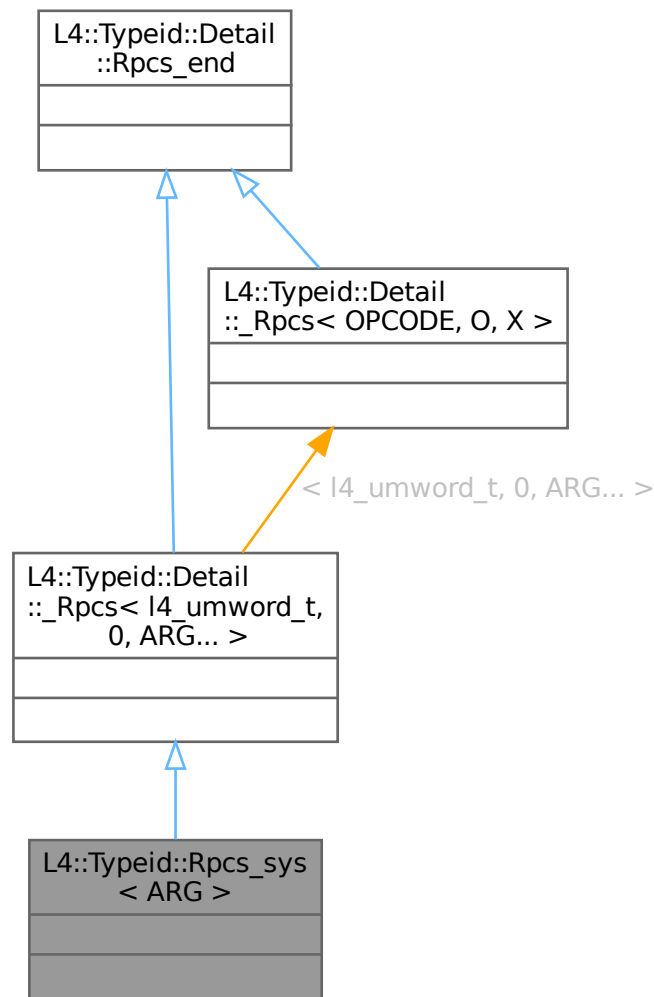
List of RPCs typically used for kernel interfaces.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpcsys< ARG >:



Collaboration diagram for L4::Typeid::Rpcsys< ARG >:



15.218.1 Detailed Description

```
template<typename ... ARG>
struct L4::Typeid::Rpcsys< ARG >
```

List of RPCs typically used for kernel interfaces.

Template Parameters

<i>RPCS</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.

Definition at line 475 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

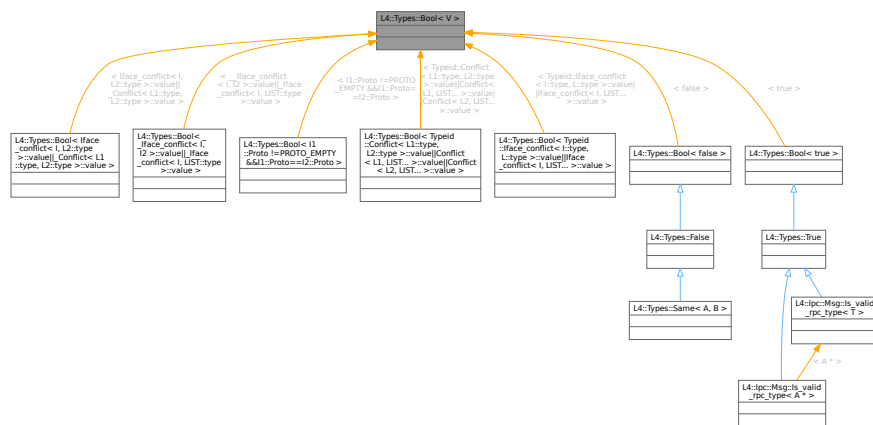
- `l4/sys/__typeinfo.h`

15.219 L4::Types::Bool< V > Struct Template Reference

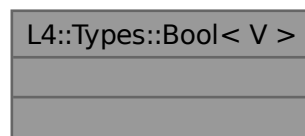
Boolean meta type.

```
#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.

15.219.1 Detailed Description

template<bool V>

```
struct L4::Types::Bool< V >
```

Boolean meta type.

Template Parameters

V	The boolean value
---	-------------------

Definition at line 300 of file [types](#).

The documentation for this struct was generated from the following file:

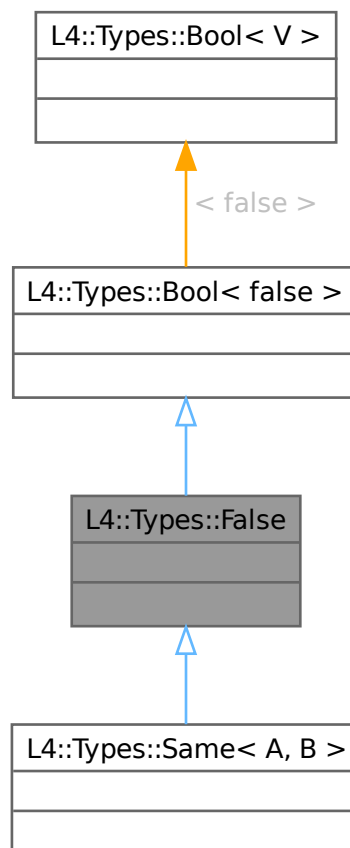
- [l4/sys/cxx/types](#)

15.220 L4::Types::False Struct Reference

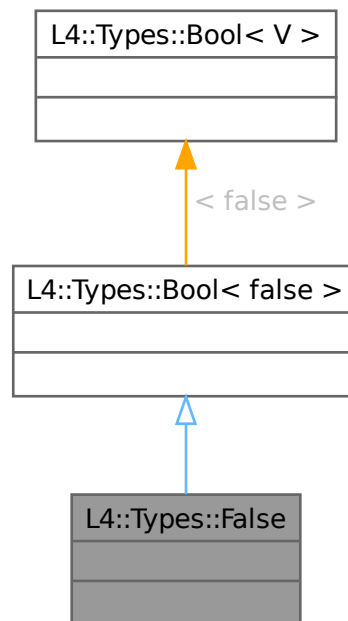
[False](#) meta value.

```
#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.

15.220.1 Detailed Description

[False](#) meta value.

Definition at line [308](#) of file [types](#).

The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

15.221 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 >
<ul style="list-style-type: none"> + Flags() + Flags() + Flags() + operator Private_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 Private_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.

15.221.1 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](#).

15.221.2 Member Enumeration Documentation

15.221.2.1 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 80 of file [types](#).

15.221.3 Constructor & Destructor Documentation

15.221.3.1 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 90 of file [types](#).

15.221.3.2 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 103 of file [types](#).

15.221.4 Member Function Documentation

15.221.4.1 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 154 of file [types](#).

References [L4::Types::Flags< BITS_ENUM, UNDERLYING >::operator&=\(\)](#).

Here is the call graph for this function:



15.221.4.2 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 110 of file [types](#).

The documentation for this class was generated from the following file:

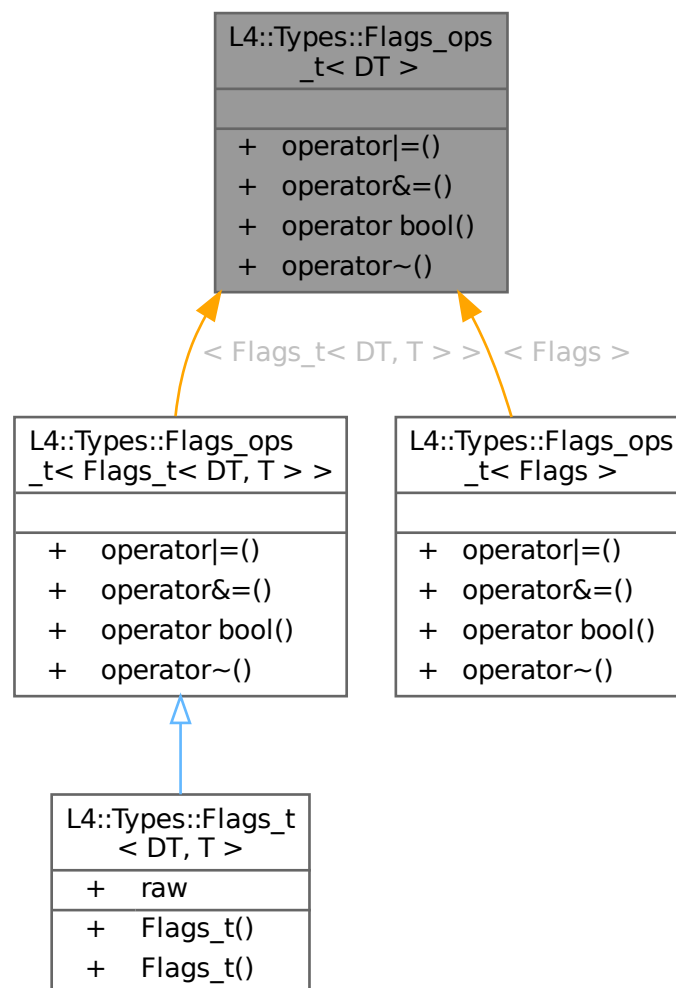
- [l4/sys/cxx/types](#)

15.222 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"> + operator =() + operator&=() + operator bool() + operator~()

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

15.222.1 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 Flags_ops_t with a member <i>raw</i> of enum or integral type.
-----------	---

Definition at line 232 of file [types](#).

The documentation for this struct was generated from the following file:

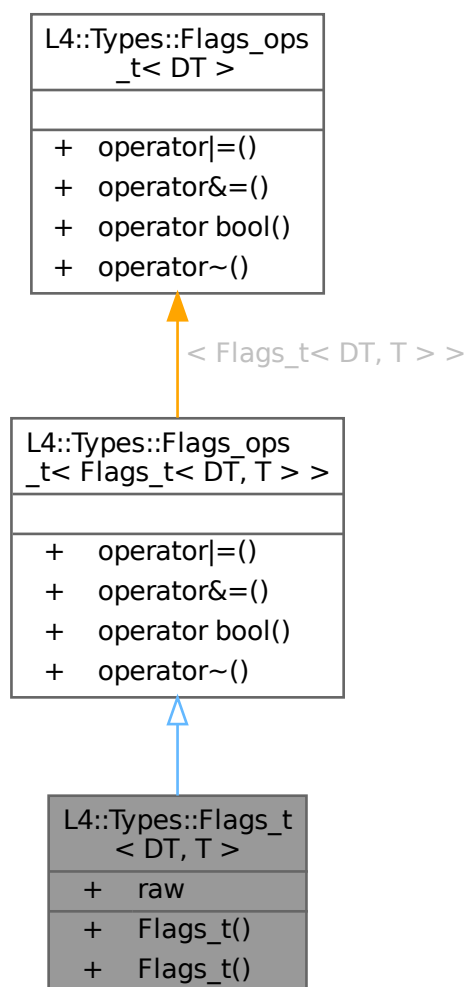
- [l4/sys/cxx/types](#)

15.223 L4::Types::Flags_t< DT, T > Struct Template Reference

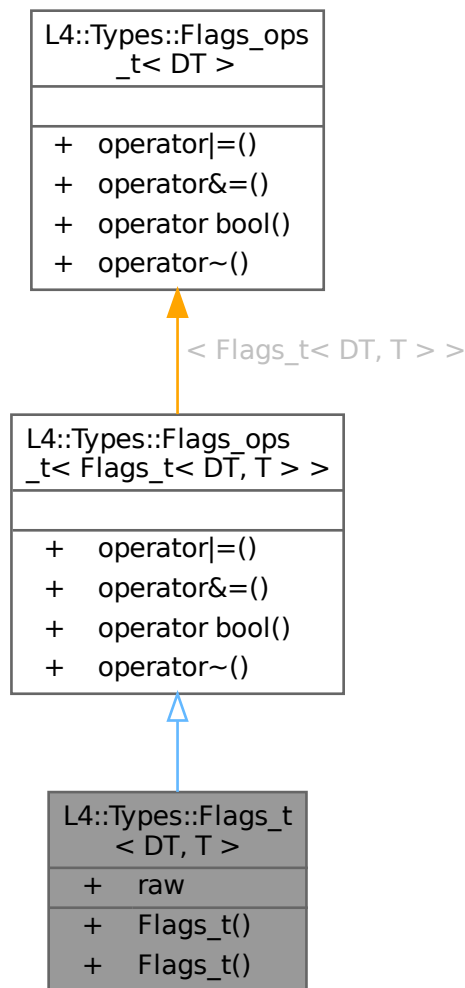
Template type to define a flags type with bitwise operations.

```
#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.

15.223.1 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 284 of file [types](#).

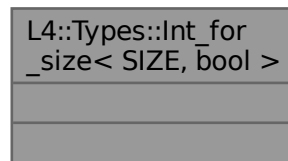
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

15.224 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 >:



15.224.1 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 165 of file [types](#).

The documentation for this struct was generated from the following file:

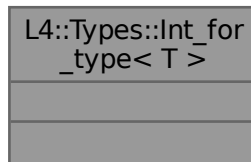
- [l4/sys/cxx/types](#)

15.225 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.

15.225.1 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 192 of file [types](#).

The documentation for this struct was generated from the following file:

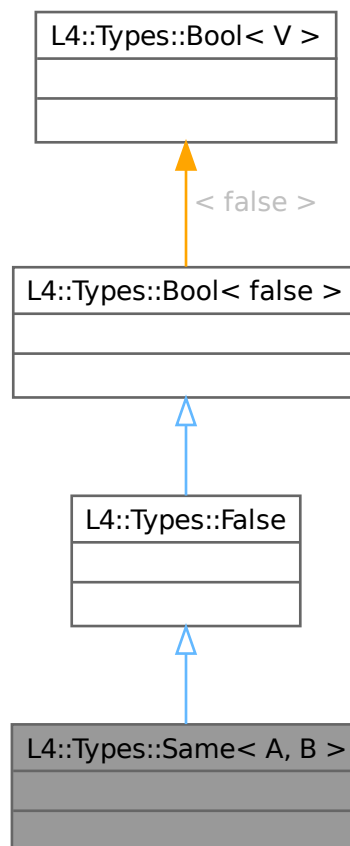
- [l4/sys/cxx/types](#)

15.226 L4::Types::Same< A, B > Struct Template Reference

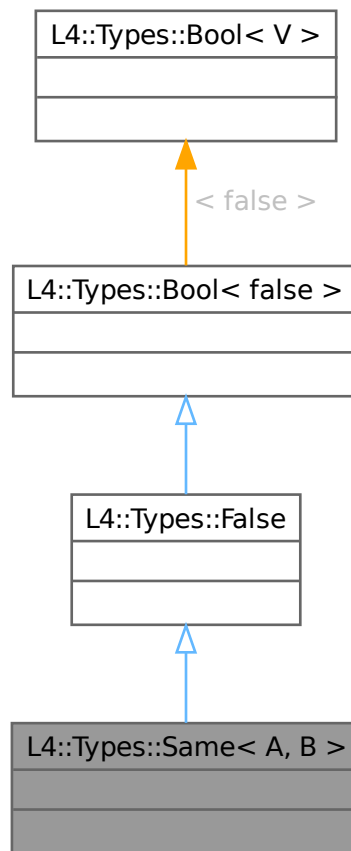
Compare two data types for equality.

```
#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.

15.226.1 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 324 of file [types](#).

The documentation for this struct was generated from the following file:

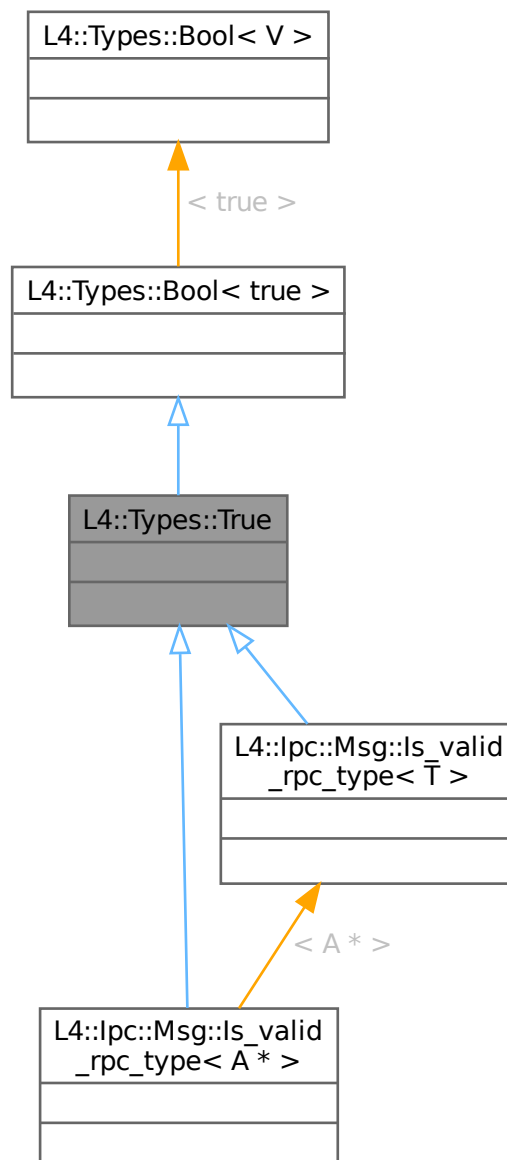
- [l4/sys/cxx/types](#)

15.227 L4::Types::True Struct Reference

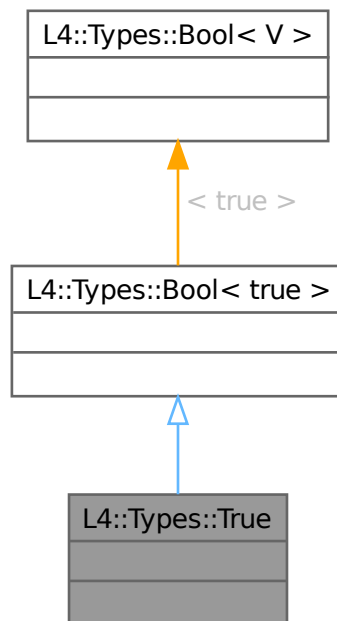
[True](#) meta value.

```
#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.

15.227.1 Detailed Description

True meta value.

Definition at line 312 of file [types](#).

The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

15.228 L4::Uart Class Reference

[Uart](#) driver abstraction.

```
#include <uart_base.h>
```

Inherited by L4::Uart_16550, L4::Uart_cadence, L4::Uart_dcc_v6, L4::Uart_dm, L4::Uart_dummy, L4::Uart_geni, L4::Uart_imx, L4::Uart_leon3, L4::Uart_linfex, L4::Uart_lpuart, L4::Uart_mvebu, L4::Uart_of, L4::Uart_omap35x, L4::Uart_pl011, L4::Uart_s3c, L4::Uart_sa1000, and L4::Uart_sh.

Collaboration diagram for L4::Uart:

L4::Uart
<ul style="list-style-type: none"> + startup() + shutdown() + change_mode() + get_char() + char_avail() + write() + irq_ack() + enable_rx_irq() + mode() + rate() # generic_write()

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.

15.228.1 Detailed Description

[Uart](#) driver abstraction.

Definition at line 25 of file [uart_base.h](#).

15.228.2 Member Function Documentation

15.228.2.1 `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.

15.228.2.2 `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.

15.228.2.3 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.

Definition at line 116 of file [uart_base.h](#).

15.228.2.4 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.

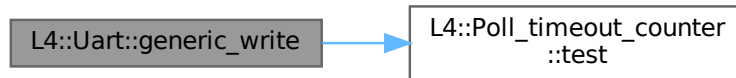
Returns

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:



15.228.2.5 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.

15.228.2.6 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](#).

15.228.2.7 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](#).

15.228.2.8 shutdown()

```
virtual void L4::Uart::shutdown ( ) [pure virtual]
```

Terminate the UART driver.

This includes disabling of interrupts.

15.228.2.9 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.

15.228.2.10 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.

The documentation for this class was generated from the following file:

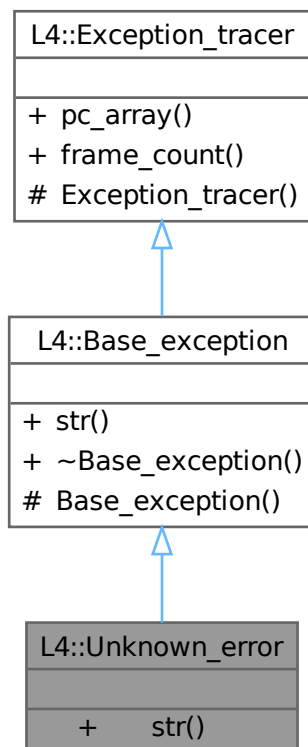
- pkg/drivers-frst/uart/include/uart_base.h

15.229 L4::Unknown_error Class Reference

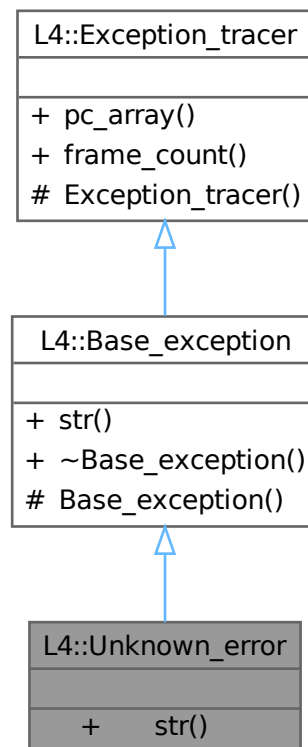
[Exception](#) for an unknown condition.

```
#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 ()` `throw ()`
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.

15.229.1 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:

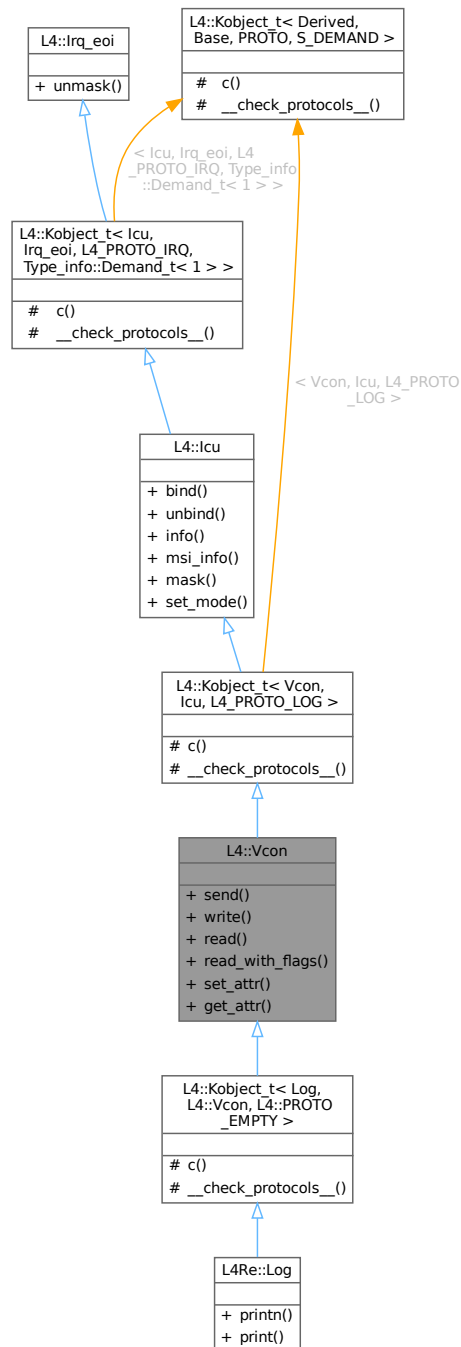
- [l4/cxx/exceptions](#)

15.230 L4::Vcon Class Reference

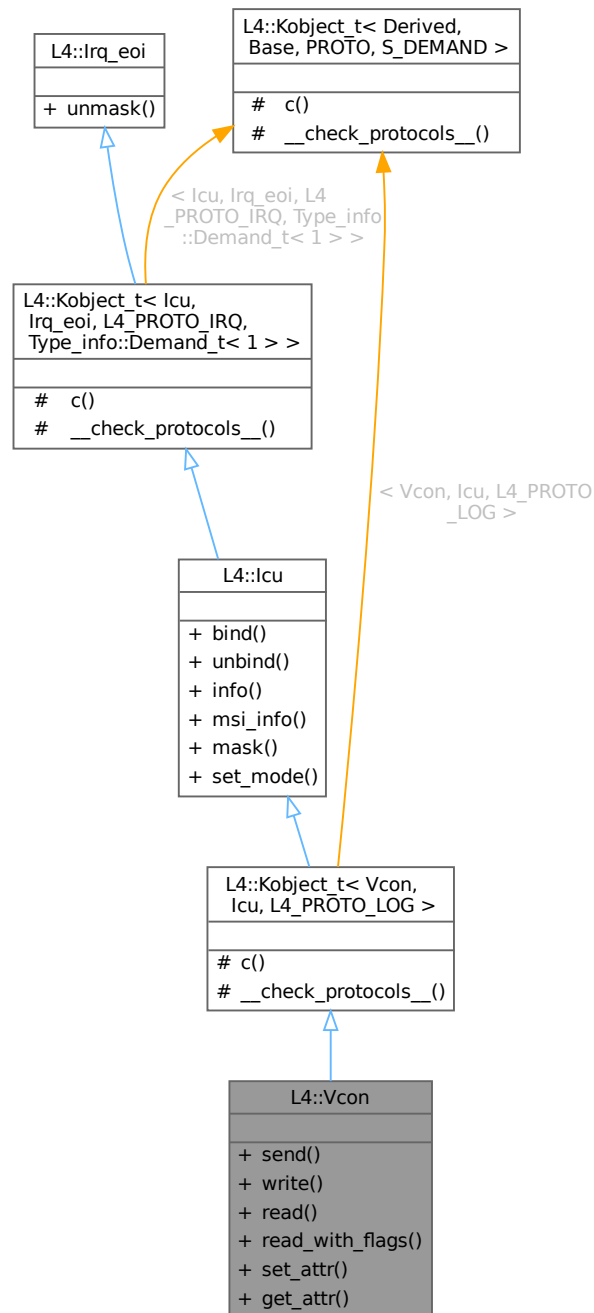
C++ [L4 Vcon](#) interface, see [Virtual Console](#) for the C interface.

```
#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.

15.230.1 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](#).

15.230.2 Member Function Documentation

15.230.2.1 [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 l4_utcb .

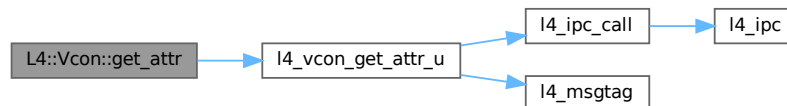
Returns

Syscall return tag.

Definition at line 162 of file [vcon](#).

References [l4_vcon_get_attr_u\(\)](#).

Here is the call graph for this function:



15.230.2.2 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 l4_utcb .

Return values

-L4_EPERM	The Vcon instance requires the L4_CAP_FPAGE_W 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:



15.230.2.3 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 l4_utcb .

Return values

<code>-L4_EPERM</code>	The Vcon instance requires the L4_CAP_FPAGE_W right on the capability used to invoke this operation and this right is not present.
<code>>size</code>	More bytes to read, <i>size</i> bytes are in the buffer <i>buf</i> .
<code><=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](#).

15.230.2.4 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 l4_utcb .

Returns

Syscall return tag

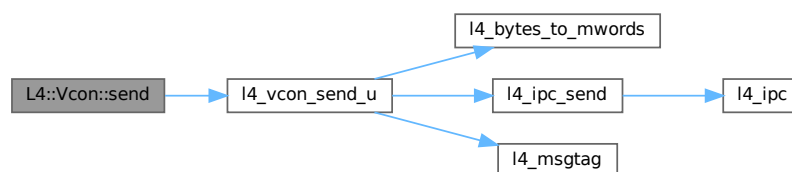
Note

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:



15.230.2.5 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 l4_utcb .

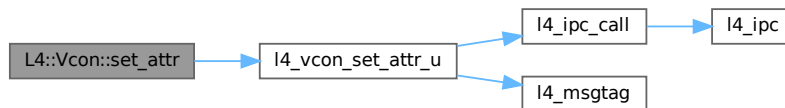
Returns

Syscall return tag.

Definition at line 149 of file [vcon](#).

References [l4_vcon_set_attr_u\(\)](#).

Here is the call graph for this function:



15.230.2.6 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 l4_utcb .

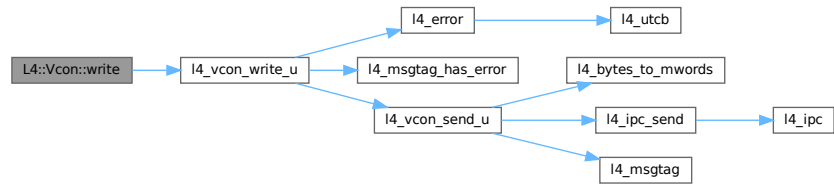
Return values

<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:

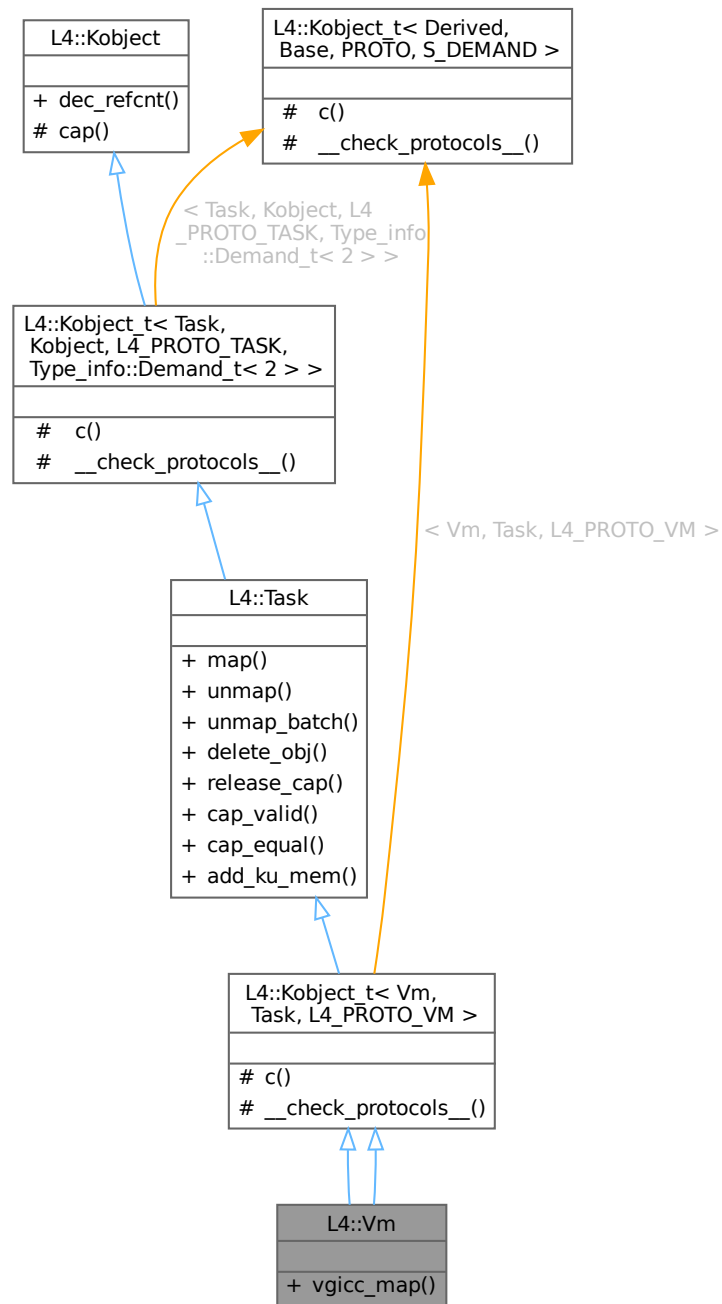
- `l4/sys/vcon`

15.231 L4::Vm Class Reference

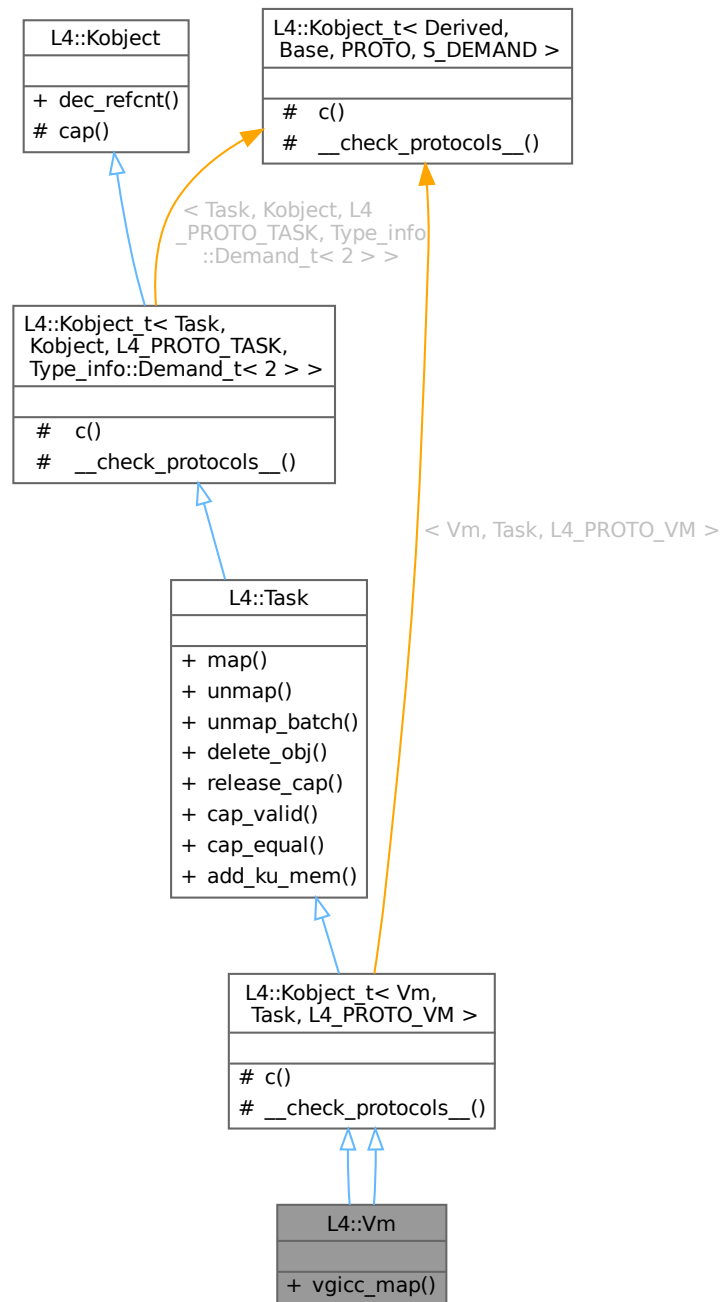
Virtual machine host address space.

```
#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 const &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.

15.231.1 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](#).

15.231.2 Member Function Documentation

15.231.2.1 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 l4_utcb .

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:

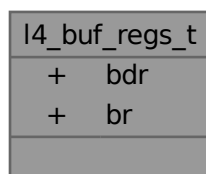
- [l4/sys/__vm-arm.h](#)
- [l4/sys/vm](#)

15.232 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.

15.232.1 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](#)

15.233 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

15.233.1 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](#).

15.233.2 Field Documentation

15.233.2.1 flags

```
l4\_umword\_t l4_exc_regs_t::flags
```

rflags

eflags

Definition at line 80 of file [utcb.h](#).

15.233.2.2 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:

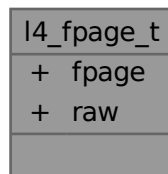
- [arm/l4/sys/utcb.h](#)
- [amd64/l4/sys/utcb.h](#)
- [x86/l4/sys/utcb.h](#)

15.234 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.

15.234.1 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:

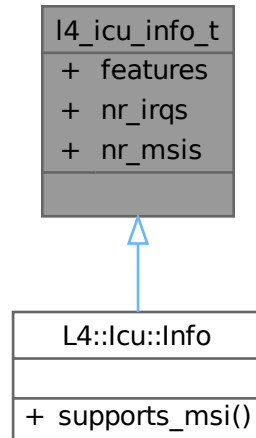
- [l4/sys/__l4_fpage.h](#)

15.235 I4_icu_info_t Struct Reference

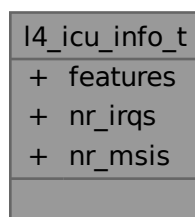
Info structure for an ICU.

```
#include <icu.h>
```

Inheritance diagram for I4_icu_info_t:



Collaboration diagram for I4_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,.

15.235.1 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](#).

15.235.2 Field Documentation

15.235.2.1 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](#)

15.236 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](#):

l4_icu_msi_info_t
<ul style="list-style-type: none">+ msi_addr+ msi_data

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.

15.236.1 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:

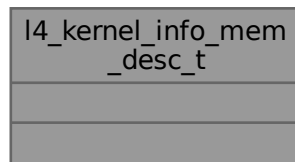
- [l4/sys/icu.h](#)

15.237 l4_kernel_info_mem_desc_t Struct Reference

Memory descriptor data structure.

```
#include <memdesc.h>
```

Collaboration diagram for `l4_kernel_info_mem_desc_t`:



15.237.1 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](#)

15.238 l4_kernel_info_t Struct Reference

[L4 Kernel Interface Page](#).

```
#include <__kip-32bit.h>
```

Collaboration diagram for l4_kernel_info_t:

l4_kernel_info_t
+ magic
+ version
+ offset_version_strings
+ fill0
+ kip_sys_calls
+ fill1
+ scheduler_granularity
+ _res00
+ sigma0_esp
+ sigma0_eip
and 23 more...

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** [3]
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** [7]
reserved
- [l4_umword_t](#) **_res_a0** [1]
reserved
- [l4_umword_t](#) **_res_b0** [2]
reserver

15.238.1 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:

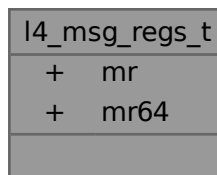
- [l4/sys/__kip-32bit.h](#)
- [l4/sys/__kip-64bit.h](#)

15.239 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.

15.239.1 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](#)

15.240 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

15.240.1 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 [162](#) of file [types.h](#).

15.240.2 Member Function Documentation

15.240.2.1 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 [180](#) of file [types.h](#).

References [raw](#).

The documentation for this struct was generated from the following file:

- [l4/sys/types.h](#)

15.241 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.

15.241.1 Detailed Description

CPU sets.

Examples

[examples/sys/migrate/thread_migrate.cc](#).

Definition at line 69 of file [scheduler.h](#).

15.241.2 Member Function Documentation

15.241.2.1 granularity()

```
unsigned char l4_sched_cpu_set_t::granularity ( ) const [inline]
```

Returns

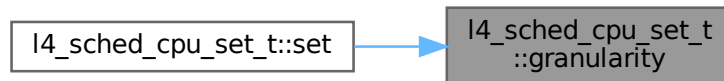
Get granularity value

Definition at line 92 of file [scheduler.h](#).

References [gran_offset](#).

Referenced by [set\(\)](#).

Here is the caller graph for this function:



15.241.2.2 offset()

```
unsigned l4_sched_cpu_set_t::offset ( ) const [inline]
```

Returns

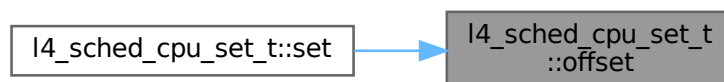
Get offset value

Definition at line 94 of file [scheduler.h](#).

References [gran_offset](#).

Referenced by [set\(\)](#).

Here is the caller graph for this function:



15.241.2.3 set()

```
void l4_sched_cpu_set_t::set (
    unsigned char granularity,
    unsigned offset ) [inline]
```

Set offset and granularity.

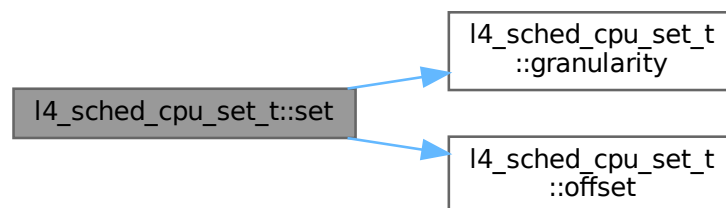
Parameters

<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:



15.241.3 Field Documentation

15.241.3.1 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:

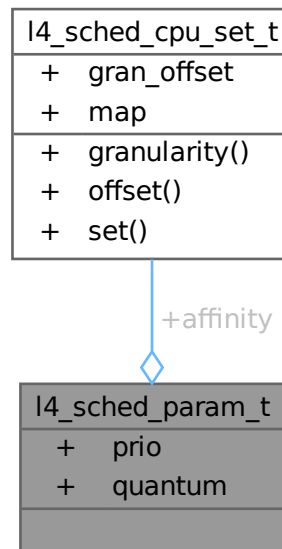
- [l4/sys/scheduler.h](#)

15.242 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.

15.242.1 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 [179](#) of file [scheduler.h](#).

The documentation for this struct was generated from the following file:

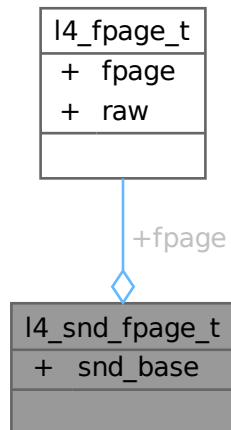
- [l4/sys/scheduler.h](#)

15.243 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.

15.243.1 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:

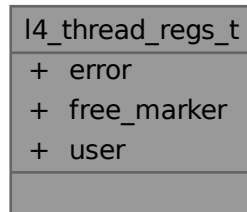
- `l4/sys/__l4_fpage.h`

15.244 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)

15.244.1 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:

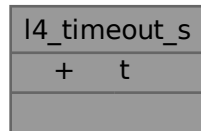
- [l4/sys/utcb.h](#)

15.245 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

15.245.1 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 47 of file [__timeout.h](#).

The documentation for this struct was generated from the following file:

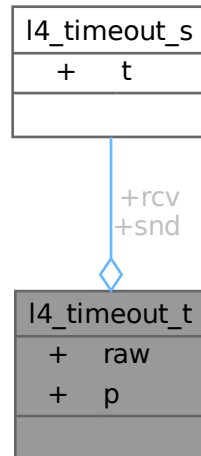
- `l4/sys/__timeout.h`

15.246 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

15.246.1 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 59 of file `__timeout.h`.

The documentation for this union was generated from the following file:

- `l4/sys/__timeout.h`

15.247 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

15.247.1 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](#).

15.247.2 Member Function Documentation

15.247.2.1 `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:

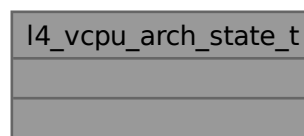
- [l4/sys/vcon.h](#)

15.248 `l4_vcpu_arch_state_t` Struct Reference

Architecture-specific vCPU state.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for `l4_vcpu_arch_state_t`:



15.248.1 Detailed Description

Architecture-specific vCPU state.

Definition at line 62 of file [__vcpu-arch.h](#).

The documentation for this struct was generated from the following files:

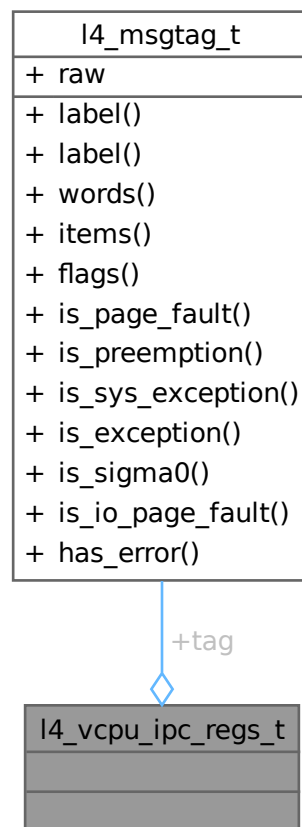
- [arm/l4/sys/__vcpu-arch.h](#)
- [amd64/l4/sys/__vcpu-arch.h](#)

15.249 l4_vcpu_ipc_regs_t Struct Reference

vCPU message registers.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for l4_vcpu_ipc_regs_t:



15.249.1 Detailed Description

vCPU message registers.

Definition at line 71 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](#)

15.250 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

15.250.1 Detailed Description

vCPU registers.

Definition at line 43 of file `__vcpu-arch.h`.

15.250.2 Field Documentation

15.250.2.1 ax

`l4_umword_t l4_vcpu_regs_t::ax`

rax register

eax register

Definition at line 75 of file `__vcpu-arch.h`.

15.250.2.2 bp

`l4_umword_t l4_vcpu_regs_t::bp`

rbp register

ebp register

Definition at line 70 of file `__vcpu-arch.h`.

15.250.2.3 bx

`l4_umword_t l4_vcpu_regs_t::bx`

rbx register

ebx register

Definition at line 72 of file `__vcpu-arch.h`.

15.250.2.4 cx

`l4_umword_t l4_vcpu_regs_t::cx`

rcx register

ecx register

Definition at line 74 of file `__vcpu-arch.h`.

15.250.2.5 di

`l4_umword_t l4_vcpu_regs_t::di`

rdi register

edi register

Definition at line 68 of file [__vcpu-arch.h](#).

15.250.2.6 dx

`l4_umword_t l4_vcpu_regs_t::dx`

rdx register

edx register

Definition at line 73 of file [__vcpu-arch.h](#).

15.250.2.7 si

`l4_umword_t l4_vcpu_regs_t::si`

rsi register

esi register

Definition at line 69 of file [__vcpu-arch.h](#).

The documentation for this struct was generated from the following files:

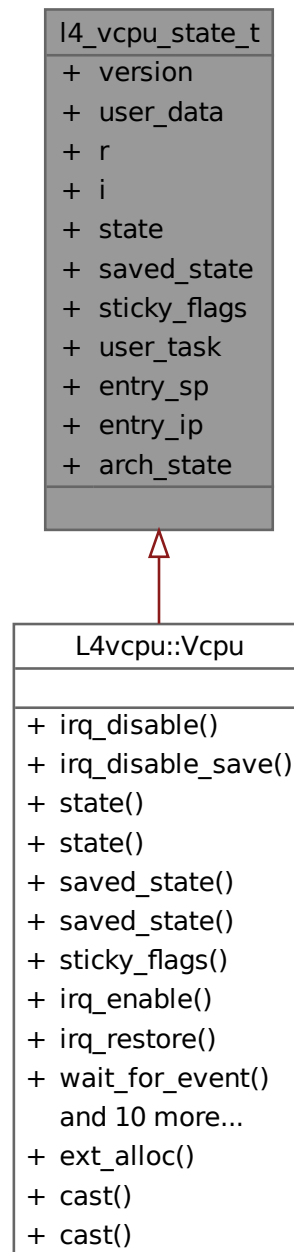
- [arm/l4/sys/__vcpu-arch.h](#)
- [amd64/l4/sys/__vcpu-arch.h](#)
- [x86/l4/sys/__vcpu-arch.h](#)

15.251 l4_vcpu_state_t Struct Reference

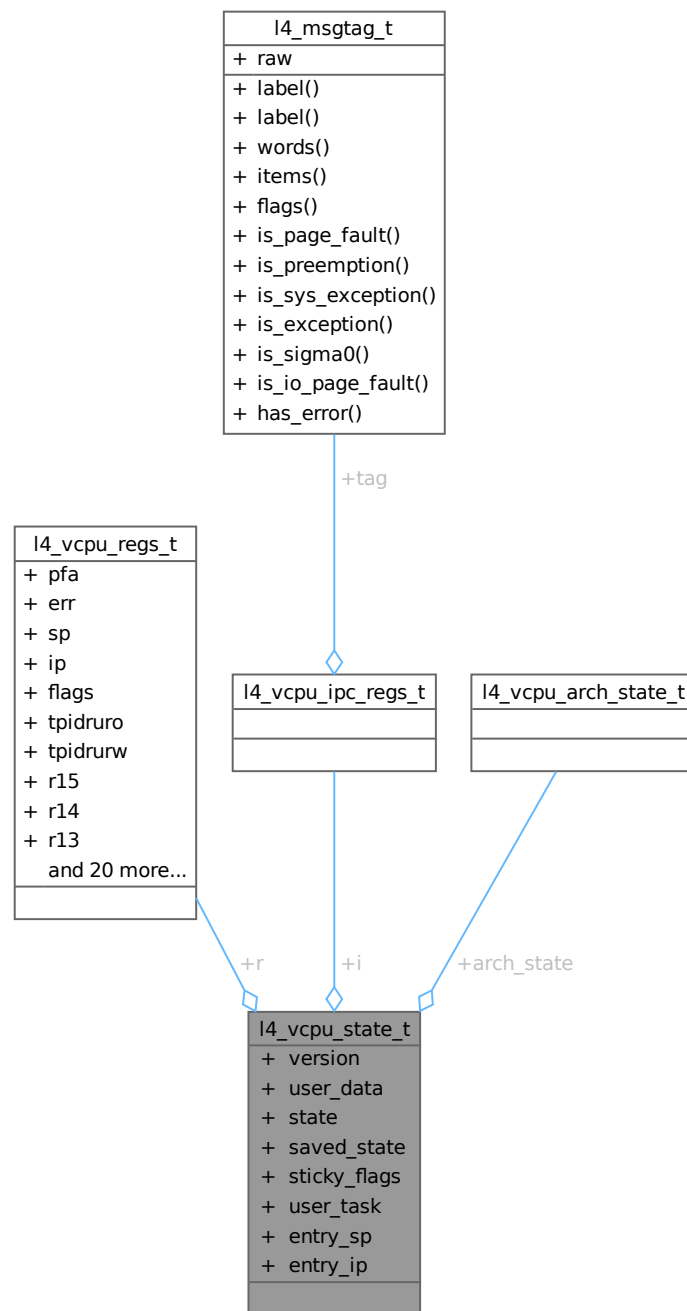
State of a vCPU.

```
#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](#)

- [l4_vcpu_ipc_regs_t](#) **i**
Register state.
- [l4_uint16_t](#) **state**
IPC state.
- [l4_uint16_t](#) **current_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.

15.251.1 Detailed Description

State of a vCPU.

Definition at line 86 of file [vcpu.h](#).

15.251.2 Field Documentation

15.251.2.1 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:

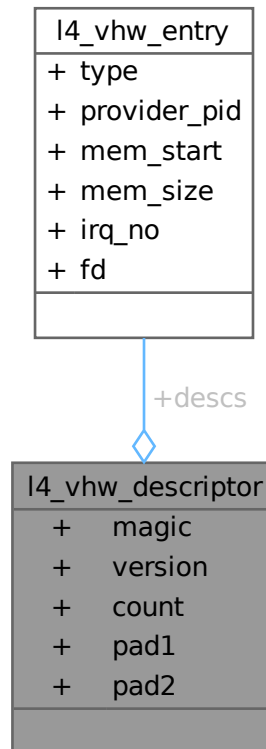
- [l4/sys/vcpu.h](#)

15.252 l4_vhw_descriptor Struct Reference

Virtual hardware devices description.

```
#include <vhw.h>
```

Collaboration diagram for l4_vhw_descriptor:



Data Fields

- [l4_uint32_t](#) **magic**
Magic.
- [l4_uint8_t](#) **version**
Version of the descriptor.
- [l4_uint8_t](#) **count**
Number of entries.
- [l4_uint8_t](#) **pad1**
padding
- [l4_uint8_t](#) **pad2**
padding
- struct [l4_vhw_entry](#) **descs** []
Array of device descriptions.

15.252.1 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](#)

15.253 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.

15.253.1 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:

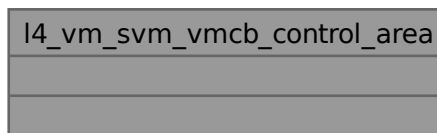
- [l4/sys/vhw.h](#)

15.254 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:



15.254.1 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:

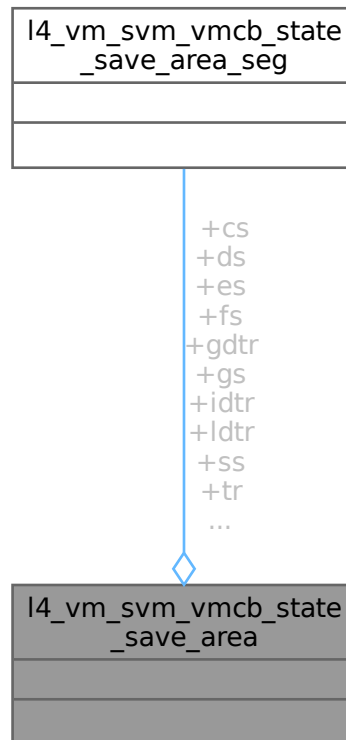
- [l4/sys/__vm-svm.h](#)

15.255 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:



15.255.1 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:

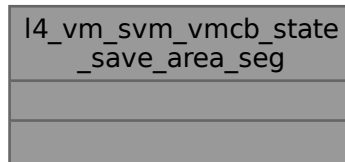
- `l4/sys/__vm-svm.h`

15.256 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:



15.256.1 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:

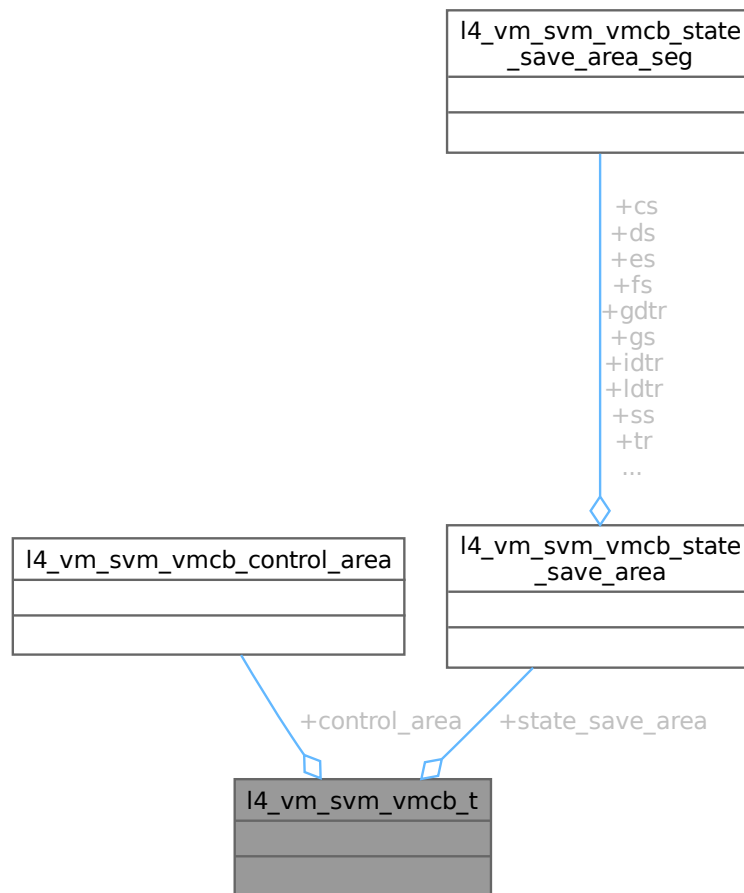
- l4/sys/__vm-svm.h

15.257 l4_vm_svm_vmcb_t Struct Reference

Control structure for SVM VMs.

```
#include <__vm-svm.h>
```

Collaboration diagram for `l4_vm_svm_vmcb_t`:



15.257.1 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:

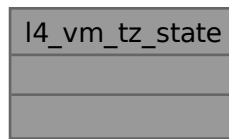
- `l4/sys/__vm-svm.h`

15.258 l4_vm_tz_state Struct Reference

state structure for TrustZone VMs

```
#include <vm.h>
```


Collaboration diagram for l4_vm_tz_state:



15.258.1 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:

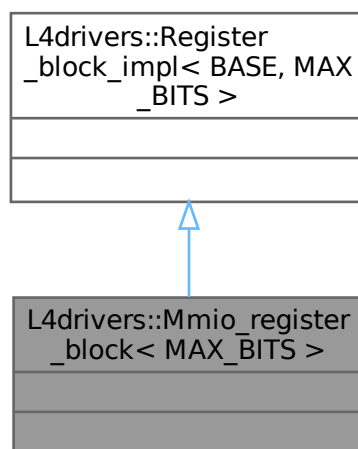
- [arm/l4/sys/vm.h](#)

15.259 L4drivers::Mmio_register_block< MAX_BITS > Struct Template Reference

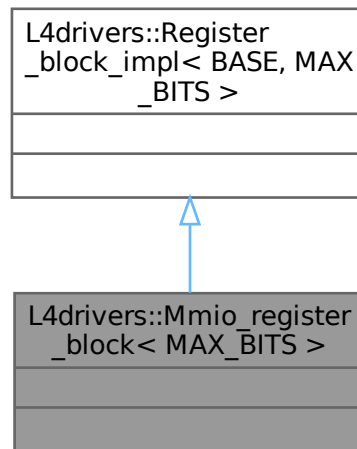
An MMIO block with up to 64-bit register access (32-bit default) and little endian byte order.

```
#include <hw_mmio_register_block>
```

Inheritance diagram for L4drivers::Mmio_register_block< MAX_BITS >:



Collaboration diagram for L4drivers::Mmio_register_block< MAX_BITS >:



15.259.1 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:

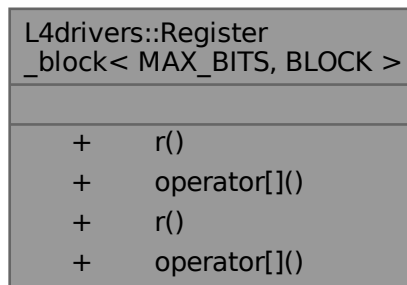
- pkg/drivers-frst/include/hw_mmio_register_block

15.260 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.

15.260.1 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](#).

15.260.2 Member Function Documentation

15.260.2.1 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](#).

15.260.2.2 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](#).

15.260.2.3 `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](#).

15.260.2.4 `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](#).

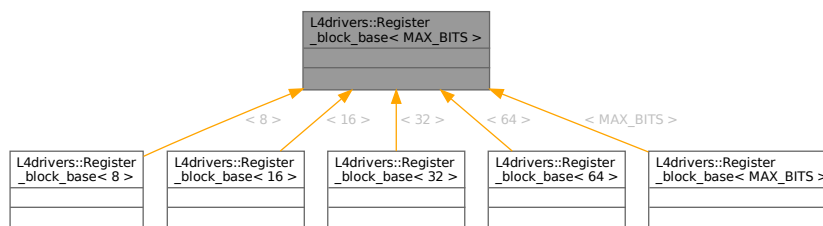
The documentation for this class was generated from the following file:

- pkg/drivers-frst/include/hw_register_block

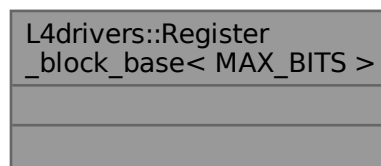
15.261 L4drivers::Register_block_base< MAX_BITS > Struct Template Reference

Abstract register block interface.

Inheritance diagram for L4drivers::Register_block_base< MAX_BITS >:



Collaboration diagram for L4drivers::Register_block_base< MAX_BITS >:



15.261.1 Detailed Description

```
template<unsigned MAX_BITS = 32>
struct L4drivers::Register_block_base< MAX_BITS >
```

Abstract register block interface.

Template Parameters

<code>MAX_BITS</code>	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](#).

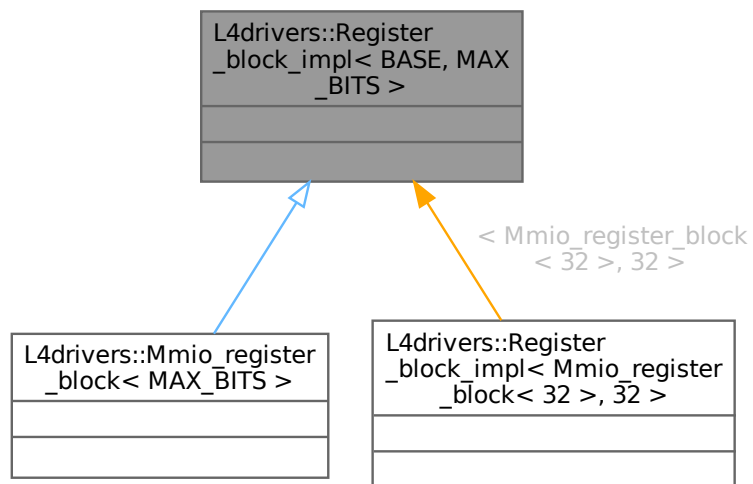
The documentation for this struct was generated from the following file:

- `pkg/drivers-frst/include/hw_register_block`

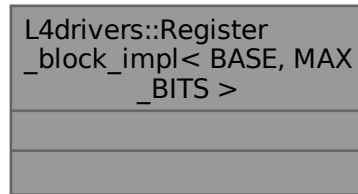
15.262 L4drivers::Register_block_impl< BASE, MAX_BITS > Struct Template Reference

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 >:



15.262.1 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 Register_block_impl .
<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:

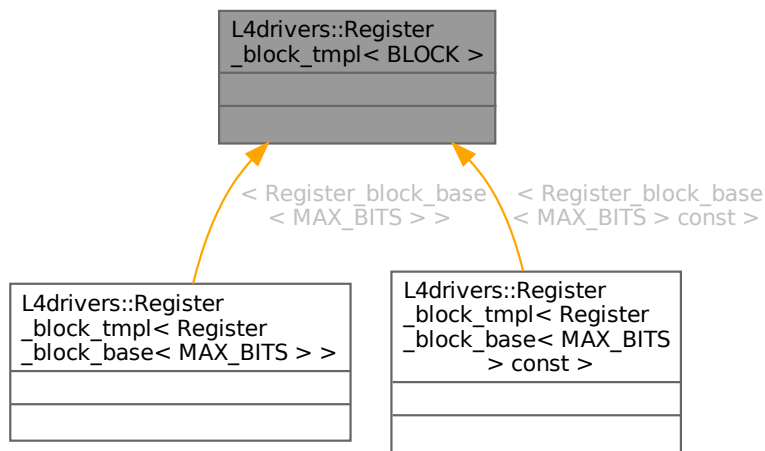
- pkg/drivers-frst/include/hw_register_block

15.263 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 >:



15.263.1 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 Register_block_base 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:

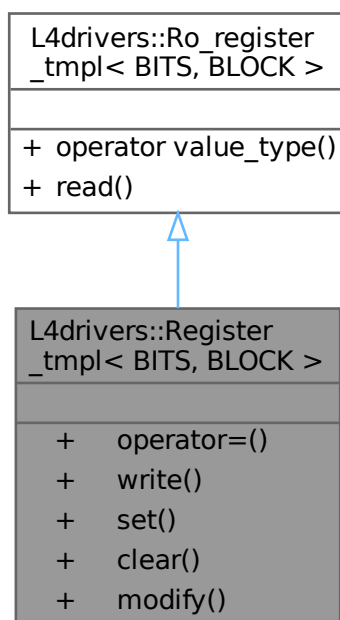
- pkg/drivers-frst/include/hw_register_block

15.264 L4drivers::Register_tmpl< BITS, BLOCK > Class Template Reference

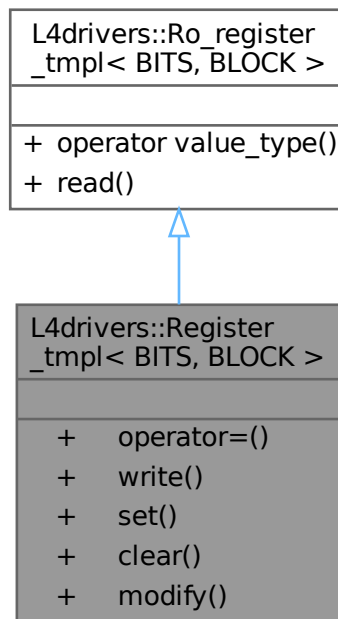
Single hardware register inside a [Register_block_base](#) interface.

```
#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.

15.264.1 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 Register_block_base 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](#).

15.264.2 Member Function Documentation

15.264.2.1 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](#).

15.264.2.2 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](#).

15.264.2.3 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](#).

15.264.2.4 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](#).

15.264.2.5 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:

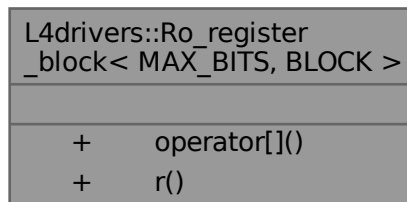
- pkg/drivers-frst/include/hw_register_block

15.265 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.

15.265.1 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](#).

15.265.2 Member Function Documentation

15.265.2.1 `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](#).

15.265.2.2 `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:

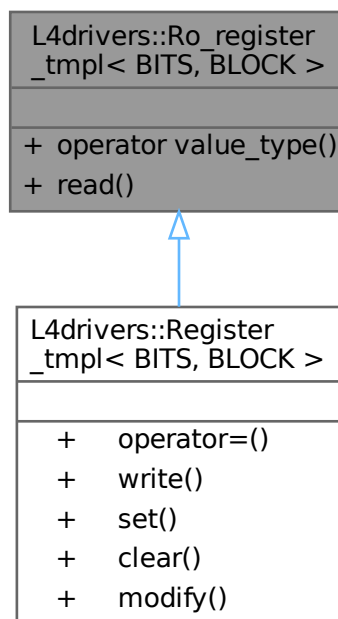
- pkg/drivers-frst/include/hw_register_block

15.266 L4drivers::Ro_register_tmpl< BITS, BLOCK > Class Template Reference

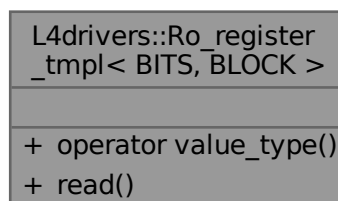
Single read only register inside a [Register_block_base](#) interface.

```
#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.

15.266.1 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 Register_block_base 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](#).

15.266.2 Member Function Documentation

15.266.2.1 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](#).

15.266.2.2 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](#)

15.267 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"> + alloc() + alloc() + free() + ~Cap_alloc() + get_cap_alloc()

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.

15.267.1 Detailed Description

Capability allocator interface.

Definition at line [41](#) of file [cap_alloc](#).

15.267.2 Member Function Documentation

15.267.2.1 `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:



15.267.2.2 alloc() [2/2]

```
virtual L4::Cap< void > L4Re::Cap_alloc::alloc ( ) [pure virtual], [noexcept]
```

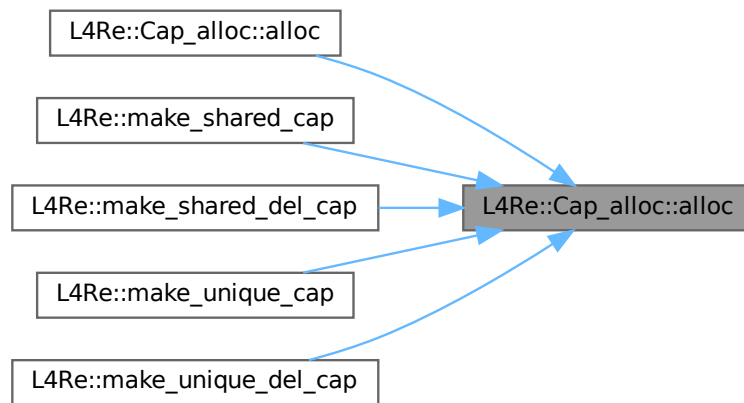
Allocate a capability.

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:

**15.267.2.3 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> .

15.267.2.4 `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

<code>ca</code>	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:

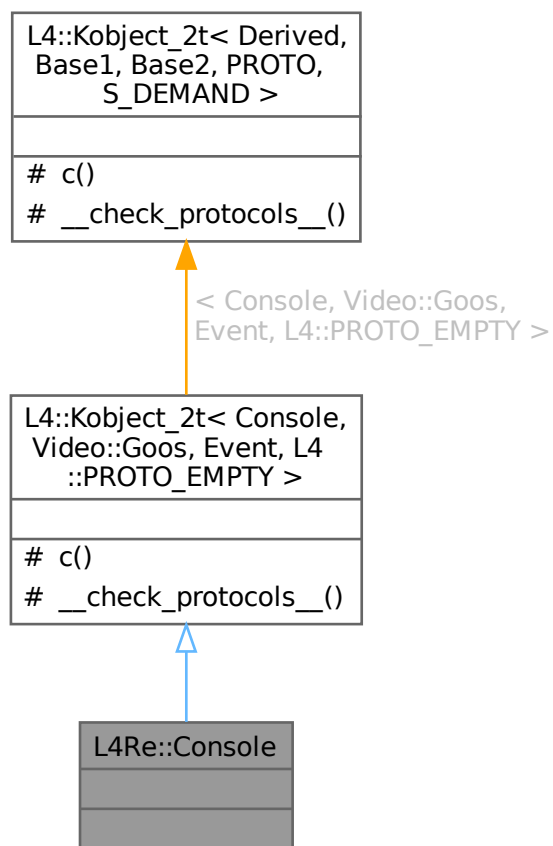
- [l4/re/cap_alloc](#)

15.268 L4Re::Console Class Reference

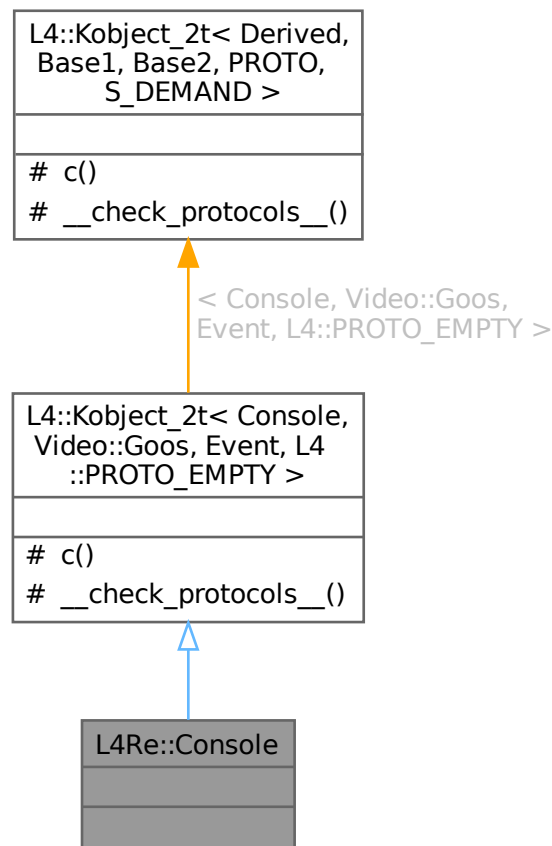
[Console](#) class.

```
#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 > [__lface](#)
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< [__lface](#) >, Typeid::Merge_list< typename Base1::__lface_list, typename Base2::__lface_list > > [__lface_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.

15.268.1 Detailed Description

`Console` class.

Definition at line 39 of file `console`.

The documentation for this class was generated from the following file:

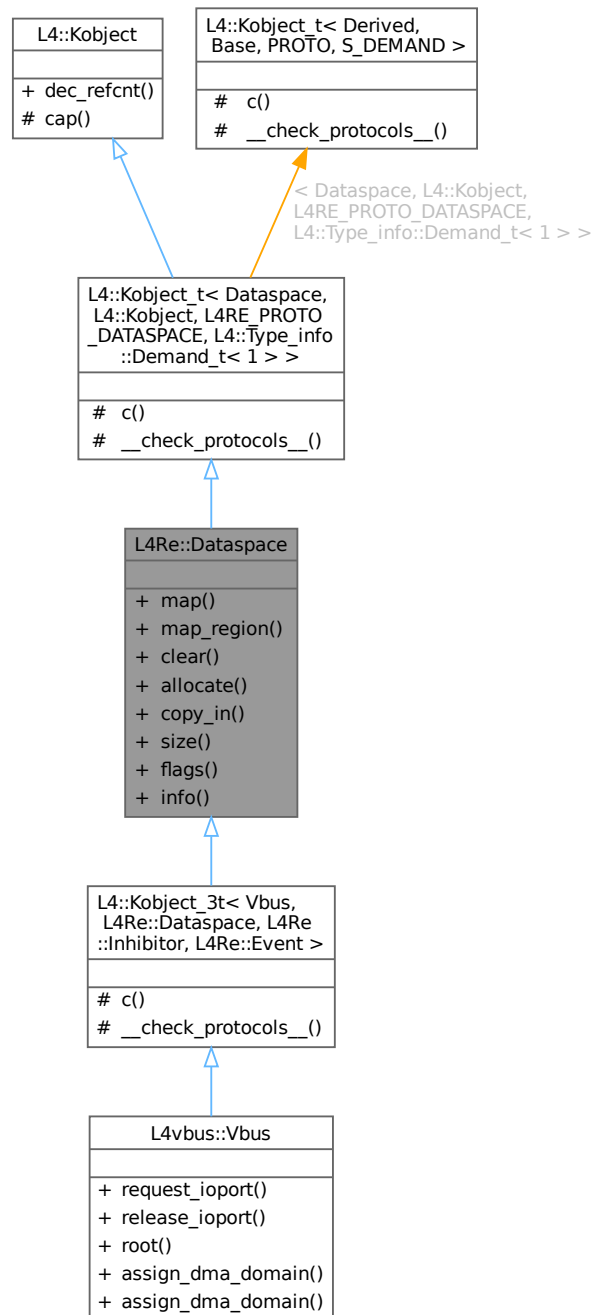
- `l4/re/console`

15.269 L4Re::Dataspace Class Reference

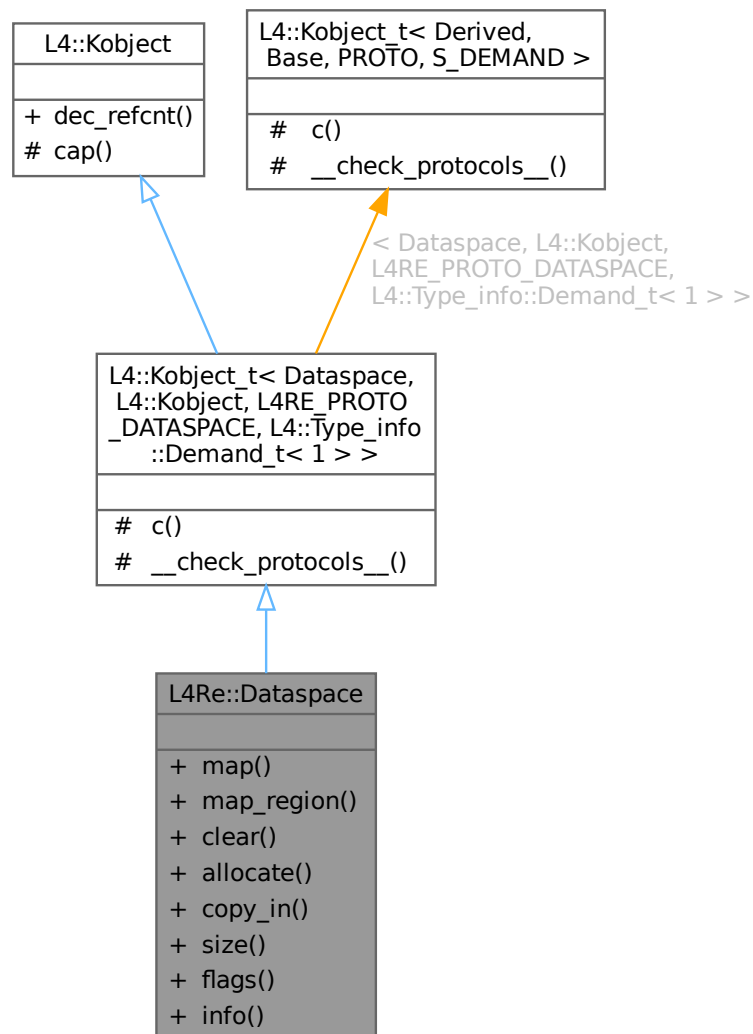
Interface for memory-like objects.

```
#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) 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) 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.

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.

15.269.1 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 60 of file [dataspace](#).

15.269.2 Member Function Documentation

15.269.2.1 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.
<i><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.

15.269.2.2 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

≥ 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

Zeroes out the memory. Depending on the type of memory the memory could also be deallocated and replaced by a shared zero-page.

15.269.2.3 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
<code>-L4_EACCESS</code>	No <code>L4_CAP_FPAGE_W</code> right on the destination dataspace.
<code>-L4_EINVAL</code>	Invalid parameter supplied.
< 0	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.

15.269.2.4 flags()

```
Dataspace::Flags L4Re::Dataspace::flags ( ) const [noexcept]
```

Get flags of the dataspace.

Return values

≥ 0	Flags of the dataspace
< 0	IPC errors

See also

[L4Re::Dataspace::F::Flags](#)

Definition at line 119 of file [dataspace_impl.h](#).

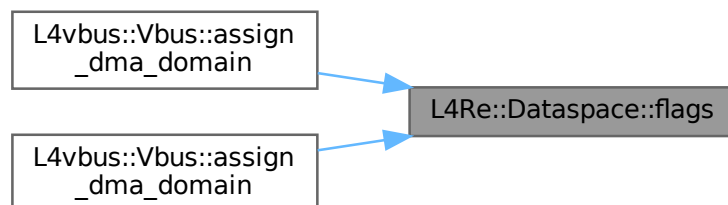
References [L4Re::Dataspace::Stats::flags](#), and [info\(\)](#).

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:



15.269.2.5 info()

```
long L4Re::Dataspace::info (  
    Stats * stats )
```

Get information on the dataspace.

Parameters

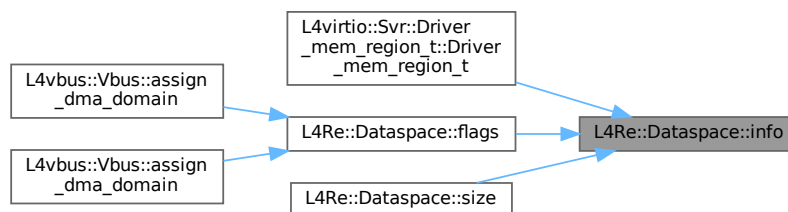
out	stats	Dataspace information
-----	-------	---------------------------------------

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:



15.269.2.6 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 ) const [noexcept]

```

Request a flex-page mapping from the dataspace.

Parameters

<i>offset</i>	Offset to start within dataspace
<i>flags</i>	Dataspace flags, see L4Re::Dataspace::F::Flags .
<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.)

Return values

<i>L4_EOK</i>	Success
<i>-L4_ERANGE</i>	Invalid offset.
<i>-L4_EPERM</i>	Insufficient permission to map with requested rights.
<0	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 94 of file [dataspace_impl.h](#).

References [L4_LOG2_PAGESIZE](#).

15.269.2.7 map_region()

```
long L4Re::Dataspace::map_region (
    Dataspace::Offset offset,
    Dataspace::Flags flags,
    Dataspace::Map_addr min_addr,
    Dataspace::Map_addr max_addr ) 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>	Dataspace flags, see L4Re::Dataspace::F::Flags .
<i>min_addr</i>	(Inclusive) start of the receive area.
<i>max_addr</i>	(Exclusive) end of receive area.

Return values

<i>L4_EOK</i>	Success
<i>-L4_ERANGE</i>	Invalid offset or receive area larger than the dataspace.
<i>-L4_EPERM</i>	Insufficient permission to map with requested rights.
<i>< 0</i>	IPC errors

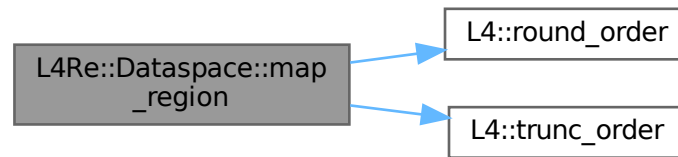
This is a convenience function which maps flex-pages consecutively into the given memory area in the local task. The area is expected to be filled completely. If the dataspace is not large enough to provide the mappings for the entire size of the area, then an error is returned. Mappings may or may not 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 55 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:



15.269.2.8 size()

```
Dataspace::Size L4Re::Dataspace::size ( ) const [noexcept]
```

Get size of a dataspace.

Returns

Size of the dataspace in bytes.

Definition at line 109 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:

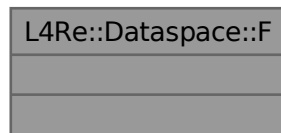
- [l4/re/dataspace](#)
- [l4/re/impl/dataspace_impl.h](#)

15.270 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.

15.270.1 Detailed Description

[Dataspace](#) flags definitions.

Definition at line 67 of file [dataspace](#).

15.270.2 Member Enumeration Documentation

15.270.2.1 anonymous enum

anonymous enum

Enumerator

Caching_shift	shift value for caching flags
---------------	-------------------------------

Definition at line 69 of file [dataspace](#).

15.270.2.2 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 80 of file [dataspace](#).

The documentation for this struct was generated from the following file:

- [l4/re/dataspace](#)

15.271 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

15.271.1 Detailed Description

Information about the dataspace.

Definition at line 135 of file [dataspace](#).

The documentation for this struct was generated from the following file:

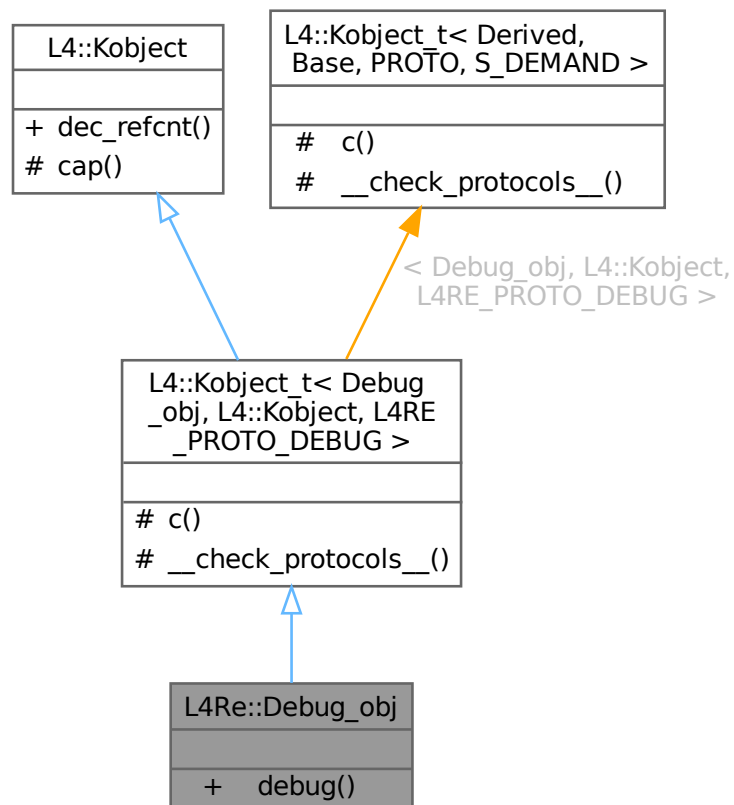
- [l4/re/dataspace](#)

15.272 L4Re::Debug_obj Class Reference

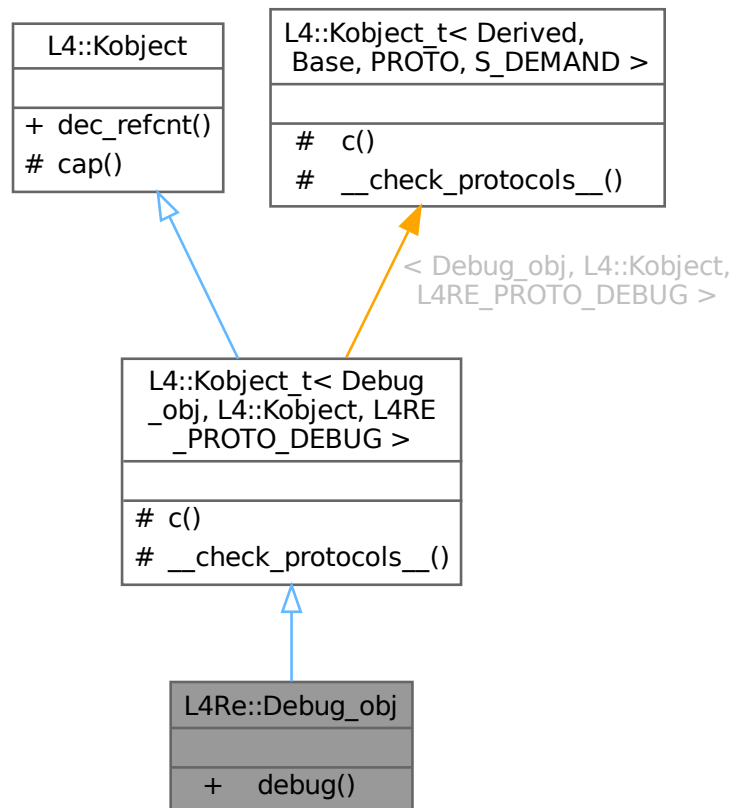
Debug interface.

```
#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.

15.272.1 Detailed Description

Debug interface.

See also

[Debugging API](#) .

Definition at line 51 of file [debug](#).

15.272.2 Member Function Documentation

15.272.2.1 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](#)

15.273 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.

15.273.1 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](#)

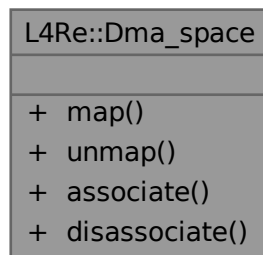
15.274 L4Re::Dma_space Class Reference

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::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.
- 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.

15.274.1 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](#).

15.274.2 Member Typedef Documentation

15.274.2.1 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](#).

15.274.3 Member Enumeration Documentation

15.274.3.1 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 Dma_space . 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 No_sync attribute).
---------	---

Definition at line 87 of file [dma_space](#).

15.274.3.2 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](#).

15.274.3.3 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](#).

15.274.4 Member Function Documentation**15.274.4.1 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) DMA space 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 L4Re::Dma_space::Phys_space 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 L4Re::Dma_space::Space_attrib .

Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EPERM</i>	No L4_CAP_FPAGE_W right on the Dma_space capability.
<i>-L4_EINVAL</i>	
<i>-L4_ENOENT</i>	

Precondition

requires capability rights: {RW}

15.274.4.2 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 L4_CAP_FPAGE_W right on the Dma_space capability.
<i>-L4_ENOENT</i>	

Precondition

requires capability rights: {RW}

15.274.4.3 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 Dma_space::Attribute 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 L4_CAP_FPAGE_W 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\(\)](#)).

15.274.4.4 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 Dma_space::map()).
<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](#)

15.275 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
<ul style="list-style-type: none"> + parent() + mem_alloc() + user_factory() + rm() + log() + main_thread() + task() + factory() + first_free_cap() + utcb_area() and 17 more... + env()

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.

15.275.1 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	L4Re::Parent	L4Re::Env::parent()
user_factory	L4::Factory	L4Re::Env::user_factory()
log	L4Re::Log	L4Re::Env::log()
main_thread	L4::Thread	L4Re::Env::main_thread()
rm	L4Re::Rm	L4Re::Env::rm()
factory	L4::Factory	L4Re::Env::factory()
task	L4::Task	L4Re::Env::task()
scheduler	L4::Scheduler	L4Re::Env::scheduler()

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](#).

15.275.2 Member Function Documentation

15.275.2.1 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>()`

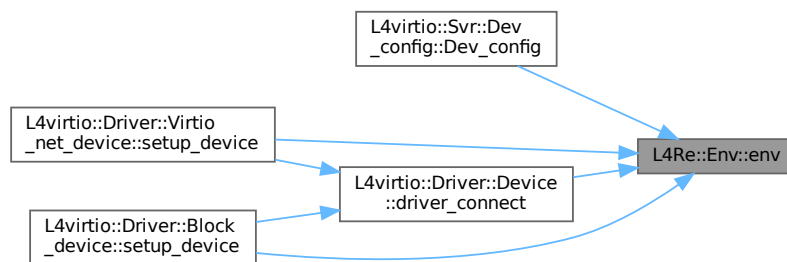
Examples

`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:



15.275.2.2 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`.

15.275.2.3 factory() [2/2]

```
void L4Re::Env::factory (
    L4::Cap< L4::Factory > const & c ) [inline], [noexcept]
```

Set factory object-capability.

Parameters

c	Factory object-capability
---	---------------------------

Definition at line 256 of file [env](#).

References [l4re_env_t::factory](#).

15.275.2.4 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](#).

15.275.2.5 first_free_cap() [2/2]

```
void L4Re::Env::first_free_cap (
    l4_cap_idx_t c ) [inline], [noexcept]
```

Set first available capability selector.

Parameters

c	First capability selector available to the application.
---	---

Definition at line 262 of file [env](#).

References [l4re_env_t::first_free_cap](#).

15.275.2.6 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](#).

15.275.2.7 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](#).

15.275.2.8 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:



15.275.2.9 `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](#).

15.275.2.10 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](#).

15.275.2.11 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](#).

15.275.2.12 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](#).

15.275.2.13 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](#).

15.275.2.14 log() [2/2]

```
void L4Re::Env::log (
    L4::Cap< Log > const & c ) [inline], [noexcept]
```

Set log object-capability.

Parameters

<i>c</i>	Log object-capability
----------	---------------------------------------

Definition at line 244 of file [env](#).

References [l4re_env_t::log](#).

15.275.2.15 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](#).

15.275.2.16 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

c	First thread's object-capability
----------	----------------------------------

Definition at line 250 of file [env](#).

References [l4re_env_t::main_thread](#).

15.275.2.17 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](#).

15.275.2.18 mem_alloc() [2/2]

```
void L4Re::Env::mem_alloc (
    L4::Cap< Mem_alloc > const & c ) [inline], [noexcept]
```

Set memory allocator object-capability.

Parameters

c	Memory allocator object-capability
----------	------------------------------------

Definition at line 232 of file [env](#).

References [l4re_env_t::mem_alloc](#).

15.275.2.19 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](#).

15.275.2.20 parent() [2/2]

```
void L4Re::Env::parent (
    L4::Cap< Parent > const & c ) [inline], [noexcept]
```

Set parent object-capability.

Parameters

c	Parent object-capability
---	--

Definition at line 226 of file [env](#).

References [l4re_env_t::parent](#).

15.275.2.21 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](#).

15.275.2.22 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](#).

15.275.2.23 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](#).

15.275.2.24 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](#).

15.275.2.25 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](#).

15.275.2.26 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](#).

15.275.2.27 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:

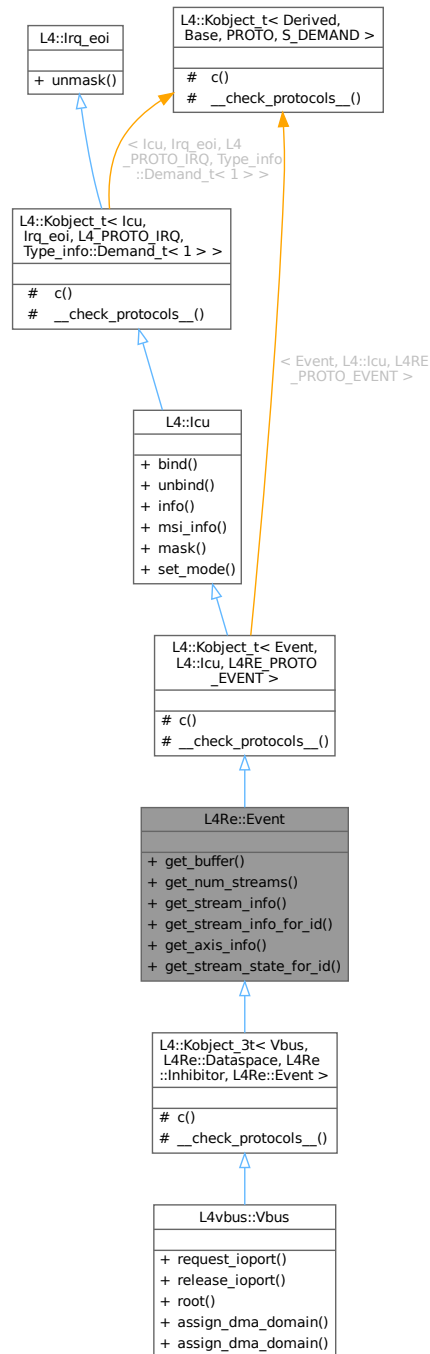
- [l4/re/env](#)

15.276 L4Re::Event Class Reference

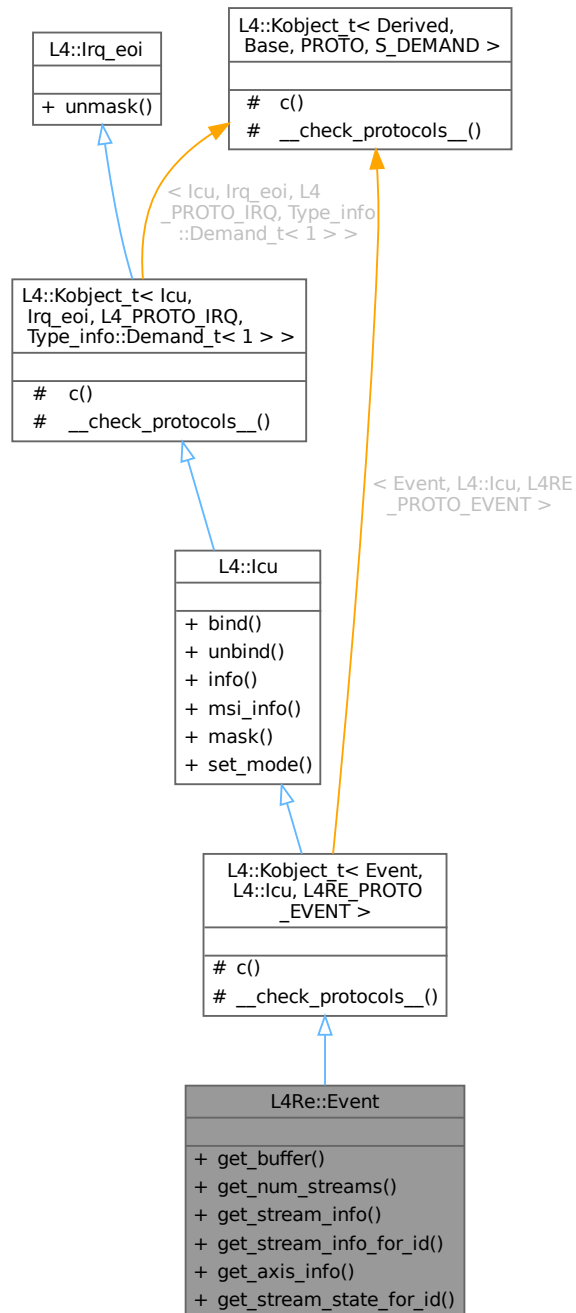
[Event](#) class.

```
#include <event>
```

Inheritance diagram for L4Re::Event:



Collaboration diagram for L4Re::Event:



Public Member Functions

- long `get_buffer` (L4::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::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](#)< [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](#), [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< 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.

15.276.1 Detailed Description

[Event](#) class.

See also

[L4Re Event API](#)

Definition at line 148 of file [event](#).

15.276.2 Member Function Documentation

15.276.2.1 [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</i> ↔ <i>_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

≥ 0	Number of returned axes infos.
< 0	Error code.

Definition at line 208 of file [event](#).

15.276.2.2 get_buffer()

```
long L4Re::Event::get_buffer (
    L4::Ipc::Out< L4::Cap< Dataspace > > ds )
```

Get event signal buffer.

Parameters

out	<i>ds</i>	Event buffer.
-----	-----------	-------------------------------

Return values

0	Success
< 0	Error

15.276.2.3 get_num_streams()

```
long L4Re::Event::get_num_streams ( )
```

Get number of event streams.

Return values

≥ 0	Number of streams.
< 0	Error code.

15.276.2.4 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>	Event stream info.

Return values

0	Success
<0	Error

15.276.2.5 [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>	Event stream info.

Return values

0	Success
<0	Error

15.276.2.6 [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>	Event stream state.

Return values

0	Success
<0	Error

The documentation for this class was generated from the following file:

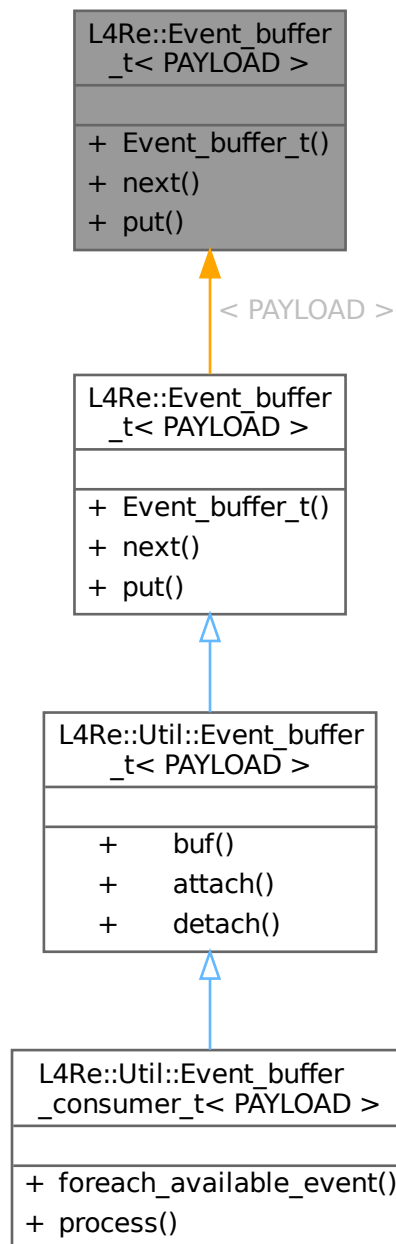
- l4/re/event

15.277 L4Re::Event_buffer_t< PAYLOAD > Class Template Reference

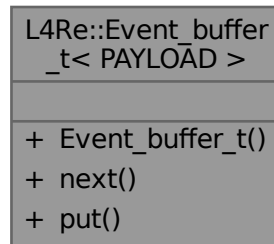
[Event](#) buffer class.

```
#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.

15.277.1 Detailed Description

```
template<typename PAYLOAD = Default_event_payload>
class L4Re::Event_buffer_t< PAYLOAD >
```

[Event](#) buffer class.

Definition at line 256 of file [event](#).

15.277.2 Constructor & Destructor Documentation

15.277.2.1 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](#).

15.277.3 Member Function Documentation

15.277.3.1 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](#).

15.277.3.2 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>	Event to put into the buffer.
-----------	---

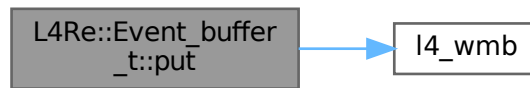
Returns

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:

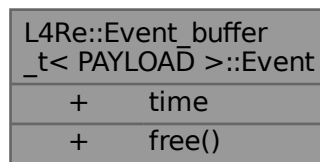
- l4/re/event

15.278 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.

15.278.1 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:

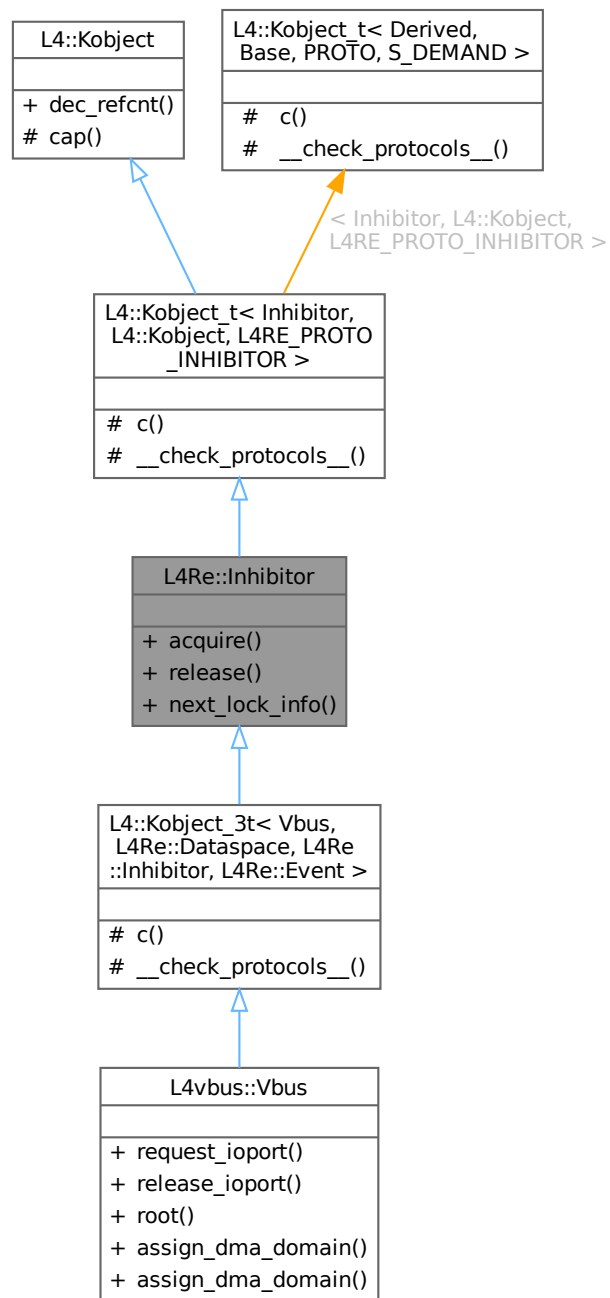
- [l4/re/event](#)

15.279 L4Re::Inhibitor Class Reference

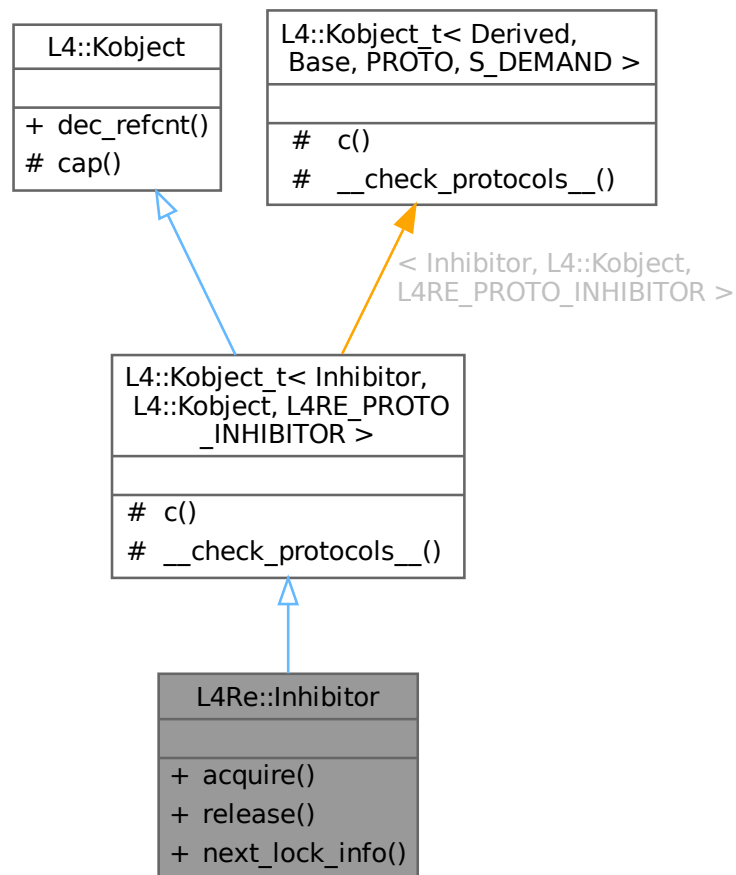
Set of inhibitor locks, which inhibit specific actions when held.

```
#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.

15.279.1 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](#).

15.279.2 Member Enumeration Documentation

15.279.2.1 anonymous enum

anonymous enum

Enumerator

Name_max	The maximum length of a lock's name.
----------	--------------------------------------

Definition at line 44 of file [inhibitor](#).

15.279.3 Member Function Documentation

15.279.3.1 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.

15.279.3.2 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 Name_max 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).
----	--

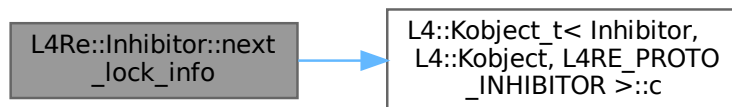
Return values

-L4_ENODEV	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:

15.279.3.3 `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

0	Success
-L4_ENODEV	Lock with the given <i>id</i> does not exist.

The documentation for this class was generated from the following file:

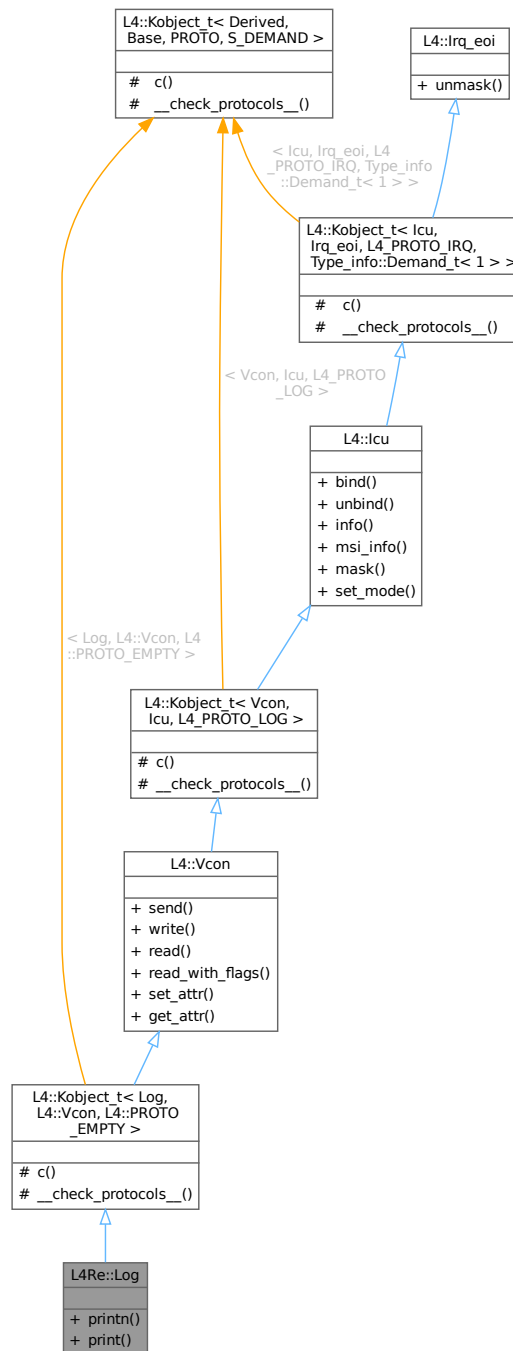
- `l4/re/inhibitor`

15.280 L4Re::Log Class Reference

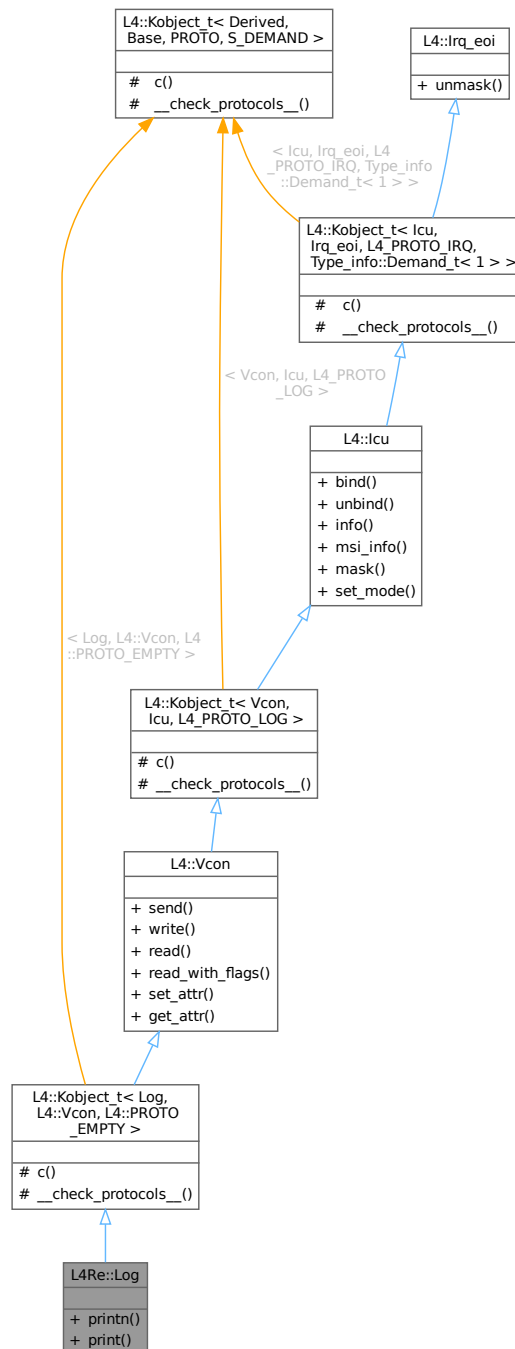
[Log](#) interface class.

```
#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.

15.280.1 Detailed Description

[Log](#) interface class.

Definition at line 44 of file [log](#).

15.280.2 Member Function Documentation

15.280.2.1 print()

```
void L4Re::Log::print (
    char const * string ) const    [noexcept]
```

Print NULL-terminated string.

Parameters

<i>string</i>	string to print
---------------	-----------------

15.280.2.2 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:

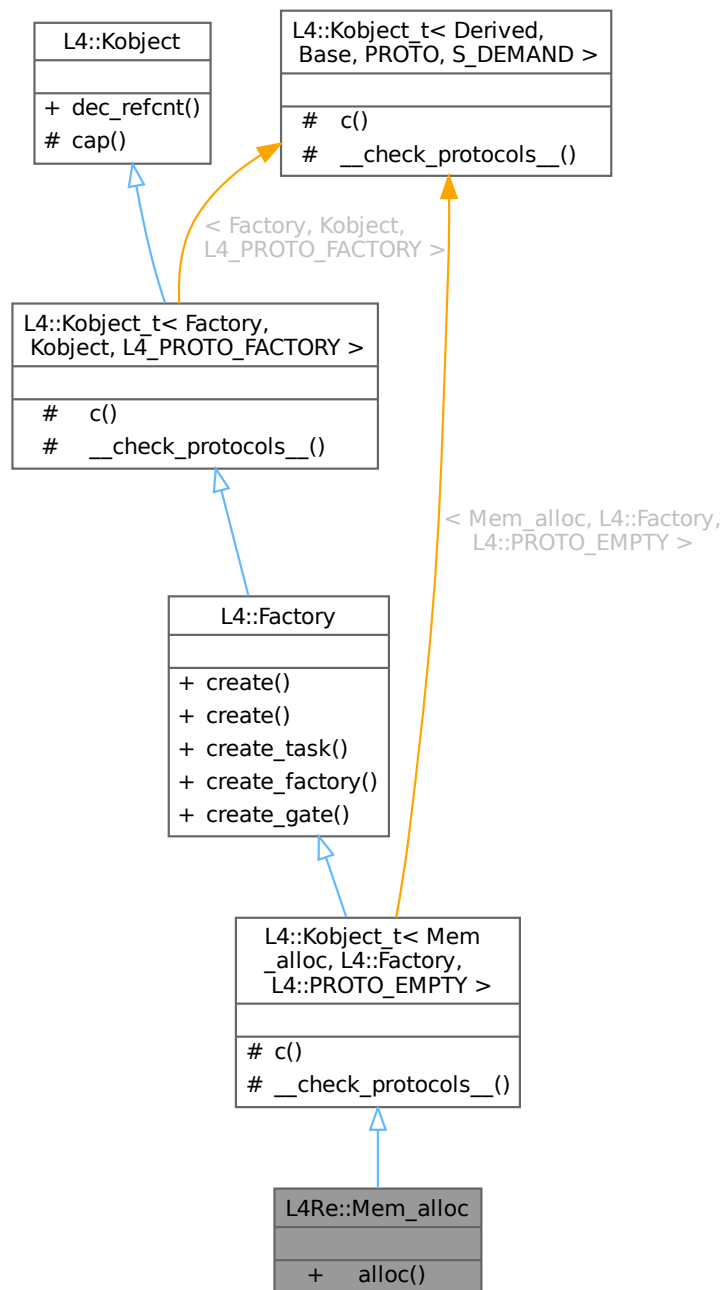
- [l4/re/log](#)

15.281 L4Re::Mem_alloc Class Reference

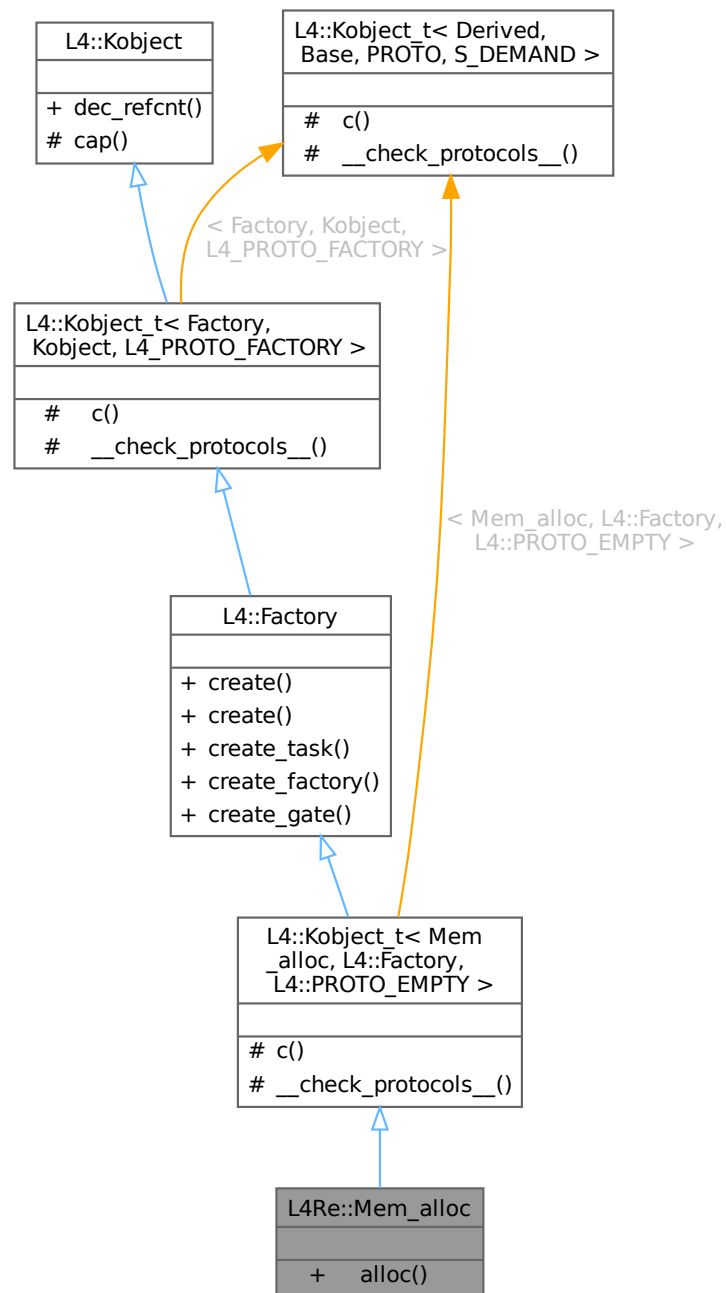
Memory allocation interface.

```
#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 }
- Flags for the allocator.*

Public Member Functions

- `long alloc` (long size, `L4::Cap< Dataspace >` mem, unsigned long flags=0, unsigned long align=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` const &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.***15.281.1 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](#).

15.281.2 Member Enumeration Documentation**15.281.2.1 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 L4Re::Dma_space instead.
Super_pages	Allocate super pages.

Definition at line 71 of file [mem_alloc](#).

15.281.3 Member Function Documentation

15.281.3.1 alloc()

```
long L4Re::Mem_alloc::alloc (
    long size,
    L4::Cap< Dataspace > mem,
    unsigned long flags = 0,
    unsigned long align = 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 Mem_alloc_flags::Continuous 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.
out	<i>mem</i>	Capability slot where the capability to the dataspace is received.
	<i>flags</i>	Special dataspace properties, see Mem_alloc_flags
	<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

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:

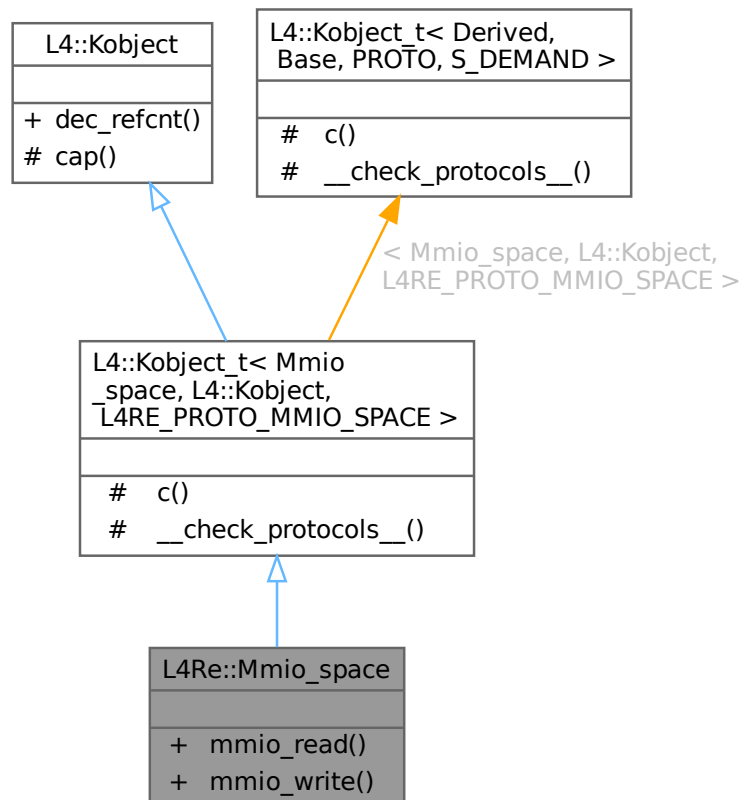
- [l4/re/mem_alloc](#)
- [l4/re/impl/mem_alloc_impl.h](#)

15.282 L4Re::Mmio_space Struct Reference

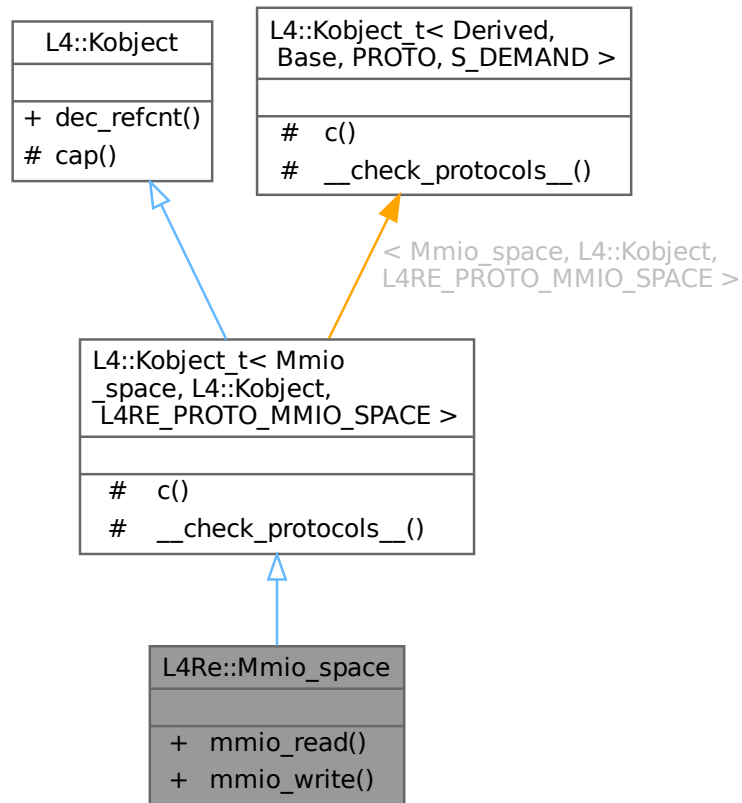
Interface for memory-like address space accessible via IPC.

```
#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 }
- typedef [l4_uint64_t](#) [Addr](#)

Actual size of the value to read or write.

Device address.

Public Member Functions

- long [mmio_read](#) ([Addr](#) addr, char width, [l4_uint64_t](#) *value)
- long [mmio_write](#) ([Addr](#) addr, char width, [l4_uint64_t](#) value)

Read a value from the given address.

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.

15.282.1 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](#).

15.282.2 Member Enumeration Documentation

15.282.2.1 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](#).

15.282.3 Member Function Documentation

15.282.3.1 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 Access_width .
out	<i>value</i>	Return value. If width is smaller than 64 bit,the upper bits are guaranteed to be 0.

Return values

L4_EOK	Success.
-L4_EPERM	Insufficient read rights.
-L4_EINVAL	Address does not exist or cannot be accessed with the given width.

15.282.3.2 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 Access_width .
<i>value</i>	Value to write. If width is smaller than 64 bit, the upper bits are ignored.

Return values

L4_EOK	Success.
-L4_EPERM	Insufficient write rights.
-L4_EINVAL	Address does not exist or cannot be accessed with the given width.

The documentation for this struct was generated from the following file:

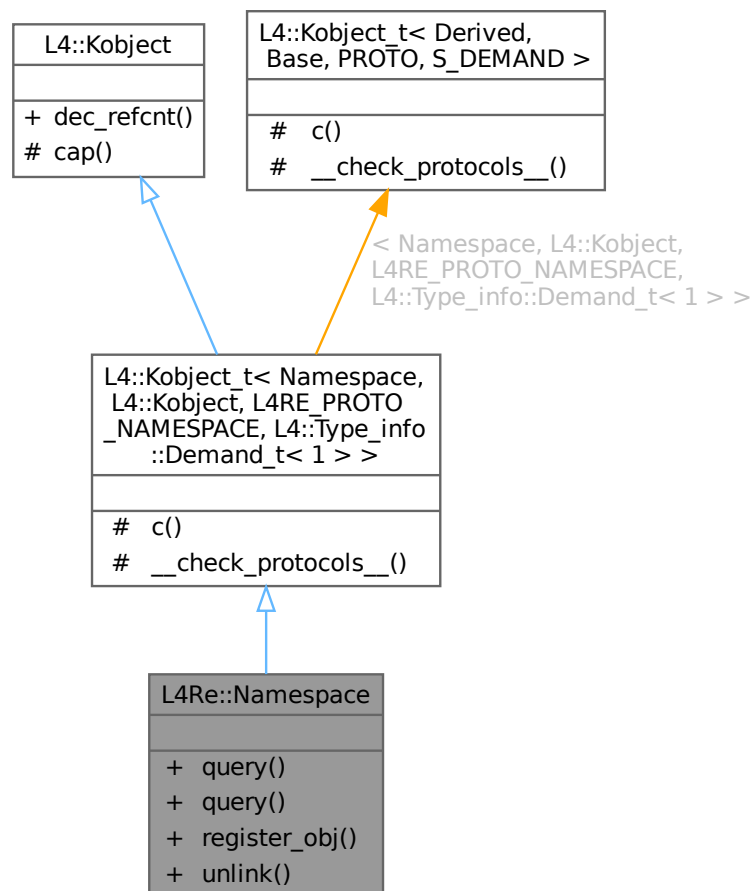
- [l4/re/mmio_space](#)

15.283 L4Re::Namespace Class Reference

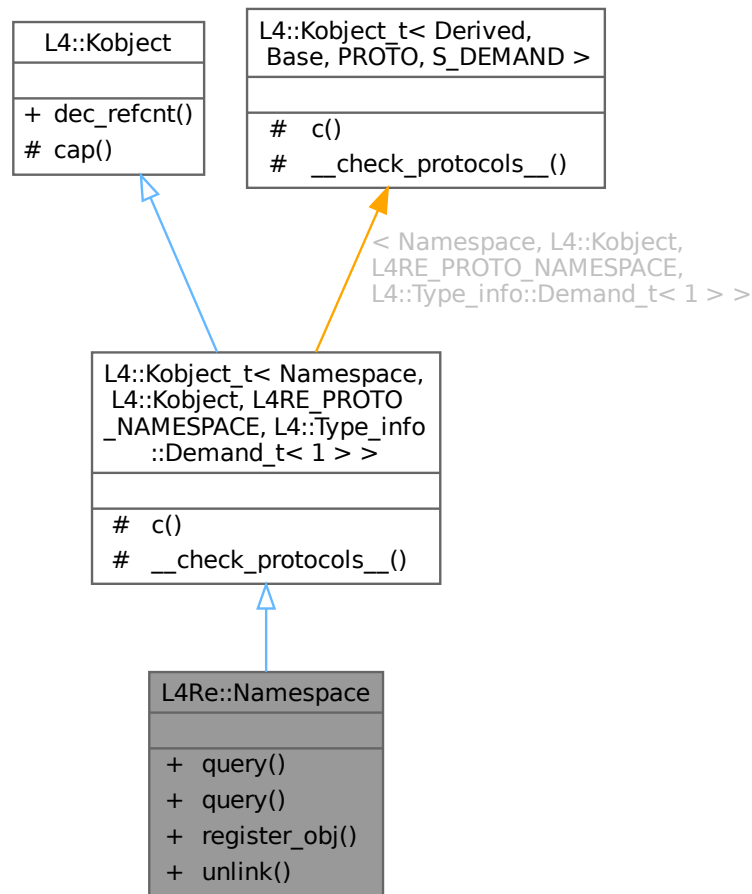
Name-space interface.

```
#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.

15.283.1 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](#).

15.283.2 Member Enumeration Documentation

15.283.2.1 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](#).

15.283.2.2 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](#).

15.283.2.3 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](#).

15.283.3 Member Function Documentation

15.283.3.1 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, L4_RCV_ITEM_LOCAL_ID 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 l4_error_code_t .

Definition at line 125 of file [namespace_impl.h](#).

15.283.3.2 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, L4_RCV_ITEM_LOCAL_ID 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.

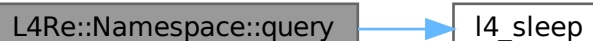
Return values

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 l4_error_code_t .

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:



15.283.3.3 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 L4Re::Namespace::Register_flags . 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 L4_CAP_FPAGE_W right on the invoked capability.
-L4_ENOMEM	Server has insufficient resources.
-L4_EINVAL	Invalid parameter.
<0	IPC errors, see l4_error_code_t .

Precondition

requires capability rights: {RW}

Definition at line 176 of file [namespace](#).

15.283.3.4 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 L4_CAP_FPAGE_W right on the invoked capability.
-L4_EACCESS	Name cannot be removed.

Return values

<code>< 0</code>	IPC errors, see l4_error_code_t .
---------------------	---

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](#)

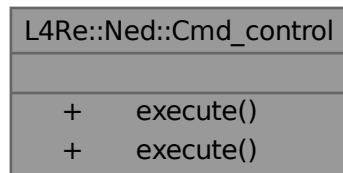
15.284 L4Re::Ned::Cmd_control Class Reference

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.

15.284.1 Detailed Description

Direct control interface for Ned.

Definition at line 20 of file [cmd_control](#).

15.284.2 Member Function Documentation

15.284.2.1 `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](#).

15.284.2.2 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:

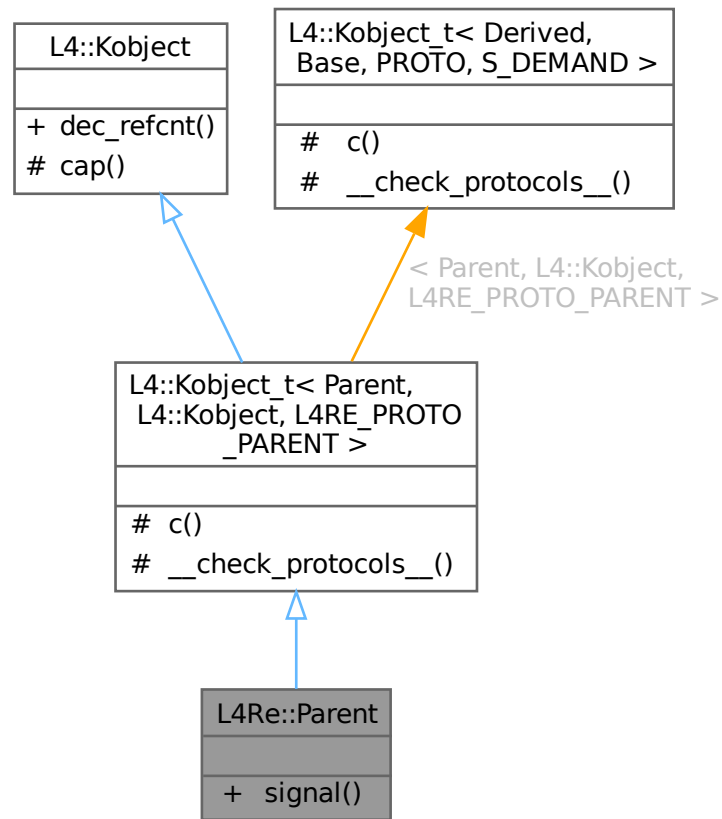
- pkg/l4re-core/ned/lib/include/cmd_control

15.285 L4Re::Parent Class Reference

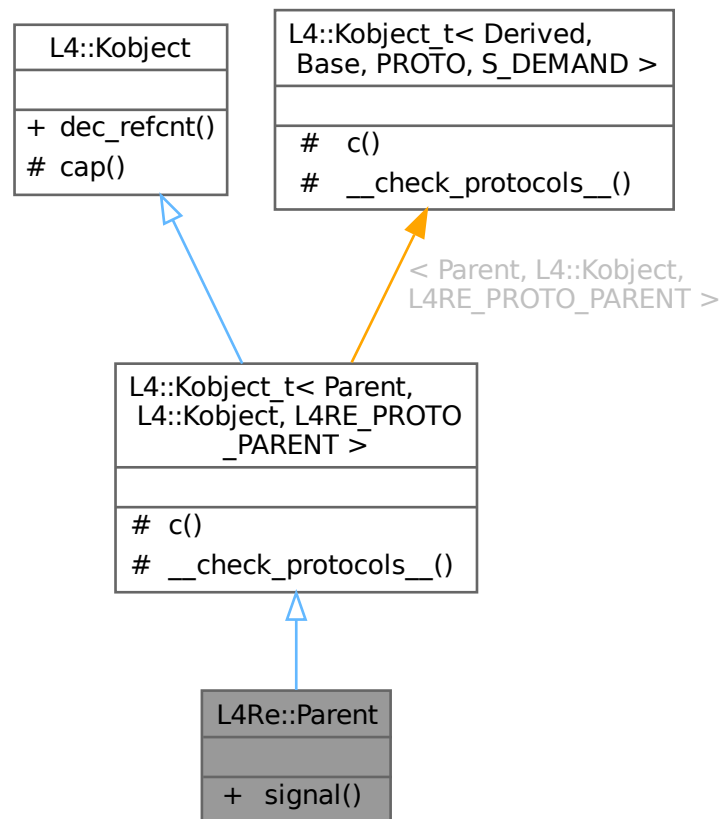
[Parent](#) interface.

```
#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.

15.285.1 Detailed Description

[Parent](#) interface.

See also

[Parent API](#) for more details about the purpose.

Definition at line 53 of file [parent](#).

15.285.2 Member Function Documentation

15.285.2.1 [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:

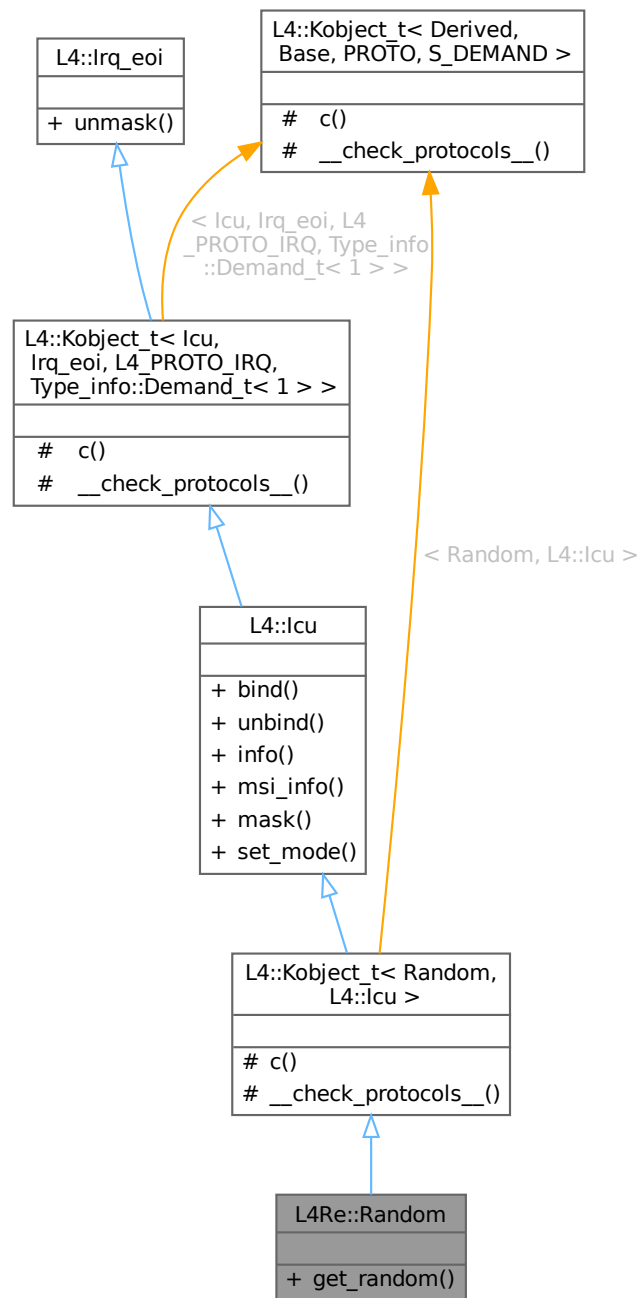
- [l4/re/parent](#)

15.286 L4Re::Random Struct Reference

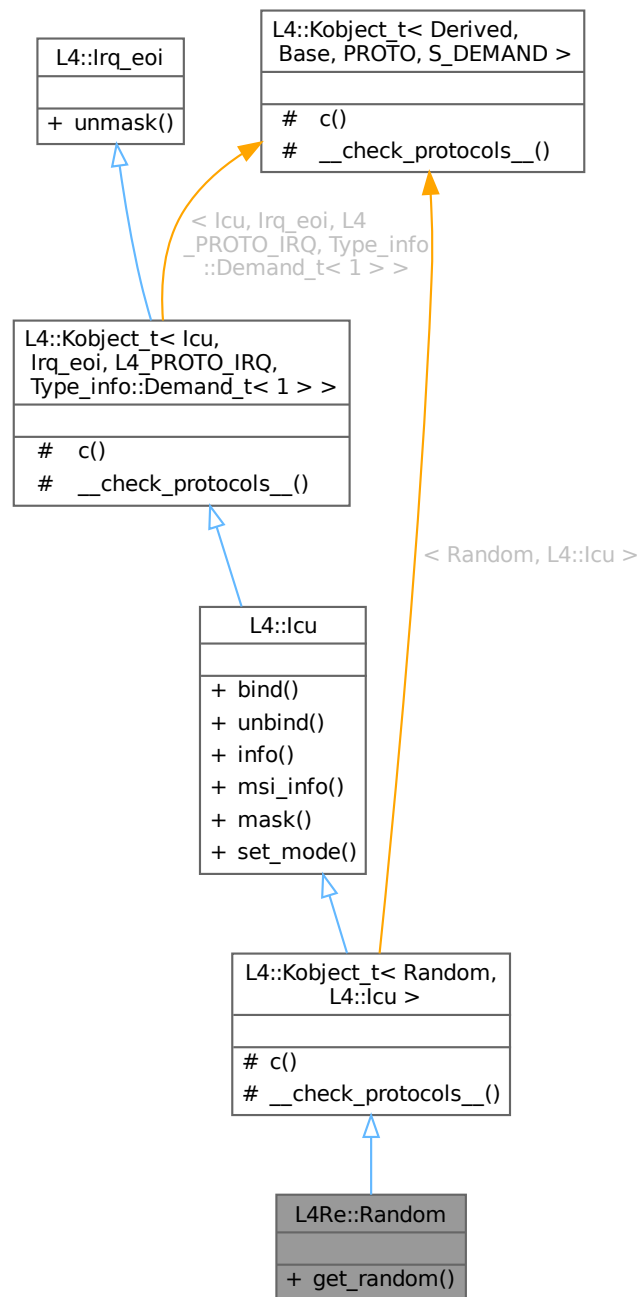
Low-bandwidth interface for random number generators.

```
#include <random>
```

Inheritance diagram for L4Re::Random:



Collaboration diagram for L4Re::Random:



Public Member Functions

- long `get_random` (`l4_size_t` size, `L4::lpc::Array< char, unsigned long > *buffer`)
Get a random number.

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::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.***15.286.1 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](#).

15.286.2 Member Function Documentation**15.286.2.1 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

≥ 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:

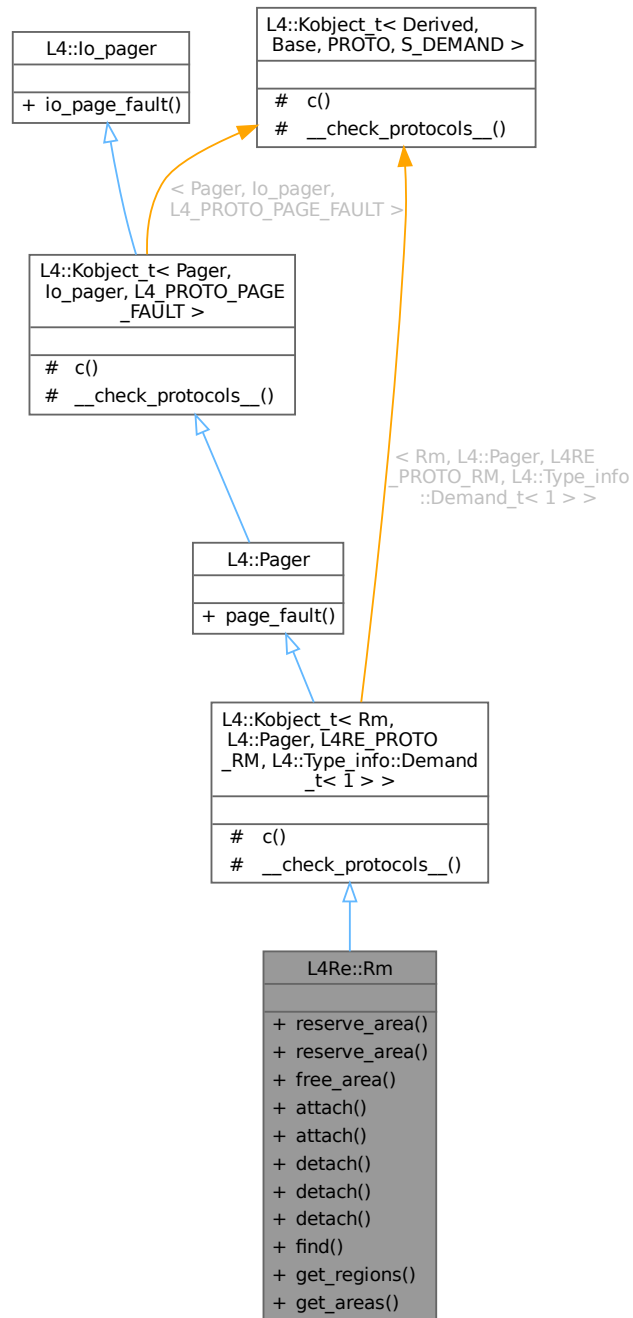
- [l4/re/random](#)

15.287 L4Re::Rm Class Reference

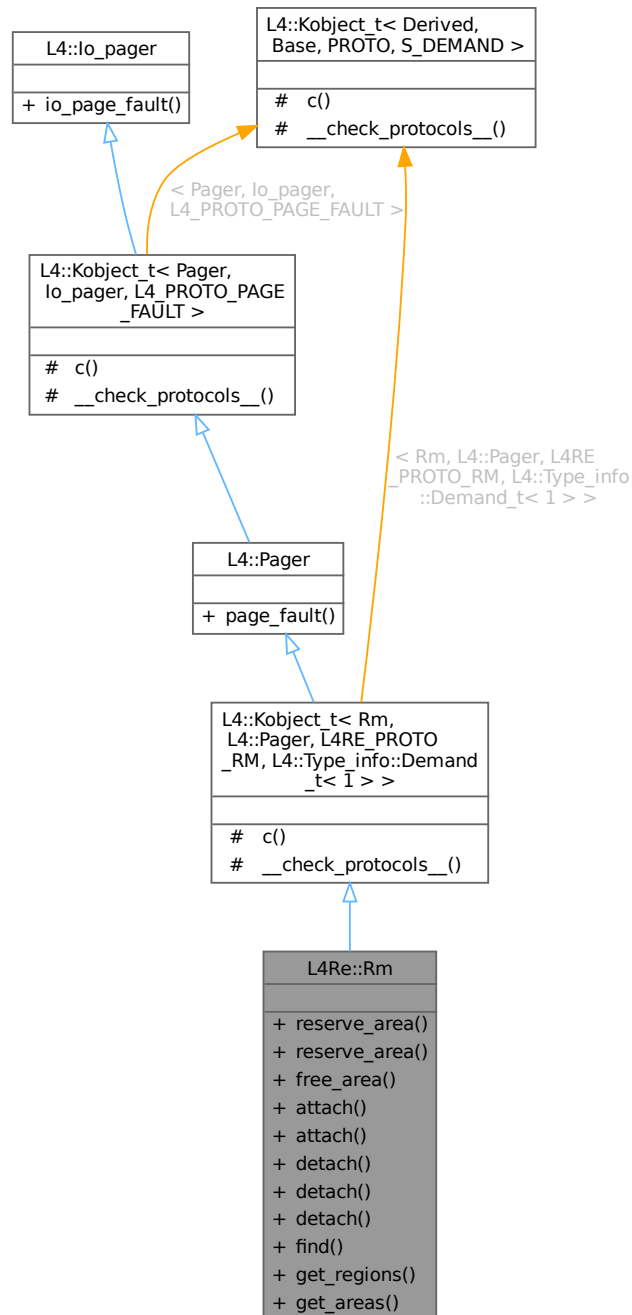
Region map.

```
#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](#)) 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](#)) 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.

15.287.1 Detailed Description

Region map.

See also

[Region map API](#) .

Definition at line 85 of file [rm](#).

15.287.2 Member Typedef Documentation**15.287.2.1 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 677 of file [rm](#).

15.287.2.2 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 669 of file [rm](#).

15.287.3 Member Enumeration Documentation**15.287.3.1 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 F::Detach_free .

Definition at line 219 of file [rm](#).

15.287.3.2 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 93 of file [rm](#).

15.287.3.3 Region_flag_shifts

```
enum L4Re::Rm::Region_flag_shifts
```

Region flag shifts.

Enumerator

Caching_shift	Start of Rm cache bits.
---------------	---

Definition at line 105 of file [rm](#).

15.287.4 Member Function Documentation**15.287.4.1 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 ) 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 L4Re::Rm::F::Search_addr 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 L4Re::Rm::F::In_area is given the value is used as a selector for the area (see L4Re::Rm::reserve_area) 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 L4Re::Rm::F::Attach_flags and L4Re::Rm::F::Region_flags . The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the F::Eager_map flag is set this function may also return L4Re::Dataspace::map 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 (L4_PAGESHIFT). This is only meaningful if the L4Re::Rm::F::Search_addr flag is used.

Return values

0	Success
-L4_ENOENT	No area could be found (see L4Re::Rm::F::In_area)
-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](#)).

Definition at line 43 of file [rm_impl.h](#).

15.287.4.2 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 ) 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 L4Re::Rm::F::Search_addr 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 L4Re::Rm::F::In_area is given the value is used as a selector for the area (see L4Re::Rm::reserve_area) 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 L4Re::Rm::F::Attach_flags and L4Re::Rm::F::Region_flags . The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the F::Eager_map flag is set this function may also return L4Re::Dataspace::map 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 (L4_PAGESHIFT). This is only meaningful if the L4Re::Rm::F::Search_addr flag is used.

Return values

0	Success
-L4_ENOENT	No area could be found (see L4Re::Rm::F::In_area)
-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](#)).

Definition at line 391 of file [rm](#).

15.287.4.3 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>	Dataspace that is affected. Give 0 if not interested.
	<i>task</i>	This argument specifies the task where the pages are unmapped. Provide L4::Cap<L4::Task>::Invalid for none. The default is the current task.

Return values

L4Re::Rm::Detach_result	On success.
<code>-L4_ENOENT</code>	No region found.
<code><0</code>	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 722 of file [rm](#).

15.287.4.4 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>	Dataspace that is affected. Give 0 if not interested.
	<i>task</i>	This argument specifies the task where the pages are unmapped. Provide L4::Cap<L4::Task>::Invalid for none. The default is the current task.

Return values

L4Re::Rm::Detach_result	On success.
<code>-L4_ENOENT</code>	No region found.
<code><0</code>	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 732 of file [rm](#).

15.287.4.5 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>	Dataspace that is affected. Give 0 if not interested.
	<i>task</i>	This argument specifies the task where the pages are unmapped. Provide L4::Cap<L4::Task>::Invalid for none. The default is the current task.

Return values

L4Re::Rm::Detach_result	On success.
-L4_ENOENT	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 [727](#) of file [rm](#).

15.287.4.6 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 F::Region_flags (and F::In_area).
out	<i>m</i>	Associated dataspace or paging service.

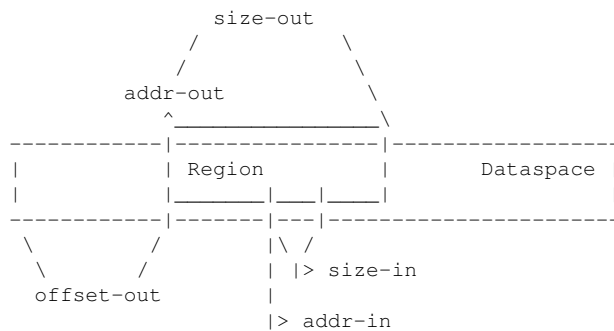
Return values

0	Success
-L4_EPERM	Operation not allowed.

Return values

<code>-L4_ENOENT</code>	No region found.
<code><0</code>	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 644 of file [rm](#).

15.287.4.7 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

<code>0</code>	Success
<code>-L4_ENOENT</code>	No area found.
<code><0</code>	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.

15.287.4.8 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

≥ 0	Number of returned areas in the <code>areas</code> array.
< 0	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.

15.287.4.9 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

≥ 0	Number of returned regions in the <code>regions</code> array.
< 0	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.

15.287.4.10 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 L4Re::Rm::F::Region_flags and L4Re::Rm::F::Attach_flags).
	<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 278 of file [rm](#).

15.287.4.11 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

<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 F::Region_flags and F::Attach_flags).
	<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><0</code>	IPC errors

For more information, please refer to the analogous function

See also

[L4Re::Rm::reserve_area](#).

Definition at line [304](#) of file [rm](#).

The documentation for this class was generated from the following files:

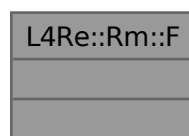
- [l4/re/rm](#)
- [l4/re/impl/rm_impl.h](#)

15.288 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 , [Attach_mask](#) = 0xf0000 }
- 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 }

15.288.1 Detailed Description

[Rm](#) flags definitions.

Definition at line 112 of file [rm](#).

15.288.2 Member Enumeration Documentation

15.288.2.1 [Attach_flags](#)

```
enum L4Re::Rm::F::Attach_flags : l4_uint32_t
```

Flags for attach operation.

Enumerator

Search_addr	Search for a suitable address range.
In_area	Search only in area, or map into area.
Eager_map	Eagerly map the attached data space in.
Attach_mask	Mask of all attach flags.

Definition at line 115 of file [rm](#).

15.288.2.2 [Region_flags](#)

```
enum L4Re::Rm::F::Region_flags : l4_uint16_t
```

Enumerator

Rights_mask	Region rights.
R	Readable region.
W	Writable region.
X	Executable region.
RW	Readable and writable region.
RX	Readable and executable region.

Enumerator

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 Rm 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 [129](#) of file [rm](#).

The documentation for this struct was generated from the following file:

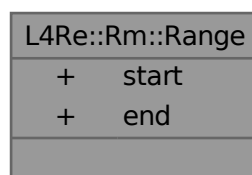
- [l4/re/rm](#)

15.289 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.

15.289.1 Detailed Description

A range of virtual addresses.

Definition at line 656 of file [rm](#).

The documentation for this struct was generated from the following file:

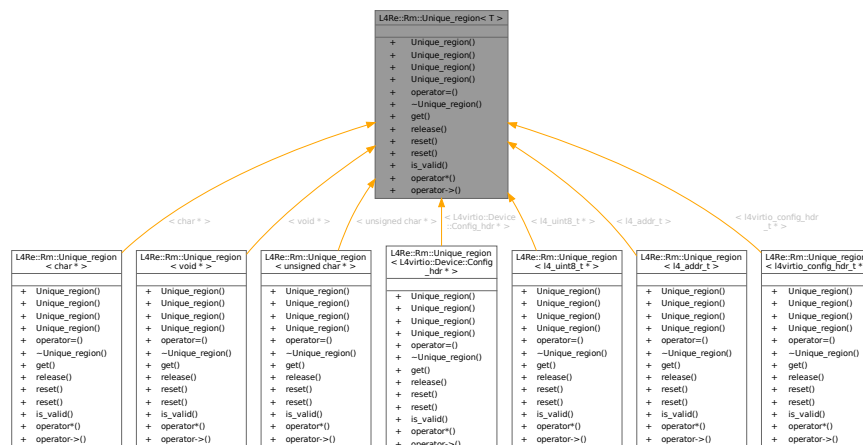
- [l4/re/rm](#)

15.290 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.

15.290.1 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 412 of file [rm](#).

15.290.2 Constructor & Destructor Documentation

15.290.2.1 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 433 of file [rm](#).

15.290.2.2 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 442 of file [rm](#).

15.290.2.3 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 450 of file [rm](#).

15.290.2.4 ~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 475 of file [rm](#).

References [L4::Cap_base::is_valid\(\)](#).

Here is the call graph for this function:



15.290.3 Member Function Documentation

15.290.3.1 get()

```
template<typename T >
T L4Re::Rm::Unique_region< T >::get ( ) const [inline], [noexcept]
```

Return the address.

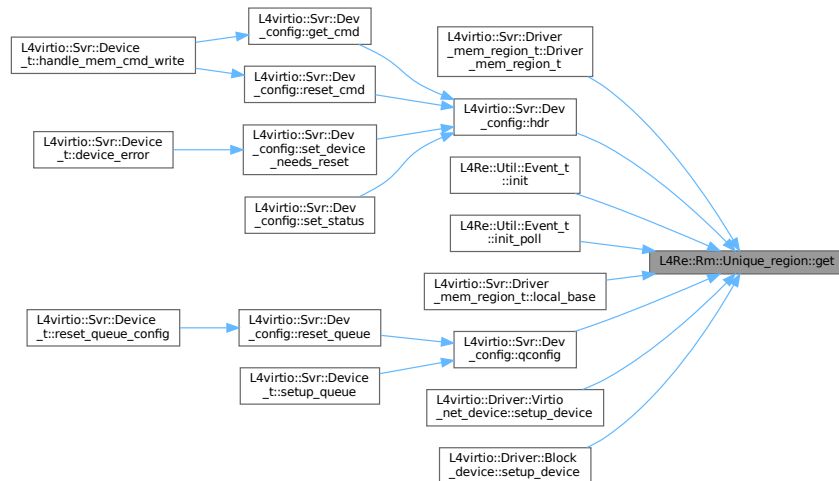
Returns

the address

Definition at line 486 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:

**15.290.3.2 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 526 of file [rm](#).

References [L4::Cap_base::is_valid\(\)](#).

Here is the call graph for this function:



15.290.3.3 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 [458](#) of file [rm](#).

References [L4::Cap_base::is_valid\(\)](#).

Here is the call graph for this function:



15.290.3.4 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 [494](#) of file [rm](#).

15.290.3.5 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 506 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:

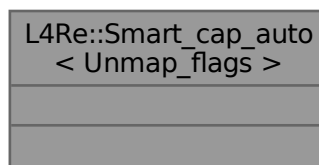
- [l4/re/rm](#)

15.291 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 >:



15.291.1 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](#)

15.292 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](#).

15.292.1 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 182 of file [cap_alloc](#).

The documentation for this class was generated from the following file:

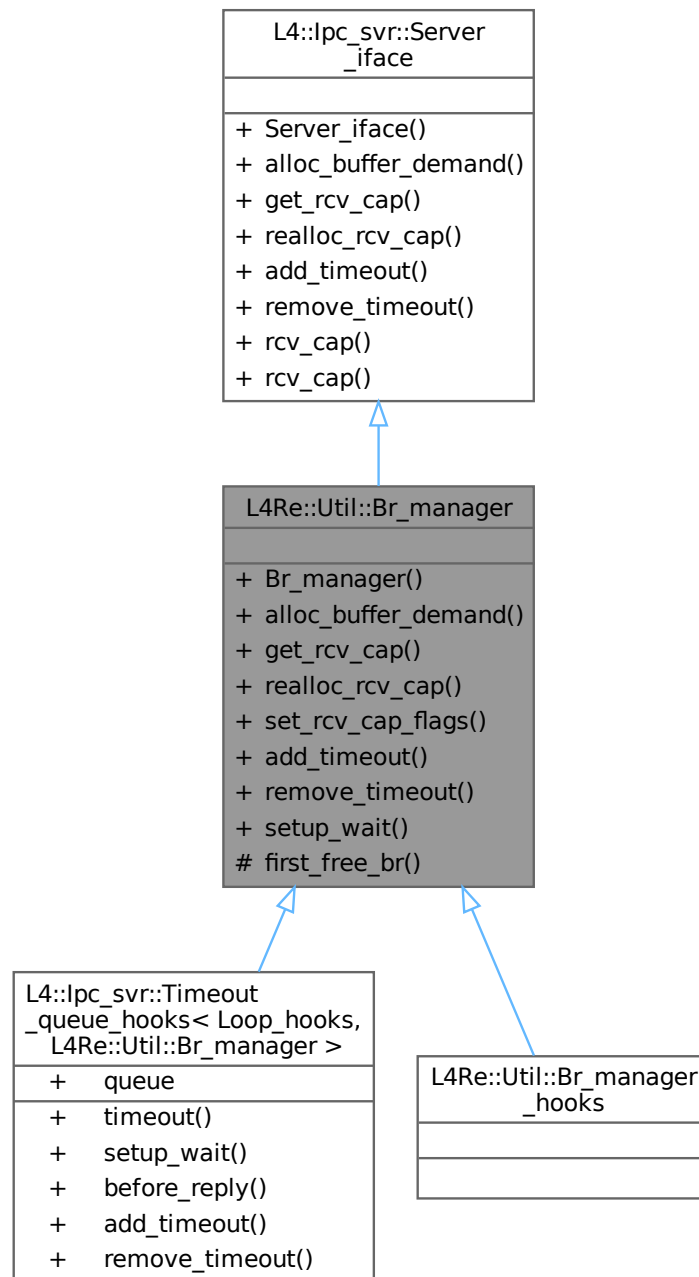
- [l4/re/cap_alloc](#)

15.293 L4Re::Util::Br_manager Class Reference

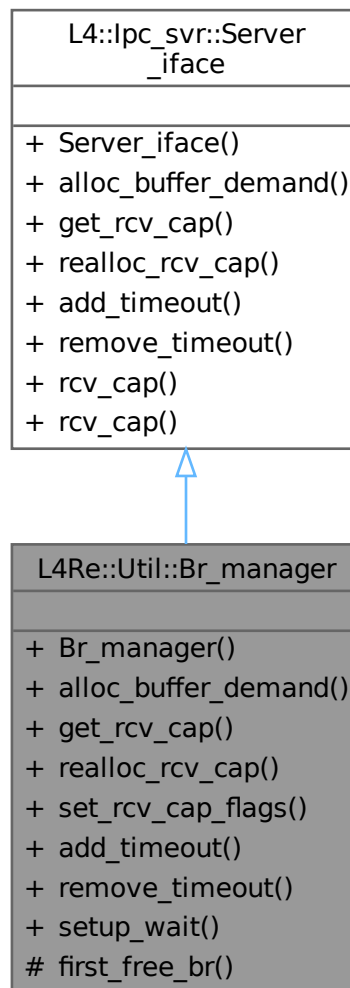
Buffer-register (BR) manager for [L4::Server](#).

```
#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.

15.293.1 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`.

15.293.2 Member Function Documentation

15.293.2.1 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.
---------------	---

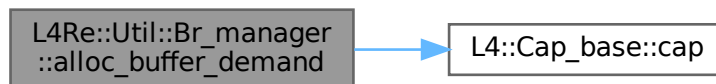
This function is not called by user applications directly. Usually the server implementation or the registry implementation calls this function whenever a new object is registered at the server.

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:



15.293.2.2 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 alloc_buffer_demand()).
--------------	---

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](#).

15.293.2.3 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 alloc_buffer_demand()).
--------------	---

Precondition

$0 \leq \text{index} < \text{caps}$ registered with [alloc_buffer_demand\(\)](#)

Returns

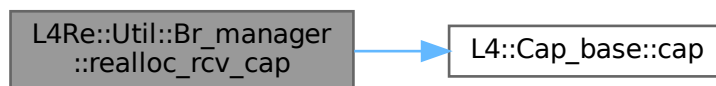
0 on success, < 0 on error.

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:



15.293.2.4 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 l4_msg_item_consts_t .
--------------	--

Definition at line 129 of file [br_manager](#).

References [l4_assert](#).

The documentation for this class was generated from the following file:

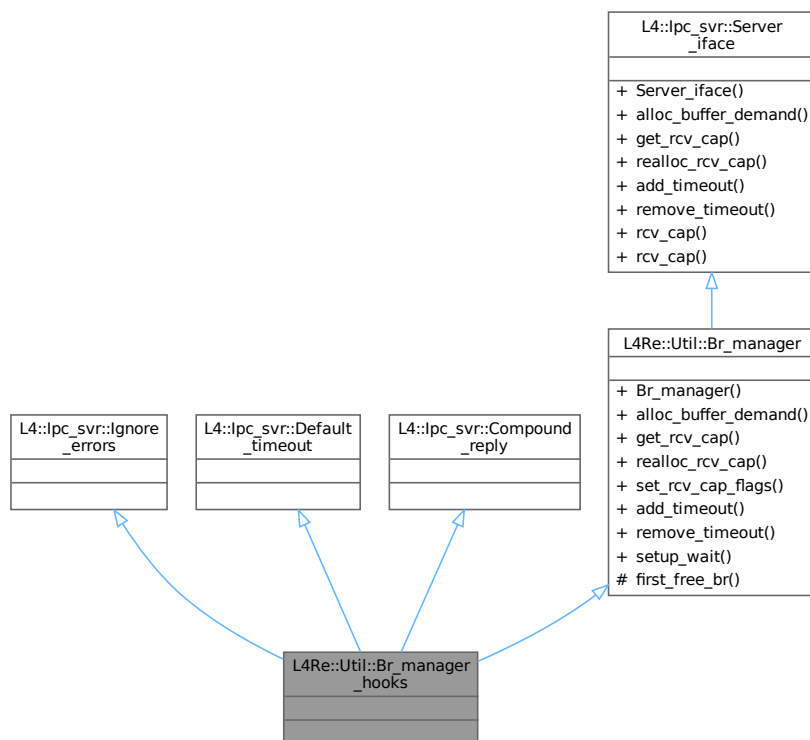
- [l4/re/util/br_manager](#)

15.294 L4Re::Util::Br_manager_hooks Struct Reference

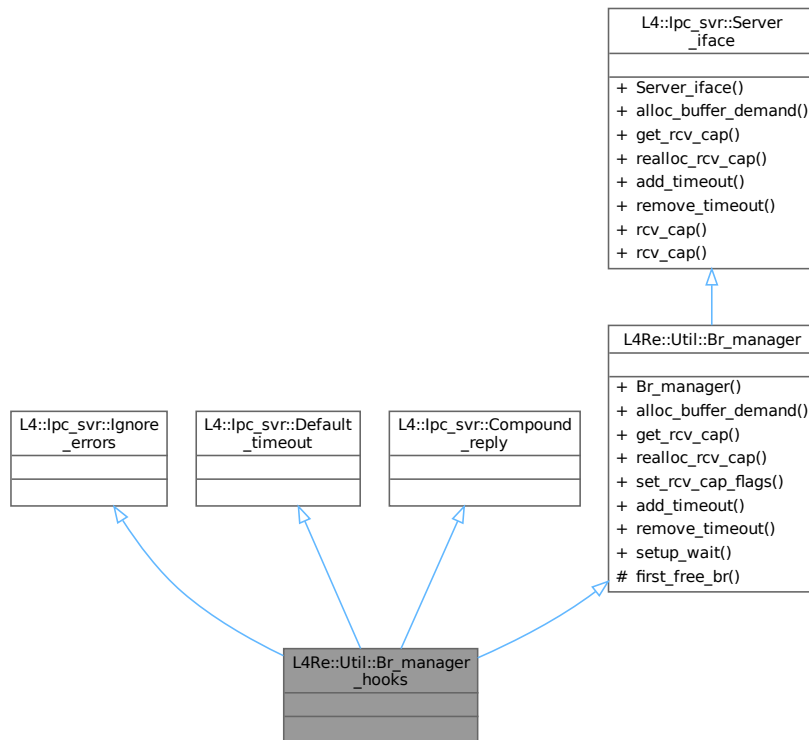
Predefined server-loop hooks for a server loop using the [Br_manager](#).

```
#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.

15.294.1 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:

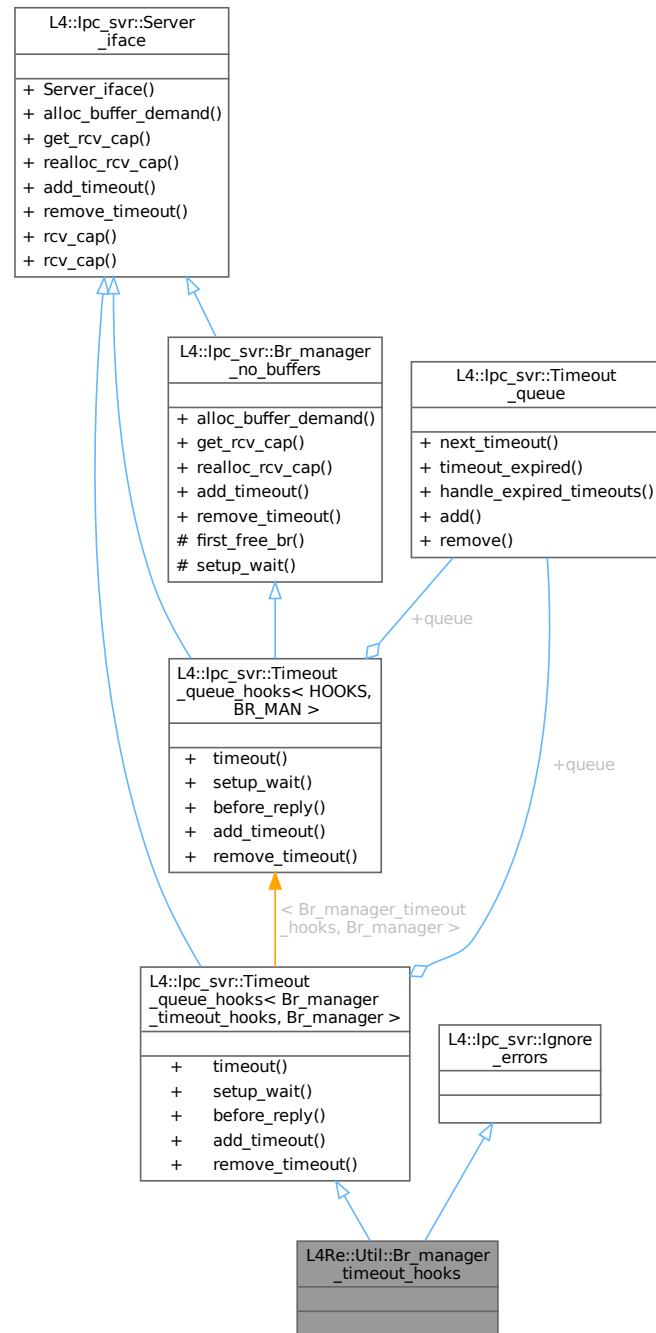
- `l4/re/util/br_manager`

15.295 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>
```


Collaboration diagram for L4Re::Util::Br_manager_timeout_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::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.

15.295.1 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](#)

15.296 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.

15.296.1 Detailed Description

Capability allocator.

Definition at line 38 of file [bitmap_cap_alloc](#).

The documentation for this class was generated from the following file:

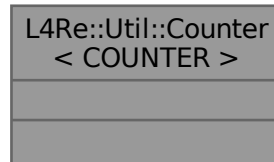
- [l4/re/util/bitmap_cap_alloc](#)

15.297 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 >:



15.297.1 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](#).

The documentation for this struct was generated from the following file:

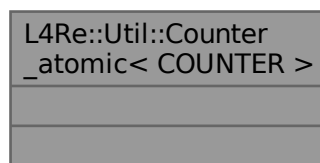
- [l4/re/util/counting_cap_alloc](#)

15.298 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 >:



15.298.1 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 73 of file [counting_cap_alloc](#).

The documentation for this struct was generated from the following file:

- [l4/re/util/counting_cap_alloc](#)

15.299 L4Re::Util::Counting_cap_alloc< COUNTERTYPE > Class Template Reference

Internal reference-counting cap allocator.

```
#include <counting_cap_alloc>
```

Collaboration diagram for L4Re::Util::Counting_cap_alloc< COUNTERTYPE >:

L4Re::Util::Counting _cap_alloc< COUNTERTYPE >
<ul style="list-style-type: none"> + alloc() + alloc() + take() + free() + release() + last() # Counting_cap_alloc() # setup()

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) noexcept
Set up the backing memory for the allocator and the area of managed capability slots.

15.299.1 Detailed Description

```
template<typename COUNTERTYPE>
class L4Re::Util::Counting_cap_alloc< COUNTERTYPE >
```

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 129 of file [counting_cap_alloc](#).

15.299.2 Constructor & Destructor Documentation

15.299.2.1 Counting_cap_alloc()

```
template<typename COUNTERTYPE >  
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::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 158 of file [counting_cap_alloc](#).

15.299.3 Member Function Documentation

15.299.3.1 alloc() [1/2]

```
template<typename COUNTERTYPE >  
L4::Cap< void > L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc ( ) [inline], [noexcept]
```

Allocate a new capability slot.

Returns

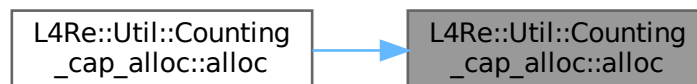
The newly allocated capability slot, invalid if the allocator was exhausted.

Definition at line 189 of file [counting_cap_alloc](#).

References [L4_CAP_SHIFT](#).

Referenced by [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc\(\)](#).

Here is the caller graph for this function:



15.299.3.2 alloc() [2/2]

```
template<typename COUNTERTYPE >
template<typename T >
L4::Cap< T > L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc ( ) [inline], [noexcept]
```

Allocate a new capability slot.

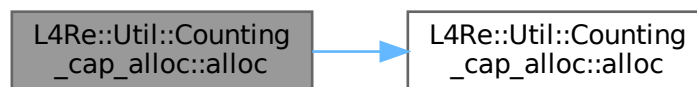
Returns

The newly allocated capability slot, invalid if the allocator was exhausted.

Definition at line 214 of file [counting_cap_alloc](#).

References [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc\(\)](#).

Here is the call graph for this function:



15.299.3.3 free()

```
template<typename COUNTERTYPE >
bool L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::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.

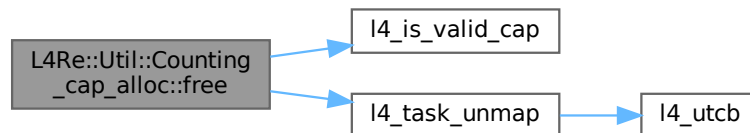
Returns

True, if the capability was managed by this allocator.

Definition at line 252 of file [counting_cap_alloc](#).

References [l4_assert](#), [l4_is_valid_cap\(\)](#), and [l4_task_unmap\(\)](#).

Here is the call graph for this function:

**15.299.3.4 release()**

```

template<typename COUNTERTYPE >
bool L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::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 l4_unmap_flags_t .

Precondition

The capability has been allocated. Calling release on a free capability results in undefined behaviour.

Returns

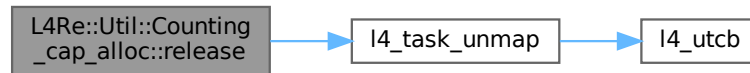
True, if the capability was freed as a result of this operation. If false is returned the capability is either still in use or is not managed by this allocator.

Does nothing apart from returning false if the capability is not managed by this allocator.

Definition at line 290 of file [counting_cap_alloc](#).

References [l4_assert](#), [L4_INVALID_CAP](#), and [l4_task_unmap\(\)](#).

Here is the call graph for this function:



15.299.3.5 setup()

```

template<typename COUNTERTYPE >
void L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::setup (
    void * m,
    long capacity,
    long bias ) [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.

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 175 of file [counting_cap_alloc](#).

15.299.3.6 take()

```

template<typename COUNTERTYPE >
void L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::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 229 of file [counting_cap_alloc](#).

The documentation for this class was generated from the following file:

- [l4/re/util/counting_cap_alloc](#)

15.300 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()

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, Dataspace::Flags flags, Dataspace::Map_addr min, Dataspace::Map_addr max)
A hook that is called as the first operation in each map request.
- 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.

15.300.1 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](#).

15.300.2 Member Function Documentation

15.300.2.1 `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 157 of file [dataspace_svr](#).

References [L4_ENODEV](#).

15.300.2.2 `clear()`

```
virtual long L4Re::Util::Dataspace_svr::clear (  
    unsigned long offs,  
    unsigned long size ) const [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

15.300.2.3 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

>=0	Number of bytes copied
<0	An error occured. The error code may depend on the implementation.

Definition at line 128 of file [dataspace_svr](#).

References [L4_ENODEV](#).

15.300.2.4 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 173 of file [dataspace_svr](#).

15.300.2.5 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 )
```

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>	Dataspace flags, see L4Re::Dataspace::F::Flags .
	<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

15.300.2.6 map_hook()

```
virtual int L4Re::Util::Dataspace_svr::map_hook (
    Dataspace::Offset offs,
    Dataspace::Flags flags,
    Dataspace::Map_addr min,
    Dataspace::Map_addr max ) [inline], [virtual]
```

A hook that is called as the first operation in each map request.

Parameters

<i>offs</i>	Offs param to map
<i>flags</i>	Flags param to map
<i>min</i>	Min param to map
<i>max</i>	Max param to map

Return values

<0	Error and the map request will be aborted with that error.
>=0	Success

See also

[map](#)

Definition at line 89 of file [dataspace_svr](#).

15.300.2.7 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 165 of file [dataspace_svr](#).

References [L4_LOG2_PAGESIZE](#).

15.300.2.8 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 113 of file [dataspace_svr](#).

15.300.2.9 take()

```
virtual void L4Re::Util::Dataspace_svr::take ( ) [inline], [virtual], [noexcept]
```

Take a reference to this dataspace.

Default does nothing.

Definition at line 103 of file [dataspace_svr](#).

The documentation for this class was generated from the following file:

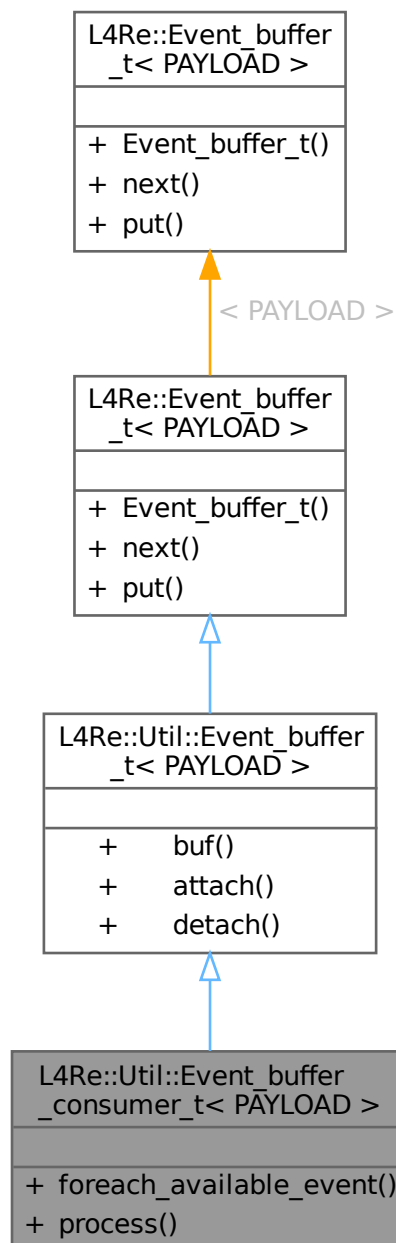
- [l4/re/util/dataspace_svr](#)

15.301 L4Re::Util::Event_buffer_consumer_t< PAYLOAD > Class Template Reference

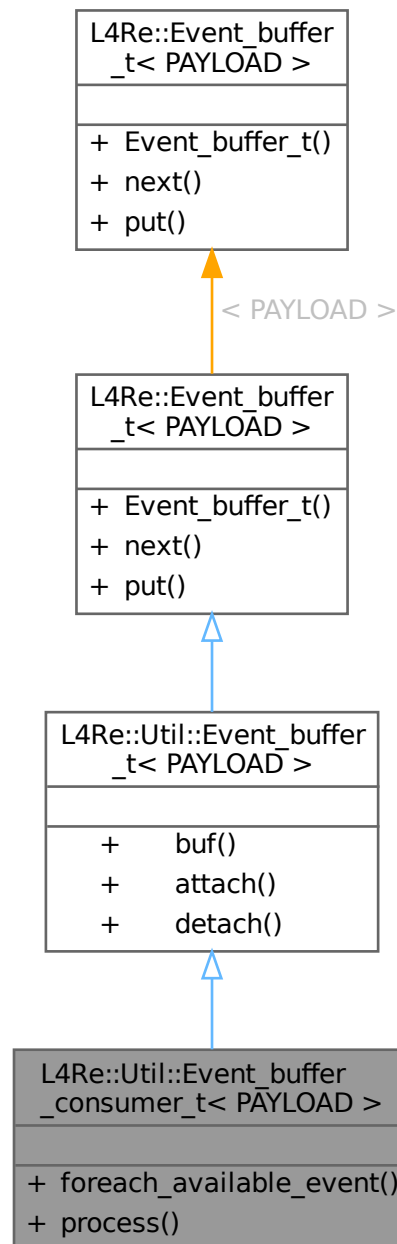
An event buffer consumer.

```
#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.

15.301.1 Detailed Description

template<typename PAYLOAD>
class [L4Re::Util::Event_buffer_consumer_t< PAYLOAD >](#)

An event buffer consumer.

Definition at line 93 of file [event_buffer](#).

15.301.2 Member Function Documentation

15.301.2.1 [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

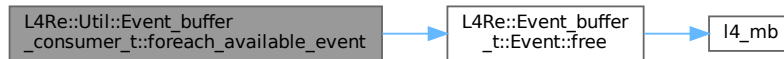
<i>cb</i>	Function callback.
<i>data</i>	Data to pass as an argument to the callback.

Definition at line 104 of file [event_buffer](#).

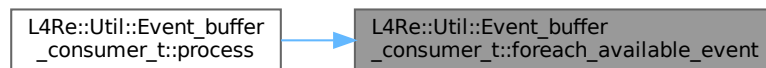
References [L4Re::Event_buffer_t< PAYLOAD >::Event::free\(\)](#).

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:



15.301.2.2 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>	Event 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.

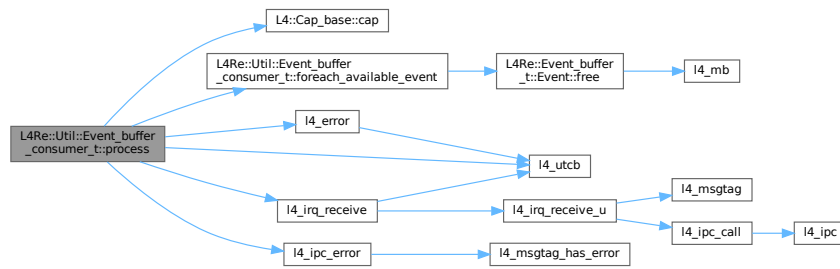
Note

This function never returns.

Definition at line 125 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:

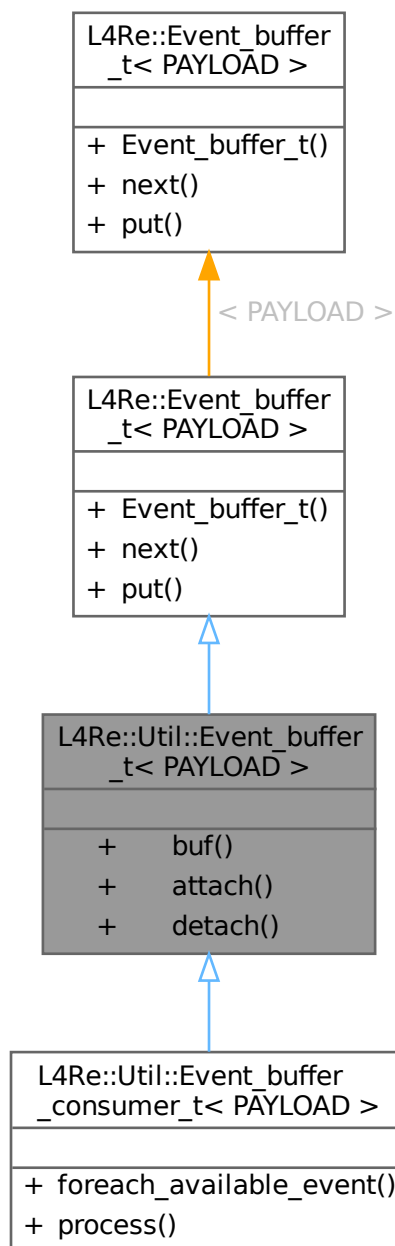
- `I4/re/util/event_buffer`

15.302 L4Re::Util::Event_buffer_t< PAYLOAD > Class Template Reference

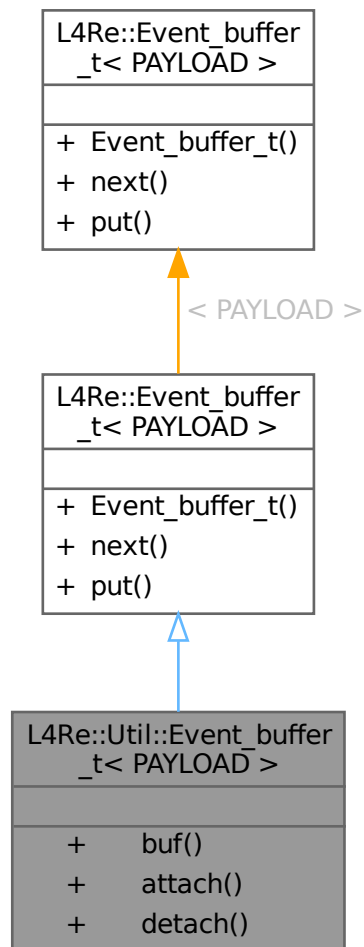
Event_buffer utility class.

```
#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.

15.302.1 Detailed Description

```
template<typename PAYLOAD>
class L4Re::Util::Event_buffer_t< PAYLOAD >
```

Event_buffer utility class.

Definition at line 36 of file [event_buffer](#).

15.302.2 Member Function Documentation

15.302.2.1 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>	Dataspace 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:



15.302.2.2 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](#).

15.302.2.3 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 78 of file [event_buffer](#).

The documentation for this class was generated from the following file:

- I4/re/util/event_buffer

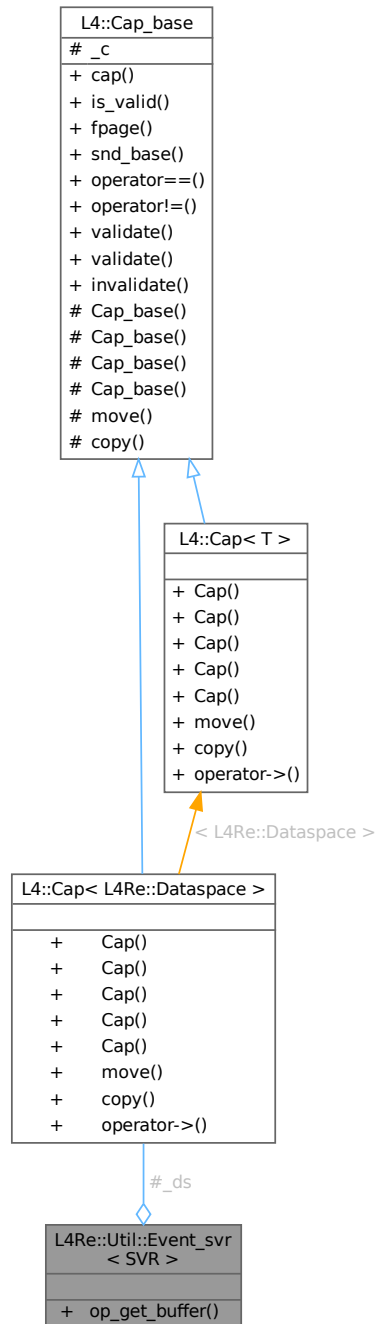
15.303 L4Re::Util::Event_svr< SVR > Class Template Reference

Convenience wrapper for implementing an event server.

```
#include <event_svr>
```

Inherits L4Re::Util::lcu_cap_array_svr< ICU >.

Collaboration diagram for L4Re::Util::Event_svr< SVR >:



Public Member Functions

- `long op_get_buffer (L4Re::Event::Rights, L4Re::Cap< L4Re::Dataspace > &ds)`
Handle [L4Re::Event](#) protocol.

15.303.1 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:

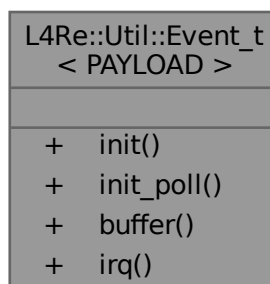
- [l4/re/util/event_svr](#)

15.304 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.

15.304.1 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`.

15.304.2 Member Enumeration Documentation

15.304.2.1 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`.

15.304.3 Member Function Documentation

15.304.3.1 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](#).

15.304.3.2 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 L4::Triggerable .
-----------------	---

Parameters

<i>event</i>	Capability to event.
<i>env</i>	Pointer to L4Re-Environment
<i>ca</i>	Pointer to capability allocator.

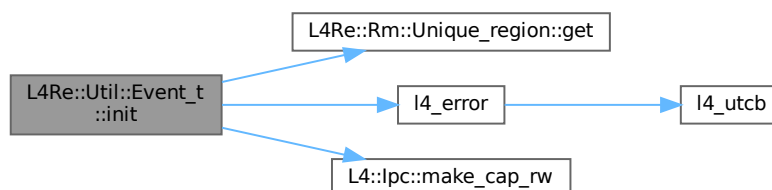
Return values

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:



15.304.3.3 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.

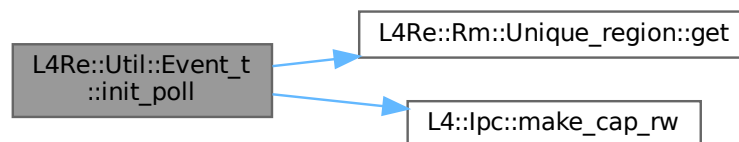
Return values

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:



15.304.3.4 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](#)

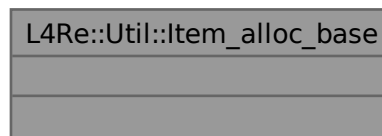
15.305 L4Re::Util::Item_alloc_base Class Reference

Item allocator.

```
#include <item_alloc>
```

Inherited by L4Re::Util::Item_alloc< Bits >.

Collaboration diagram for L4Re::Util::Item_alloc_base:



15.305.1 Detailed Description

Item allocator.

Definition at line 38 of file [item_alloc](#).

The documentation for this class was generated from the following file:

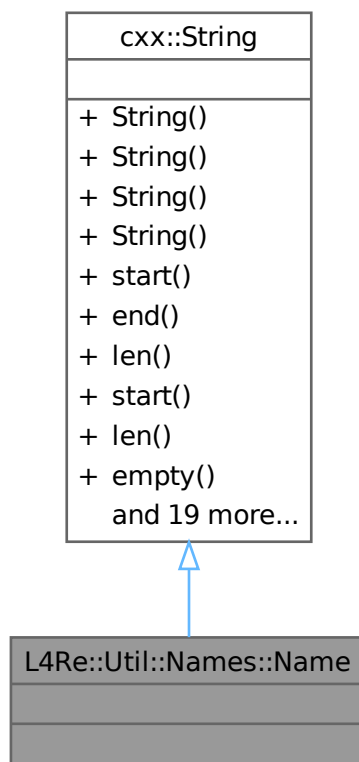
- [l4/re/util/item_alloc](#)

15.306 L4Re::Util::Names::Name Class Reference

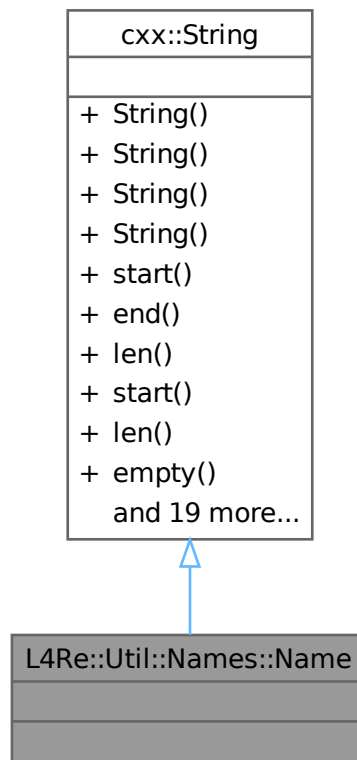
[Name](#) class.

```
#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.

15.306.1 Detailed Description

[Name](#) class.

Definition at line 42 of file [name_space_svr](#).

The documentation for this class was generated from the following file:

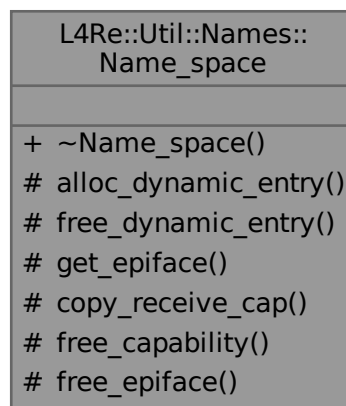
- [l4/re/util/name_space_svr](#)

15.307 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\(\)](#).

15.307.1 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 186 of file [name_space_svr](#).

15.307.2 Member Function Documentation

15.307.2.1 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>	Name of the entry, must be copied.
<i>flags</i>	Entry flags, see Obj::Flags .

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.

15.307.2.2 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

<i>out</i>	<i>cap</i>	Capability slot with the received capability. Must be permanently available until free_capability() 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.

15.307.2.3 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	<i>cap</i>	Capability to free.
----	------------	---------------------

Counterpart of [copy_receive_cap](#). Free the capability slot when the entry is deleted or changed.

15.307.2.4 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

<i>e</i>	Entry to free.
----------	----------------

15.307.2.5 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.

15.307.2.6 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:

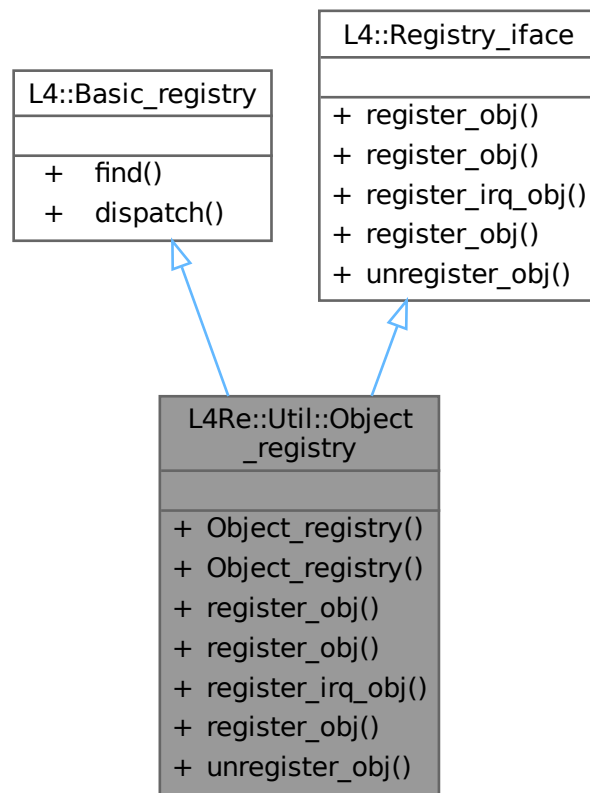
- `l4/re/util/name_space_svr`

15.308 L4Re::Util::Object_registry Class Reference

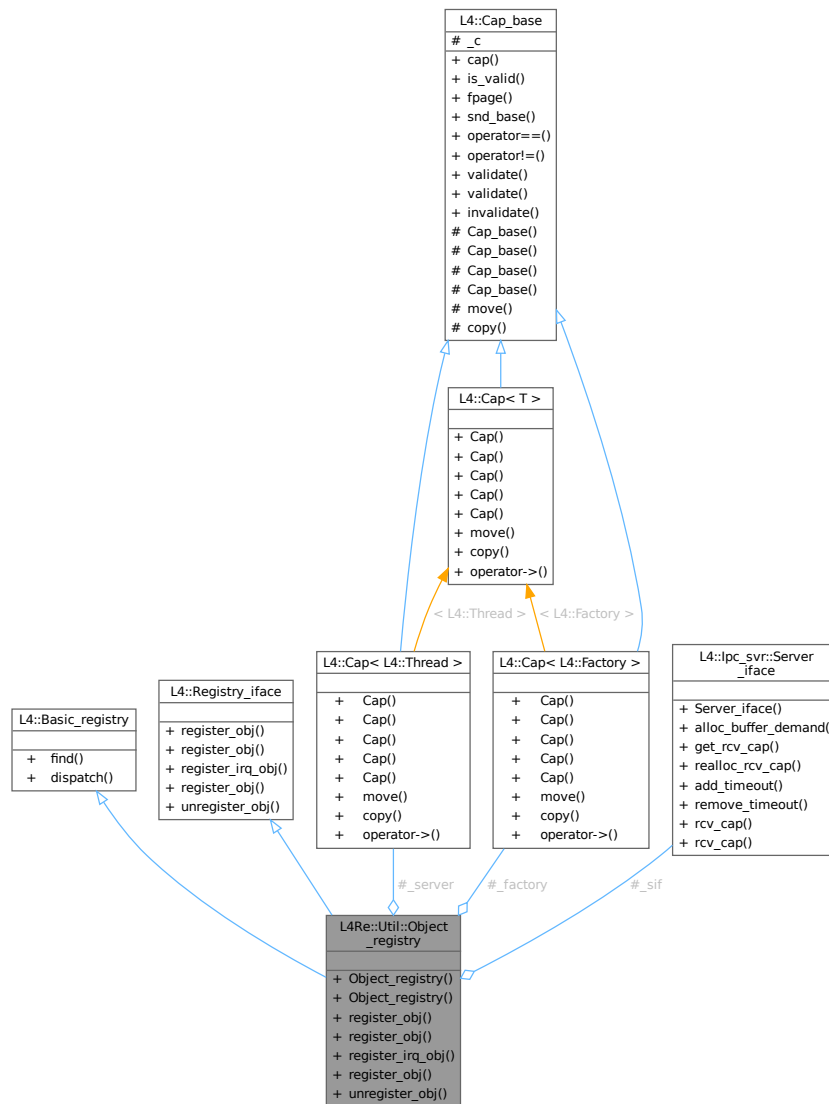
A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread.

```
#include <object_registry>
```

Inheritance diagram for L4Re::Util::Object_registry:



Collaboration diagram for L4Re::Util::Object_registry:



Public Member Functions

- `Object_registry (L4::ipc_svr::Server_iface *sif)`
Create a registry for the main thread of the task using the default factory.
- `Object_registry (L4::ipc_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.

15.308.1 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`.

15.308.2 Constructor & Destructor Documentation

15.308.2.1 `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`.

15.308.2.2 `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](#).

15.308.3 Member Function Documentation

15.308.3.1 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<L4::Irq></i>	Capability to a new IRQ object on success.
<i>L4::Cap<L4::Irq>::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](#).

Examples

[examples/libs/l4re/c++/shared_ds/ds_srv.cc](#).

Definition at line 238 of file [object_registry](#).

15.308.3.2 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<void></i>	A valid capability to a new IPC gate.
<i>L4::Cap<void>::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](#).

15.308.3.3 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<void></i>	The capability known as <i>service</i> on success.
<i>L4::Cap<void>::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](#), [examples/libs/l4re/c++/shared_ds/ds_srv.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 205 of file [object_registry](#).

15.308.3.4 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<L4::Rcv_endpoint></i>	Capability <i>ep</i> on success.
<i>L4::Cap<L4::Rcv_endpoint>::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](#).

15.308.3.5 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.

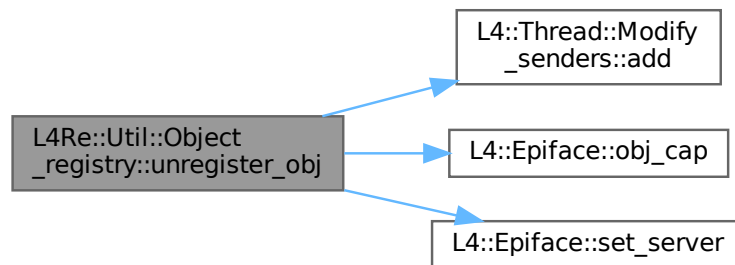
The capability used by the server object will be unmapped if *unmap* is true.

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:

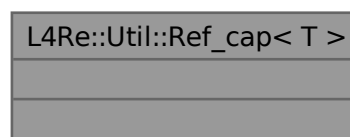
- `l4/re/util/object_registry`

15.309 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 >`:



15.309.1 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

T	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 165 of file [cap_alloc](#).

The documentation for this struct was generated from the following file:

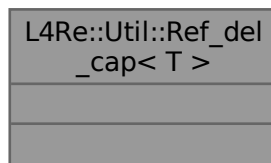
- [l4/re/util/cap_alloc](#)

15.310 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 >:



15.310.1 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

T	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 206 of file [cap_alloc](#).

The documentation for this struct was generated from the following file:

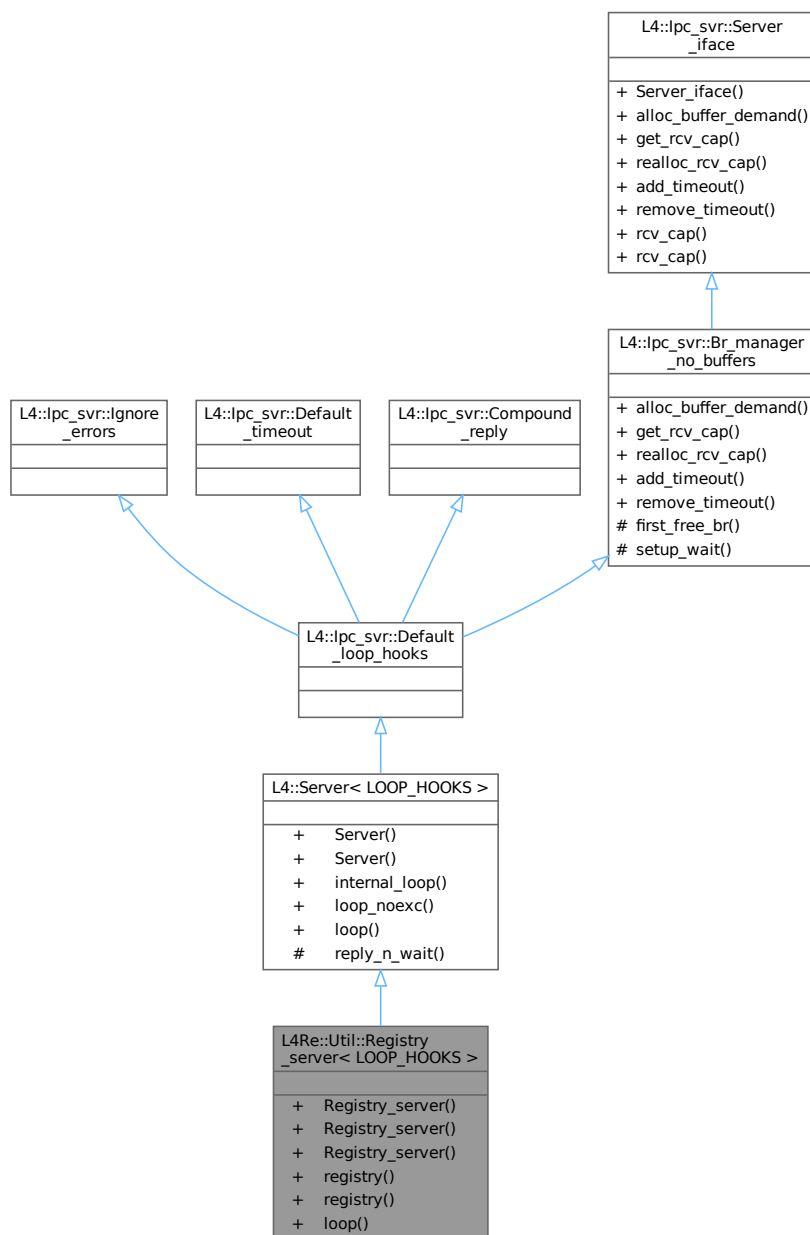
- [l4/re/util/cap_alloc](#)

15.311 L4Re::Util::Registry_server< LOOP_HOOKS > Class Template Reference

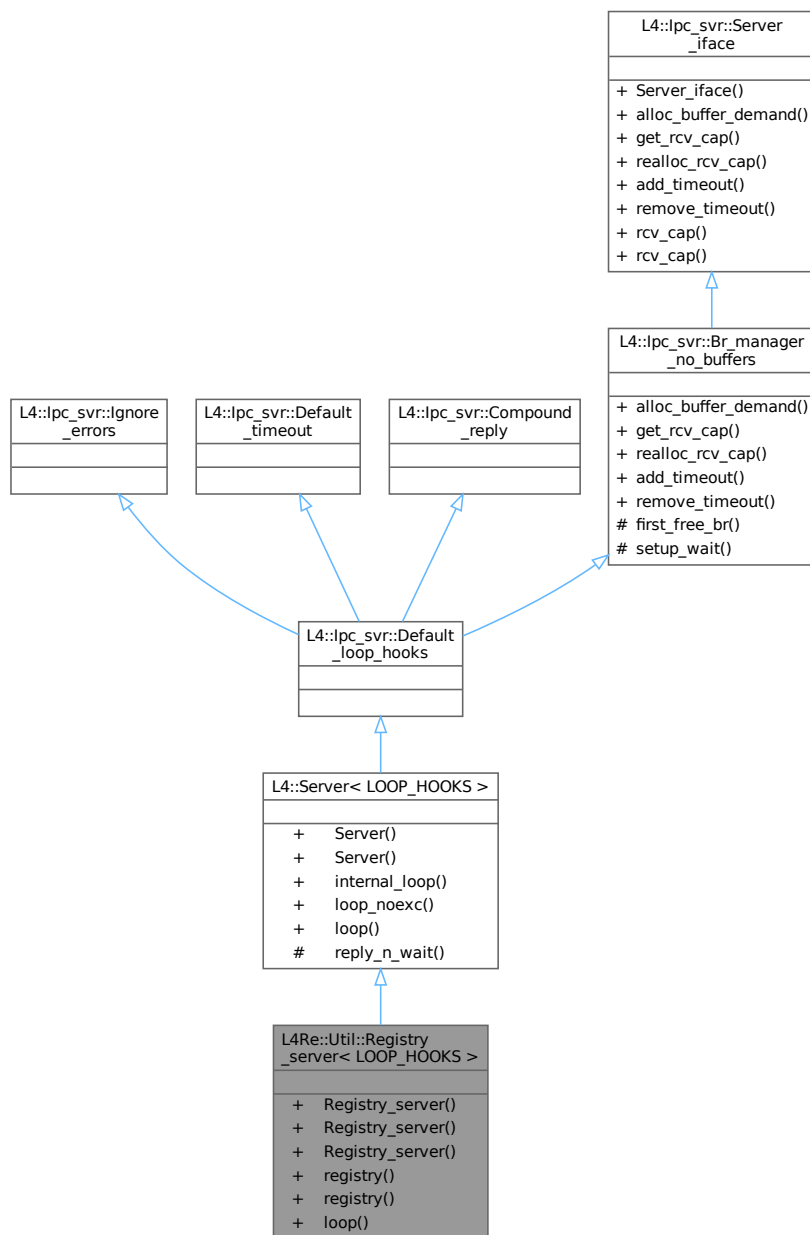
A server loop object which has a [Object_registry](#) included.

```
#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<>).

15.311.1 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 306 of file [object_registry](#).

15.311.2 Constructor & Destructor Documentation

15.311.2.1 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 318 of file [object_registry](#).

15.311.2.2 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 330 of file [object_registry](#).

15.311.2.3 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 341 of file [object_registry](#).

15.311.3 Member Function Documentation**15.311.3.1 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 l4_utcb() .
-------------	---

Examples

[examples/clntsrv/server.cc](#), [examples/libs/l4re/c++/shared_ds/ds_srv.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 357 of file [object_registry](#).

The documentation for this class was generated from the following file:

- [l4re/util/object_registry](#)

15.312 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()
+	copy()

Static Public Member Functions

- static void **free** ([L4::Cap_base](#) &c)
Free operation for [L4::Smart_cap](#).
- static void **invalidate** ([L4::Cap_base](#) &c)
Invalidate operation for [L4::Smart_cap](#).
- static [L4::Cap_base](#) **copy** ([L4::Cap_base](#) const &src)
Copy operation for [L4::Smart_cap](#).

15.312.1 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 59 of file [cap_alloc](#).

The documentation for this class was generated from the following file:

- [l4/re/util/cap_alloc](#)

15.313 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).

15.313.1 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 99 of file [cap_alloc](#).

The documentation for this class was generated from the following file:

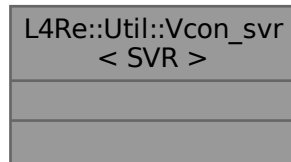
- [l4/re/util/cap_alloc](#)

15.314 L4Re::Util::Vcon_svr< SVR > Class Template Reference

[Console](#) server template class.

```
#include <vcon_svr>
```

Collaboration diagram for L4Re::Util::Vcon_svr< SVR >:



15.314.1 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 46 of file [vcon_svr](#).

The documentation for this class was generated from the following file:

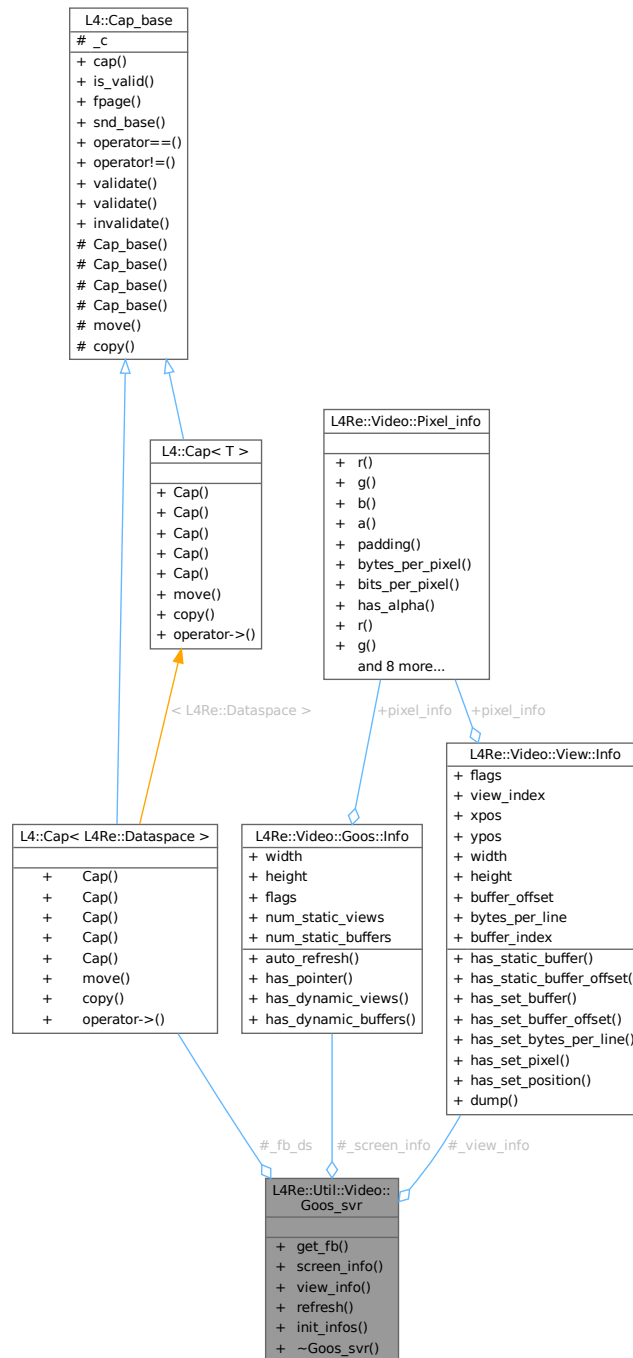
- `l4/re/util/vcon_svr`

15.315 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.

15.315.1 Detailed Description

Goos server class.

Definition at line 36 of file [goos_svr](#).

15.315.2 Member Function Documentation

15.315.2.1 [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](#).

15.315.2.2 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](#).

15.315.2.3 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](#).

15.315.2.4 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](#).

15.315.2.5 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:

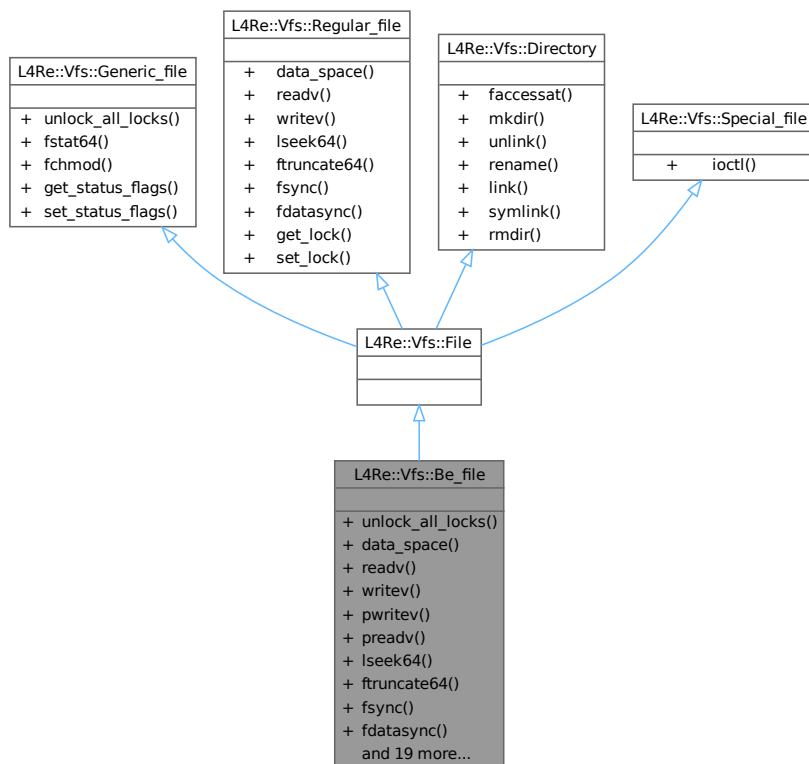
- [l4/re/util/video/goos_svr](#)

15.316 L4Re::Vfs::Be_file Class Reference

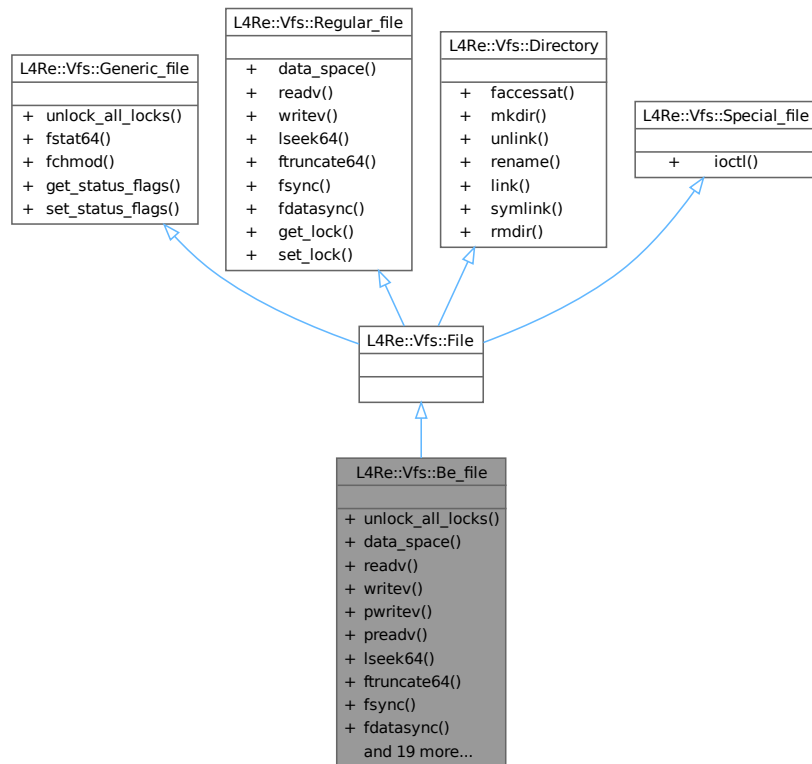
Boiler plate class for implementing an open file for [L4Re::Vfs](#).

```
#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 ()` const 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 fdatsync ()` const noexcept override
Default backend for POSIX fdatsync.

- 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.

15.316.1 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](#).

15.316.2 Member Function Documentation

15.316.2.1 data_space()

```
L4::Cap< L4Re::Dataspace > L4Re::Vfs::Be_file::data_space ( ) const [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](#).

15.316.2.2 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](#).

15.316.2.3 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:

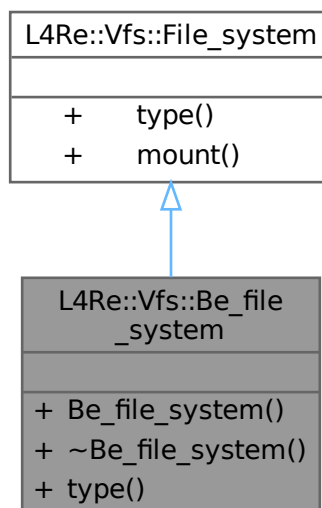
- l4/l4re_vfs/backend

15.317 L4Re::Vfs::Be_file_system Class Reference

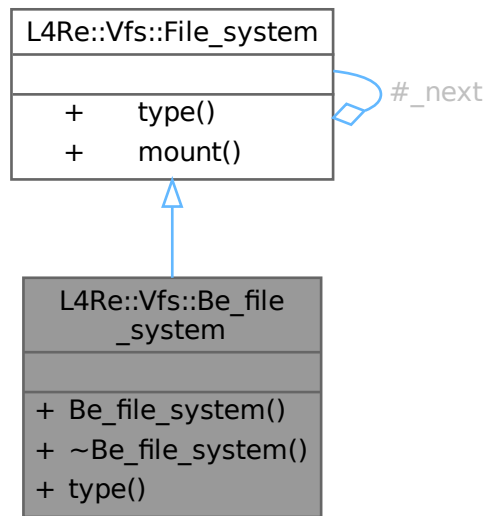
Boilerplate class for implementing a [L4Re::Vfs::File_system](#).

```
#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.

15.317.1 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 308 of file [backend](#).

15.317.2 Constructor & Destructor Documentation

15.317.2.1 `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 type() shall return.
---------------	--

This constructor takes care of registering the file system in the registry of `L4Re::Vfs::vfs_ops`.

Definition at line 322 of file [backend](#).

15.317.2.2 `~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 334 of file [backend](#).

15.317.3 Member Function Documentation

15.317.3.1 `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 344 of file [backend](#).

The documentation for this class was generated from the following file:

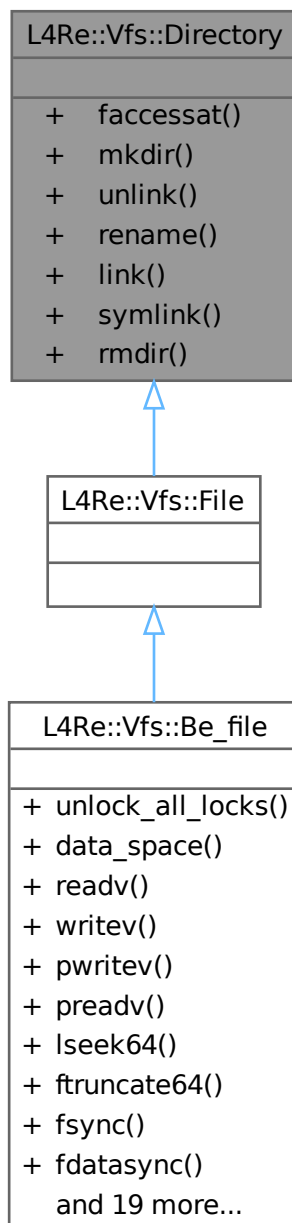
- `l4/l4re_vfs/backend`

15.318 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.

15.318.1 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](#).

15.318.2 Member Function Documentation

15.318.2.1 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](#).

15.318.2.2 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](#).

15.318.2.3 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](#).

15.318.2.4 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](#).

15.318.2.5 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](#).

15.318.2.6 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](#).

15.318.2.7 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:

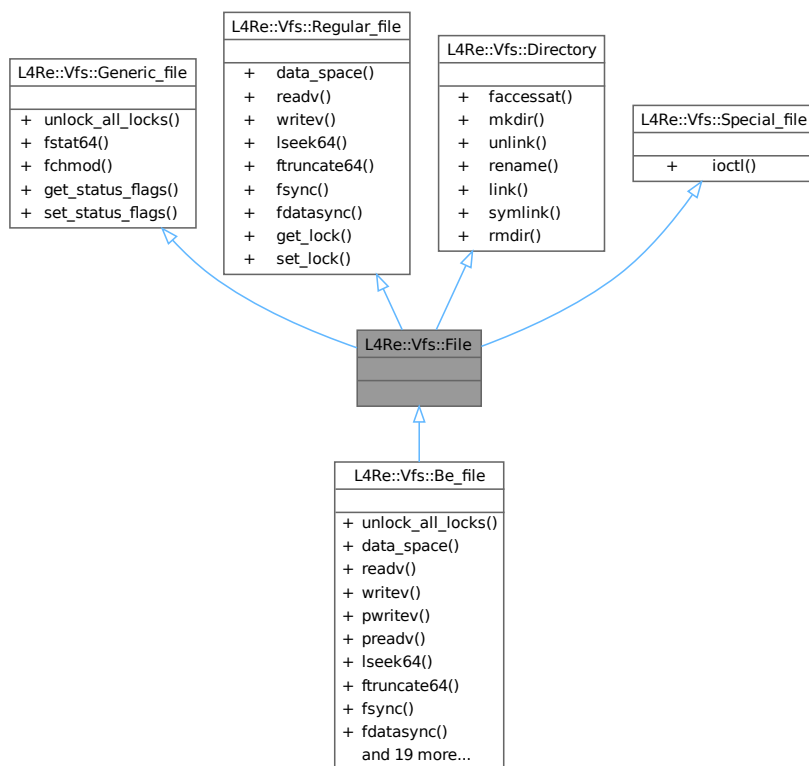
- l4/l4re_vfs/vfs.h

15.319 L4Re::Vfs::File Class Reference

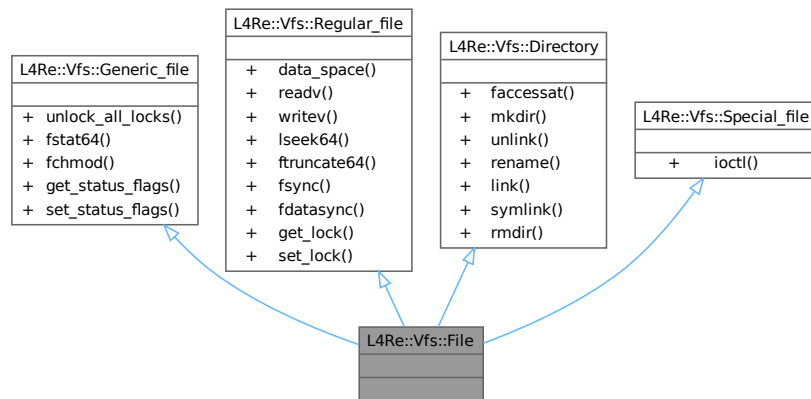
The basic interface for an open POSIX file.

```
#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](#) () const 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 [fdatsync](#) () 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.

15.319.1 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:

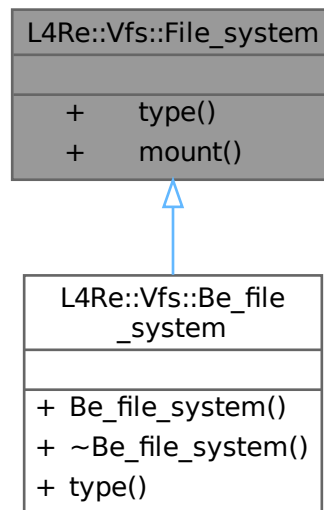
- [l4/l4re_vfs/vfs.h](#)

15.320 L4Re::Vfs::File_system Class Reference

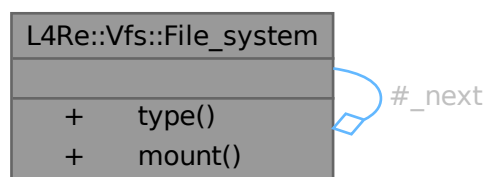
Basic interface for an [L4Re::Vfs](#) file system.

```
#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.

15.320.1 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 828 of file [vfs.h](#).

15.320.2 Member Function Documentation

15.320.2.1 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

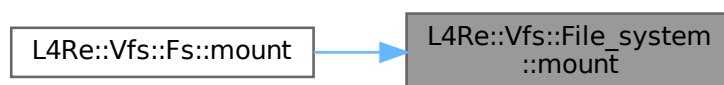
	<i>source</i>	The path to the source device to mount. This may also be some URL or anything file-system specific.
	<i>mountflags</i>	The mount flags as specified in the POSIX mount call.
	<i>data</i>	The data as specified in the POSIX mount call. The contents are file-system specific.
out	<i>dir</i>	A new directory object representing the file-system root directory.

Returns

0 on success, and <0 on error (e.g. -EINVAL).

Referenced by [L4Re::Vfs::Fs::mount\(\)](#).

Here is the caller graph for this function:



15.320.2.2 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:

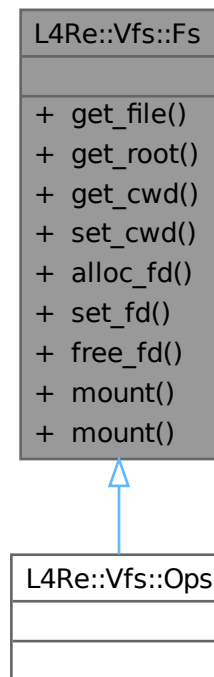
- l4/l4re_vfs/vfs.h

15.321 L4Re::Vfs::Fs Class Reference

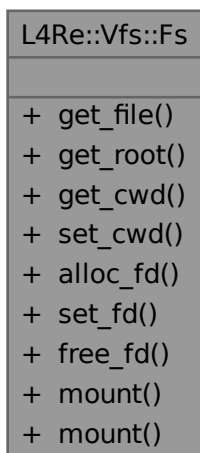
POSIX File-system related functionality.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Fs:



Collaboration diagram for L4Re::Vfs::Fs:



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.

15.321.1 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 881 of file [vfs.h](#).

15.321.2 Member Function Documentation**15.321.2.1 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.

15.321.2.2 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*.

15.321.2.3 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.

15.321.2.4 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.

15.321.2.5 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 file:

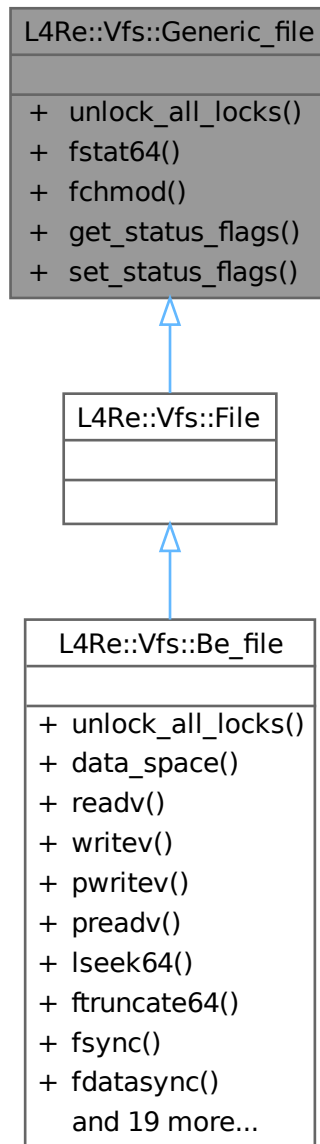
- l4/l4re_vfs/vfs.h

15.322 L4Re::Vfs::Generic_file Class Reference

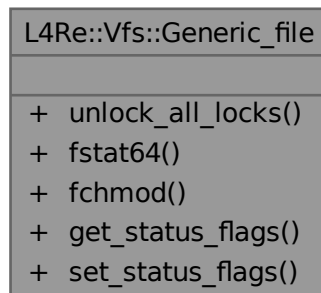
The common interface for an open POSIX file.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Generic_file:



Collaboration diagram for L4Re::Vfs::Generic_file:



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).

15.322.1 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](#).

15.322.2 Member Function Documentation

15.322.2.1 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](#).

15.322.2.2 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](#).

15.322.2.3 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](#).

15.322.2.4 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](#).

15.322.2.5 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:

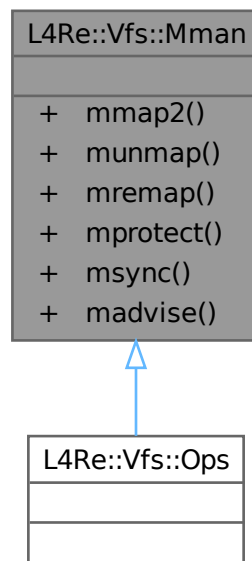
- `l4/l4re_vfs/vfs.h`

15.323 L4Re::Vfs::Mman Class Reference

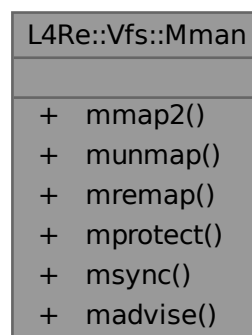
Interface for POSIX memory management.

```
#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.

15.323.1 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 744 of file [vfs.h](#).

The documentation for this class was generated from the following file:

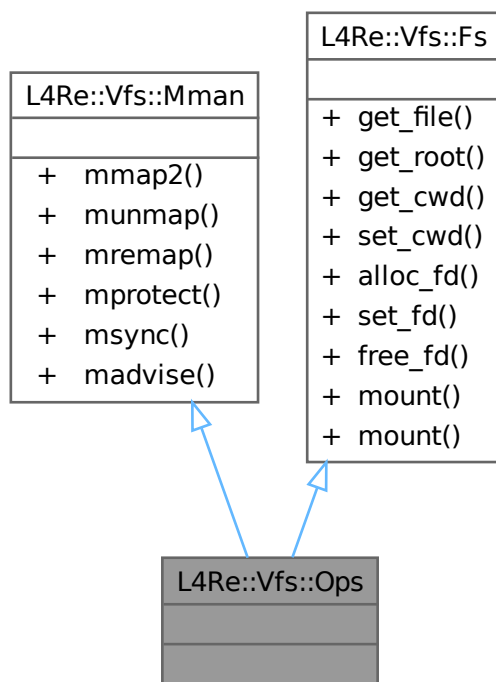
- l4/l4re_vfs/vfs.h

15.324 L4Re::Vfs::Ops Class Reference

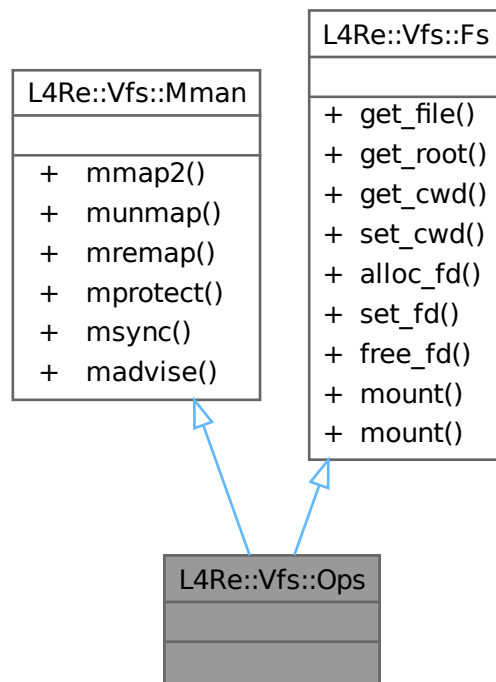
Interface for the POSIX backends of an application.

```
#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.

15.324.1 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 1007 of file `vfs.h`.

The documentation for this class was generated from the following file:

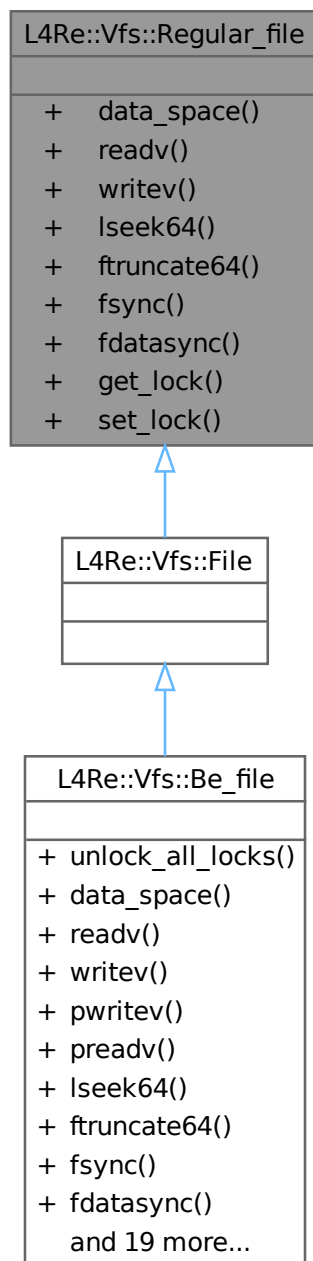
- `l4/l4re_vfs/vfs.h`

15.325 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"> + data_space() + readv() + writev() + lseek64() + ftruncate64() + fsync() + fdatsync() + get_lock() + set_lock()

Public Member Functions

- virtual [L4::Cap](#)< [L4Re::Dataspace](#) > [data_space](#) () const 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 [fdatsync](#) () 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.

15.325.1 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](#).

15.325.2 Member Function Documentation

15.325.2.1 data_space()

```
virtual L4::Cap< L4Re::Dataspace > L4Re::Vfs::Regular_file::data_space ( ) const [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](#).

15.325.2.2 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](#).

15.325.2.3 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](#).

15.325.2.4 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](#).

15.325.2.5 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](#).

15.325.2.6 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](#).

15.325.2.7 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](#).

15.325.2.8 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](#).

15.325.2.9 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:

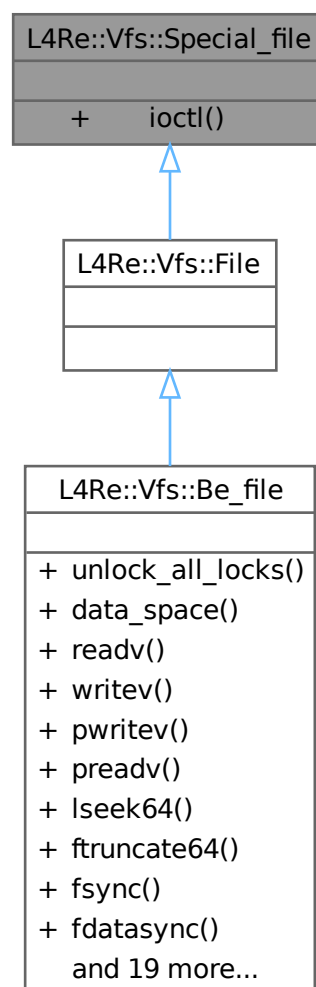
- l4/l4re_vfs/vfs.h

15.326 L4Re::Vfs::Special_file Class Reference

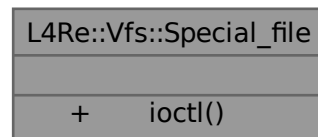
Interface for a POSIX file that provides special file semantics.

```
#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.

15.326.1 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](#).

15.326.2 Member Function Documentation

15.326.2.1 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

15.327 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"> + Color_component() + Color_component() + size() + shift() + operator==() + get() + set() + dump()

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.

15.327.1 Detailed Description

A color component.

Definition at line 32 of file [colors](#).

15.327.2 Constructor & Destructor Documentation

15.327.2.1 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](#).

15.327.3 Member Function Documentation

15.327.3.1 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
----------	--------

Returns

The stream

Definition at line 94 of file [colors](#).

15.327.3.2 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](#).

15.327.3.3 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](#).

15.327.3.4 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 85 of file [colors](#).

15.327.3.5 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:

**15.327.3.6 size()**

```
unsigned char L4Re::Video::Color_component::size ( ) const [inline]
```

Return the number of bits used by the component.

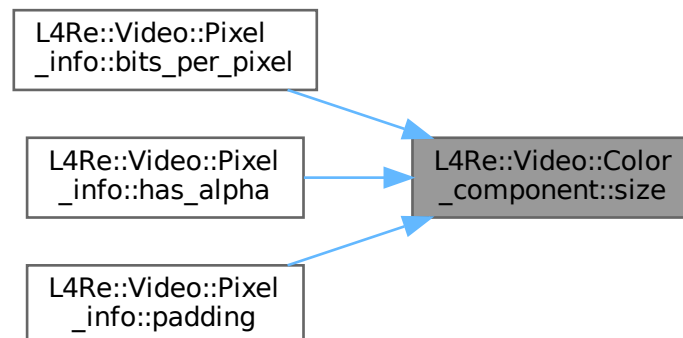
Returns

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:

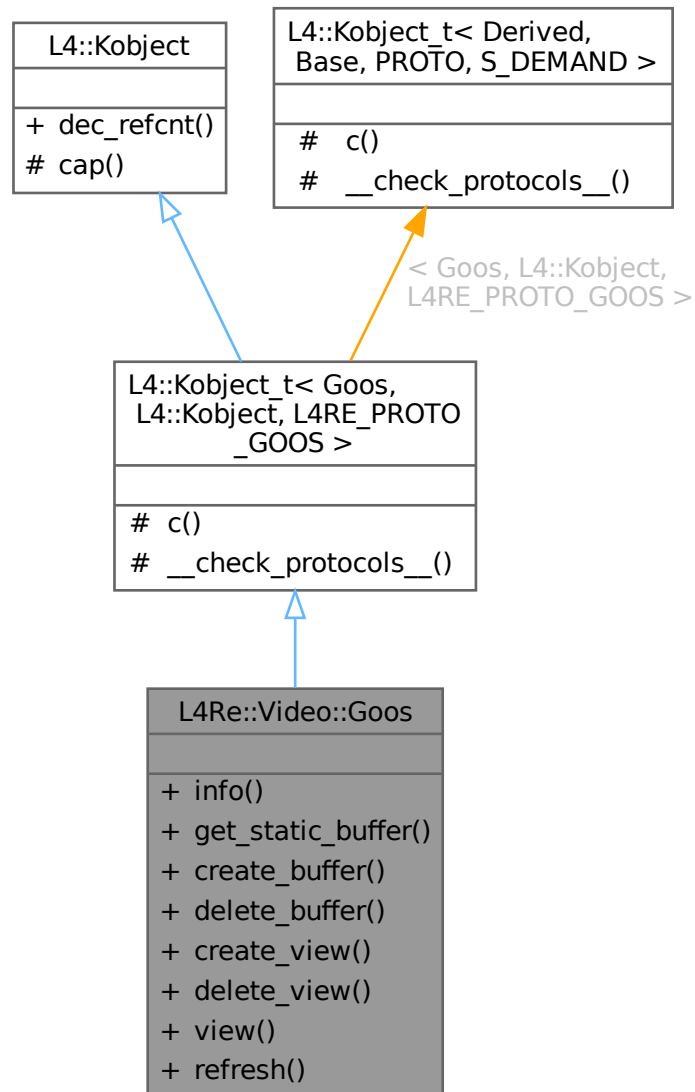
- `I4/re/video/colors`

15.328 L4Re::Video::Goos Class Reference

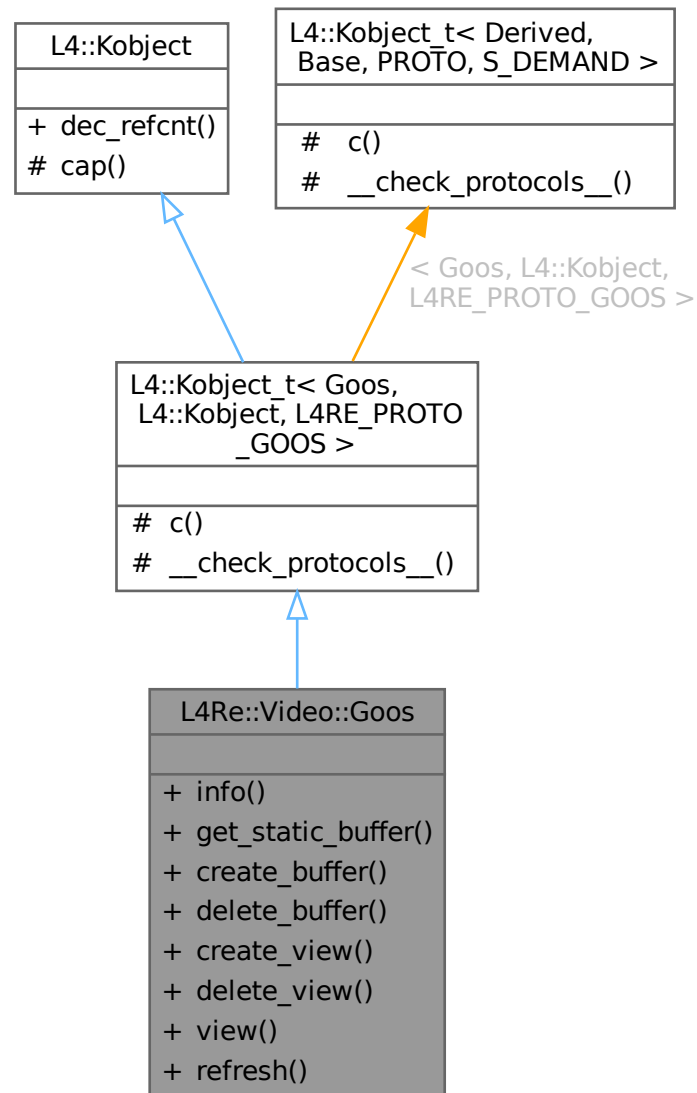
Class that abstracts framebuffers.

```
#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.

15.328.1 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](#).

15.328.2 Member Enumeration Documentation

15.328.2.1 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](#).

15.328.3 Member Function Documentation

15.328.3.1 `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

≥ 0	Success, the value returned is the buffer index.
< 0	Error

15.328.3.2 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

≥ 0	Success, the value returned is the view index.
< 0	Error

Definition at line 315 of file [goos](#).

15.328.3.3 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

15.328.3.4 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](#).

15.328.3.5 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

15.328.3.6 info()

```
long L4Re::Video::Goos::info (
    Info * info )
```

Return the [Goos](#) information of the [Goos](#).

Parameters

out	<i>info</i>	Goos information structure pointer.
-----	-------------	---

Return values

0	Success
<0	Error

15.328.3.7 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:

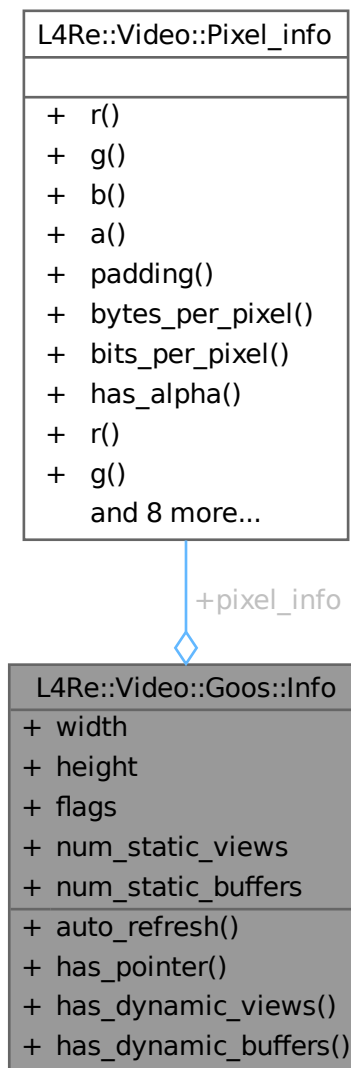
- [l4/re/video/goos](#)

15.329 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.

15.329.1 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

15.330 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"> + r() + g() + b() + a() + padding() + bytes_per_pixel() + bits_per_pixel() + has_alpha() + r() + g() and 8 more...

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](#) ()=default
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.

15.330.1 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 107 of file [colors](#).

15.330.2 Constructor & Destructor Documentation

15.330.2.1 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 225 of file [colors](#).

15.330.2.2 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 237 of file [colors](#).

15.330.3 Member Function Documentation

15.330.3.1 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 136 of file [colors](#).

15.330.3.2 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 200 of file [colors](#).

15.330.3.3 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 130 of file [colors](#).

15.330.3.4 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 194 of file [colors](#).

15.330.3.5 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 169 of file [colors](#).

References [L4Re::Video::Color_component::size\(\)](#).

Here is the call graph for this function:



15.330.3.6 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 163 of file [colors](#).

15.330.3.7 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 206 of file [colors](#).

15.330.3.8 dump()

```
template<typename OUT >
void L4Re::Video::Pixel_info::dump (
    OUT & s ) const [inline]
```

Dump information on the pixel to a stream.

Parameters

s	Stream
---	--------

Returns

The stream

Definition at line 260 of file [colors](#).

Referenced by [L4Re::Video::View::Info::dump\(\)](#).

Here is the caller graph for this function:



15.330.3.9 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 124 of file [colors](#).

15.330.3.10 g() [2/2]

```
void L4Re::Video::Pixel_info::g (
    Color_component const & c ) [inline]
```

Set the green color component of the pixel.

Parameters

c	Green color component.
---	------------------------

Definition at line 188 of file [colors](#).

15.330.3.11 has_alpha()

```
bool L4Re::Video::Pixel_info::has_alpha ( ) const [inline]
```

Return whether the pixel has an alpha channel.

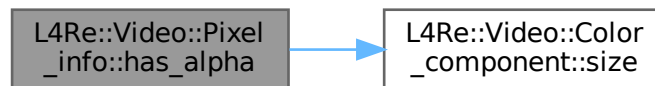
Returns

True if the pixel has an alpha channel, false if not.

Definition at line 176 of file [colors](#).

References [L4Re::Video::Color_component::size\(\)](#).

Here is the call graph for this function:

**15.330.3.12 operator==()**

```
bool L4Re::Video::Pixel_info::operator== (
    Pixel_info const & o ) const [inline]
```

Compare for complete equality of the color space.

Parameters

o	A Pixel_info to compare to.
---	---

Returns

true if the both [Pixel_info](#)'s are equal, false if not.

Definition at line 249 of file [colors](#).

15.330.3.13 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.

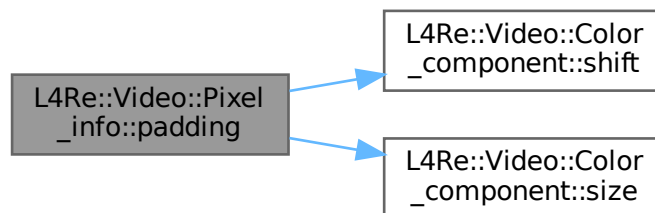
Returns

Padding pseudo component.

Definition at line 144 of file [colors](#).

References [L4Re::Video::Color_component::shift\(\)](#), and [L4Re::Video::Color_component::size\(\)](#).

Here is the call graph for this function:

**15.330.3.14 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 118 of file [colors](#).

15.330.3.15 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 182 of file [colors](#).

The documentation for this class was generated from the following file:

- [l4/re/video/colors](#)

15.331 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"> + info() + set_info() + set_viewport() + stack() + push_top() + push_bottom() + refresh() + valid()

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.

15.331.1 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](#).

15.331.2 Member Enumeration Documentation

15.331.2.1 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	View 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](#).

15.331.2.2 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](#).

15.331.3 Member Function Documentation

15.331.3.1 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](#).

15.331.3.2 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](#).

15.331.3.3 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](#).

15.331.3.4 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](#).

15.331.3.5 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>	View 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:

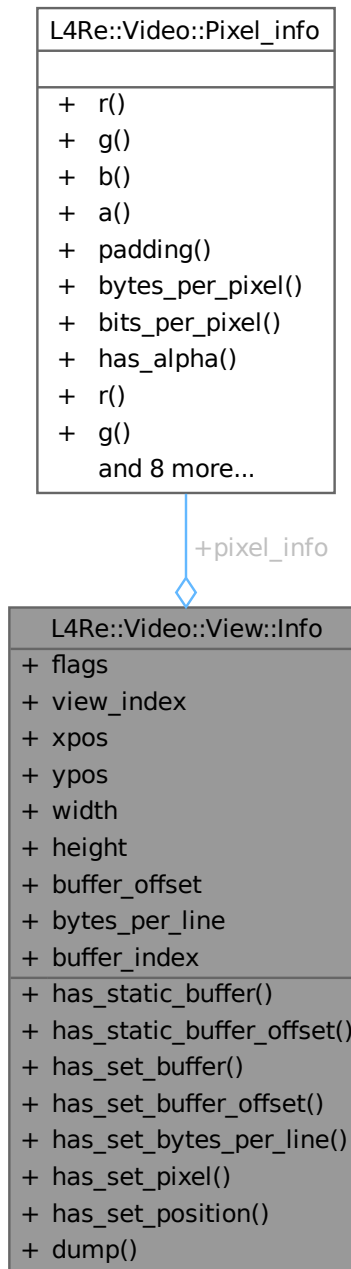
- [l4/re/video/goos](#)

15.332 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**

Flags, see [Flags](#) and [V_flags](#).

- unsigned **view_index**

Index of the view.

- unsigned long **xpos**

X position in pixels of the view in the [Goos](#).

- unsigned long **ypos**

Y position in pixels of the view in the [Goos](#).

- unsigned long **width**

Width of the view in pixels.

- unsigned long **height**

Height of the view in pixels.

- unsigned long **buffer_offset**

Offset in the memory buffer in bytes.

- unsigned long **bytes_per_line**

Bytes per line.

- [Pixel_info](#) **pixel_info**

Pixel information.

- unsigned **buffer_index**

Number of the buffer used for this view.

15.332.1 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

15.333 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

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.

15.333.1 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](#)

15.334 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

15.334.1 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](#)

15.335 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	

15.335.1 Detailed Description

Auxiliary vector element for a single unsigned data word.

Definition at line 124 of file [elf_aux.h](#).

The documentation for this struct was generated from the following file:

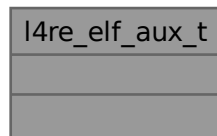
- [l4re/elf_aux.h](#)

15.336 l4re_elf_aux_t Struct Reference

Generic header for each auxiliary vector element.

```
#include <elf_aux.h>
```

Collaboration diagram for l4re_elf_aux_t:



15.336.1 Detailed Description

Generic header for each auxiliary vector element.

Definition at line 104 of file [elf_aux.h](#).

The documentation for this struct was generated from the following file:

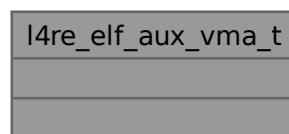
- [l4re/elf_aux.h](#)

15.337 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:



15.337.1 Detailed Description

Auxiliary vector element for a reserved virtual memory area.

Definition at line 113 of file [elf_aux.h](#).

The documentation for this struct was generated from the following file:

- [l4/re/elf_aux.h](#)

15.338 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:

l4re_env_cap_entry_t
+ cap
+ flags
+ name
+ l4re_env_cap_entry_t()
+ l4re_env_cap_entry_t()

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.

15.338.1 Detailed Description

Entry in the [L4Re](#) environment array for the named initial objects.

Definition at line [50](#) of file [env.h](#).

15.338.2 Constructor & Destructor Documentation

15.338.2.1 `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](#).

15.338.3 Field Documentation

15.338.3.1 `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:

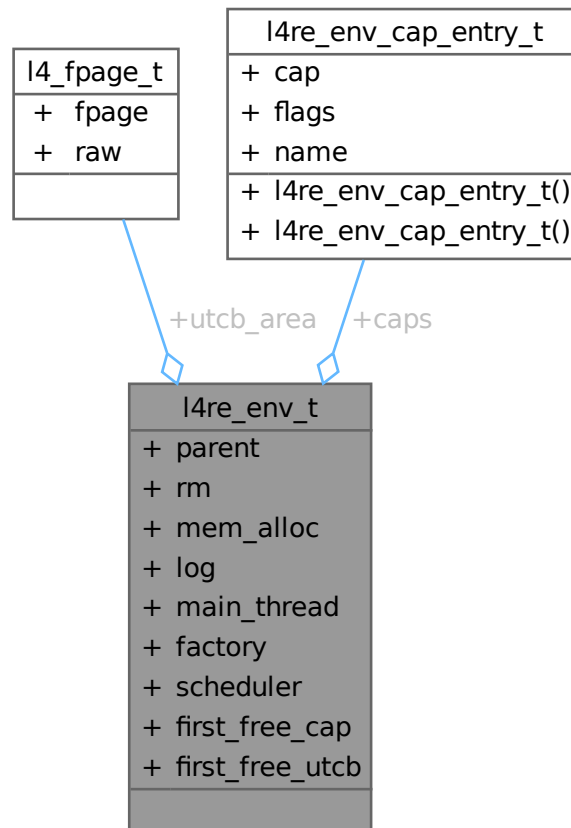
- [l4/re/env.h](#)

15.339 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.

15.339.1 Detailed Description

Initial environment data structure.

See also

[Initial environment](#)

Definition at line 109 of file [env.h](#).

15.339.2 Field Documentation

15.339.2.1 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:

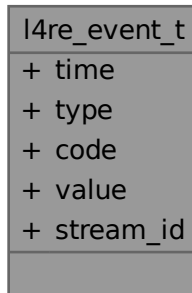
- [l4/re/env.h](#)

15.340 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.

15.340.1 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:

- [l4/re/c/event.h](#)

15.341 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

15.341.1 Detailed Description

Color component structure.

Definition at line 31 of file [colors.h](#).

The documentation for this struct was generated from the following file:

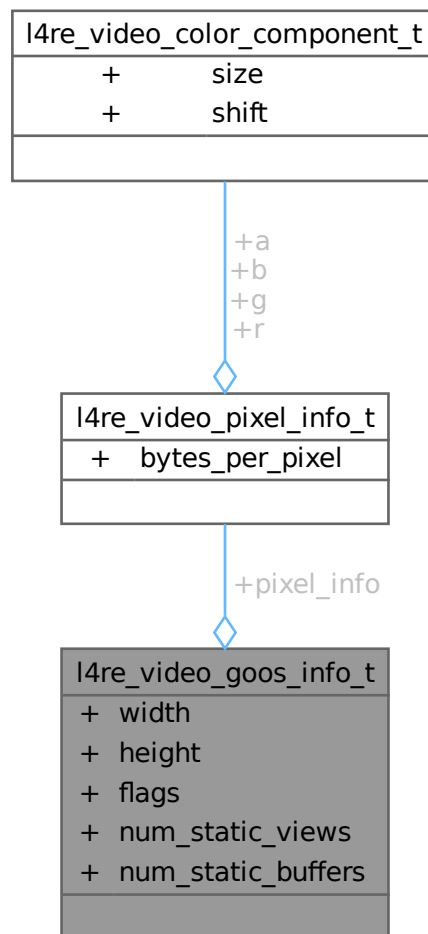
- [l4re/c/video/colors.h](#)

15.342 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.

15.342.1 Detailed Description

Goos information structure.

Definition at line 51 of file [goos.h](#).

The documentation for this struct was generated from the following file:

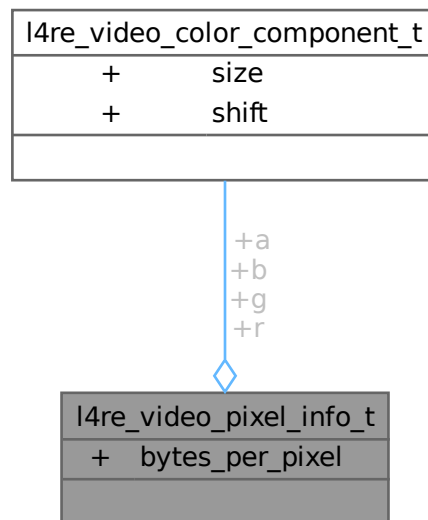
- [l4re/c/video/goos.h](#)

15.343 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.

15.343.1 Detailed Description

Pixel_info structure.

Definition at line 41 of file [colors.h](#).

The documentation for this struct was generated from the following file:

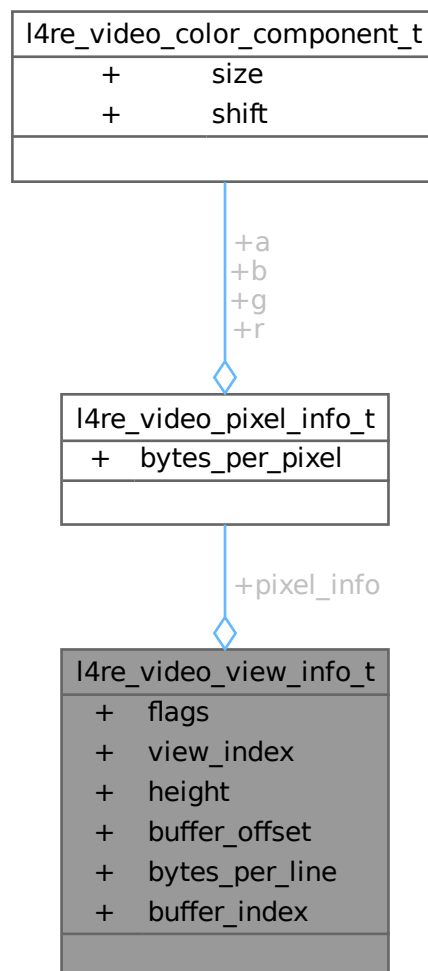
- [l4re/c/video/colors.h](#)

15.344 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.

15.344.1 Detailed Description

View information structure.

Definition at line 59 of file [view.h](#).

The documentation for this struct was generated from the following file:

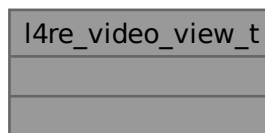
- [l4re/c/video/view.h](#)

15.345 l4re_video_view_t Struct Reference

C representation of a goos view.

```
#include <view.h>
```

Collaboration diagram for `l4re_video_view_t`:



15.345.1 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](#)

15.346 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

15.346.1 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:

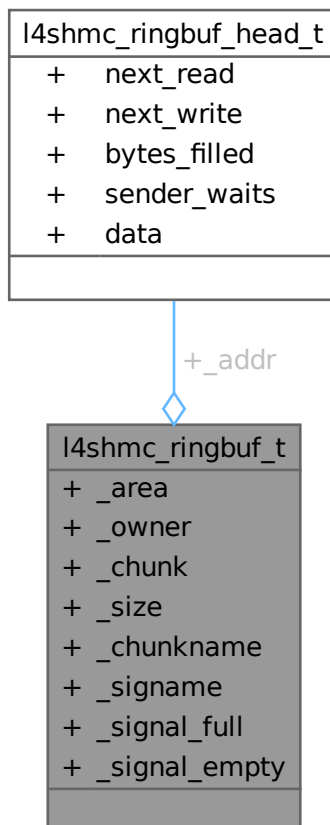
- [l4shmc/ringbuf.h](#)

15.347 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

15.347.1 Detailed Description

Ring buffer.

Definition at line 84 of file [ringbuf.h](#).

The documentation for this struct was generated from the following file:

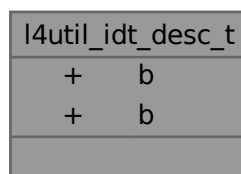
- [l4/shmc/ringbuf.h](#)

15.348 l4util_idt_desc_t Struct Reference

IDT entry.

```
#include <idt.h>
```

Collaboration diagram for `l4util_idt_desc_t`:



Data Fields

- [l4_uint64_t](#) **b**
see Intel doc
- [l4_uint32_t](#) **b**
see Intel doc

15.348.1 Detailed Description

IDT entry.

Definition at line 33 of file [idt.h](#).

The documentation for this struct was generated from the following files:

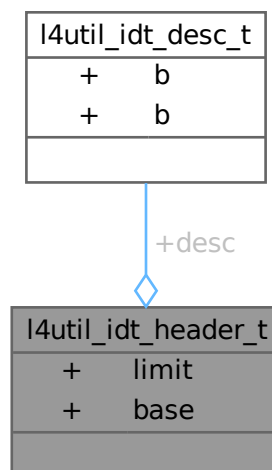
- [amd64/l4/util/idt.h](#)
- [x86/l4/util/idt.h](#)

15.349 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)

15.349.1 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](#)

15.350 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.

15.350.1 Detailed Description

Base module structure.

Definition at line 35 of file [l4mod.h](#).

15.350.2 Field Documentation

15.350.2.1 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](#)

15.351 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.

15.351.1 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](#)

15.352 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

15.352.1 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:

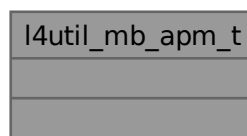
- [l4/util/mb_info.h](#)

15.353 l4util_mb_apm_t Struct Reference

APM BIOS info.

```
#include <mb_info.h>
```

Collaboration diagram for l4util_mb_apm_t:



15.353.1 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](#)

15.354 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

15.354.1 Detailed Description

Drive Info structure.

Definition at line 75 of file [mb_info.h](#).

15.354.2 Field Documentation

15.354.2.1 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](#).

15.354.2.2 drive_mode

`l4_uint8_t l4util_mb_drive_t::drive_mode`

<The BIOS drive number.

Definition at line 79 of file [mb_info.h](#).

15.354.2.3 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](#)

15.355 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.

15.355.1 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](#)

15.356 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:

l4util_mb_mod_t
+ mod_start
+ mod_end
+ cmdline
+ pad

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

15.356.1 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:

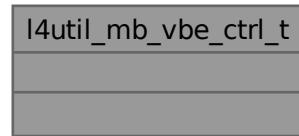
- [l4/util/mb_info.h](#)

15.357 l4util_mb_vbe_ctrl_t Struct Reference

VBE controller information.

```
#include <mb_info.h>
```

Collaboration diagram for `l4util_mb_vbe_ctrl_t`:



15.357.1 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](#)

15.358 l4util_mb_vbe_mode_t Struct Reference

VBE mode information.

```
#include <mb_info.h>
```

Collaboration diagram for l4util_mb_vbe_mode_t:

l4util_mb_vbe_mode_t
+ 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
+ reversed2
+ 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
* reversed2
* 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)

15.358.1 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:

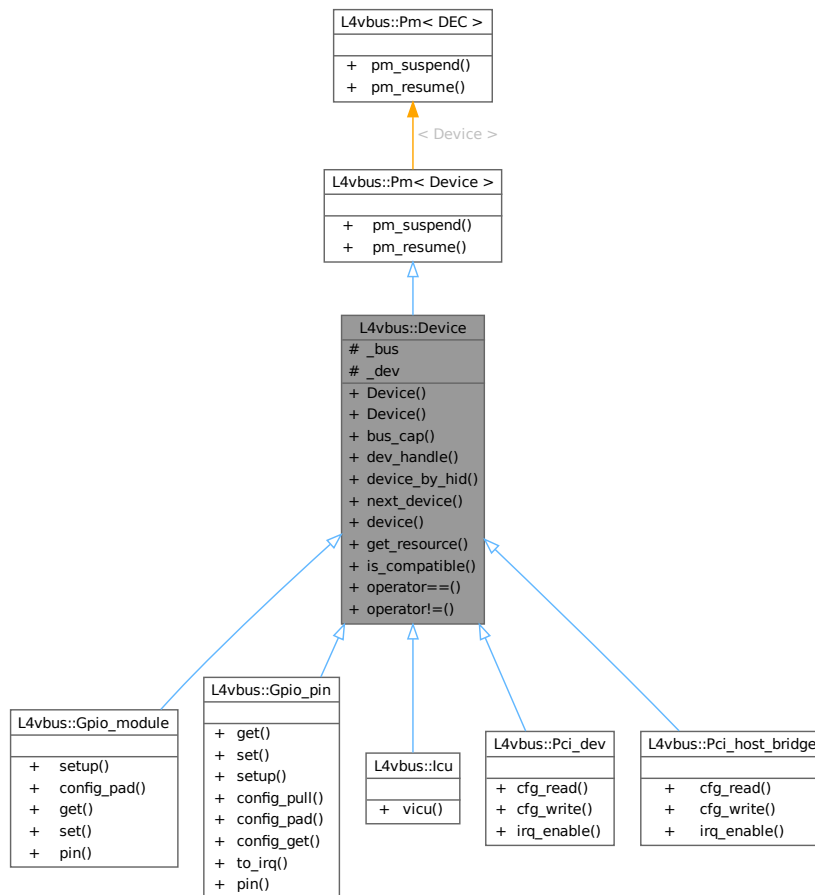
- [l4/util/mb_info.h](#)

15.359 L4vbus::Device Class Reference

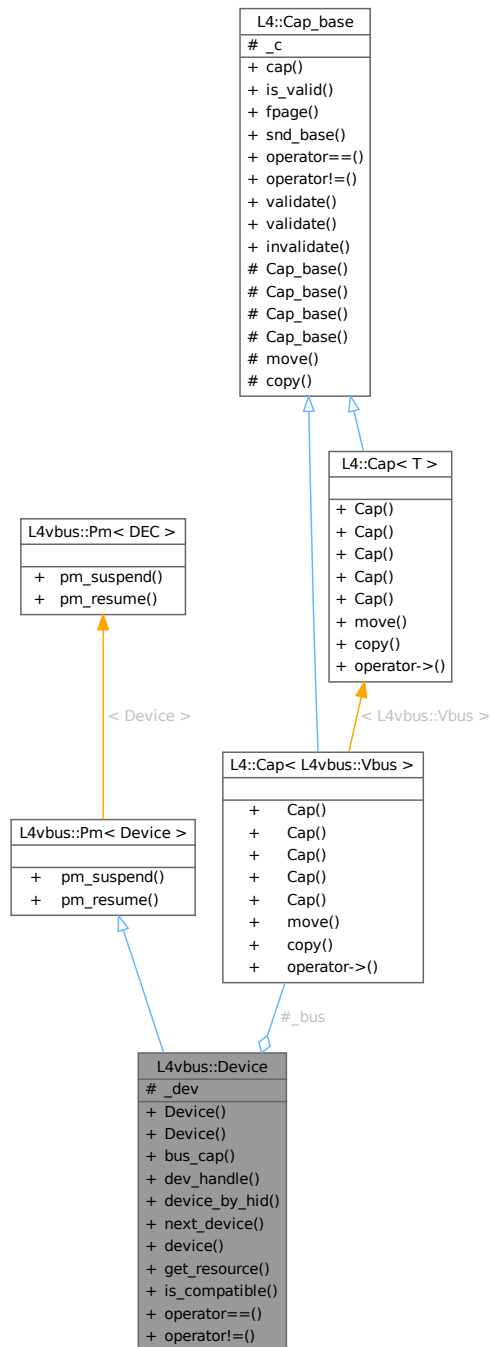
Device on a [L4vbus::Vbus](#).

```
#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 (int 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.

15.359.1 Detailed Description

`Device` on a `L4vbus::Vbus`.

Definition at line 85 of file `vbus`.

15.359.2 Constructor & Destructor Documentation

15.359.2.1 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](#).

15.359.3 Member Function Documentation

15.359.3.1 bus_cap()

```
L4::Cap< Vbus > L4vbus::Device::bus_cap ( ) const [inline]
```

Access the [Vbus](#) capability of the underlying virtual bus.

Returns

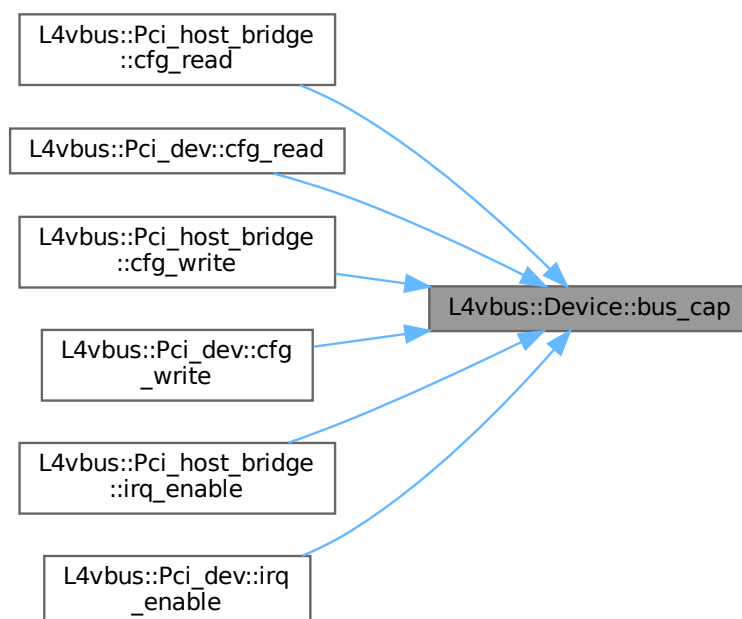
the capability to the underlying [Vbus](#).

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:



15.359.3.2 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](#).

15.359.3.3 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 <code>dev</code> 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:



15.359.3.4 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 (L4VBUS_NULL). 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>	Device information structure (might be NULL)

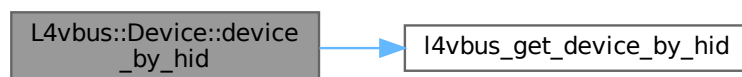
Return values

≥ 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:



15.359.3.5 `get_resource()`

```
int L4vbus::Device::get_resource (
    int 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 l4vbus_device_t structure that is returned by L4vbus::Device::device_by_hid() and L4vbus::Device::next_device() .
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.

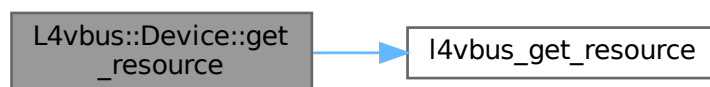
Return values

<i>0</i>	Success.
<i>-L4_ENOENT</i>	Invalid resource index <code>res_idx</code> .

Definition at line 211 of file [vbus](#).

References [_bus](#), [_dev](#), and [l4vbus_get_resource\(\)](#).

Here is the call graph for this function:



15.359.3.6 `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
------------	------------------------------

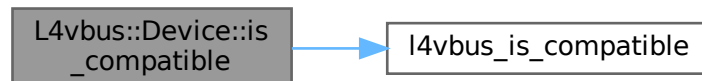
Returns

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:

**15.359.3.7 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 L4VBUS_NULL (Device::Device). 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)

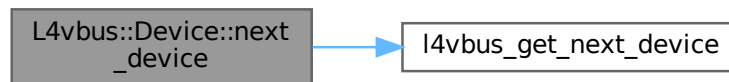
Returns

0 on success, else failure

Definition at line 173 of file [vbus](#).

References [_bus](#), [_dev](#), and [l4vbus_get_next_device\(\)](#).

Here is the call graph for this function:



15.359.3.8 `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](#).

15.359.3.9 `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:

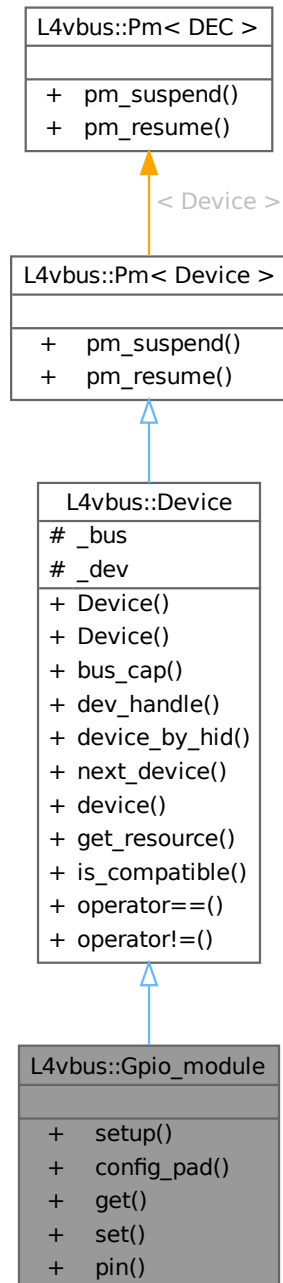
- [l4/vbus/vbus](#)

15.360 L4vbus::Gpio_module Class Reference

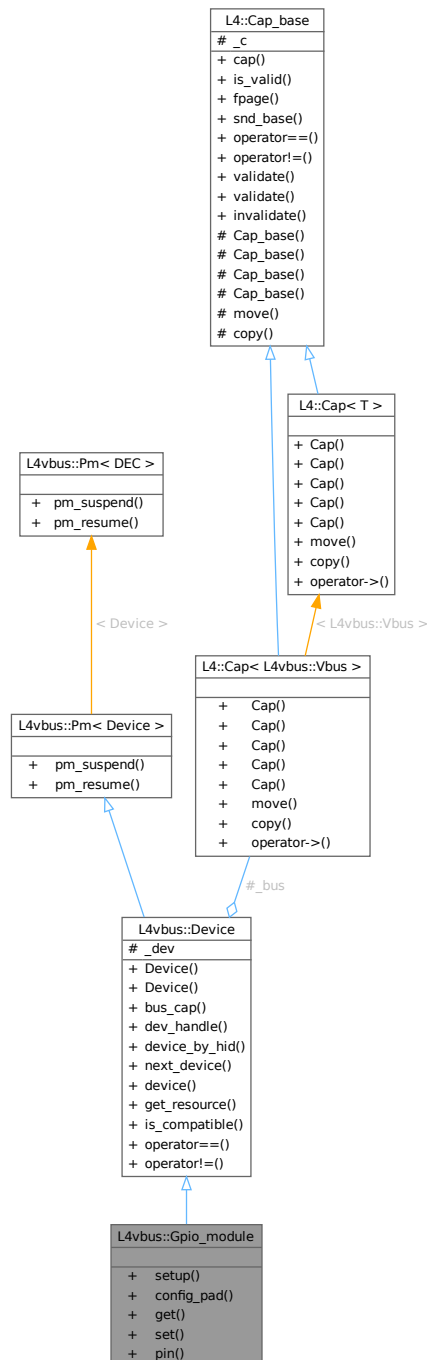
A [Gpio_module](#) groups multiple GPIO pins together.

```
#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 (int 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.

15.360.1 Detailed Description

A [Gpio_module](#) groups multiple GPIO pins together.

Definition at line 135 of file [vbus_gpio](#).

15.360.2 Member Function Documentation

15.360.2.1 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

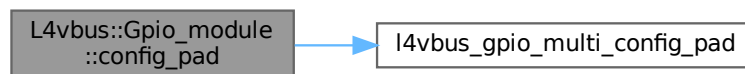
Returns

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:



15.360.2.2 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:



15.360.2.3 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 Gpio_pin for.
------------	--

Returns

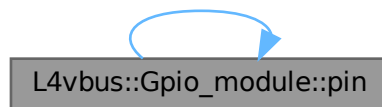
[Gpio_pin](#)

Definition at line 231 of file [vbus_gpio](#).

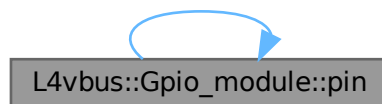
References [pin\(\)](#).

Referenced by [pin\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.360.2.4 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:



15.360.2.5 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 L4vbus_gpio_generic_func 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

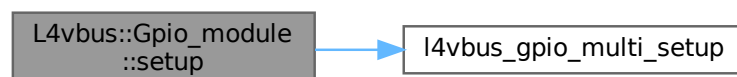
Returns

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:

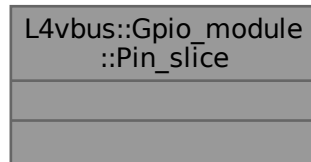
- [l4/vbus/vbus_gpio](#)

15.361 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:



15.361.1 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:

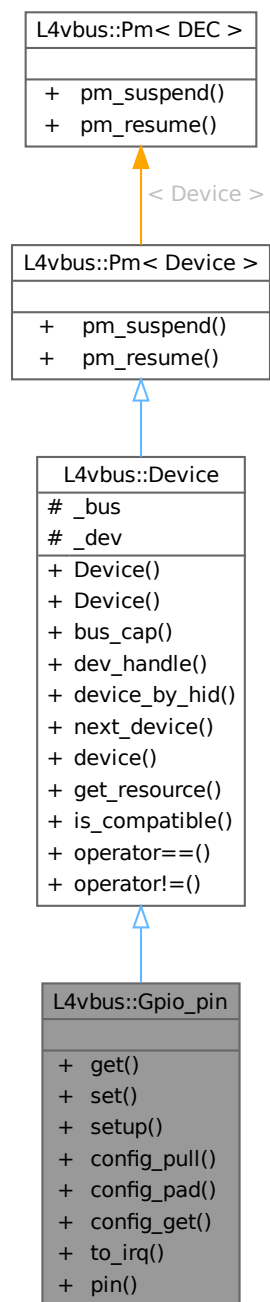
- l4/vbus/vbus_gpio

15.362 L4vbus::Gpio_pin Class Reference

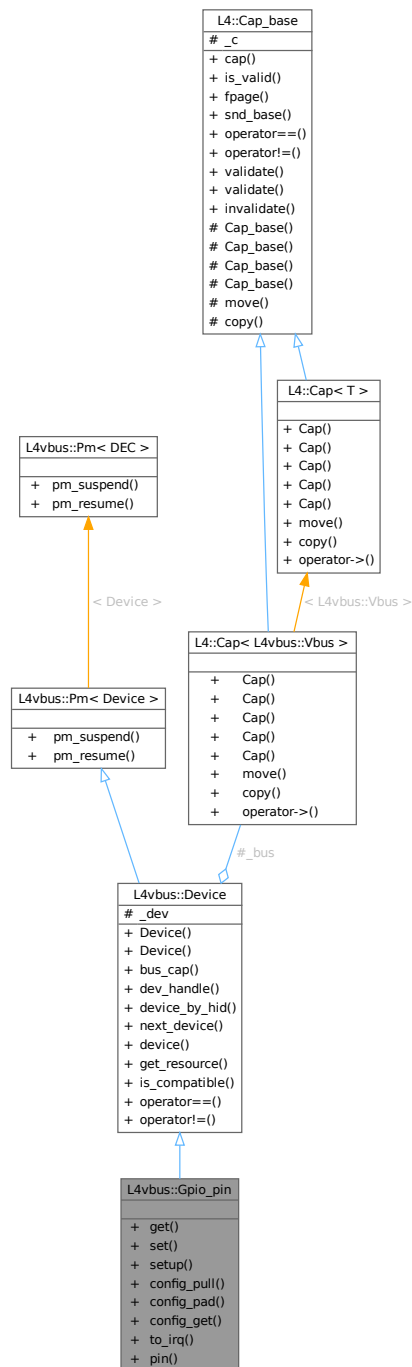
A GPIO pin.

```
#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` (int 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.

15.362.1 Detailed Description

A GPIO pin.

Definition at line 28 of file [vbus_gpio](#).

15.362.2 Member Function Documentation

15.362.2.1 `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.
out	<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:



15.362.2.2 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:



15.362.2.3 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 L4vbus_gpio_pull_modes
-------------	---

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:



15.362.2.4 `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:



15.362.2.5 `pin()`

```
unsigned L4vbus::Gpio_pin::pin ( ) const [inline]
```

Get pin number.

Returns

GPIO pin number

Definition at line 124 of file `vbus_gpio`.

15.362.2.6 `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:



15.362.2.7 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 L4vbus_gpio_generic_func 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:



15.362.2.8 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:

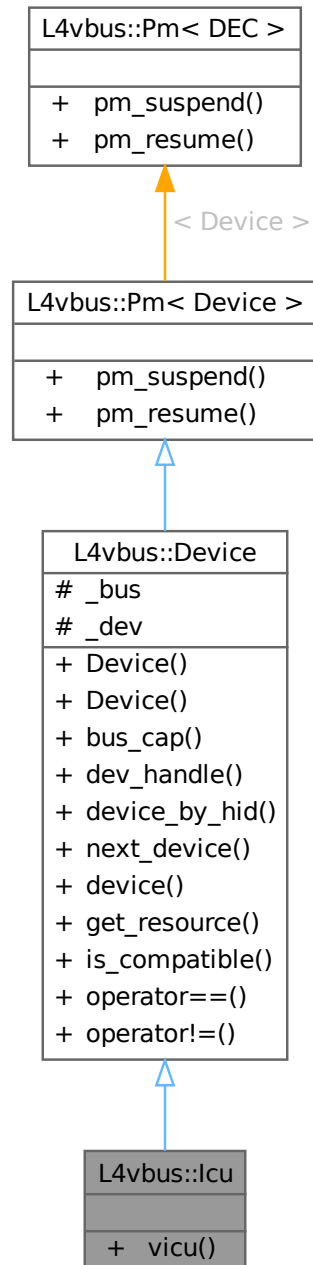
- [l4/vbus/vbus_gpio](#)

15.363 L4vbus::lcu Class Reference

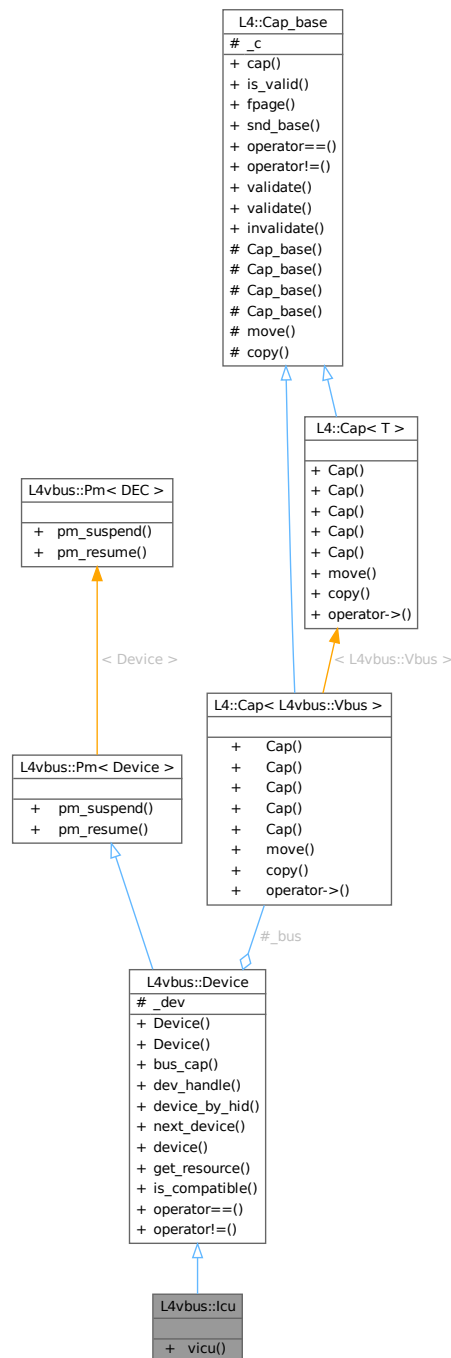
Vbus Interrupt controller API.

```
#include <vbus>
```

Inheritance diagram for L4vbus::lcu:



Collaboration diagram for L4vbus::Icu:



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::Icu > icu) const

Request an [L4::lcu](#) 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](#) (int 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.

15.363.1 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::lcu](#) object which can then be used to interface with the IRQs.

See also

[L4::lcu](#)

Definition at line 262 of file [vbus](#).

15.363.2 Member Enumeration Documentation

15.363.2.1 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 L4::lcu::msi_info() to denote that the ICU should interpret the source ID as a device handle.
----------------	---

Definition at line 266 of file [vbus](#).

15.363.3 Member Function Documentation

15.363.3.1 vicu()

```
int L4vbus::Icu::vicu (
    L4::Cap< L4::Icu > icu ) const [inline]
```

Request an [L4::lcu](#) capability for this [Vbus](#)'s virtual ICU.

Parameters

out	icu	Capability slot where the L4::lcu capability shall be stored.
-----	-----	---

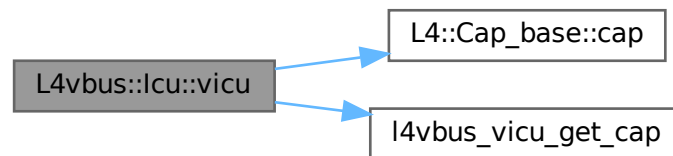
Return values

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:

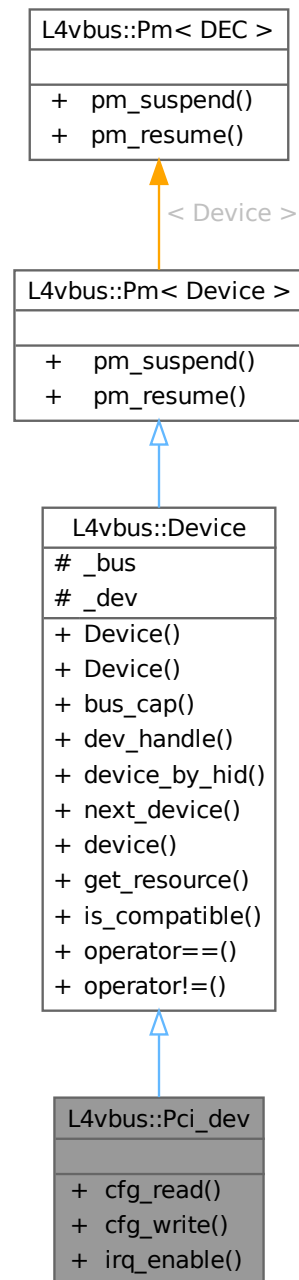
- `I4/vbus/vbus`

15.364 L4vbus::Pci_dev Class Reference

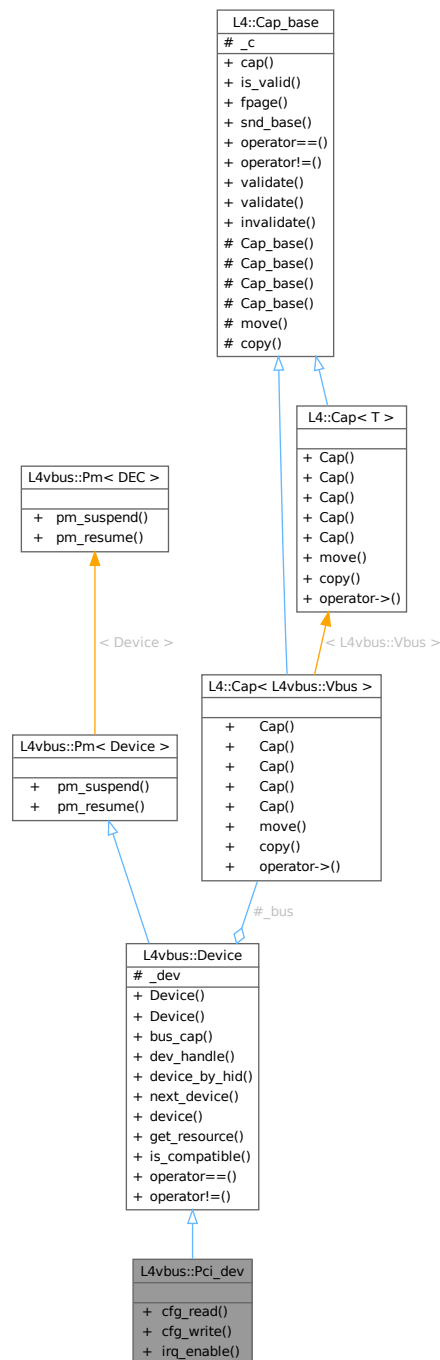
A PCI device.

```
#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** (int 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.

15.364.1 Detailed Description

A PCI device.

Definition at line 95 of file [vbus_pci](#).

15.364.2 Member Function Documentation

15.364.2.1 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)

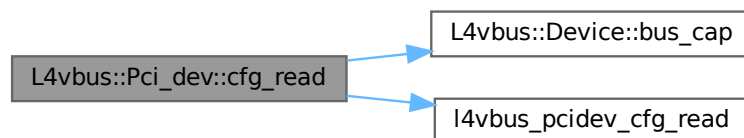
Returns

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:



15.364.2.2 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)

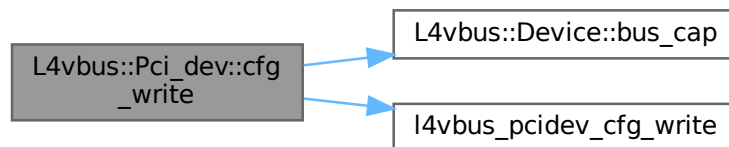
Returns

0 on success, else failure

Definition at line 123 of file [vbus_pci](#).

References [L4vbus::Device::_dev](#), [L4vbus::Device::bus_cap\(\)](#), and [l4vbus_pciddev_cfg_write\(\)](#).

Here is the call graph for this function:

**15.364.2.3 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

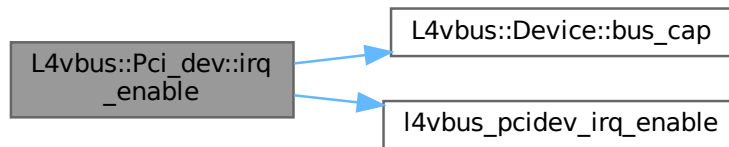
Returns

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_pciddev_irq_enable\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

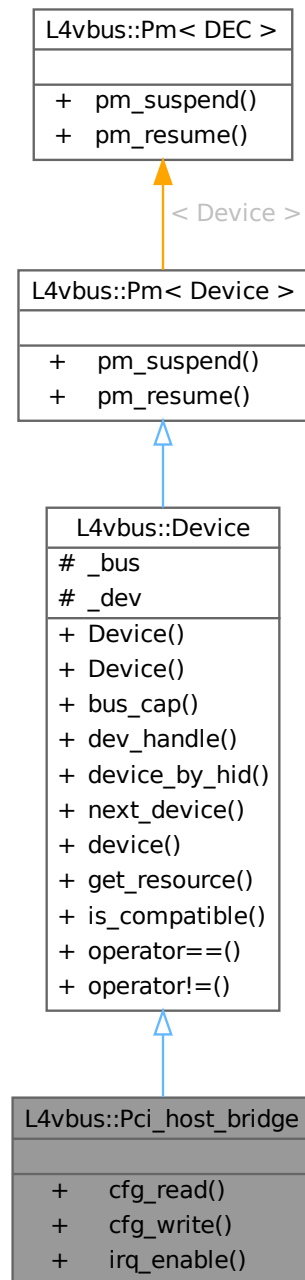
- l4/vbus/vbus_pci

15.365 L4vbus::Pci_host_bridge Class Reference

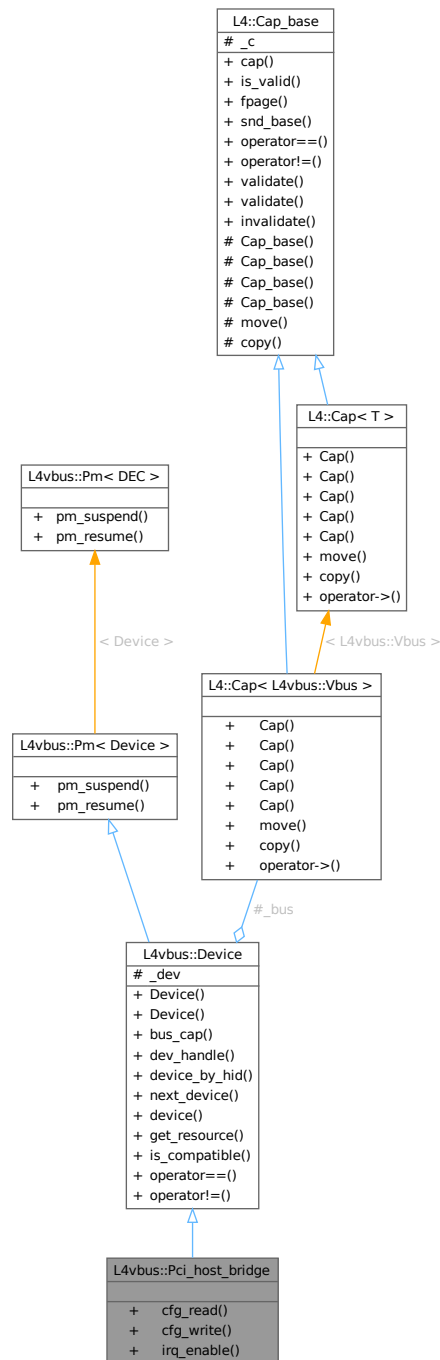
A Pci host bridge.

```
#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` (int 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.

15.365.1 Detailed Description

A Pci host bridge.

Definition at line 27 of file [vbus_pci](#).

15.365.2 Member Function Documentation

15.365.2.1 `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>	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)

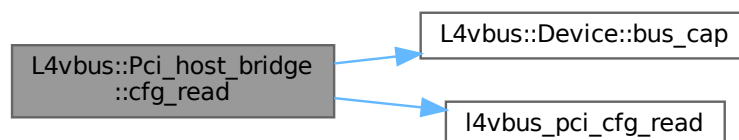
Returns

0 on success, else failure

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:



15.365.2.2 `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>	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)

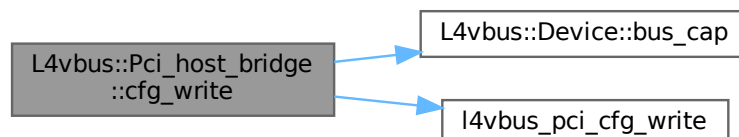
Returns

0 on success, else failure

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:



15.365.2.3 `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>	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

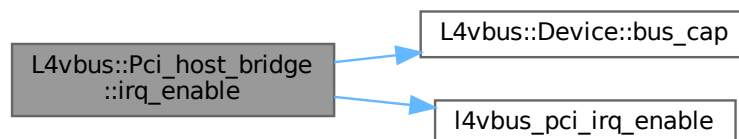
Returns

On success: Interrupt line to be used, else failure

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:

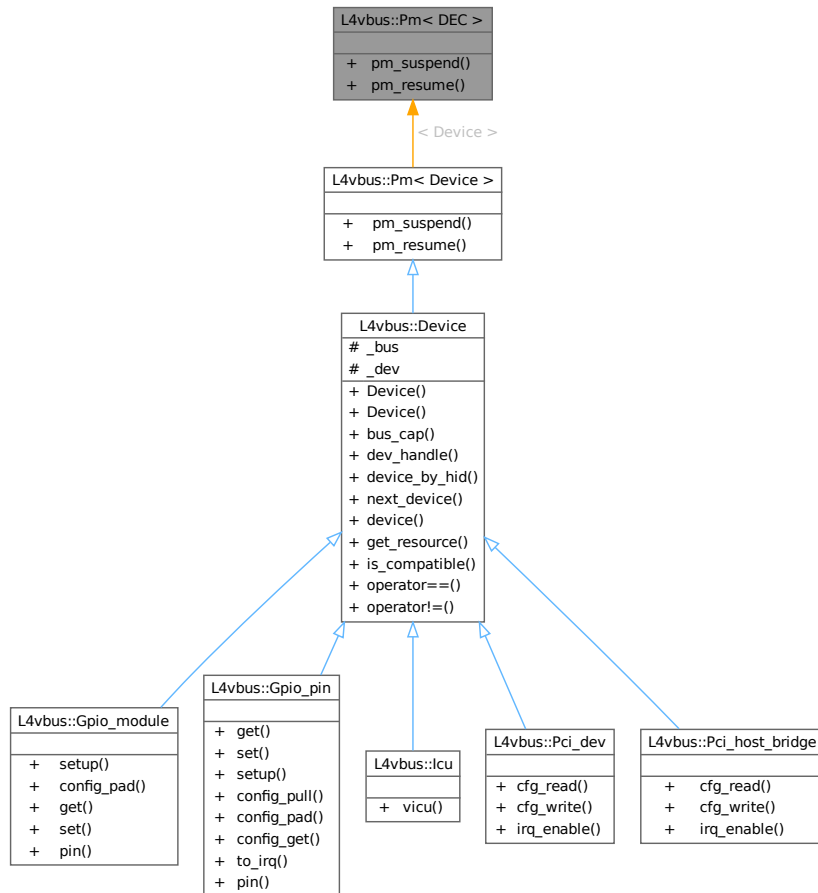
- `I4/vbus/vbus_pci`

15.366 L4vbus::Pm< DEC > Class Template Reference

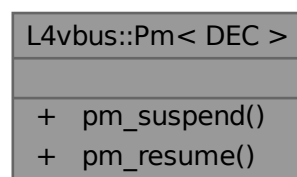
Power-management API mixin.

```
#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.

15.366.1 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](#).

15.366.2 Member Function Documentation

15.366.2.1 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:



15.366.2.2 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:



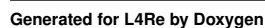
The documentation for this class was generated from the following file:

- `l4/vbus/vbus`

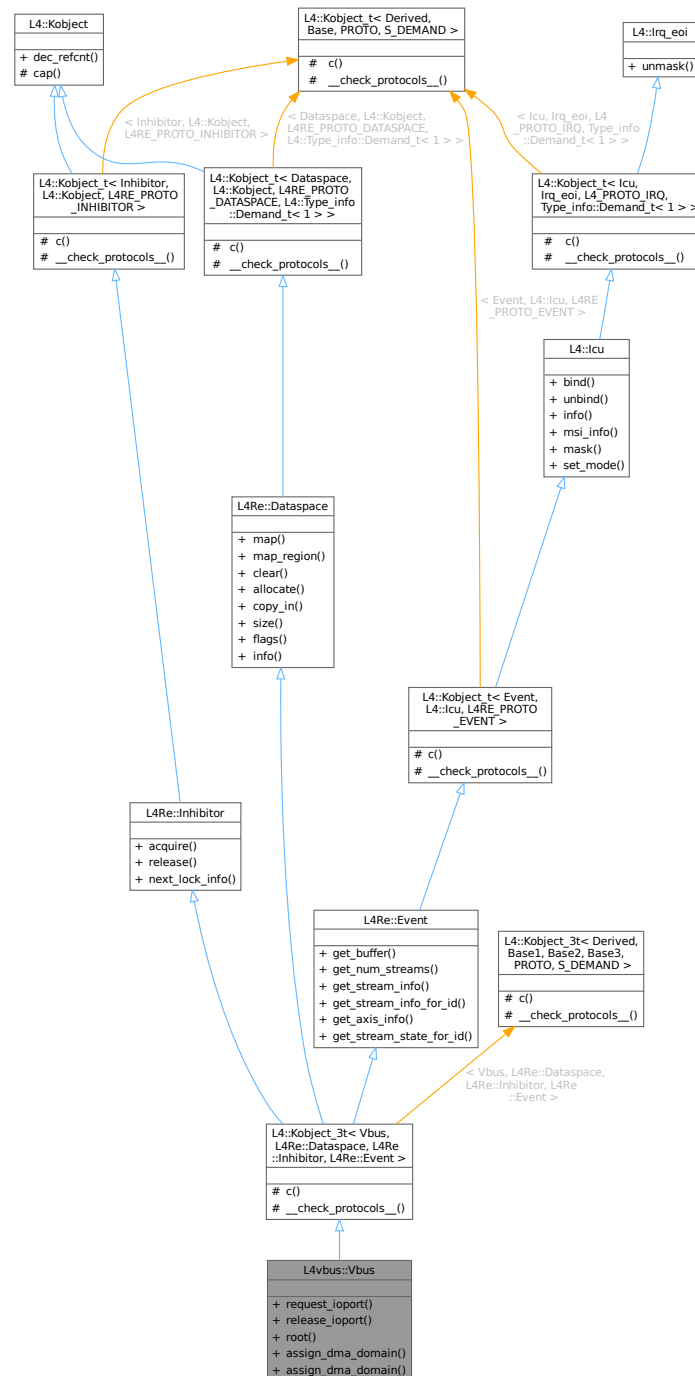
15.367 L4vbus::Vbus Class Reference

The virtual bus ([Vbus](#)) interface.

```
#include <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) 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) 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.

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::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_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.

15.367.1 Detailed Description

The virtual bus ([Vbus](#)) interface.

See also

[L4Re Vbus API](#)

Definition at line [300](#) of file [vbus](#).

15.367.2 Member Function Documentation**15.367.2.1 `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 L4vbus_dma_domain_assign_flags .
<i>dma_space</i>	The DMA space capability to bind or unbind, this must be a kernel DMA space (L4::Task created with L4_PROTO_DMA_SPACE)

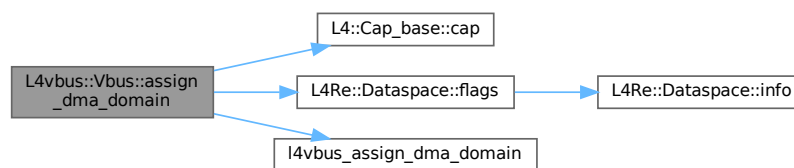
Return values

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:



15.367.2.2 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 L4vbus_dma_domain_assign_flags .
<i>dma_space</i>	The DMA space capability to bind or unbind, this must be an L4Re::Dma_space

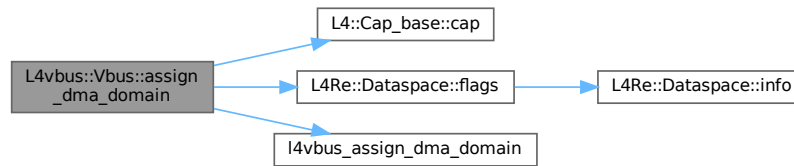
Return values

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:



15.367.2.3 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

≥ 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:



15.367.2.4 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:



15.367.2.5 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:

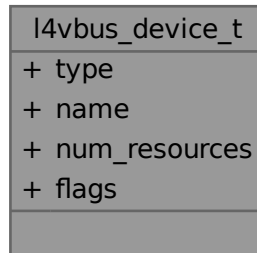
- `I4/vbus/vbus`

15.368 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](#).

15.368.1 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](#)

15.369 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.

15.369.1 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:

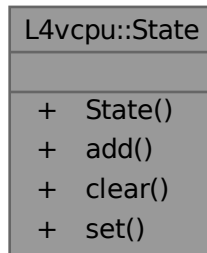
- [l4/vbus/vbus_types.h](#)

15.370 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.

15.370.1 Detailed Description

C++ implementation of state word in the vCPU area.

Definition at line [35](#) of file [vcpu](#).

15.370.2 Constructor & Destructor Documentation

15.370.2.1 State()

```
L4vcpu::State::State (  
    unsigned v ) [inline], [explicit]
```

Initialize state.

Parameters

<i>v</i>	Initial state.
----------	----------------

Definition at line 45 of file [vcpu](#).

15.370.3 Member Function Documentation

15.370.3.1 `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](#).

15.370.3.2 `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](#).

15.370.3.3 `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:

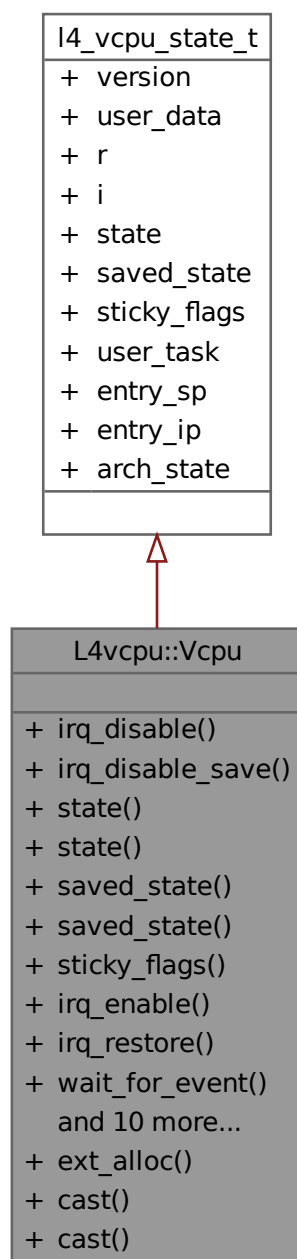
- [l4/vcpu/vcpu](#)

15.371 L4vcpu::Vcpu Class Reference

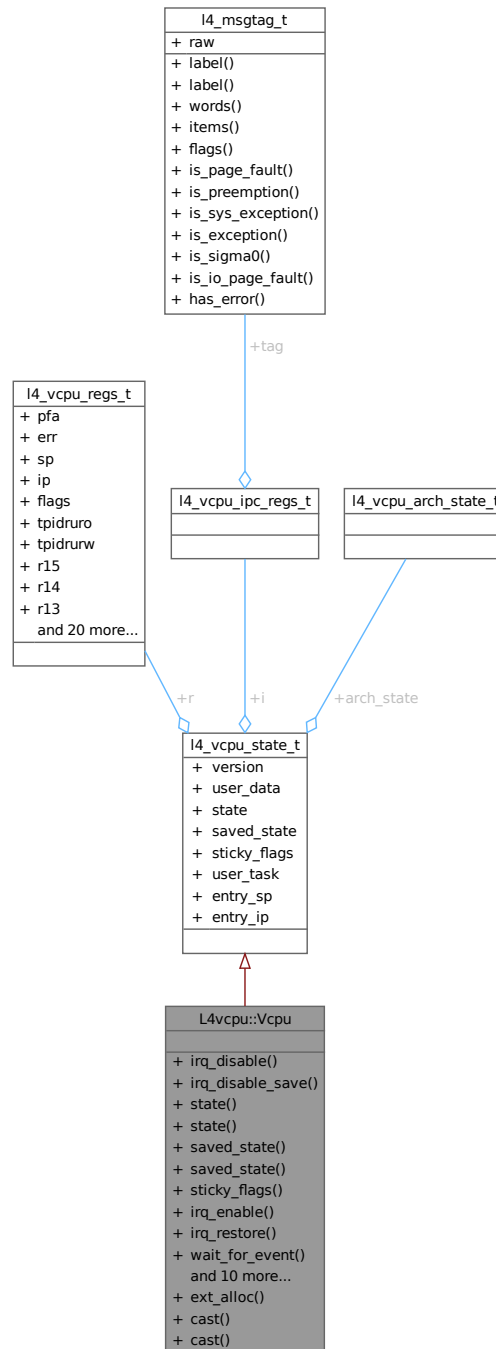
C++ implementation of the vCPU save state area.

```
#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.

15.371.1 Detailed Description

C++ implementation of the vCPU save state area.

Definition at line 76 of file [vcpu](#).

15.371.2 Member Function Documentation

15.371.2.1 `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](#).

15.371.2.2 `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](#).

15.371.2.3 `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](#).

15.371.2.4 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](#).

15.371.2.5 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

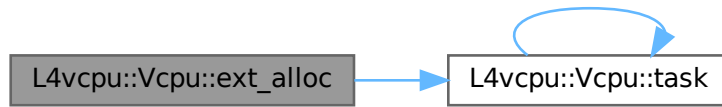
out	<i>vcpu</i>	Allocated vcpu-state area.
out	<i>ext_state</i>	Allocated extended vcpu-state area.
	<i>task</i>	Task to use for allocation, defaults to own task.
	<i>rm</i>	Region manager to use for allocation defaults to standard region manager.

Returns

0 for success, error code otherwise

References [task\(\)](#).

Here is the call graph for this function:



15.371.2.6 `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`.

15.371.2.7 `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`.

15.371.2.8 `irq_disable_save()`

```
unsigned L4vcpu::Vcpu::irq_disable_save ( ) throw ( ) [inline]
```

Disable the vCPU for event delivery and return previous state.

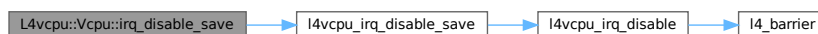
Returns

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:



15.371.2.9 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.

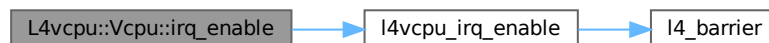
Parameters

<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:



15.371.2.10 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.

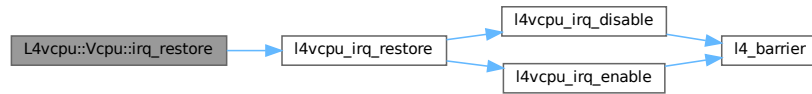
Parameters

<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:



15.371.2.11 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:



15.371.2.12 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:



15.371.2.13 `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`.

15.371.2.14 `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`.

15.371.2.15 `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`.

15.371.2.16 `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`.

15.371.2.17 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](#).

15.371.2.18 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](#).

15.371.2.19 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:



15.371.2.20 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

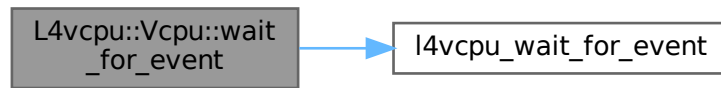
<i>utcb</i>	The UTCB to use.
<i>do_event_work_cb</i>	Call-back function that is called in case an event (such as an interrupt) is pending.
<i>setup_ipc</i>	Call-back function that is called before an IPC operation is called.

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:

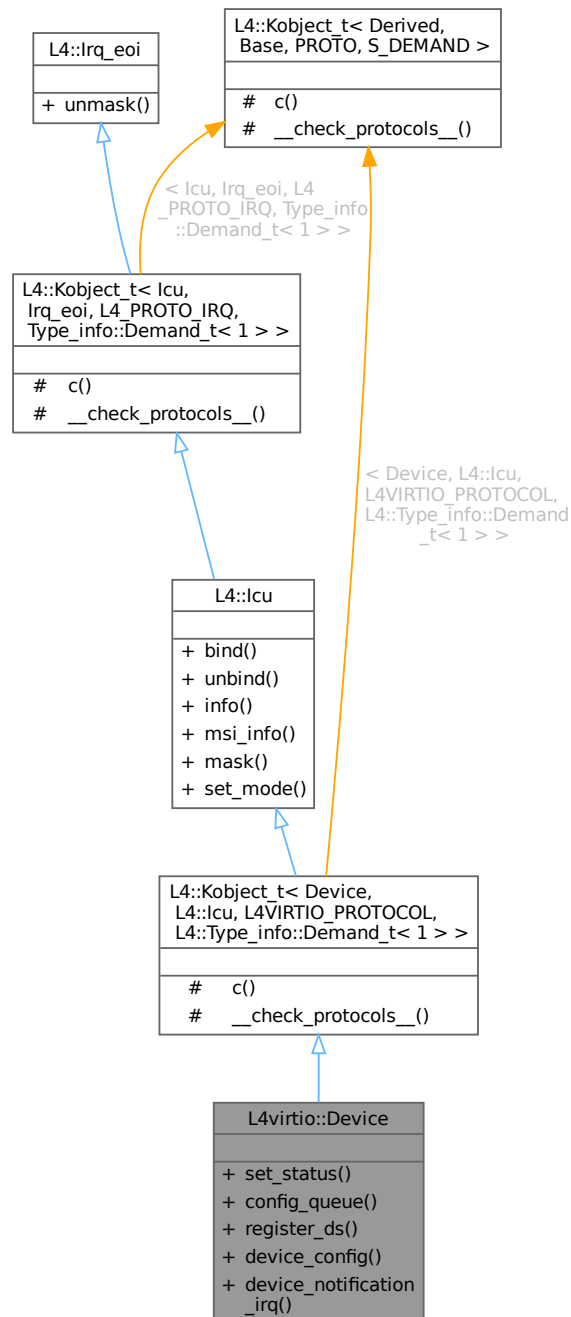
- [l4/vcpu/vcpu](#)

15.372 L4virtio::Device Class Reference

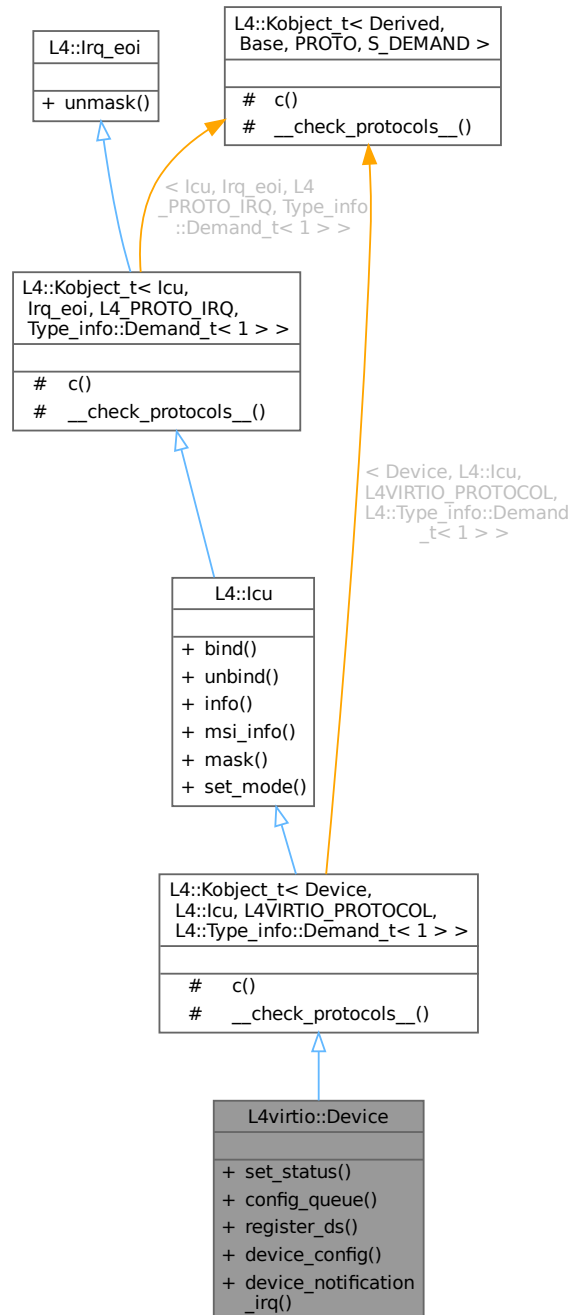
IPC interface for virtio over [L4](#) IPC.

```
#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, 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< 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, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >**

- static void **__check_protocols__()** noexcept
Helper to check for protocol conflicts.

15.372.1 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](#).

15.372.2 Member Function Documentation**15.372.2.1 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

0	on success.
-L4_EIO	The queue's status is invalid.
-L4_ERANGE	The queue index exceeds the number of queues.
-L4_EINVAL	Otherwise.

Referenced by [L4virtio::Driver::Device::config_queue\(\)](#).

Here is the caller graph for this function:



15.372.2.2 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:



15.372.2.3 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><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:



15.372.2.4 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><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:



15.372.2.5 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

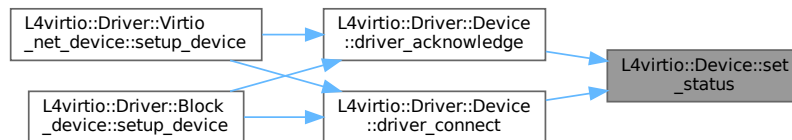
<i>0</i>	on success.
----------	-------------

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:

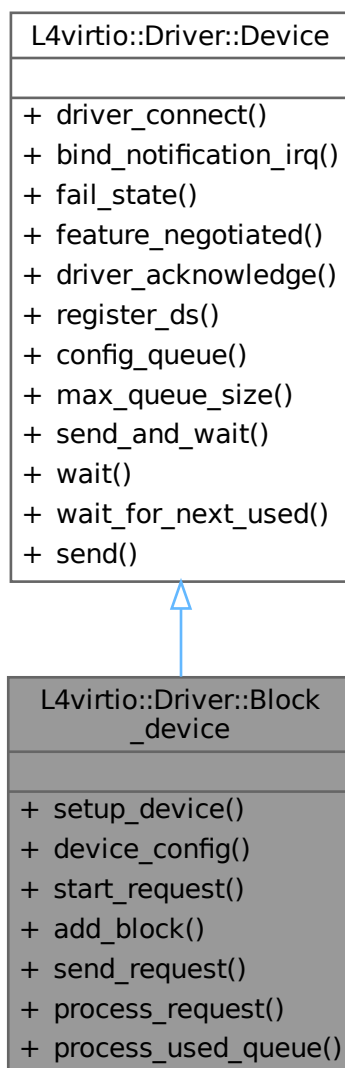
- `l4/l4virtio/l4virtio`

15.373 L4virtio::Driver::Block_device Class Reference

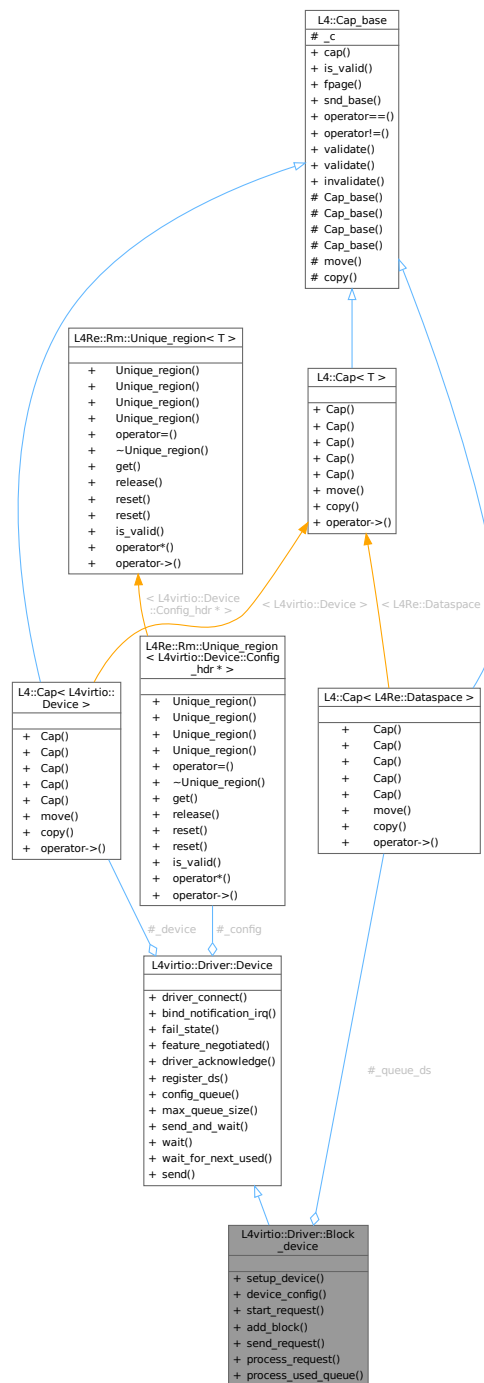
Simple class for accessing a virtio block device synchronously.

```
#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.

15.373.1 Detailed Description

Simple class for accessing a virtio block device synchronously.

Definition at line 36 of file [virtio-block](#).

15.373.2 Member Function Documentation

15.373.2.1 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>	Handle to request previously set up with start_request() .
<i>addr</i>	Address of data block in device address space.
<i>size</i>	Size of data block.

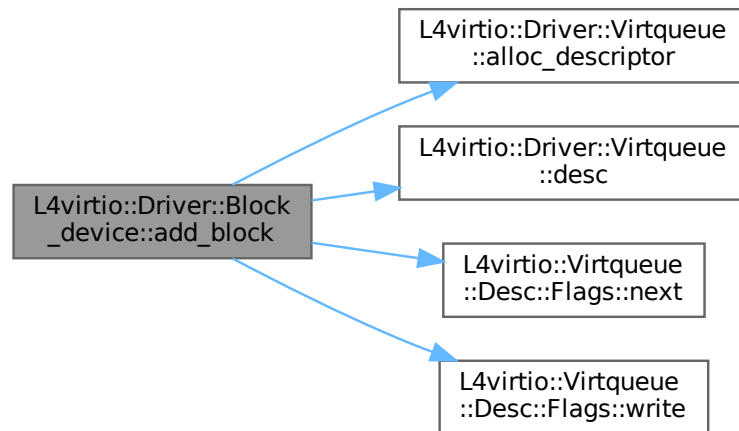
Return values

<i>L4_OK</i>	Block was successfully added.
<i>-L4_EAGAIN</i>	No descriptors available. Try again later.

Definition at line 228 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:



15.373.2.2 process_request()

```
int L4virtio::Driver::Block_device::process_request (
    Handle handle ) [inline]
```

Process request synchronously.

Parameters

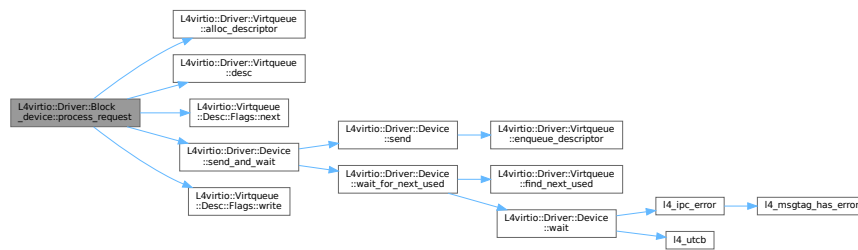
<i>handle</i>	Handle to request to process.
---------------	-------------------------------

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 299 of file `virtio-block`.

References `L4virtio::Virtqueue::Desc::addr`, `L4virtio::Driver::Virtqueue::alloc_descriptor()`, `L4virtio::Driver::Virtqueue::desc()`, `L4virtio::Virtqueue::Desc::flags`, `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:



15.373.2.3 process_used_queue()

```
void L4virtio::Driver::Block_device::process_used_queue ( ) [inline]
```

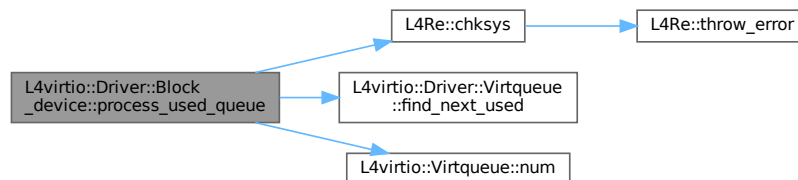
Process and free all items in the used queue.

If the request has a callback registered it is called after the item has been removed from the queue.

Definition at line 350 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:



15.373.2.4 send_request()

```
int L4virtio::Driver::Block_device::send_request (
    Handle handle ) [inline]
```

Process request asynchronously.

Parameters

<i>handle</i>	Handle to request to send to the device
---------------	---

Return values

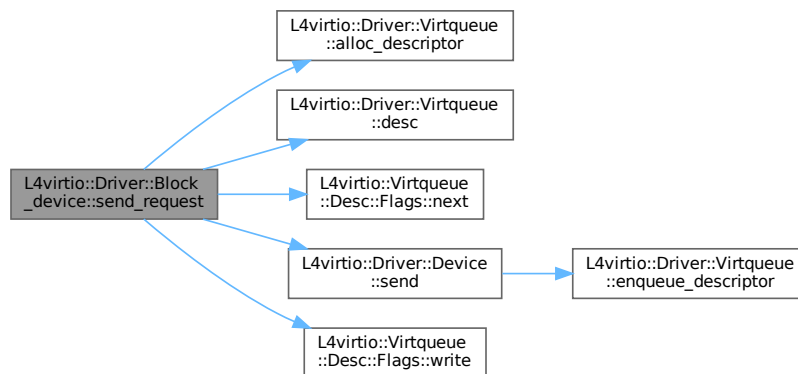
<code>L4_OK</code>	Request was successfully scheduled.
<code>-L4_EAGAIN</code>	No descriptors available. Try again later.

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 264 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:



15.373.2.5 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.
<code>userdata</code>	Pointer to the region of user-usable memory.
<code>user_devaddr</code>	Address of user-usable memory in device address space.
<code>qds</code>	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.
<code>fmask0</code>	Feature bits 0..31 that the driver supports.
<code>fmask1</code>	Feature bits 32..63 that the driver supports.

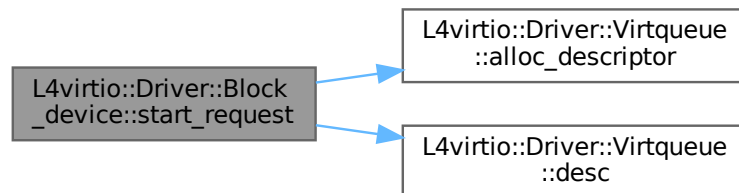
Parameters

<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 190 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:

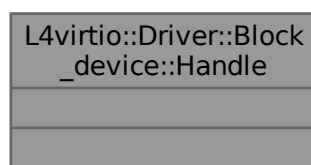
- `l4/l4virtio/client/virtio-block`

15.374 L4virtio::Driver::Block_device::Handle Class Reference

[Handle](#) to an ongoing request.

```
#include <virtio-block>
```

Collaboration diagram for L4virtio::Driver::Block_device::Handle:



15.374.1 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:

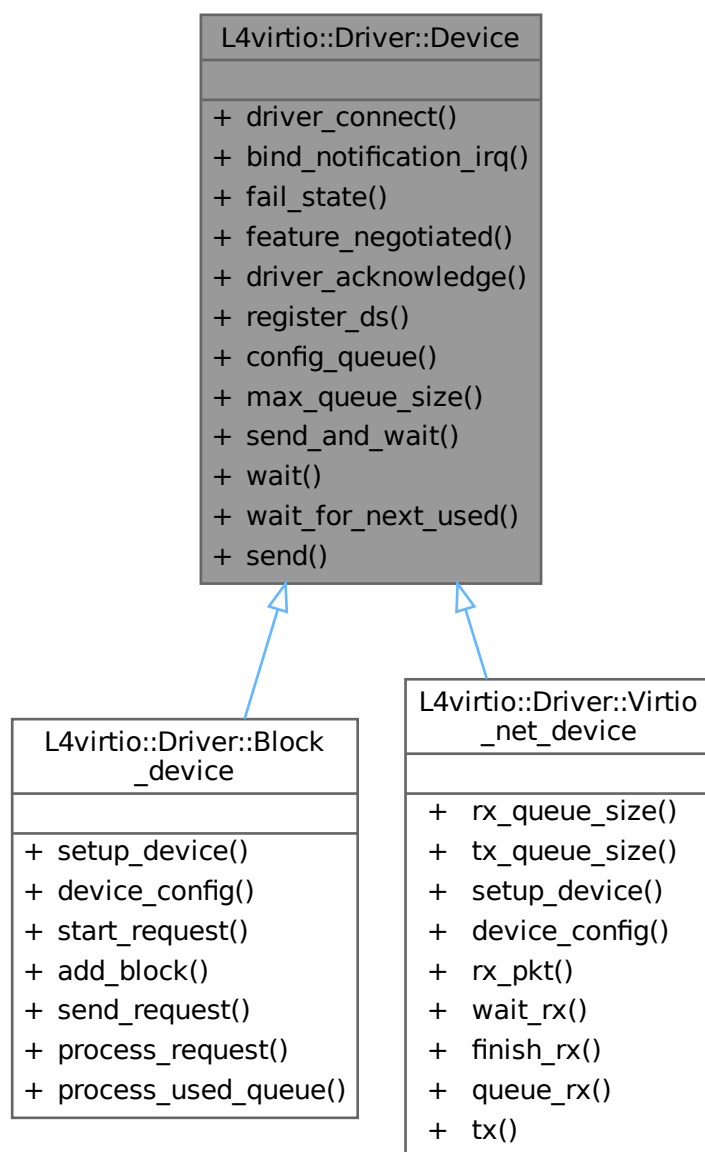
- l4/l4virtio/client/virtio-block

15.375 L4virtio::Driver::Device Class Reference

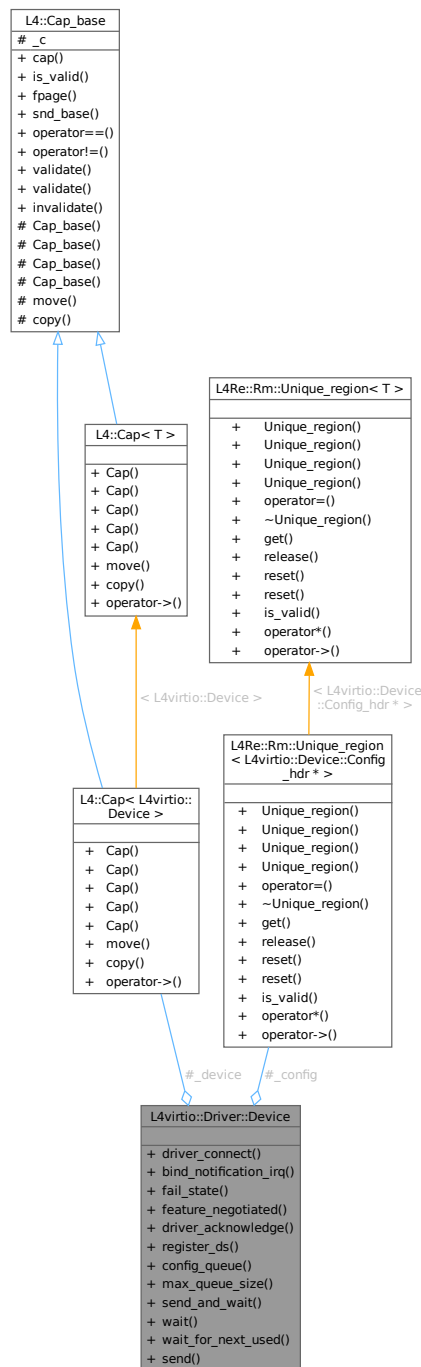
Client-side implementation for a general virtio device.

```
#include <l4virtio>
```


Inheritance diagram for L4virtio::Driver::Device:



Collaboration diagram for L4virtio::Driver::Device:



Public Member Functions

- void `driver_connect` (`L4::Cap< L4virtio::Device >` `srcap`, `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.

15.375.1 Detailed Description

Client-side implementation for a general virtio device.

Definition at line 31 of file [l4virtio](#).

15.375.2 Member Function Documentation

15.375.2.1 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.

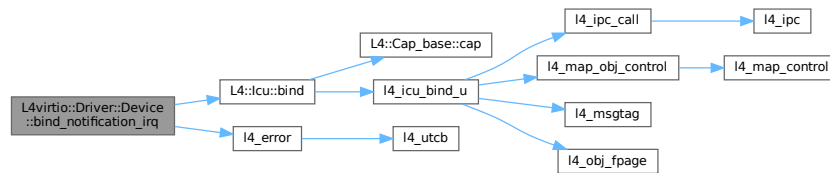
Parameters

	<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:



15.375.2.2 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](#).

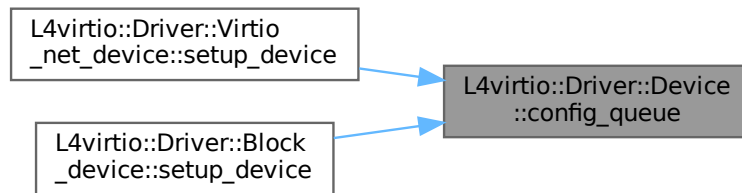
References [L4virtio::Device::config_queue\(\)](#).

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:



15.375.2.3 driver_acknowledge()

```
int L4virtio::Driver::Device::driver_acknowledge ( ) [inline]
```

Finalize handshake with the device.

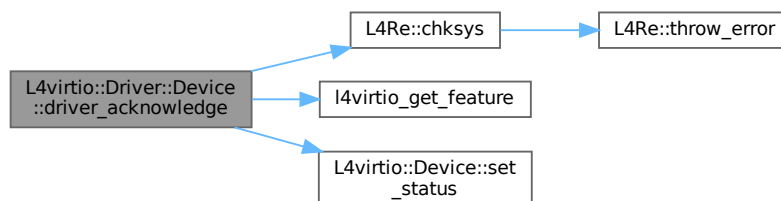
Must be called after all queues have been set up and before the first request is sent. It is still possible to add more shared dataspace after the handshake has been finished.

Definition at line 156 of file [l4virtio](#).

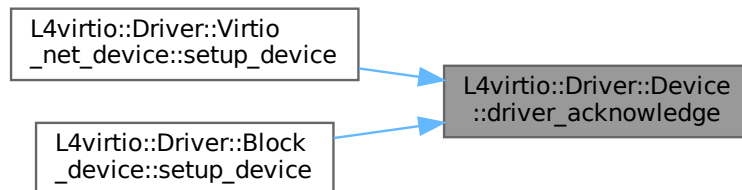
References [L4Re::chksys\(\)](#), [L4_EINVAL](#), [L4_EIO](#), [L4_ENODEV](#), [L4_EOK](#), [L4VIRTIO_FEATURE_VERSION_1](#), [l4virtio_get_feature\(\)](#), [L4VIRTIO_STATUS_DRIVER_OK](#), [L4VIRTIO_STATUS_FEATURES_OK](#), and [L4virtio::Device::set_status\(\)](#).

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:



15.375.2.4 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

L4::Runtime_error	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\(\)](#).

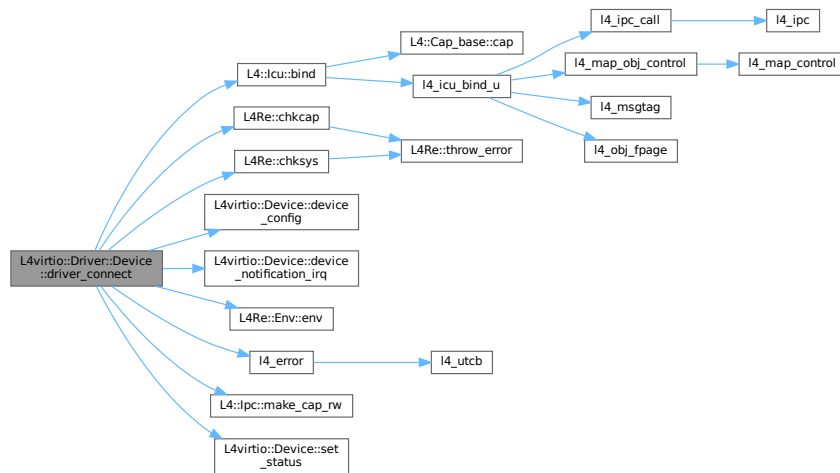
When `manage_notify` is false, then the caller may manually register and handle notification interrupts from the device. This is for example useful, when the client runs in an application with a server loop.

Definition at line 56 of file [l4virtio](#).

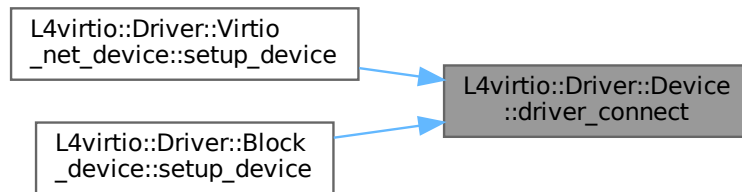
References [L4::lcu::bind\(\)](#), [L4Re::chkcapp\(\)](#), [L4Re::chksys\(\)](#), [L4virtio::Device::device_config\(\)](#), [L4virtio::Device::device_notification_irq\(\)](#), [L4Re::Env::env\(\)](#), [L4_EINVAL](#), [L4_EIO](#), [L4_ENODEV](#), [l4_error\(\)](#), [L4_PAGEMASK](#), [L4_PAGESHIFT](#), [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:



15.375.2.5 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.
-------------	---

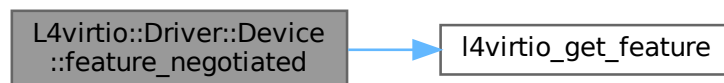
Return values

<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:



15.375.2.6 max_queue_size()

```
int L4virtio::Driver::Device::max_queue_size (
    int num ) const [inline]
```

Maximum queue size allowed by the device.

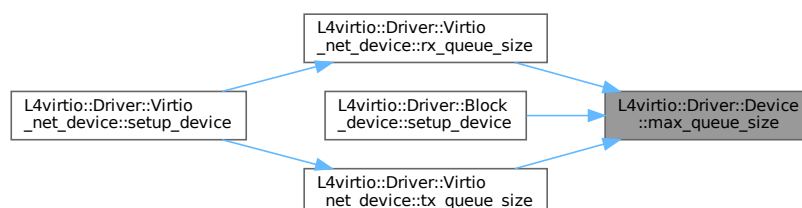
Parameters

<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:



15.375.2.7 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.

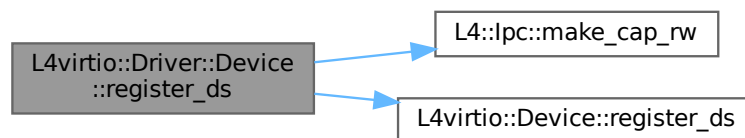
When communicating with the device, addresses must be given with respect to the device address space. This is not the same as the virtual address space of the client in order to not leak information about the address space layout.

Definition at line 196 of file [l4virtio](#).

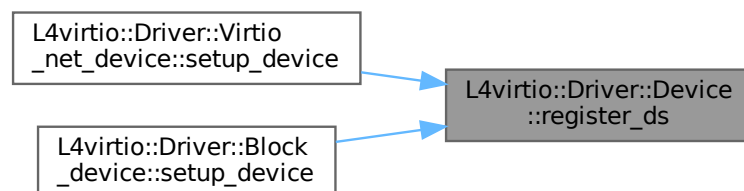
References [L4::lpc::make_cap_rw\(\)](#), and [L4virtio::Device::register_ds\(\)](#).

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:



15.375.2.8 send()

```
void L4virtio::Driver::Device::send (
    Virtqueue & queue,
    l4_uint16_t descno ) [inline]
```

Send a request to the device.

Parameters

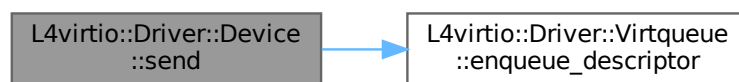
<i>queue</i>	Queue that contains the request in its descriptor table
<i>descno</i>	Index of first entry in descriptor table where

Definition at line 312 of file [l4virtio](#).

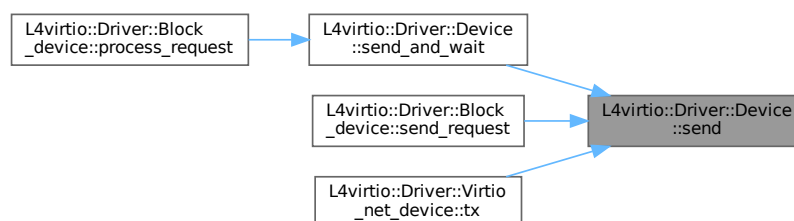
References [L4virtio::Driver::Virtqueue::enqueue_descriptor\(\)](#).

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:



15.375.2.9 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

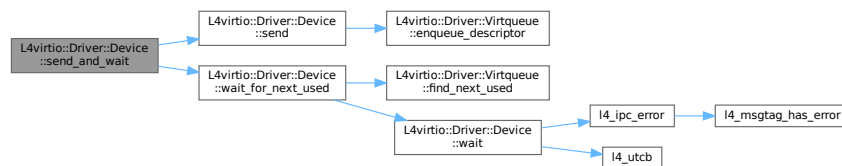
[driver_connect\(\)](#) was called with `manage_notify`.

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:



15.375.2.10 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

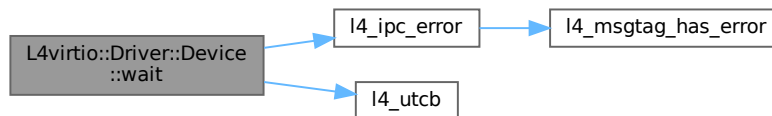
[driver_connect\(\)](#) was called with `manage_notify`.

Definition at line 268 of file [l4virtio](#).

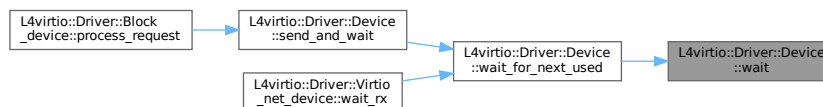
References [L4_EEXIST](#), [l4_ipc_error\(\)](#), and [l4_utcb\(\)](#).

Referenced by [wait_for_next_used\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.375.2.11 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

≥ 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

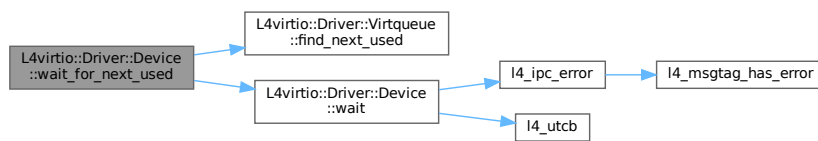
[driver_connect\(\)](#) was called with `manage_notify`.

Definition at line 291 of file [l4virtio](#).

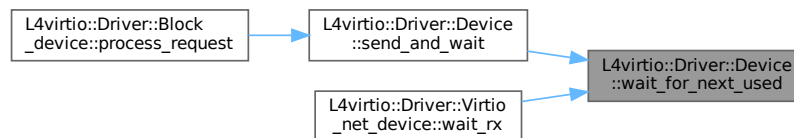
References [L4virtio::Driver::Virtqueue::find_next_used\(\)](#), and [wait\(\)](#).

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:

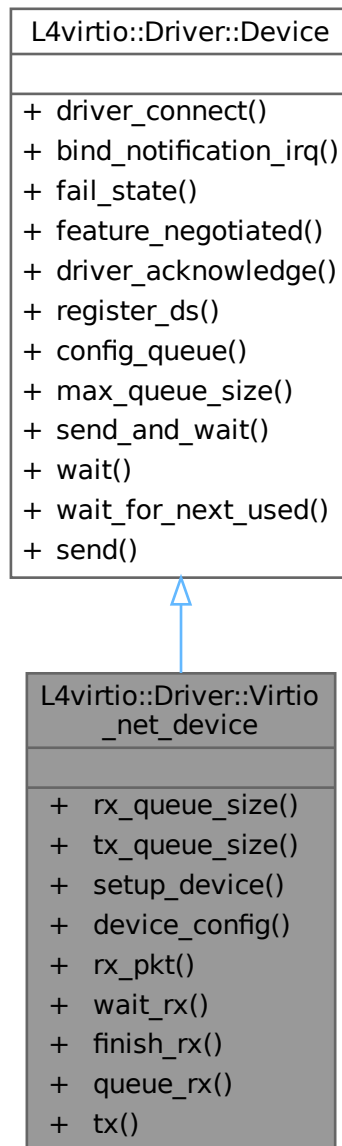
- `I4/I4virtio/client/I4virtio`

15.376 L4virtio::Driver::Virtio_net_device Class Reference

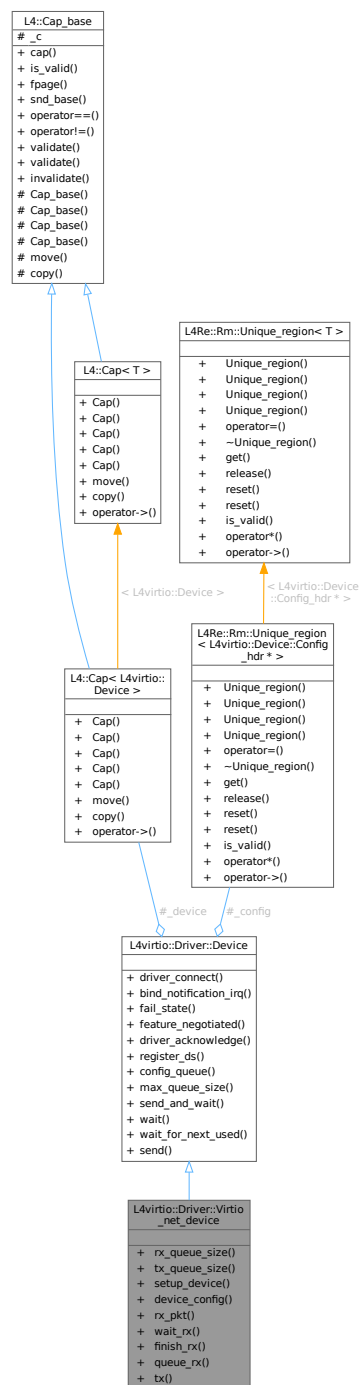
Simple class for accessing a virtio net device.

```
#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.

15.376.1 Detailed Description

Simple class for accessing a virtio net device.

Definition at line 30 of file [virtio-net](#).

15.376.2 Member Function Documentation

15.376.2.1 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

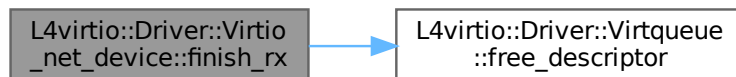
<i>descno</i>	Descriptor number in the virtio queue.
---------------	--

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:



15.376.2.2 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\(\)](#).

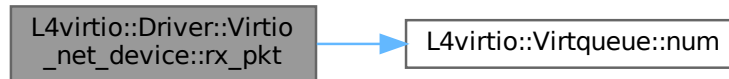
Parameters

<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:



15.376.2.3 rx_queue_size()

```
int L4virtio::Driver::Virtio_net_device::rx_queue_size ( ) const [inline]
```

Return the maximum receive queue size allowed by the device.

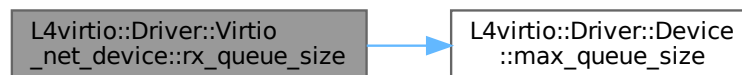
[wait_rx\(\)](#) will return a descriptor number that is smaller than this size.

Definition at line 47 of file [virtio-net](#).

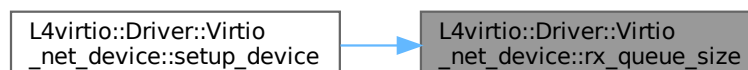
References [L4virtio::Driver::Device::max_queue_size\(\)](#).

Referenced by [setup_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.376.2.4 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

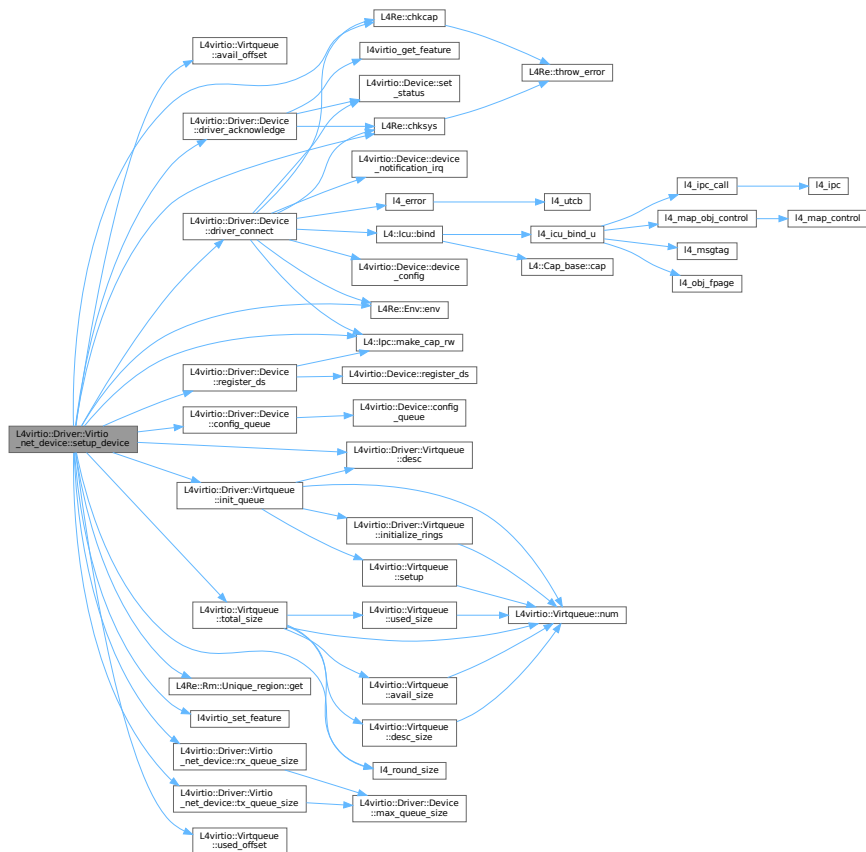
<code>srvcap</code>	IPC capability of the channel to the server.
---------------------	--

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](#), [I4virtio_set_feature\(\)](#), [L4::Ipc::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:



15.376.2.5 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

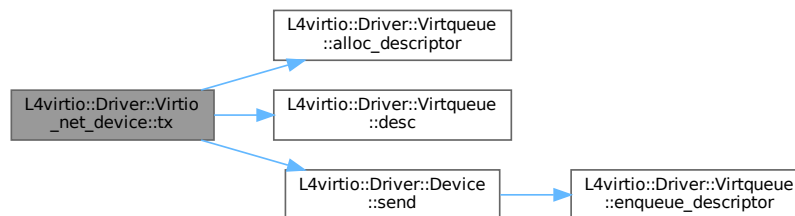
<i>true</i>	The packet was queued.
<i>false</i>	TX queue is full.

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:



15.376.2.6 tx_queue_size()

```
int L4virtio::Driver::Virtio_net_device::tx_queue_size ( ) const [inline]
```

Return the maximum transmit queue size allowed by the device.

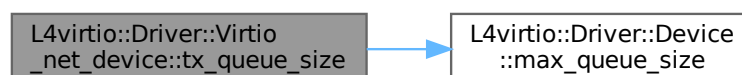
[tx\(\)](#) will fail if the amount of queued packets exceeds this size.

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:



15.376.2.7 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

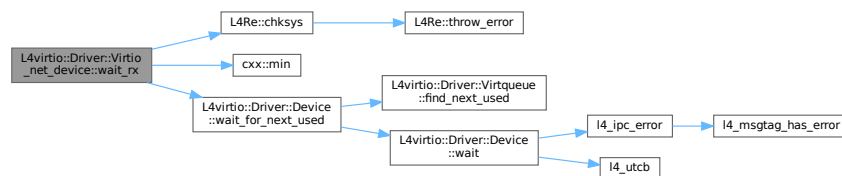
Descriptor number of received packet.

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\(\)](#), [cxx::min\(\)](#), 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:

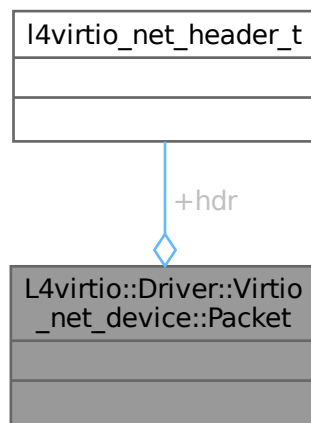
- [I4/I4virtio/client/virtio-net](#)

15.377 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:



15.377.1 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:

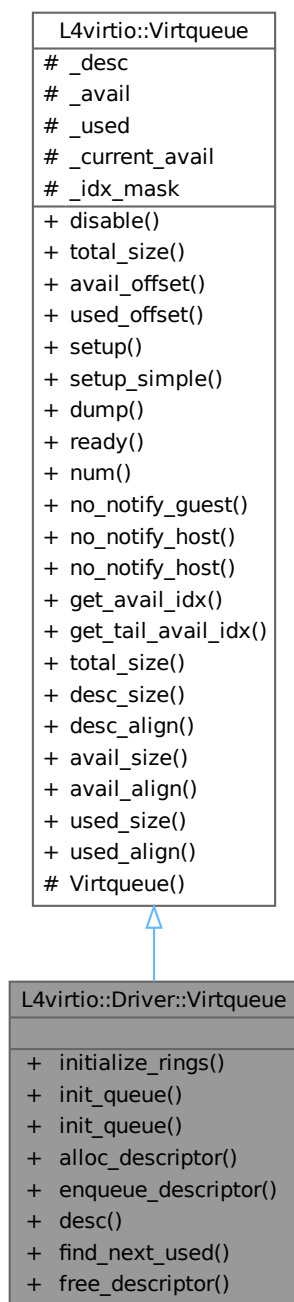
- `I4/I4virtio/client/virtio-net`

15.378 L4virtio::Driver::Virtqueue Class Reference

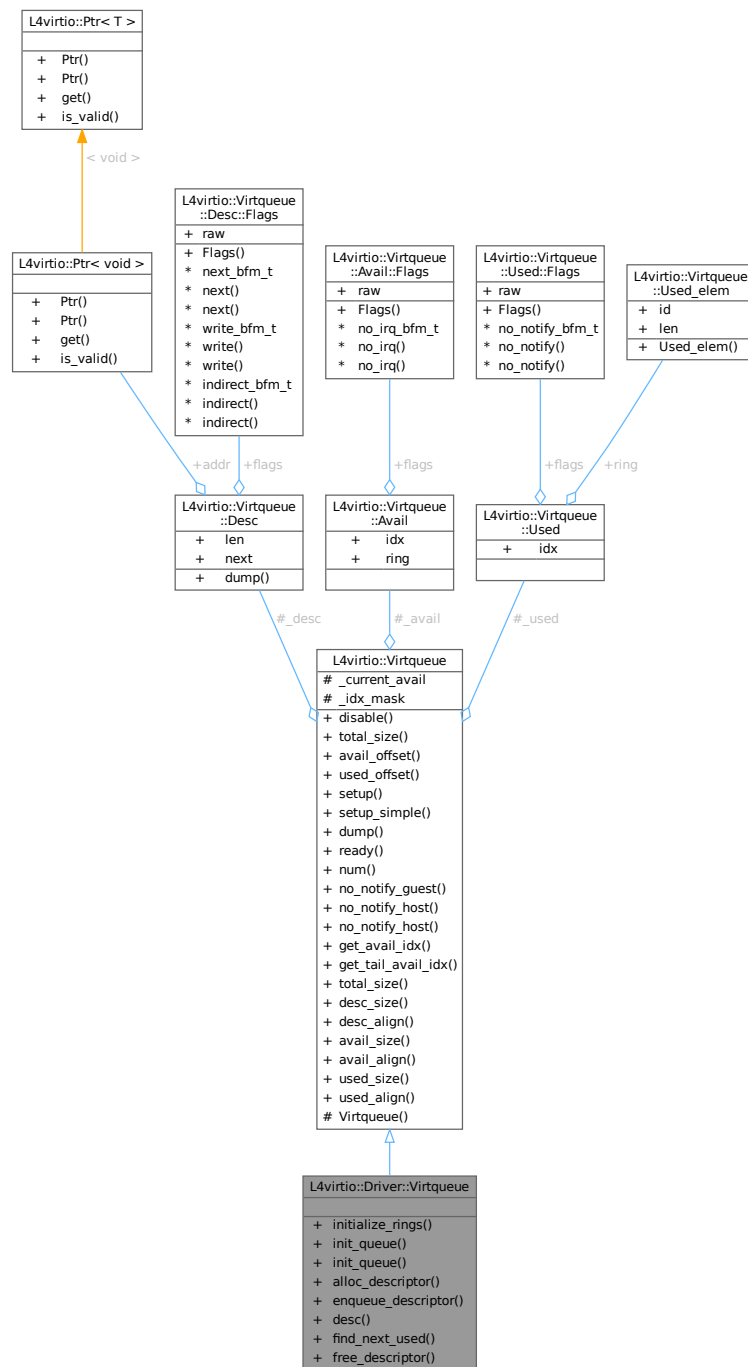
Driver-side implementation of a [Virtqueue](#).

```
#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](#) ()
Create a disabled virtqueue.

Protected Attributes inherited from [L4virtio::Virtqueue](#)

- [Desc](#) * [_desc](#)
pointer to descriptor table, NULL if queue is off.
- [Avail](#) * [_avail](#)
pointer to available ring.
- [Used](#) * [_used](#)
pointer to used ring.
- [l4_uint16_t](#) [_current_avail](#)
The life counter for the queue.
- [l4_uint16_t](#) [_idx_mask](#)
mask used for indexing into the descriptor table and the rings.

15.378.1 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 [482](#) of file [virtqueue](#).

15.378.2 Member Function Documentation

15.378.2.1 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.

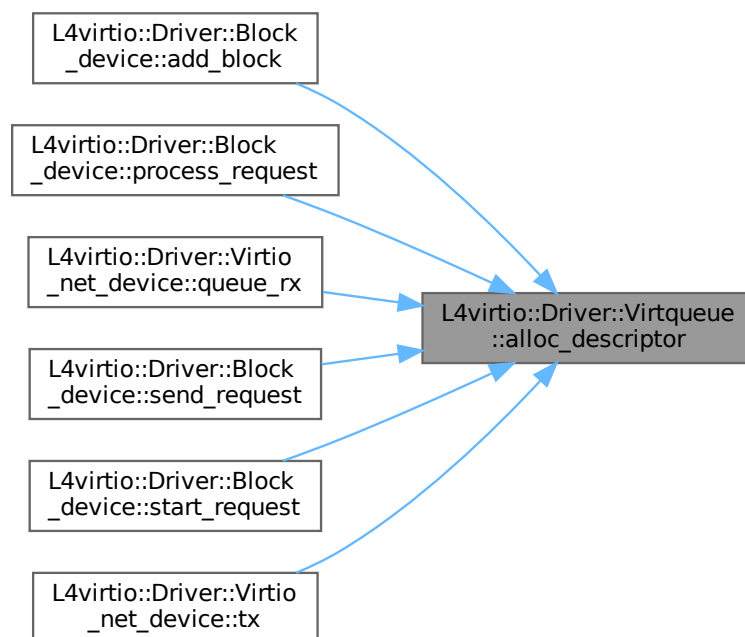
Note: the implementation uses $(2^{16} - 1)$ as the end of queue marker. That means that the final entry in the queue can not be allocated iff the queue size is 2^{16} .

Definition at line 570 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:



15.378.2.2 desc()

```
Desc & L4virtio::Driver::Virtqueue::desc (
    l4_uint16_t descno ) [inline]
```

Return a reference to a descriptor in the descriptor table.

Parameters

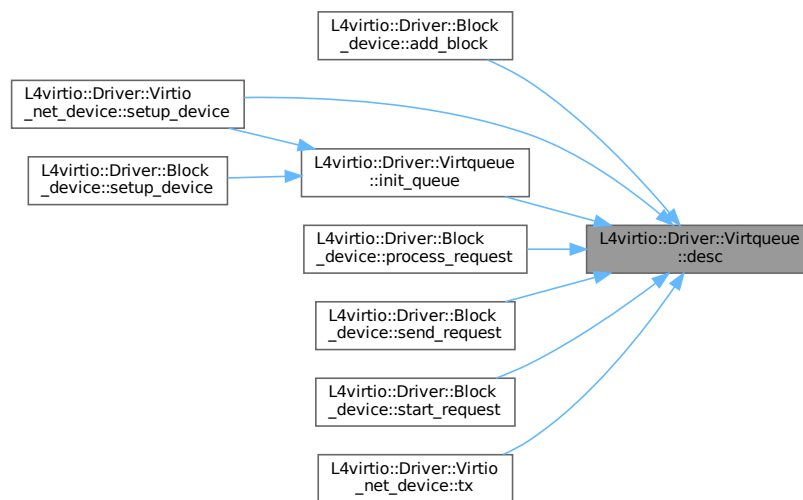
<i>descno</i>	Index of the descriptor, expected to be in correct range.
---------------	---

Definition at line 602 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:



15.378.2.3 enqueue_descriptor()

```
void L4virtio::Driver::Virtqueue::enqueue_descriptor (
    uint16_t descno ) [inline]
```

Enqueue a descriptor in the available ring.

Parameters

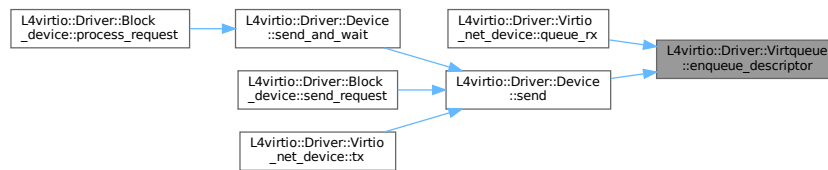
<i>descno</i>	Index of the head descriptor to enqueue.
---------------	--

Definition at line 586 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:



15.378.2.4 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

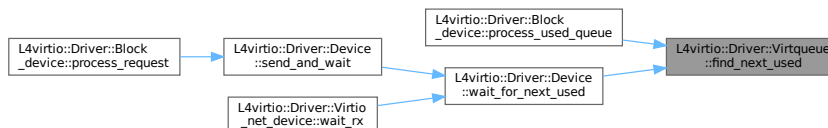
Index of the head or Virtqueue::Eoq if no used element is currently available.

Definition at line 621 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:



15.378.2.5 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 643 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:



15.378.2.6 init_queue() [1/2]

```
void L4virtio::Driver::Virtqueue::init_queue (
    unsigned num,
    void * base ) [inline]
```

Initialize this virtqueue.

Parameters

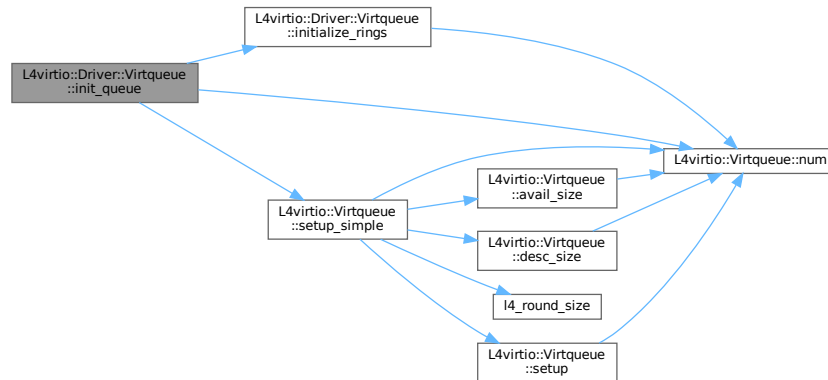
<i>num</i>	The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).
<i>base</i>	The base address for the queue data structure.

This function sets up the memory and initializes the freelist.

Definition at line 549 of file [virtqueue](#).

References [initialize_rings\(\)](#), [L4virtio::Virtqueue::num\(\)](#), and [L4virtio::Virtqueue::setup_simple\(\)](#).

Here is the call graph for this function:



15.378.2.7 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.)

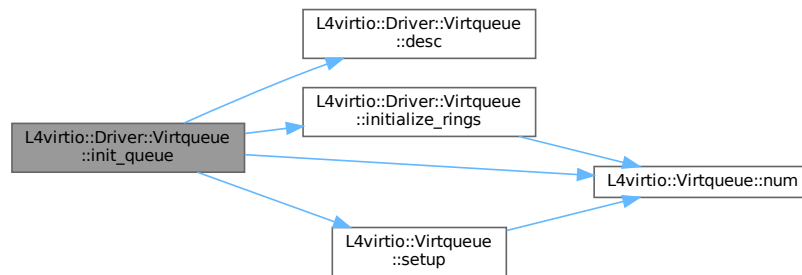
This function sets up the memory and initializes the freelist.

Definition at line 534 of file [virtqueue](#).

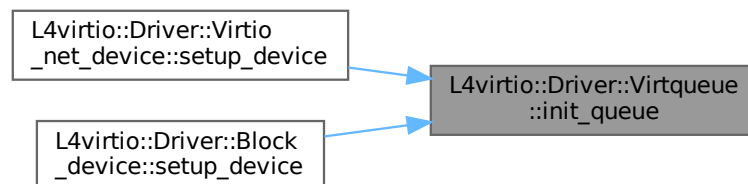
References [desc\(\)](#), [initialize_rings\(\)](#), [L4virtio::Virtqueue::num\(\)](#), and [L4virtio::Virtqueue::setup\(\)](#).

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:



15.378.2.8 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 506 of file [virtqueue](#).

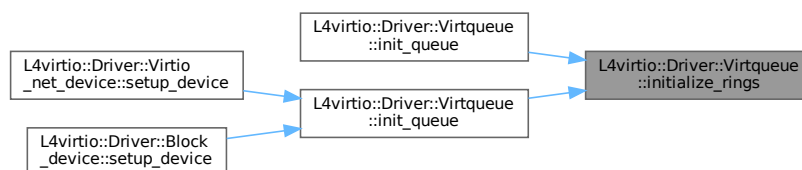
References [L4virtio::Virtqueue::_avail](#), [L4virtio::Virtqueue::_desc](#), [L4virtio::Virtqueue::_used](#), [L4virtio::Virtqueue::Avail::idx](#), [L4virtio::Virtqueue::Used::idx](#), [L4virtio::Virtqueue::Desc::next](#), and [L4virtio::Virtqueue::num\(\)](#).

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:

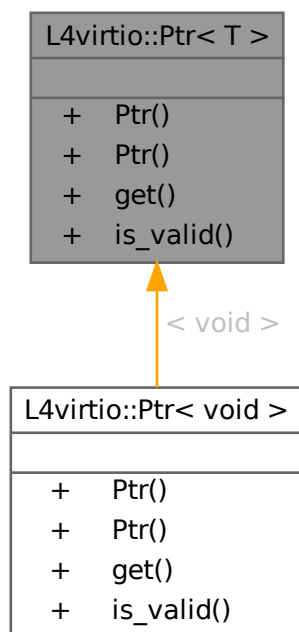
- `I4/I4virtio/virtqueue`

15.379 L4virtio::Ptr< T > Class Template Reference

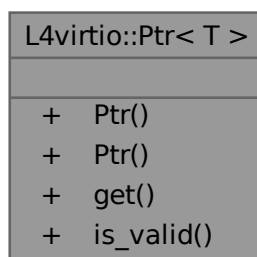
Pointer used in virtio descriptors.

```
#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

15.379.1 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 63 of file [virtqueue](#).

15.379.2 Member Enumeration Documentation

15.379.2.1 Invalid_type

```
template<typename T >
enum L4virtio::Ptr::Invalid_type
```

Type for making an invalid (NULL) [Ptr](#).

Enumerator

Invalid	Use to set a Ptr to invalid (NULL)
---------	--

Definition at line 67 of file [virtqueue](#).

15.379.3 Member Function Documentation

15.379.3.1 get()

```
template<typename T >
l4_uint64_t L4virtio::Ptr< T >::get ( ) const [inline]
```

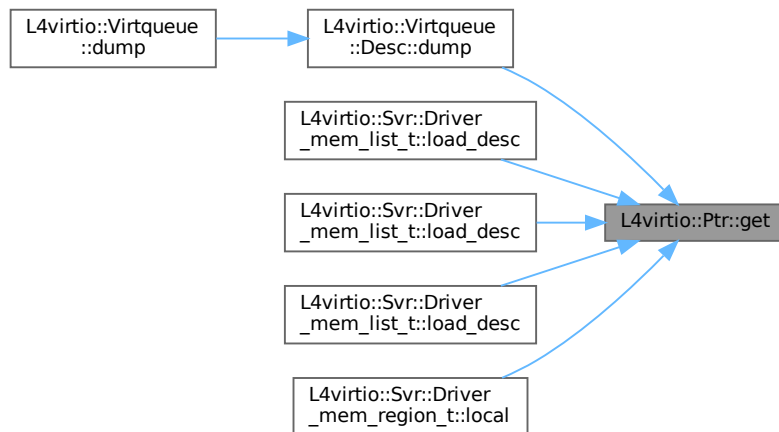
Returns

The raw 64bit address of the pointer.

Definition at line 78 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:

**15.379.3.2 is_valid()**

```

template<typename T >
bool L4virtio::Ptr< T >::is_valid ( ) const [inline]

```

Returns

true if the pointer is invalid (NULL).

Definition at line 81 of file [virtqueue](#).

The documentation for this class was generated from the following file:

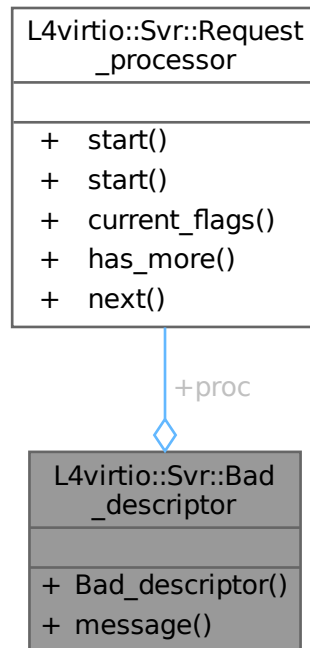
- `l4/l4virtio/virtqueue`

15.380 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.

15.380.1 Detailed Description

Exception used by Queue to indicate descriptor errors.

Definition at line 325 of file [virtio](#).

15.380.2 Member Enumeration Documentation

15.380.2.1 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 328 of file [virtio](#).

15.380.3 Constructor & Destructor Documentation

15.380.3.1 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 348 of file [virtio](#).

15.380.4 Member Function Documentation

15.380.4.1 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 357 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:

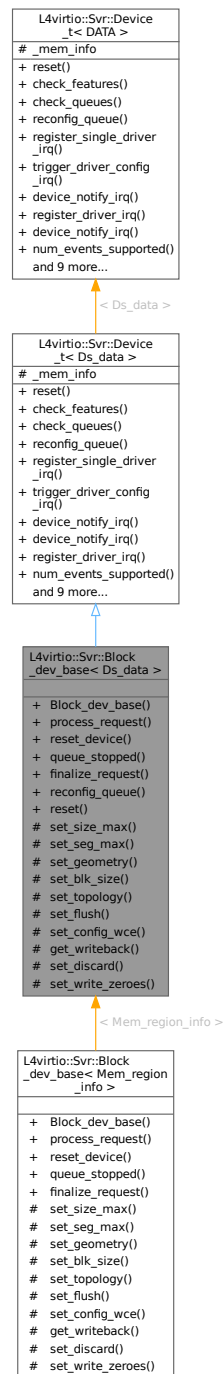
- l4/l4virtio/server/virtio

15.381 L4virtio::Svr::Block_dev_base< Ds_data > Class Template Reference

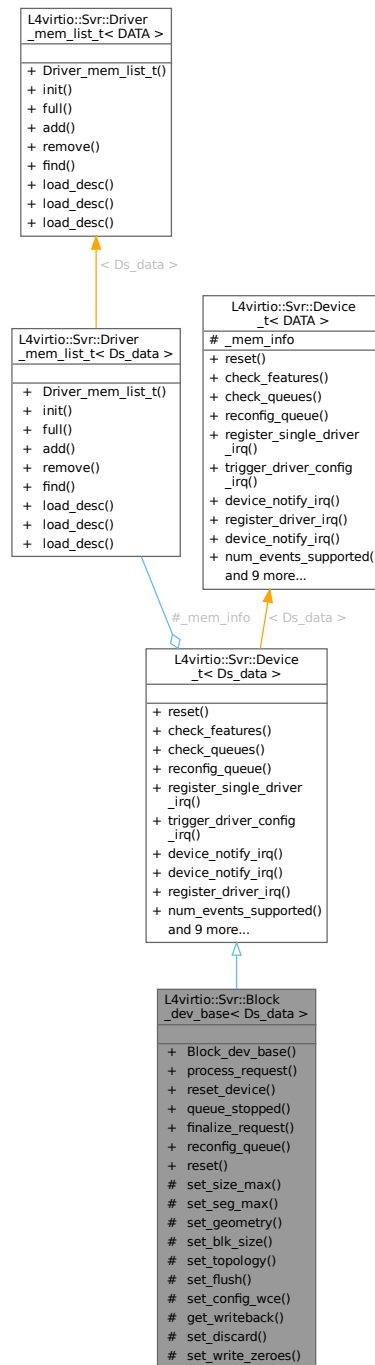
Base class for virtio block devices.

```
#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 bool **process_request** (`cx::unique_ptr< Request >` &&req)=0
Implements the actual processing of data in the 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 [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.

Additional Inherited Members

Protected Attributes inherited from `L4virtio::Svr::Device_t< Ds_data >`

- `Mem_list` `_mem_info`
Memory region list.

15.381.1 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`.

15.381.2 Constructor & Destructor Documentation

15.381.2.1 `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.

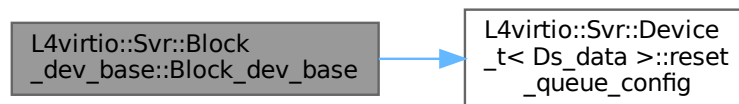
Parameters

<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 462 of file [virtio-block](#).

References [L4virtio::Svr::Device_t< Ds_data >::reset_queue_config\(\)](#).

Here is the call graph for this function:



15.381.3 Member Function Documentation

15.381.3.1 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

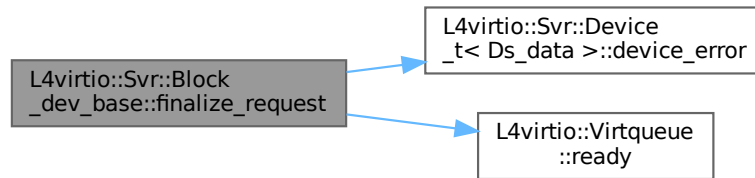
<i>req</i>	Pointer to request that has finished.
<i>sz</i>	Number of bytes consumed.
<i>status</i>	Status of request (see L4virtio_block_status).

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 515 of file [virtio-block](#).

References [L4virtio::Svr::Device_t< Ds_data >::device_error\(\)](#), and [L4virtio::Virtqueue::ready\(\)](#).

Here is the call graph for this function:



15.381.3.2 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 406 of file [virtio-block](#).

15.381.3.3 process_request()

```
template<typename Ds_data >
virtual bool L4virtio::Svr::Block_dev_base< Ds_data >::process_request (
    cxx::unique_ptr< Request > && req ) [pure virtual]
```

Implements the actual processing of data in the device.

Parameters

<i>req</i>	The request to be processed.
------------	------------------------------

Returns

If false, no further requests will be scheduled.

Synchronous and asynchronous processing of the data is supported. For asynchronous mode, the function should set up the worker and then return false. In synchronous mode, the function should return true, once processing is complete. If there is an error and processing is aborted, the status flag of `req` needs to be set accordingly and the request immediately finished with `finish_request()` if the client is to be answered.

15.381.3.4 set_blk_size()

```
template<typename Ds_data >
void L4virtio::Svr::Block_dev_base< Ds_data >::set_blk_size (
    14_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 350 of file [virtio-block](#).

15.381.3.5 set_config_wce()

```
template<typename Ds_data >
void L4virtio::Svr::Block_dev_base< Ds_data >::set_config_wce (
    14_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 393 of file [virtio-block](#).

15.381.3.6 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 420 of file [virtio-block](#).

15.381.3.7 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 308 of file [virtio-block](#).

15.381.3.8 set_topology()

```
template<typename Ds_data >
void L4virtio::Svr::Block_dev_base< Ds_data >::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 ) [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 366 of file [virtio-block](#).

15.381.3.9 set_write_zeroes()

```
template<typename Ds_data >
void L4virtio::Svr::Block_dev_base< Ds_data >::set_write_zeroes (
    l4_uint32_t max_write_zeroes_sectors,
    l4_uint32_t max_write_zeroes_seg,
    l4_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 440 of file [virtio-block](#).

The documentation for this class was generated from the following file:

- l4/l4virtio/server/virtio-block

15.382 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 >	
+	data_size()
+	has_more()
+	next_block()
+	header()

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.

15.382.1 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](#).

15.382.2 Member Function Documentation

15.382.2.1 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

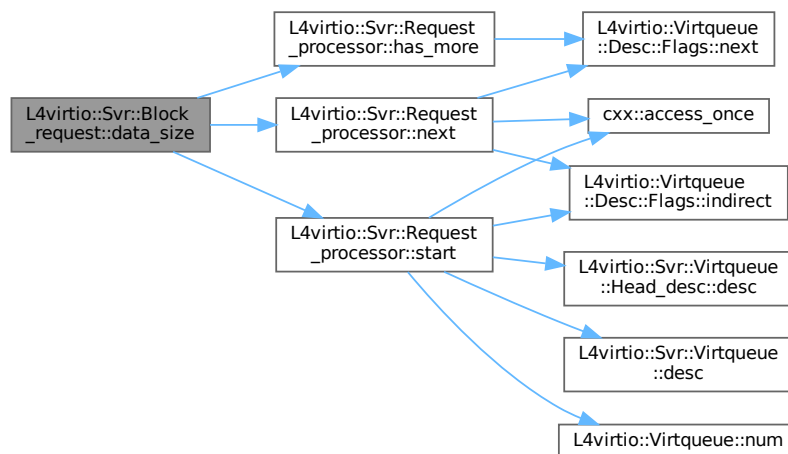
L4::Runtime_error(-L4_EIO)	Request has a bad format.
--	---------------------------

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:



15.382.2.2 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.

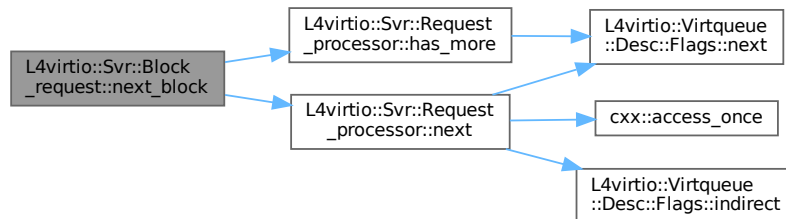
Exceptions

L4::Runtime_error	No more data block is available.
Bad_descriptor	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:

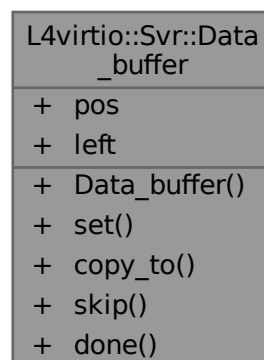
- `I4/I4virtio/server/virtio-block`

15.383 L4virtio::Svr::Data_buffer Struct Reference

Abstract data buffer.

```
#include <virtio>
```

Collaboration diagram for L4virtio::Svr::Data_buffer:



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.

15.383.1 Detailed Description

Abstract data buffer.

Definition at line 239 of file [virtio](#).

15.383.2 Constructor & Destructor Documentation

15.383.2.1 Data_buffer()

```
template<typename T >
L4virtio::Svr::Data_buffer::Data_buffer (
    T * p ) [inline], [explicit]
```

Create buffer for object `p`.

Template Parameters

<code>T</code>	type of object (implicit)
----------------	---------------------------

Parameters

<code>p</code>	pointer to object.
----------------	--------------------

The buffer shall point to the start of the object `p` and the size left is `sizeof(T)`.

Definition at line 255 of file [virtio](#).

15.383.3 Member Function Documentation

15.383.3.1 `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

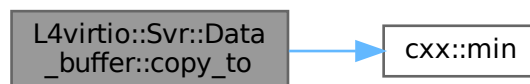
the number of bytes copied.

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 284 of file [virtio](#).

References [left](#), [cxx::min\(\)](#), and [pos](#).

Here is the call graph for this function:



15.383.3.2 `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 316 of file [virtio](#).

References [left](#).

15.383.3.3 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 268 of file [virtio](#).

References [left](#), and [pos](#).

15.383.3.4 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

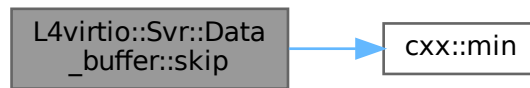
The number of bytes skipped.

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 304 of file [virtio](#).

References [left](#), [cxx::min\(\)](#), and [pos](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

- l4/l4virtio/server/virtio

15.384 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 >, 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](#) [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 [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.

15.384.1 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](#).

15.384.2 Constructor & Destructor Documentation

15.384.2.1 Dev_config() [1/2]

```
L4virtio::Svr::Dev_config::Dev_config (  
    14_uint32_t vendor,  
    14_uint32_t device,  
    unsigned cfg_size,  
    14_uint32_t num_queues = 0 ) [inline]
```

Create a L4-Virtio config data space.

Parameters

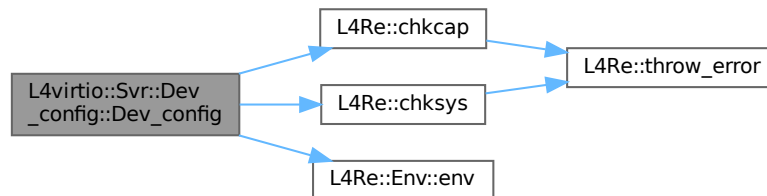
<i>vendor</i>	The vendor ID to store in config header.
<i>device</i>	The device ID to store in config header.
<i>cfg_size</i>	The size of the device-specific config data in bytes.
<i>num_queues</i>	The number of queues provided by the device.

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:



15.384.2.2 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](#).

15.384.3 Member Function Documentation

15.384.3.1 `change_queue_config()`

```
bool L4virtio::Svr::Dev_config::change_queue_config (
    l4\_uint32\_t num_queues ) [inline]
```

Setup new queue configuration.

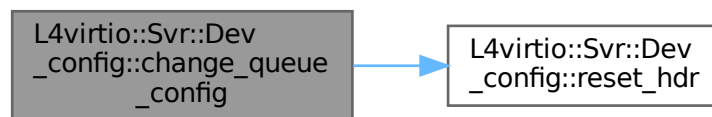
Parameters

<code>num_queues</code>	The number of queues provided by the device.
-------------------------	--

Definition at line [269](#) of file [l4virtio](#).

References [L4_PAGESIZE](#), and [reset_hdr\(\)](#).

Here is the call graph for this function:



15.384.3.2 `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 [358](#) of file [l4virtio](#).

15.384.3.3 get_cmd()

```
l4_uint32_t L4virtio::Svr::Dev_config::get_cmd ( ) const [inline]
```

Get the value from the `cmd` register.

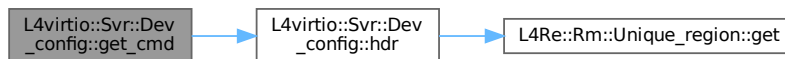
Note, the most significant eight bits are the command (0 is nothing to do). The upper eight bit are reset to zero after the command was handled.

Definition at line 224 of file [l4virtio](#).

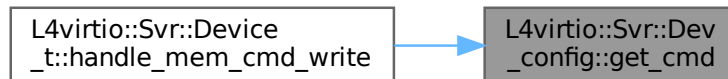
References [l4virtio_config_hdr_t::cmd](#), and [hdr\(\)](#).

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:



15.384.3.4 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 192 of file [l4virtio](#).

15.384.3.5 `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 351 of file [l4virtio](#).

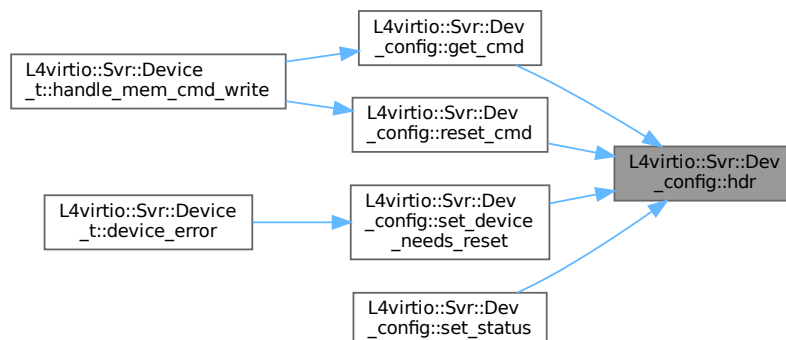
References [L4Re::Rm::Unique_region< T >::get\(\)](#).

Referenced by [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:



15.384.3.6 `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 206 of file [l4virtio](#).

15.384.3.7 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 286 of file [l4virtio](#).

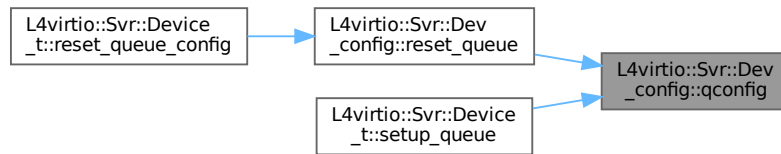
References [L4Re::Rm::Unique_region< T >::get\(\)](#), and [L4_UNLIKELY](#).

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:



15.384.3.8 reset_cmd()

```
void L4virtio::Svr::Dev_config::reset_cmd ( ) [inline]
```

Reset the `cmd` register after execution of a command.

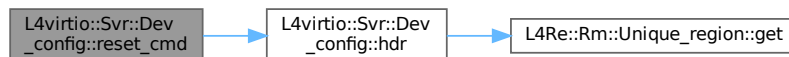
This function resets the `cmd` register in order for the client to detect that the command was executed by the device.

Definition at line 235 of file [l4virtio](#).

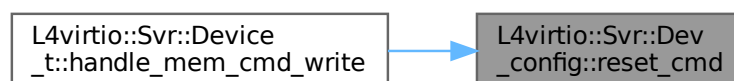
References [l4virtio_config_hdr_t::cmd](#), and [hdr\(\)](#).

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:



15.384.3.9 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

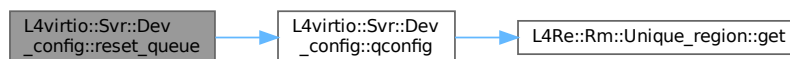
true on success, or false when *index* is out of range.

Definition at line 328 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:



15.384.3.10 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 259 of file [l4virtio](#).

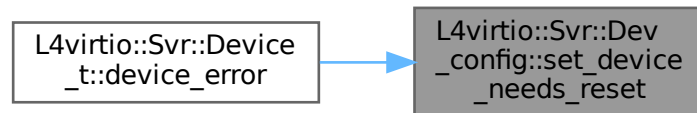
References [L4virtio::Svr::Dev_status::device_needs_reset\(\)](#), [hdr\(\)](#), [L4virtio::Svr::Dev_status::raw](#), and [l4virtio_config_hdr_t::status](#).

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:



15.384.3.11 set_status()

```
void L4virtio::Svr::Dev_config::set_status (
    Status status ) [inline]
```

Set device status register.

Parameters

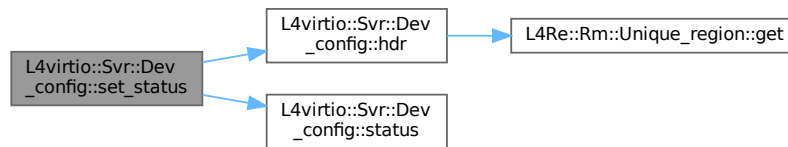
<i>status</i>	The new value for the device status register.
---------------	---

This function sets the internal status register and also the status register in the shared memory to *status*.

Definition at line 247 of file [l4virtio](#).

References [hdr\(\)](#), [L4virtio::Svr::Dev_status::raw](#), [status\(\)](#), and [l4virtio_config_hdr_t::status](#).

Here is the call graph for this function:



15.384.3.12 status()

```
Status L4virtio::Svr::Dev_config::status ( ) const [inline]
```

Get current device status (trusted).

Returns

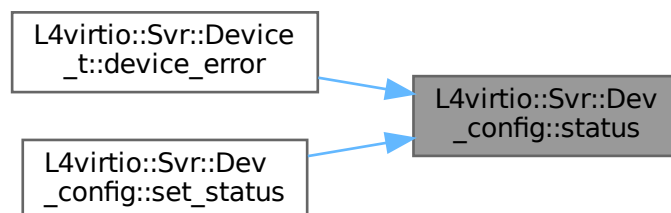
Current device status register (trusted).

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 216 of file [l4virtio](#).

Referenced by [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`

15.385 L4virtio::Svr::Dev_features Struct Reference

Type for device feature bitmap.

```
#include <virtio>
```

Inherited by L4virtio::Svr::Block_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.

15.385.1 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

15.386 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.

15.386.1 Detailed Description

Type of the device status register.

Definition at line 32 of file [virtio](#).

15.386.2 Member Function Documentation

15.386.2.1 `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:

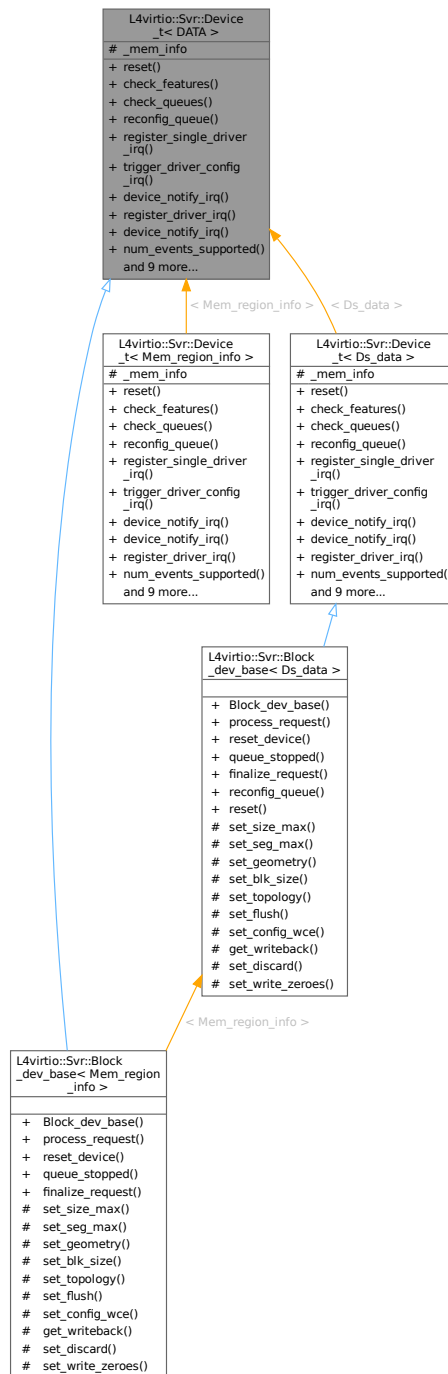
- `l4/l4virtio/server/virtio`

15.387 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() + register_single_driver_irq() + trigger_driver_config_irq() + device_notify_irq() + register_driver_irq() + device_notify_irq() + num_events_supported() and 9 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 **register_single_driver_irq** ()
callback for registering a single guest IRQ for all queues (old-style)
- virtual void **trigger_driver_config_irq** () const =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.

15.387.1 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 775 of file [l4virtio](#).

15.387.2 Member Function Documentation

15.387.2.1 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 1169 of file [l4virtio](#).

15.387.2.2 device_error()

```
template<typename DATA >
void L4virtio::Svr::Device_t< DATA >::device_error ( ) [inline]
```

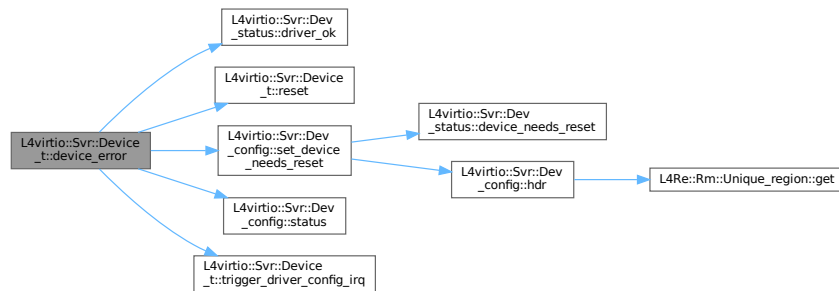
Transition device into DEVICE_NEEDS_RESET state.

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 995 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:



15.387.2.3 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 845 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:



15.387.2.4 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.

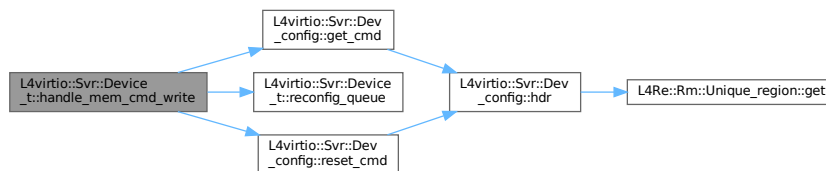
This function checks for a value in the `cmd` register and executes the command if there is any, or returns false if there was no command.

Execution of the command is signaled by a zero in the `cmd` register.

Definition at line 1129 of file `l4virtio`.

References `L4virtio::Svr::Dev_config::get_cmd()`, `L4_LIKELY`, `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:



15.387.2.5 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 983 of file `l4virtio`.

References `L4virtio::Svr::Device_t< DATA >::_mem_info`.

15.387.2.6 register_driver_irq()

```
template<typename DATA >
virtual void L4virtio::Svr::Device_t< DATA >::register_driver_irq (
    unsigned idx ) [inline], [virtual]
```

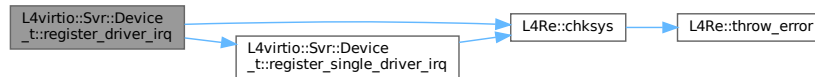
Callback for registering an notification IRQ (multi IRQ).

The default implementation maps to the implementation for single IRQ notification points.

Definition at line 831 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:



15.387.2.7 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 973 of file [l4virtio](#).

References [L4virtio::Svr::Dev_config::reset_queue\(\)](#).

Here is the call graph for this function:



15.387.2.8 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

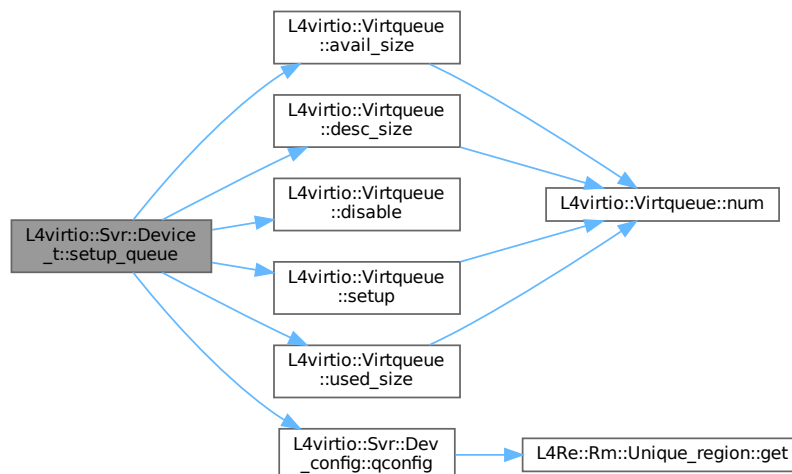
true for success.

- 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 1022 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\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

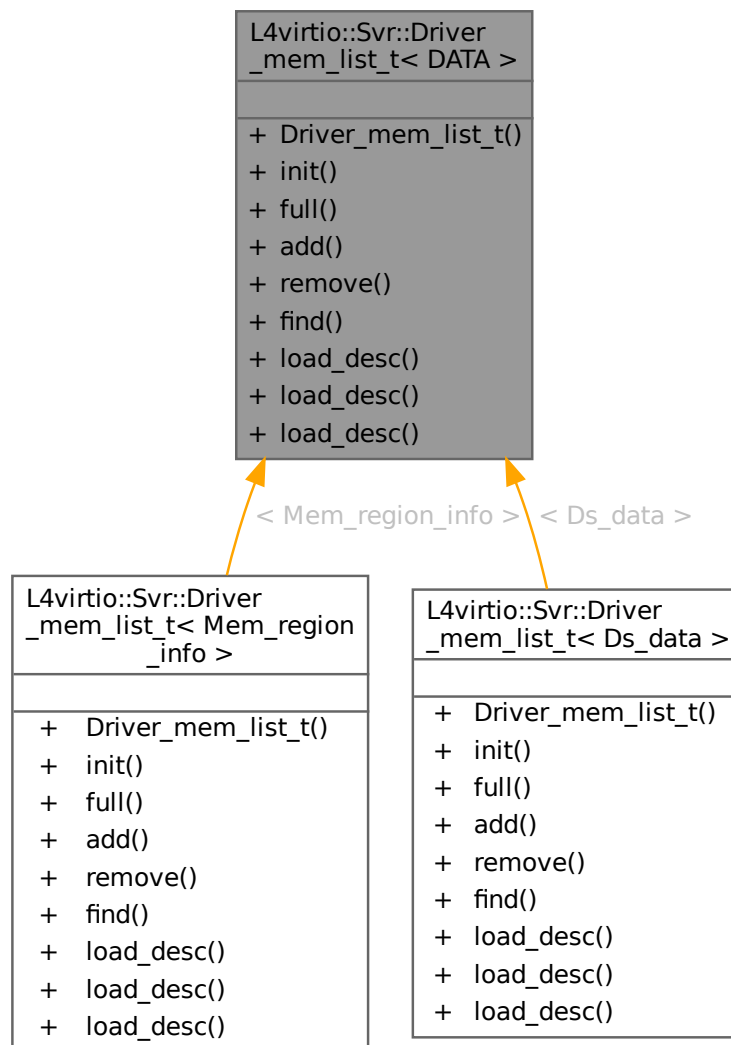
- `l4/l4virtio/server/l4virtio`

15.388 L4virtio::Svr::Driver_mem_list_t< DATA > Class Template Reference

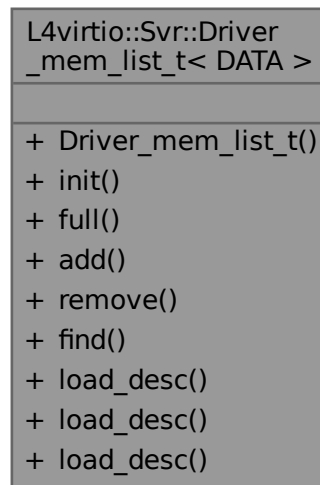
List of driver memory regions assigned to a single L4-VIRTIO transport instance.

```
#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.

15.388.1 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 609 of file [l4virtio](#).

15.388.2 Member Function Documentation

15.388.2.1 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.

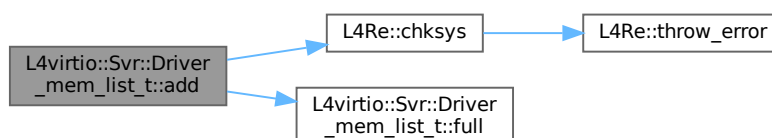
Returns

A pointer to the new region.

Definition at line 649 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:



15.388.2.2 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.

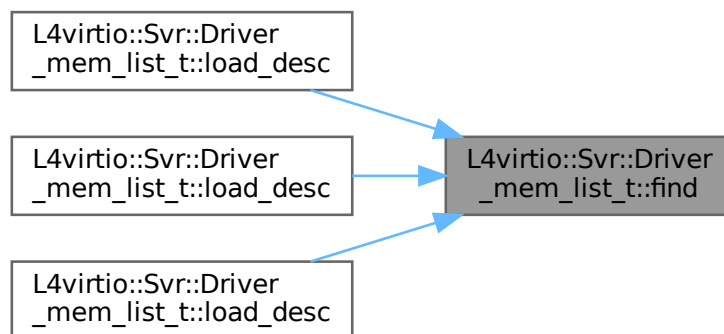
Returns

Pointer to the region containing the given region, NULL if none is found.

Definition at line 683 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:



15.388.2.3 full()

```
template<typename DATA >
bool L4virtio::Svr::Driver_mem_list_t< DATA >::full ( ) const [inline]
```

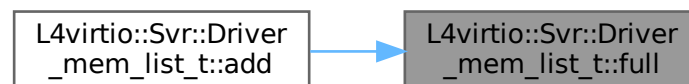
Returns

True if the remaining capacity is 0.

Definition at line 638 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Driver_mem_list_t< DATA >::add\(\)](#).

Here is the caller graph for this function:

**15.388.2.4 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 630 of file [l4virtio](#).

15.388.2.5 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 Request_processor::start() and Request_processor::next() 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 Virtqueue::Desc descriptor, and (3) a pointer to the calling Request_processor .
------------	--

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)

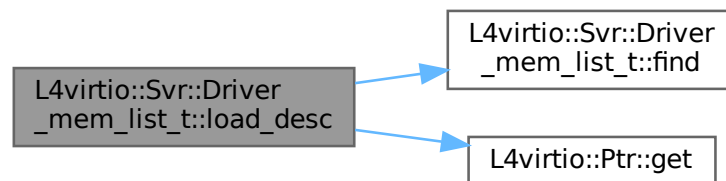
Exceptions

<i>Bad_descriptor</i>	The descriptor address could not be translated.
---------------------------------------	---

Definition at line 744 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:



15.388.2.6 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.

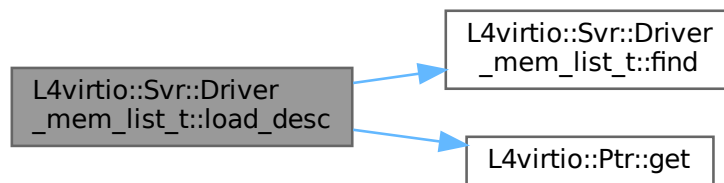
Exceptions

<i>Bad_descriptor</i>	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:



15.388.2.7 `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

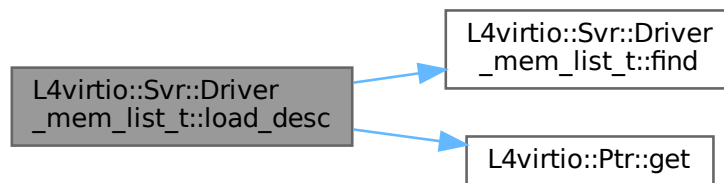
Exceptions

Bad_descriptor	The descriptor address could not be translated.
--------------------------------	---

Definition at line 697 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:



15.388.2.8 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.

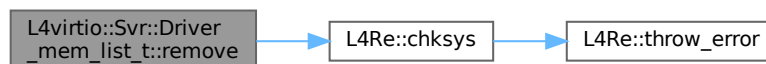
Parameters

<i>r</i>	The region to remove (result from add() , or find()).
----------	--

Definition at line 663 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:

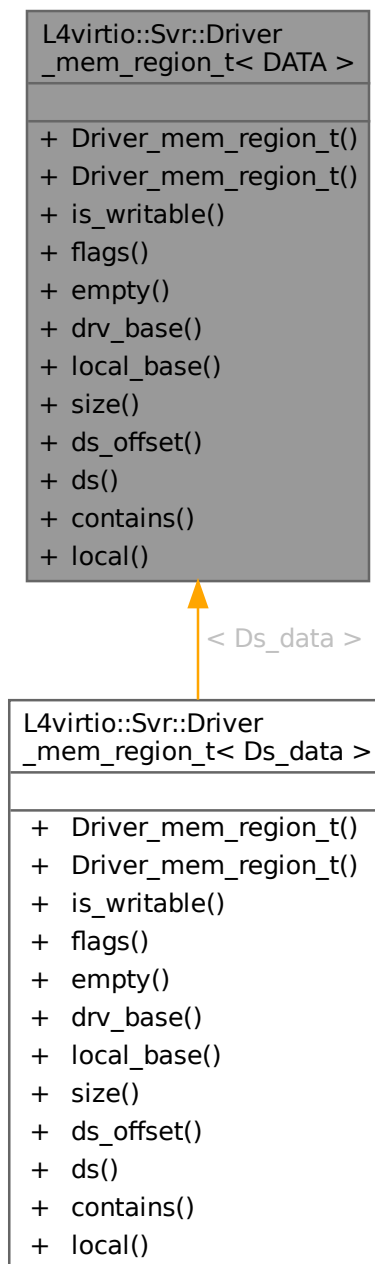
- `l4/l4virtio/server/l4virtio`

15.389 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"> + Driver_mem_region_t() + Driver_mem_region_t() + is_writable() + flags() + empty() + drv_base() + local_base() + size() + ds_offset() + ds() + contains() + local()

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
- void * **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.

15.389.1 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 433 of file [l4virtio](#).

15.389.2 Constructor & Destructor Documentation

15.389.2.1 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

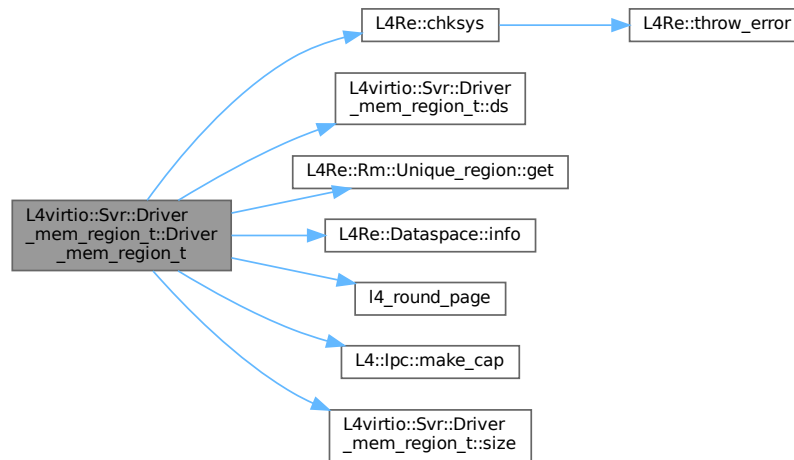
<i>drv_base</i>	Base address of the memory region used by the driver.
<i>size</i>	Size of the memory region.
<i>offset</i>	Offset within the data space that is mapped to <i>drv_base</i> within the driver.
<i>ds</i>	Data space capability backing the memory.

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 478 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:



15.389.3 Member Function Documentation

15.389.3.1 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.

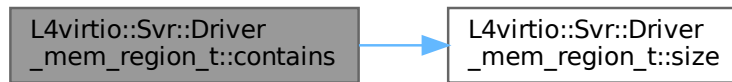
Returns

true if the given driver address region is contained in this region, false else.

Definition at line 572 of file [l4virtio](#).

References [L4virtio::Svr::Driver_mem_region_t< DATA >::size\(\)](#).

Here is the call graph for this function:



15.389.3.2 `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 551 of file [l4virtio](#).

15.389.3.3 `ds()`

```
template<typename DATA >
L4::Cap< L4Re::Dataspace > L4virtio::Svr::Driver_mem_region_t< DATA >::ds ( ) const [inline]
```

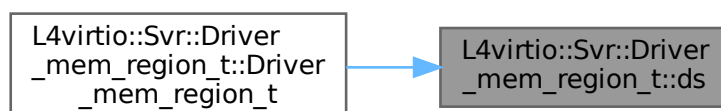
Returns

The data space capability for this region.

Definition at line 563 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Driver_mem_region_t< DATA >::Driver_mem_region_t\(\)](#).

Here is the caller graph for this function:



15.389.3.4 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 560 of file [l4virtio](#).

15.389.3.5 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 547 of file [l4virtio](#).

15.389.3.6 flags()

```
template<typename DATA >
Flags L4virtio::Svr::Driver_mem_region_t< DATA >::flags ( ) const [inline]
```

Returns

The flags for this region.

Definition at line 544 of file [l4virtio](#).

15.389.3.7 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 541 of file [l4virtio](#).

15.389.3.8 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 596 of file [l4virtio](#).

References [L4virtio::Ptr< T >::get\(\)](#).

Here is the call graph for this function:



15.389.3.9 local_base()

```
template<typename DATA >
void * L4virtio::Svr::Driver\_mem\_region\_t< DATA >::local_base ( ) const [inline]
```

Returns

The local base address.

Definition at line 554 of file [l4virtio](#).

References [L4Re::Rm::Unique_region< T >::get\(\)](#).

Here is the call graph for this function:



15.389.3.10 size()

```
template<typename DATA >
l4_umword_t L4virtio::Svr::Driver_mem_region_t< DATA >::size ( ) const [inline]
```

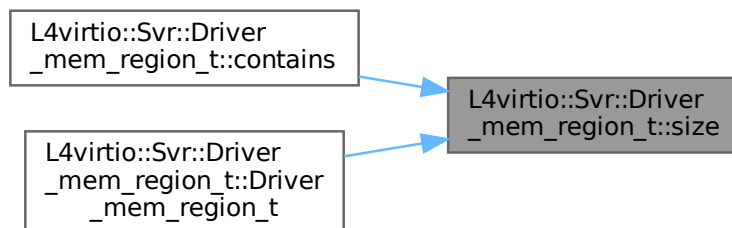
Returns

The size of the region in bytes.

Definition at line 557 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:

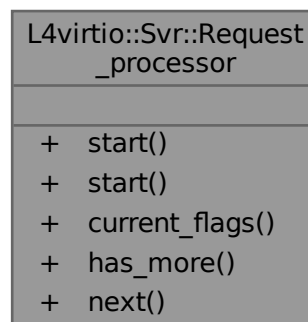
- [l4/l4virtio/server/l4virtio](#)

15.390 L4virtio::Svr::Request_processor Class Reference

Encapsulate the state for processing a VIRTIO request.

```
#include <virtio>
```

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.

15.390.1 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 399 of file [virtio](#).

15.390.2 Member Function Documentation

15.390.2.1 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 469 of file [virtio](#).

References [L4virtio::Virtqueue::Desc::flags](#).

15.390.2.2 has_more()

```
bool L4virtio::Svr::Request_processor::has_more ( ) const [inline]
```

Are there more chained descriptors?

Returns

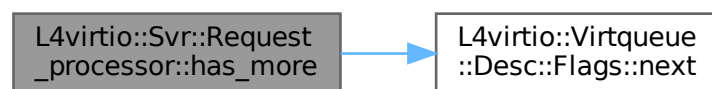
true if there are more chained descriptors in the current request.

Definition at line 476 of file [virtio](#).

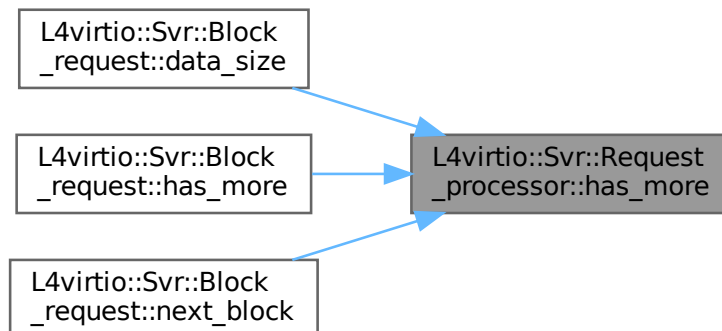
References [L4virtio::Virtqueue::Desc::flags](#), and [L4virtio::Virtqueue::Desc::Flags::next\(\)](#).

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:



15.390.2.3 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->load_desc()</code>

Return values

<i>true</i>	A next descriptor is available.
<i>false</i>	No descriptor available.

Exceptions

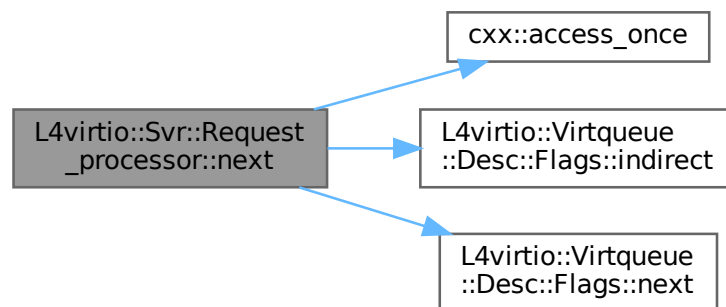
<i>Bad_descriptor</i>	The <code>next</code> index of this descriptor is invalid.
---------------------------------------	--

Definition at line 493 of file [virtio](#).

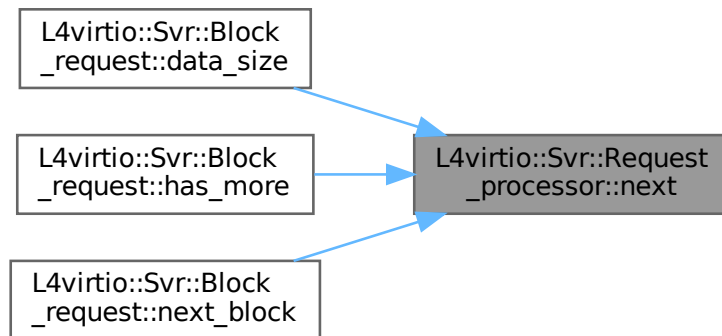
References [cxx::access_once\(\)](#), [L4virtio::Svr::Bad_descriptor::Bad_flags](#), [L4virtio::Svr::Bad_descriptor::Bad_next](#), [L4virtio::Virtqueue::Desc::flags](#), [L4virtio::Virtqueue::Desc::Flags::indirect\(\)](#), [L4_UNLIKELY](#), [L4virtio::Virtqueue::Desc::Flags::next\(\)](#), and [L4virtio::Virtqueue::Desc::next](#).

Referenced by [L4virtio::Svr::Block_request<Ds_data>::data_size\(\)](#), [L4virtio::Svr::Block_request<Ds_data>::has_more\(\)](#), 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:



15.390.2.4 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 Virtqueue::next_avail()
<i>args</i>	Extra arguments passed to <code>dm->load_desc()</code>

Precondition

The given request must be valid.

Exceptions

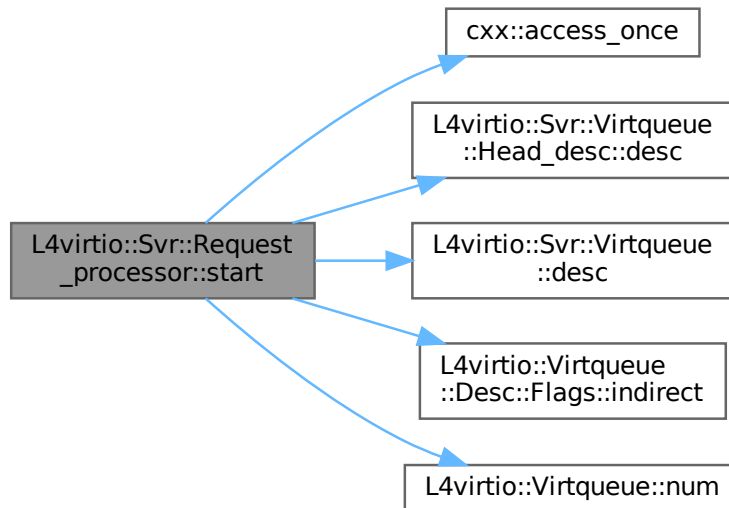
Bad_descriptor	The descriptor has an invalid size or <code>load_desc()</code> has thrown an exception by itself.
--------------------------------	---

Definition at line 428 of file [virtio](#).

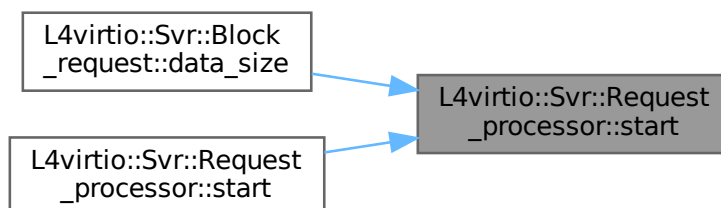
References [cxx::access_once\(\)](#), [L4virtio::Svr::Bad_descriptor::Bad_size](#), [L4virtio::Svr::Virtqueue::Head_desc::desc\(\)](#), [L4virtio::Svr::Virtqueue::desc\(\)](#), [L4virtio::Virtqueue::Desc::flags](#), [L4virtio::Virtqueue::Desc::Flags::indirect\(\)](#), [L4_UNLIKELY](#), [L4virtio::Virtqueue::Desc::len](#), and [L4virtio::Virtqueue::num\(\)](#).

Referenced by [L4virtio::Svr::Block_request<Ds_data>::data_size\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.390.2.5 `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 Virtqueue::next_avail()
<i>args</i>	Extra arguments passed to <code>dm->load_desc()</code>

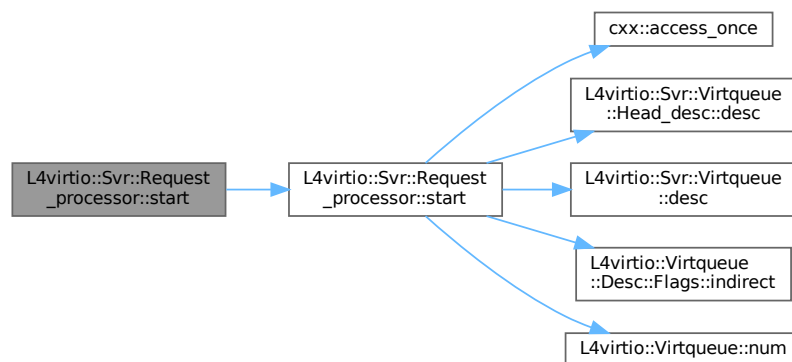
Precondition

The given request must be valid.

Definition at line 459 of file [virtio](#).

References [start\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

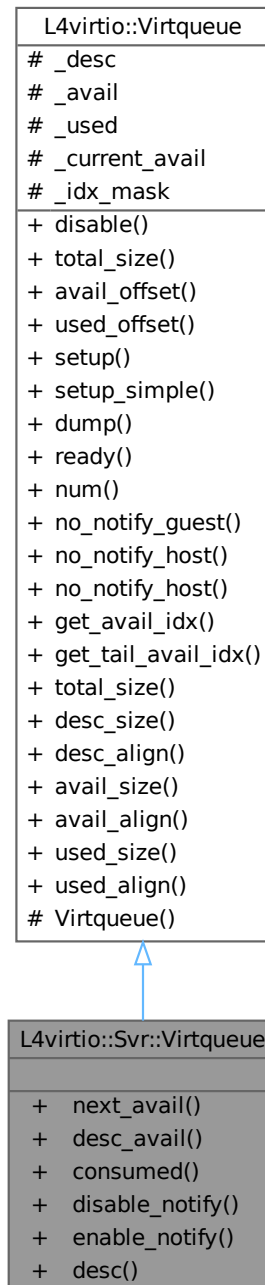
- `I4/I4virtio/server/virtio`

15.391 L4virtio::Svr::Virtqueue Class Reference

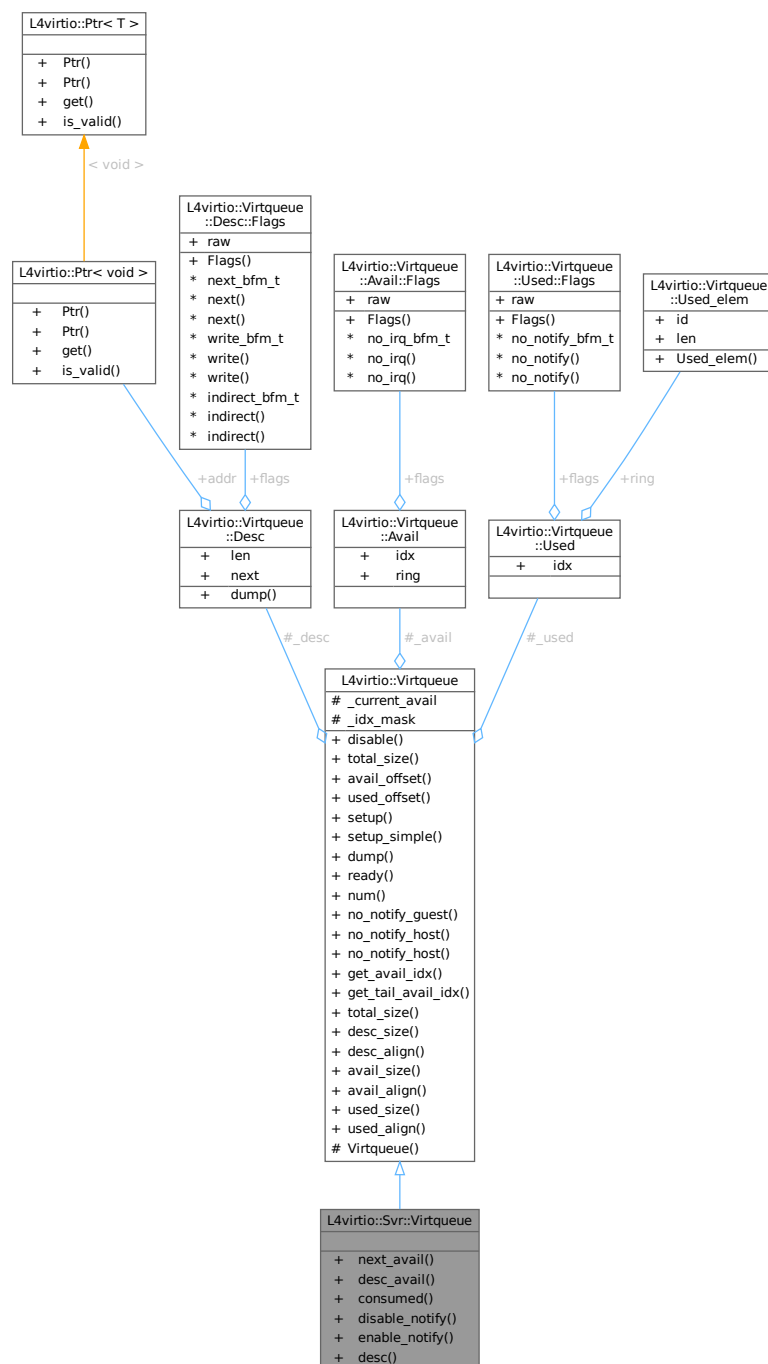
[Virtqueue](#) implementation for the device.

```
#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.
- 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` ()
Create a disabled virtqueue.

Protected Attributes inherited from L4virtio::Virtqueue

- `Desc * _desc`
pointer to descriptor table, NULL if queue is off.
- `Avail * _avail`
pointer to available ring.
- `Used * _used`
pointer to used ring.
- `l4_uint16_t _current_avail`
The life counter for the queue.
- `l4_uint16_t _idx_mask`
mask used for indexing into the descriptor table and the rings.

15.391.1 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`.

15.391.2 Member Function Documentation

15.391.2.1 consumed()

```
void L4virtio::Svr::Virtqueue::consumed (
    Head_desc const & r,
    l4_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 166 of file [virtio](#).

References [L4virtio::Virtqueue::_desc](#), [L4virtio::Virtqueue::_idx_mask](#), [L4virtio::Virtqueue::_used](#), [L4virtio::Virtqueue::Used::idx](#), and [L4virtio::Virtqueue::Used::ring](#).

15.391.2.2 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 231 of file [virtio](#).

References [L4virtio::Virtqueue::_desc](#).

Referenced by [L4virtio::Svr::Request_processor::start\(\)](#).

Here is the caller graph for this function:



15.391.2.3 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 154 of file [virtio](#).

References [L4virtio::Virtqueue::_avail](#), [L4virtio::Virtqueue::_current_avail](#), and [L4virtio::Virtqueue::Avail::idx](#).

15.391.2.4 disable_notify()

```
void L4virtio::Svr::Virtqueue::disable_notify ( ) [inline]
```

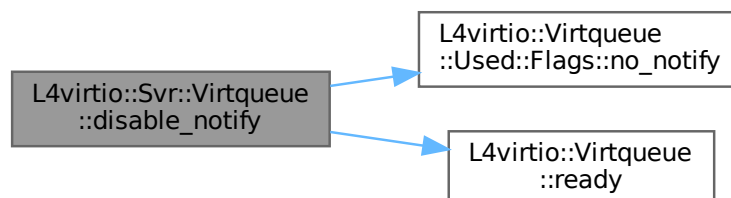
Set the 'no notify' flag for this queue.

This function may be called on a disabled queue.

Definition at line 208 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:



15.391.2.5 enable_notify()

```
void L4virtio::Svr::Virtqueue::enable_notify ( ) [inline]
```

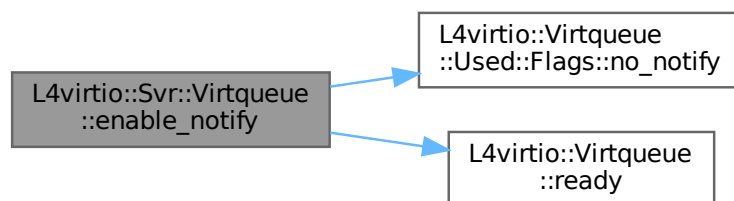
Clear the 'no notify' flag for this queue.

This function may be called on a disabled queue.

Definition at line 219 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:



15.391.2.6 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.

Note

The return value must be checked even when a previous [desc_avail\(\)](#) returned true.

Definition at line 137 of file [virtio](#).

References [L4virtio::Virtqueue::_avail](#), [L4virtio::Virtqueue::_current_avail](#), [L4virtio::Virtqueue::_idx_mask](#), [L4virtio::Virtqueue::Avail::idx](#), [L4_LIKELY](#), and [L4virtio::Virtqueue::Avail::ring](#).

The documentation for this class was generated from the following file:

- I4/I4virtio/server/virtio

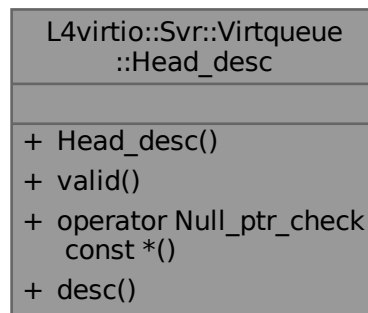
15.392 L4virtio::Svr::Virtqueue::Head_desc Class Reference

VIRTIO request, essentially a descriptor from the available ring.

```
#include <virtio>
```

Inherited by L4virtio::Svr::Virtqueue::Request.

Collaboration diagram for L4virtio::Svr::Virtqueue::Head_desc:



Public Member Functions

- **Head_desc** ()
Make invalid (NULL) request.
- bool **valid** () const
- **operator Null_ptr_check const *** () const
- **Desc** const * **desc** () const

15.392.1 Detailed Description

VIRTIO request, essentially a descriptor from the available ring.

Definition at line 93 of file [virtio](#).

15.392.2 Member Function Documentation

15.392.2.1 desc()

```
Desc const * L4virtio::Svr::Virtqueue::Head_desc::desc ( ) const [inline]
```

Returns

Pointer to the head descriptor of the request.

Definition at line 114 of file [virtio](#).

Referenced by [L4virtio::Svr::Request_processor::start\(\)](#).

Here is the caller graph for this function:

**15.392.2.2 operator Null_ptr_check const *()**

```
L4virtio::Svr::Virtqueue::Head_desc::operator Null_ptr_check const * ( ) const [inline]
```

Returns

True if the request is valid (not NULL).

Definition at line 110 of file [virtio](#).

15.392.2.3 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:

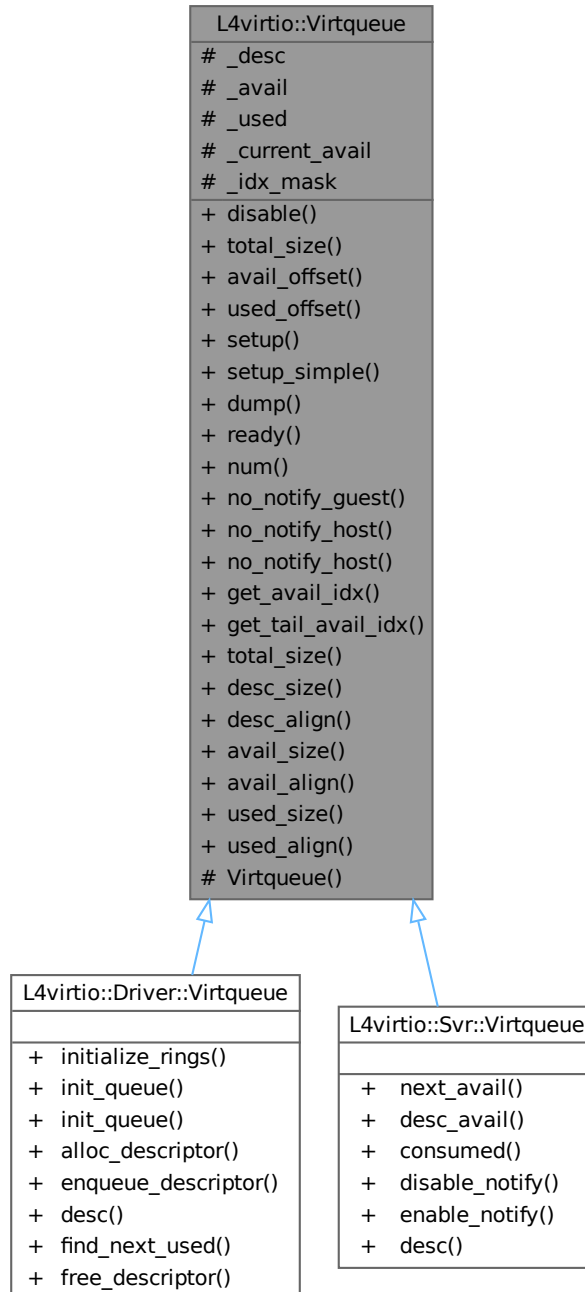
- `l4/l4virtio/server/virtio`

15.393 L4virtio::Virtqueue Class Reference

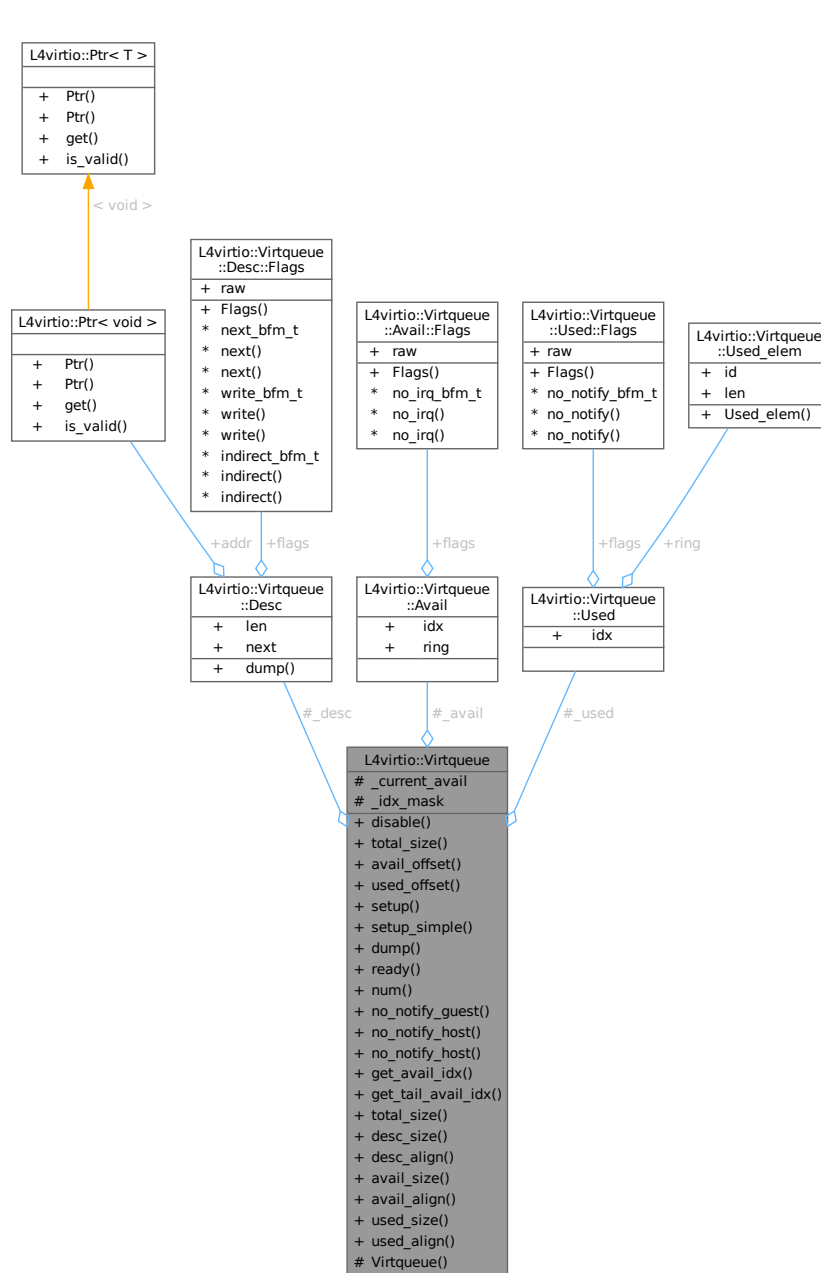
Low-level [Virtqueue](#).

```
#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 ()**
Create a disabled virtqueue.

Protected Attributes

- **Desc * _desc**
pointer to descriptor table, NULL if queue is off.
- **Avail * _avail**
pointer to available ring.
- **Used * _used**
pointer to used ring.
- **l4_uint16_t _current_avail**
The life counter for the queue.
- **l4_uint16_t _idx_mask**
mask used for indexing into the descriptor table and the rings.

15.393.1 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 96 of file [virtqueue](#).

15.393.2 Member Function Documentation

15.393.2.1 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 300 of file [virtqueue](#).

15.393.2.2 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

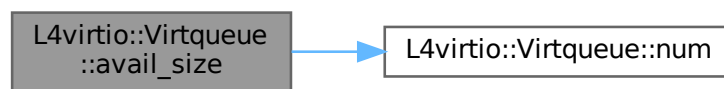
The size in bytes needed for an available ring with *num* entries.

Definition at line 292 of file [virtqueue](#).

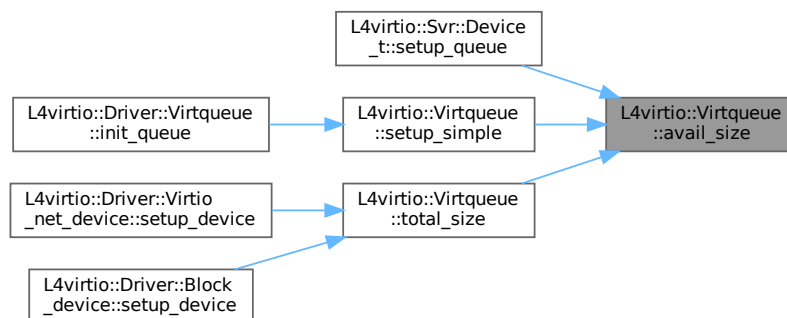
References [num\(\)](#).

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:



15.393.2.3 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 282 of file [virtqueue](#).

15.393.2.4 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

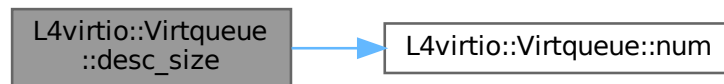
The size in bytes needed for a descriptor table with `num` entries.

Definition at line 274 of file [virtqueue](#).

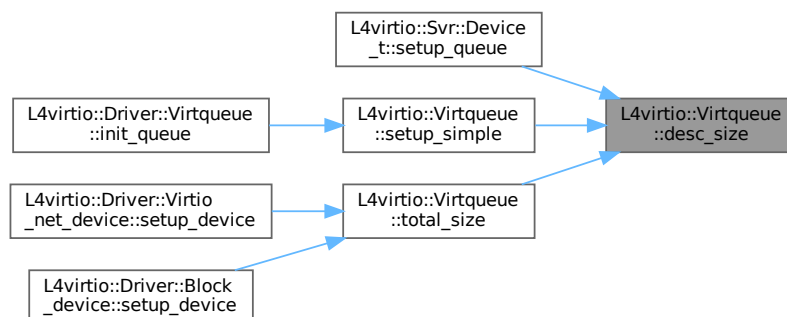
References [num\(\)](#).

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:



15.393.2.5 disable()

```
void L4virtio::Virtqueue::disable ( ) [inline]
```

Completely disable the queue.

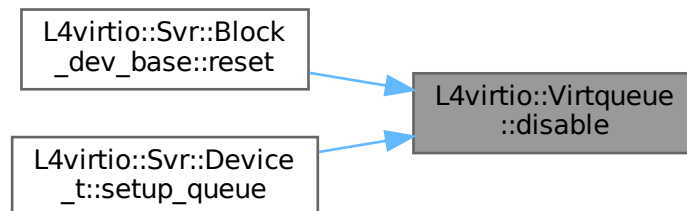
[setup\(\)](#) must be used to enable the queue again.

Definition at line 237 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:



15.393.2.6 dump()

```
void L4virtio::Virtqueue::dump (
    Desc const * d ) const [inline]
```

Dump descriptors for this queue.

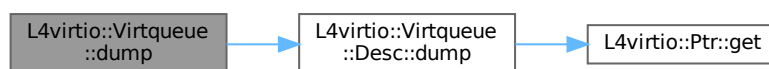
Precondition

the queue must be in working state.

Definition at line 404 of file [virtqueue](#).

References [_desc](#), and [L4virtio::Virtqueue::Desc::dump\(\)](#).

Here is the call graph for this function:



15.393.2.7 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 461 of file [virtqueue](#).

References [_avail](#), and [L4virtio::Virtqueue::Avail::idx](#).

15.393.2.8 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 468 of file [virtqueue](#).

References [_current_avail](#).

15.393.2.9 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.

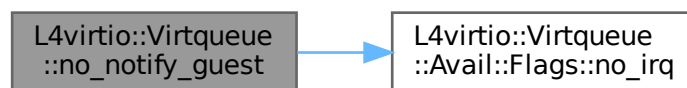
Returns

true if the guest does not want to get IRQs (currently).

Definition at line 426 of file [virtqueue](#).

References [_avail](#), [L4virtio::Virtqueue::Avail::flags](#), and [L4virtio::Virtqueue::Avail::Flags::no_irq\(\)](#).

Here is the call graph for this function:



15.393.2.10 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.

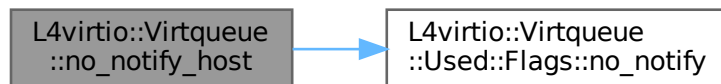
Returns

true if the host does not want to get IRQs (currently).

Definition at line 438 of file [virtqueue](#).

References [_used](#), [L4virtio::Virtqueue::Used::flags](#), and [L4virtio::Virtqueue::Used::Flags::no_notify\(\)](#).

Here is the call graph for this function:

**15.393.2.11 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 448 of file [virtqueue](#).

References [_used](#), [L4virtio::Virtqueue::Used::flags](#), and [L4virtio::Virtqueue::Used::Flags::no_notify\(\)](#).

Here is the call graph for this function:



15.393.2.12 num()

```
unsigned L4virtio::Virtqueue::num ( ) const [inline]
```

Returns

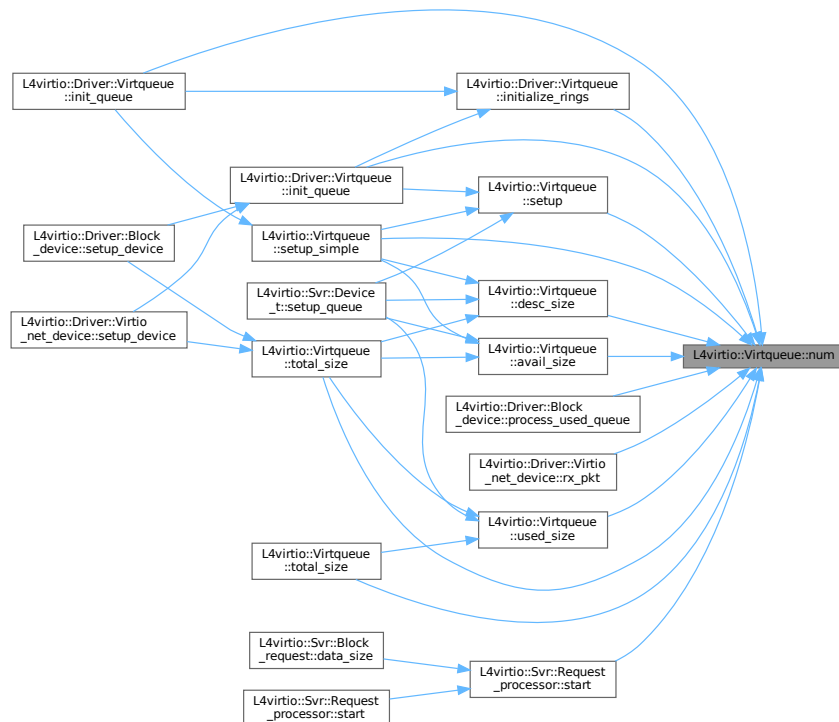
The number of entries in the ring.

Definition at line 416 of file [virtqueue](#).

References [_idx_mask](#).

Referenced by [avail_size\(\)](#), [desc_size\(\)](#), [L4virtio::Driver::Virtqueue::init_queue\(\)](#), [L4virtio::Driver::Virtqueue::init_queue\(\)](#), [L4virtio::Driver::Virtqueue::initialize_rings\(\)](#), [L4virtio::Driver::Block_device::process_used_queue\(\)](#), [L4virtio::Driver::Virtio_net_device::setup\(\)](#), [setup_simple\(\)](#), [L4virtio::Svr::Request_processor::start\(\)](#), [total_size\(\)](#), [total_size\(\)](#), and [used_size\(\)](#).

Here is the caller graph for this function:

**15.393.2.13 ready()**

```
bool L4virtio::Virtqueue::ready ( ) const [inline]
```

Test if this queue is in working state.

Returns

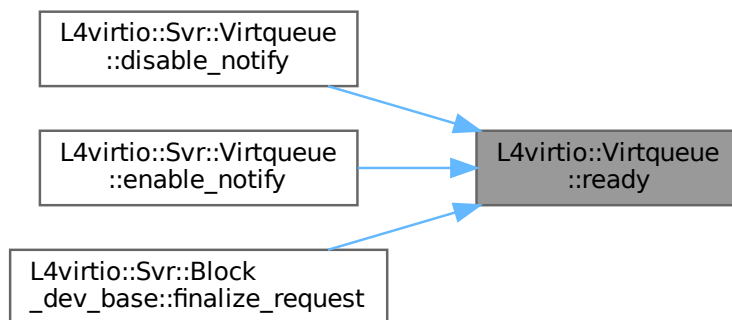
true when the queue is in working state, false else.

Definition at line 412 of file [virtqueue](#).

References [_desc](#), and [L4_LIKELY](#).

Referenced by [L4virtio::Svr::Virtqueue::disable_notify\(\)](#), [L4virtio::Svr::Virtqueue::enable_notify\(\)](#), and [L4virtio::Svr::Block_dev_base<](#)

Here is the caller graph for this function:

**15.393.2.14 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.)

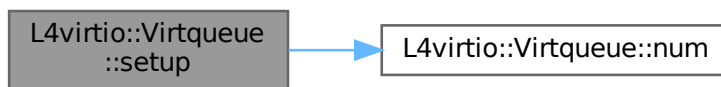
Due to the data type of the descriptors, the queue can have a maximum size of 2^{16} .

Definition at line 362 of file [virtqueue](#).

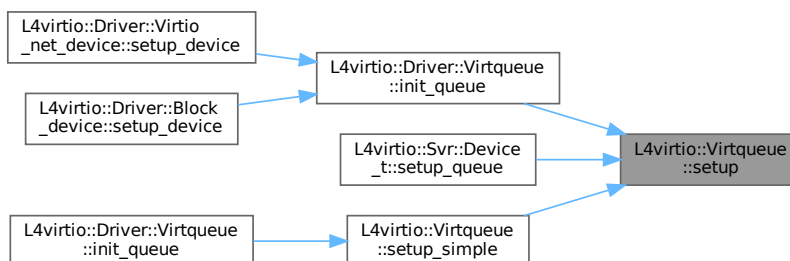
References [_avail](#), [_current_avail](#), [_desc](#), [_idx_mask](#), [_used](#), [L4_EINVAL](#), and [num\(\)](#).

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:



15.393.2.15 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 <i>ring</i> must be at least <code>total_size(num)</code> bytes in size and have an alignment of <code>Desc_align</code> (desc_align()) bits.

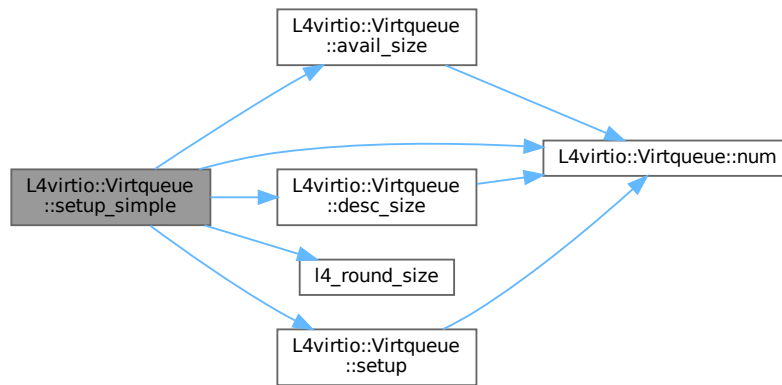
Due to the data type of the descriptors, the queue can have a maximum size of 2^{16} .

Definition at line 391 of file [virtqueue](#).

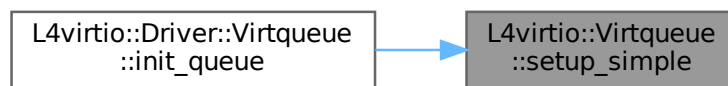
References [avail_size\(\)](#), [desc_size\(\)](#), [l4_round_size\(\)](#), [num\(\)](#), and [setup\(\)](#).

Referenced by [L4virtio::Driver::Virtqueue::init_queue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.393.2.16 total_size() [1/2]

```
unsigned long L4virtio::Virtqueue::total_size ( ) const [inline]
```

Calculate the total size of this virtqueue.

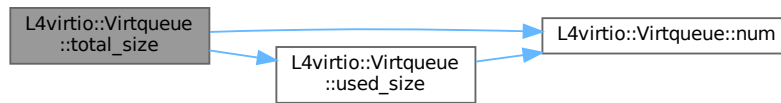
Precondition

The queue has been set up.

Definition at line 327 of file [virtqueue](#).

References [_desc](#), [_used](#), [num\(\)](#), and [used_size\(\)](#).

Here is the call graph for this function:



15.393.2.17 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

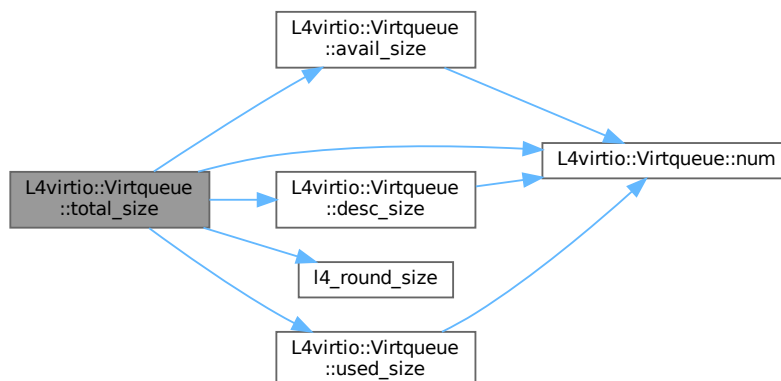
The total size in bytes of the queue data structures.

Definition at line 258 of file [virtqueue](#).

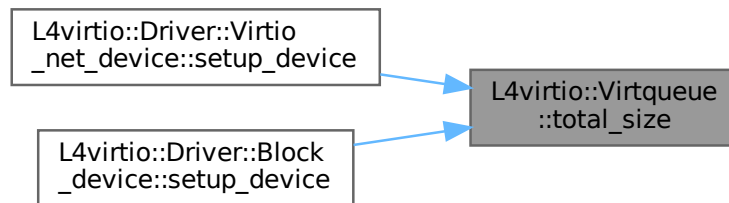
References [avail_size\(\)](#), [desc_size\(\)](#), [l4_round_size\(\)](#), [num\(\)](#), and [used_size\(\)](#).

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:



15.393.2.18 `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 319 of file [virtqueue](#).

15.393.2.19 `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

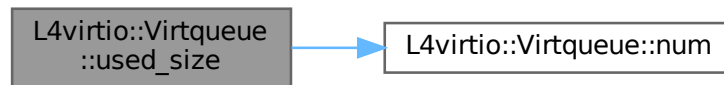
The size in bytes needed for an used ring with `num` entries.

Definition at line 311 of file [virtqueue](#).

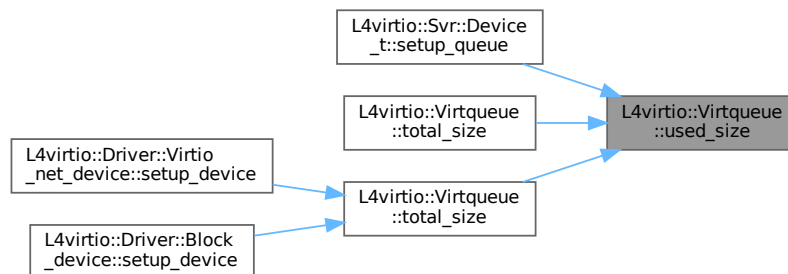
References [num\(\)](#).

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`

15.394 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.

15.394.1 Detailed Description

Type of available ring, this is read-only for the host.

Definition at line [143](#) of file [virtqueue](#).

The documentation for this class was generated from the following file:

- [l4/l4virtio/virtqueue](#)

15.395 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.

15.395.1 Detailed Description

Flags of the available ring.

Definition at line 149 of file [virtqueue](#).

15.395.2 Member Typedef Documentation

15.395.2.1 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 158 of file [virtqueue](#).

The documentation for this struct was generated from the following file:

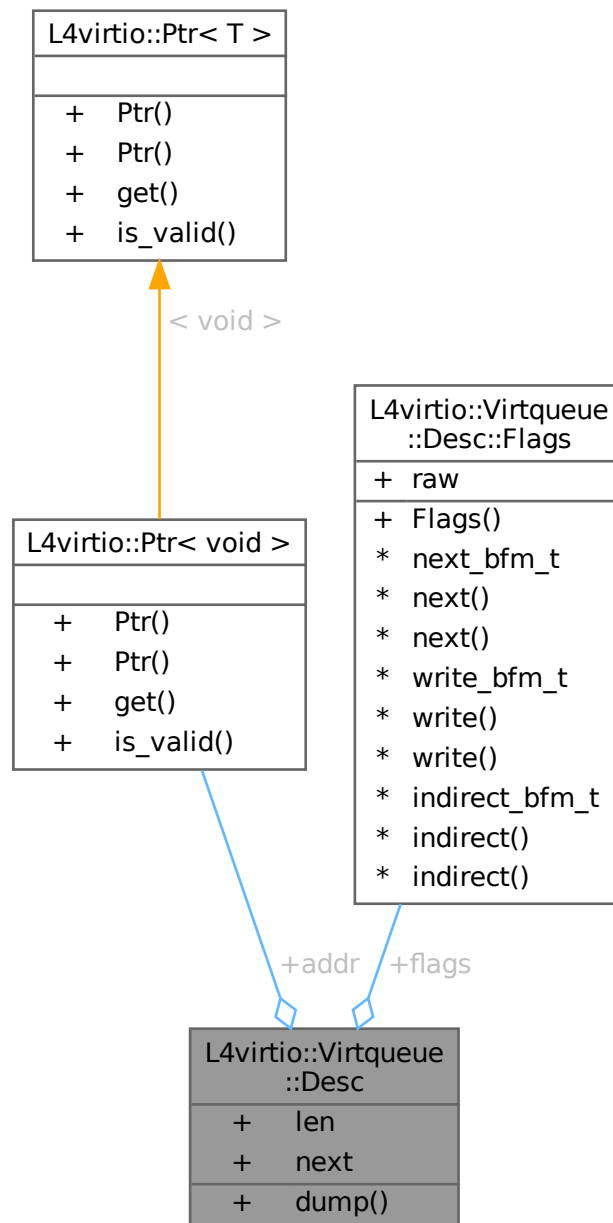
- [l4/l4virtio/virtqueue](#)

15.396 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.

15.396.1 Detailed Description

Descriptor in the descriptor table.

Definition at line 102 of file [virtqueue](#).

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

15.397 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.

15.397.1 Detailed Description

Type for descriptor flags.

Definition at line 108 of file [virtqueue](#).

15.397.2 Member Typedef Documentation

15.397.2.1 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 121 of file [virtqueue](#).

15.397.2.2 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 117 of file [virtqueue](#).

15.397.2.3 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 119 of file [virtqueue](#).

The documentation for this struct was generated from the following file:

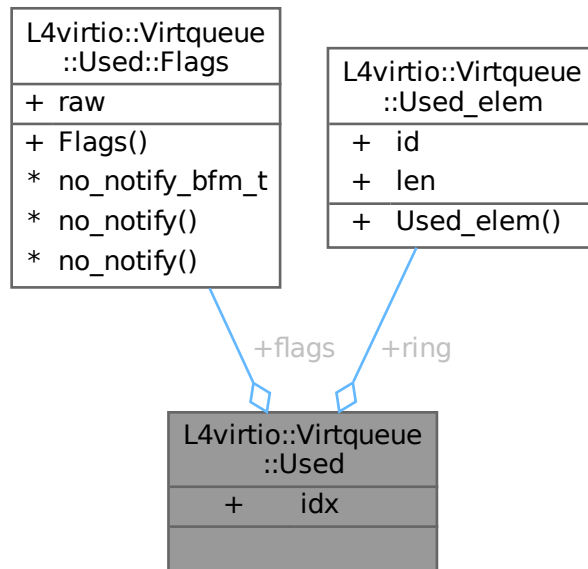
- `l4/l4virtio/virtqueue`

15.398 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.

15.398.1 Detailed Description

Used ring.

Definition at line 188 of file [virtqueue](#).

The documentation for this class was generated from the following file:

- [l4/l4virtio/virtqueue](#)

15.399 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.

15.399.1 Detailed Description

flags for the used ring.

Definition at line 194 of file [virtqueue](#).

15.399.2 Member Typedef Documentation

15.399.2.1 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 203 of file [virtqueue](#).

The documentation for this struct was generated from the following file:

- [l4/l4virtio/virtqueue](#)

15.400 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

15.400.1 Detailed Description

Type of an element of the used ring.

Definition at line 169 of file [virtqueue](#).

15.400.2 Constructor & Destructor Documentation

15.400.2.1 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 180 of file [virtqueue](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtqueue

15.401 l4virtio_block_config_t Struct Reference

Device configuration for block devices.

```
#include <virtio_block.h>
```

Collaboration diagram for l4virtio_block_config_t:

l4virtio_block_config_t	
+	capacity
+	size_max
+	seg_max
+	blk_size

Data Fields

- [l4_uint64_t](#) **capacity**
Capacity of device in 512-byte sectors.
- [l4_uint32_t](#) **size_max**
Maximum size of a single segment.
- [l4_uint32_t](#) **seg_max**
Maximum number of segments per request.
- [l4_uint32_t](#) **blk_size**
Block size of underlying disk.

15.401.1 Detailed Description

Device configuration for block devices.

Definition at line 80 of file [virtio_block.h](#).

The documentation for this struct was generated from the following file:

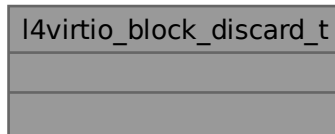
- l4/l4virtio/virtio_block.h

15.402 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:



15.402.1 Detailed Description

Structure used for the write zeroes and discard commands.

Definition at line 70 of file [virtio_block.h](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio_block.h

15.403 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.

15.403.1 Detailed Description

Header structure of a request for a block device.

Definition at line 54 of file [virtio_block.h](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio_block.h

15.404 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
+ status
+ cfg_driver_notify_index
+ cfg_device_notify_index
+ cmd

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
- [l4_uint32_t](#) **status**
Device status register (read-only).
- [l4_uint32_t](#) **cfg_driver_notify_index**

W: Event index to be used for config notifications (device to driver)

- [l4_uint32_t cfg_device_notify_index](#)

R: Event index to be used for config notifications (driver to device)

- [l4_uint32_t cmd](#)

L4 specific command register polled by the driver iff supported.

15.404.1 Detailed Description

L4-VIRTIO config header, provided in shared data space.

Definition at line 125 of file [virtio.h](#).

15.404.2 Field Documentation

15.404.2.1 status

[l4_uint32_t](#) [l4virtio_config_hdr_t::status](#)

Device status register (read-only).

The register must be written using [l4virtio_set_status\(\)](#).

must be at offset 0x70 (virtio-mmio)

Definition at line 164 of file [virtio.h](#).

Referenced by [L4virtio::Svr::Dev_config::reset_hdr\(\)](#), [L4virtio::Svr::Dev_config::set_device_needs_reset\(\)](#), and [L4virtio::Svr::Dev_config::set_status\(\)](#).

The documentation for this struct was generated from the following file:

- [l4/l4virtio/virtio.h](#)

15.405 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)

15.405.1 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 206 of file [virtio.h](#).

The documentation for this struct was generated from the following file:

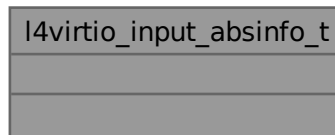
- `l4/l4virtio/virtio.h`

15.406 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:



15.406.1 Detailed Description

Information about the absolute axis in the underlying evdev implementation.

Definition at line 44 of file [virtio_input.h](#).

The documentation for this struct was generated from the following file:

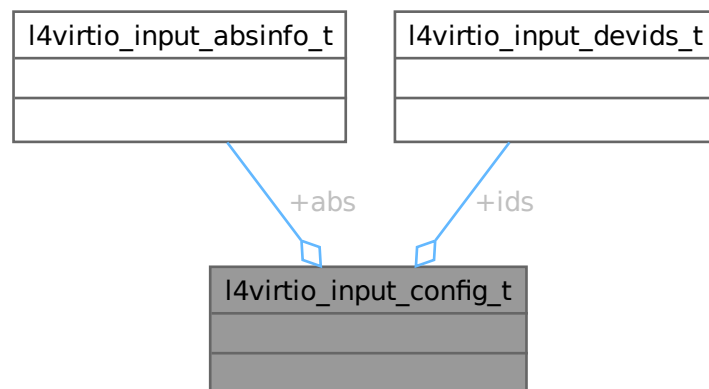
- l4/l4virtio/virtio_input.h

15.407 l4virtio_input_config_t Struct Reference

Device configuration for input devices.

```
#include <virtio_input.h>
```

Collaboration diagram for l4virtio_input_config_t:



15.407.1 Detailed Description

Device configuration for input devices.

Definition at line 67 of file [virtio_input.h](#).

The documentation for this struct was generated from the following file:

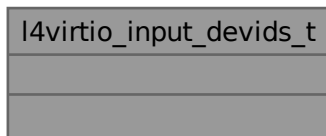
- l4/l4virtio/virtio_input.h

15.408 l4virtio_input_devids_t Struct Reference

Device ID information for the device.

```
#include <virtio_input.h>
```

Collaboration diagram for l4virtio_input_devids_t:



15.408.1 Detailed Description

Device ID information for the device.

Definition at line 56 of file [virtio_input.h](#).

The documentation for this struct was generated from the following file:

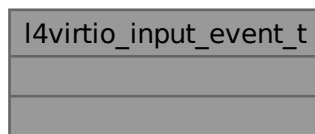
- l4/l4virtio/virtio_input.h

15.409 l4virtio_input_event_t Struct Reference

Single event in event or status queue.

```
#include <virtio_input.h>
```

Collaboration diagram for l4virtio_input_event_t:



15.409.1 Detailed Description

Single event in event or status queue.

Definition at line 85 of file [virtio_input.h](#).

The documentation for this struct was generated from the following file:

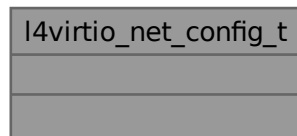
- l4/l4virtio/virtio_input.h

15.410 l4virtio_net_config_t Struct Reference

Device configuration for network devices.

```
#include <virtio_net.h>
```

Collaboration diagram for l4virtio_net_config_t:



15.410.1 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:

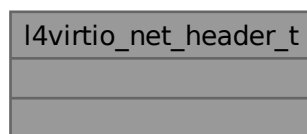
- l4/l4virtio/virtio_net.h

15.411 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:



15.411.1 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): Jakub Jermar <jakub.jermar@kernkonzept.com>
00005  */
00006
00007 #pragma once
00008
00009 #include <l4/drivers/asm_access_gen.h>

```

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/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 }

```


16.8 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 }

```

16.9 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 }

```

16.10 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 CHILD>
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         CHILD const *c = static_cast<CHILD 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 #define REGBLK_READ_TEMPLATE(sz) \
00128     template< typename T > \
00129     typename cxx::enable_if<sizeof(T) == (sz / 8), T>::type read(l4_addr_t reg) const \
00130     { \
00131         union X { T t; l4_uint##sz##_t v; } m; \
00132         m.v = _b->do_read_##sz (reg); \
00133         return m.t; \
00134     }
00135
00136 #define REGBLK_WRITE_TEMPLATE(sz) \
00137     template< typename T > \
00138     void write(T value, l4_addr_t reg, typename cxx::enable_if<sizeof(T) == (sz / 8), T>::type = T()) \
00139     const \
00140     { \
00141         union X { T t; l4_uint##sz##_t v; } m; \
00142         m.t = value; \
00143         _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 with 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 /**
00313  * \brief Handles a reference to a register block of the given
00314  *       maximum access width.
00315  * \tparam MAX_BITS Maximum access width for the registers in this
00316  *       block.
00317  * \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  * \Provides read only access to registers in this block via r<WIDTH>()
00393  * and operator[]().
00394  */
00395 template<
00396     unsigned MAX_BITS,
00397     typename BLOCK = Register_block_tmpl<
00398         Register_block_base<MAX_BITS> const
00399     >
00400 >
00401 class Ro_register_block
00402 {
00403 private:
00404     template< unsigned B, typename BLK > friend class Ro_register_block;
00405     typedef BLOCK Block;
00406     Block _b;
00407
00408 public:
00409     Ro_register_block() = default;
00410     Ro_register_block(BLOCK const &blk) : _b(blk) {}
00411
00412     template< unsigned BITS >
00413     Ro_register_block(Register_block<BITS> const &blk) : _b(blk._b) {}
00414
00415
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  * \brief Implementation helper for register blocks.
00441  * \param BASE The class implementing read<> and write<> template functions
00442  *           for accessing the registers. This class must inherit from
00443  *           Register_block_impl.
00444  * \param MAX_BITS The maximum access width for the register file.
00445  *           Supported values are 8, 16, 32, or 64.
00446  *
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 }

```

16.11 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:
00026         virtual unsigned char  read8(unsigned long reg) const = 0;
00027
00031         virtual unsigned short read16(unsigned long reg) const = 0;
00032
00036         virtual unsigned int   read32(unsigned long reg) const = 0;
00037
00038         /*
00039          * \brief Read register with an 8 byte access.
00040          */
00041         //virtual unsigned long long read64(unsigned long reg) const = 0;
00042
00046         virtual void write8(unsigned long reg, unsigned char value) const = 0;
00047
00051         virtual void write16(unsigned long reg, unsigned short value) const = 0;
00052
00056         virtual void write32(unsigned long reg, unsigned int value) const = 0;
00057
00058         /*
00059          * \brief Write register with an 8 byte access.
00060          */
00061         //virtual void write64(unsigned long reg, unsigned long long value) const = 0;
00062
00066         virtual unsigned long addr(unsigned long reg) const = 0;
00067
00073         virtual void delay() const = 0;
00074
00075         virtual ~Io_register_block() = 0;
00076
00084         template< typename R >
00085         R read(unsigned long reg) const
00086         {
00087             switch (sizeof(R))
00088             {
00089                 case 1: return read8(reg);
00090                 case 2: return read16(reg);
00091                 case 4: return read32(reg);
00092                 default: static_assert(sizeof(R) == 1 || sizeof(R) == 2 || sizeof(R) == 4,
00093                                     "Invalid size");
00094             };
00095         }
00096
00104         template< typename R >
00105         void write(unsigned long reg, R value) const
00106         {
00107             switch (sizeof(R))
00108             {
00109                 case 1: write8(reg, value); return;
00110                 case 2: write16(reg, value); return;
00111                 case 4: write32(reg, value); return;
00112                 default: static_assert(sizeof(R) == 1 || sizeof(R) == 2 || sizeof(R) == 4,
00113                                     "Invalid size");
00114             };
00115         }
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 }

```

16.12 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 }

```

16.13 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
00083   bool timed_out() const { return _c == 0; }
00084
00085 private:
00086   unsigned _c;
00087 };
00088
00089 }

```

16.14 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 }

```

16.15 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 }

```

16.16 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     protected:
00065         template <typename Uart_driver>
00066         int generic_write(char const *s, unsigned long count,
00067                          bool blocking = true) const
00068         {
00069             auto *self = static_cast<Uart_driver const*>(this);
00070

```

```

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 }

```

16.17 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 };

```

16.18 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 };

```

16.19 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 };

```

16.20 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 };

```

16.21 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 };

```

16.22 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 };

```

16.23 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 };

```

16.24 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 };

```

16.25 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 };

```

16.26 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 };

```

16.27 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 };

```

```

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 };

```

16.28 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 };

```

16.29 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) {}

```

```

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 enable) 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 private:
00037     void set_rate(Baud_rate r);
00038     unsigned _freq;
00039 };
00040 };

```

16.30 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
00029     public:
00030         explicit Uart_s3c(Uart_type type) : _type(type) {}
00031         explicit Uart_s3c(Uart_type type, unsigned /*base_rate*/) : _type(type) {}
00032         bool startup(Io_register_block const *) override;
00033         void shutdown() override;
00034         bool change_mode(Transfer_mode m, Baud_rate r) override;
00035         int get_char(bool blocking = true) const override;
00036         int char_avail() const override;
00037         int tx_avail() const;
00038         void wait_tx_done() const;
00039         inline void out_char(char c) const;
00040         int write(char const *s, unsigned long count,
00041                  bool blocking = true) const override;
00042         void fifo_reset();
00043
00044     protected:
00045         virtual void ack_rx_irq() const = 0;
00046         virtual void wait_for_empty_tx_fifo() const = 0;
00047         virtual unsigned is_rx_fifo_non_empty() const = 0;
00048         virtual unsigned is_tx_fifo_not_full() const = 0;
00049
00050     private:
00051         Uart_type _type;
00052     };
00053
00054     class Uart_s3c2410 : public Uart_s3c
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 };

```

16.31 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 };

```

16.32 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  };

```

16.33 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

```

16.34 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

```

16.35 Makefile

```

00001 PKGDIR = ../..
00002 L4DIR ?= $(PKGDIR)/../..
00003
00004 PKGNAME = drivers
00005
00006 include $(L4DIR)/mk/include.mk

```

16.36 Makefile

```

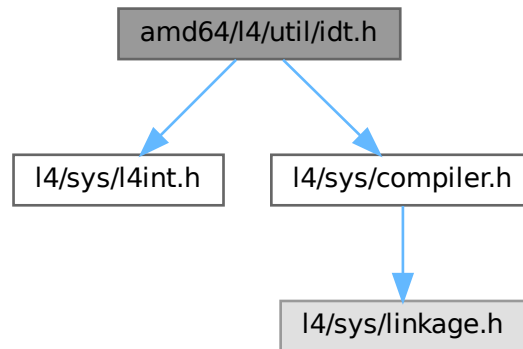
00001 PKGDIR = ../..
00002 L4DIR ?= $(PKGDIR)/../..
00003
00004 EXTRA_TARGET += cmd_control
00005
00006 include $(L4DIR)/mk/include.mk

```

16.37 amd64/I4/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.

16.37.1 Detailed Description

IDT related functions.

Date

2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [idt.h](#).

16.38 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

```

16.39 x86/l4/util/idt.h File Reference

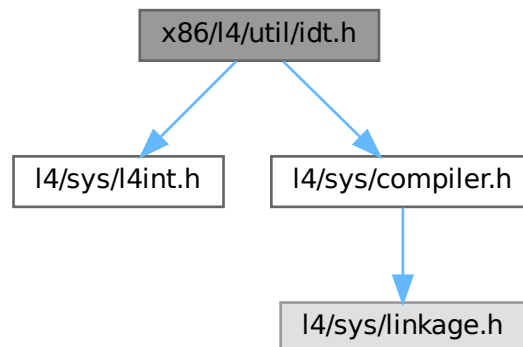
IDT related functions.

```

#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.

16.39.1 Detailed Description

IDT related functions.

Date

2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [idt.h](#).

16.40 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

```

16.41 amd64/l4/util/perform.h File Reference

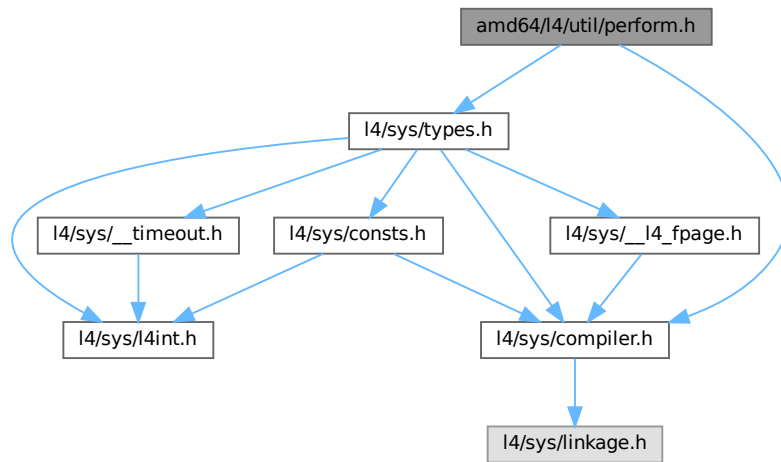
Performance Monitoring using P5/P6 Measurement Counters.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for perform.h:



16.41.1 Detailed Description

Performance Monitoring using P5/P6 Measurement Counters.

Define either `CPU_PENTIUM` or `CPU_P6`

Definition in file [perform.h](#).

16.42 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

```


Generated for L4Re by Doxygen

```

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

```

16.43 x86/i4/util/perform.h File Reference

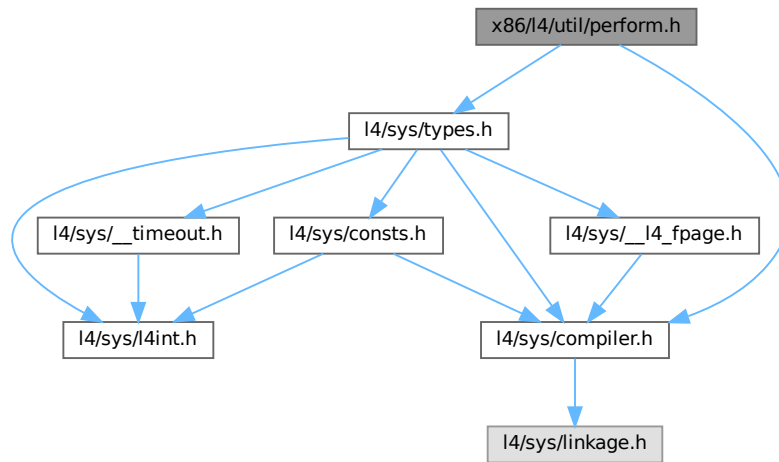
Perfomance Monitoring using P5/P6 Measurement Counters.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for perform.h:



16.43.1 Detailed Description

Performance Monitoring using P5/P6 Measurement Counters.

Define either `CPU_PENTIUM` or `CPU_P6`

Definition in file [perform.h](#).

16.44 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

```

16.45 amd64/l4/util/rdtsc.h File Reference

Timestamp counter related functions.

```

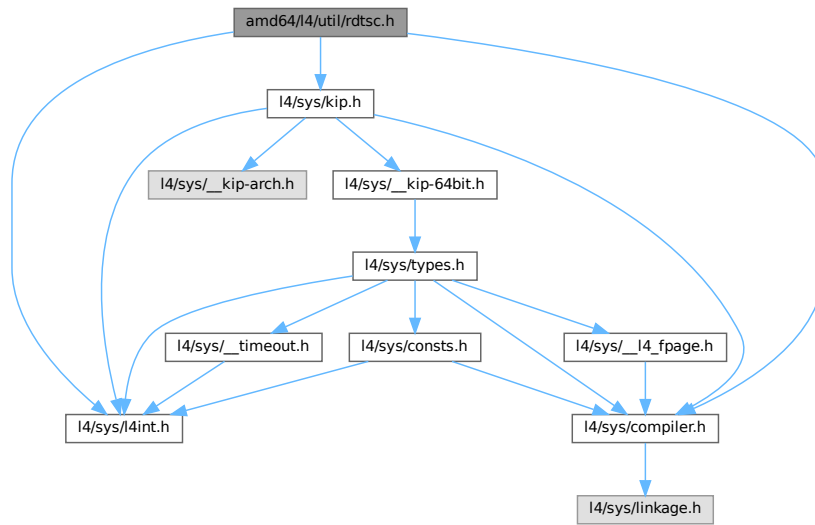
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>

```



```
#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.

16.45.1 Detailed Description

Timestamp counter related functions.

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [rdtsc.h](#).

16.46 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

```

16.47 x86/l4/util/rdtsc.h File Reference

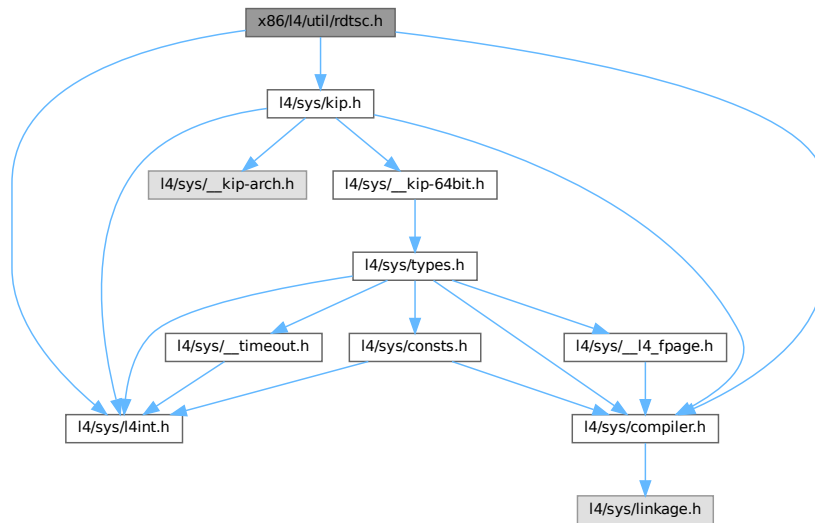
Timestamp counter related functions.

```

#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.

16.47.1 Detailed Description

Timestamp counter related functions.

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [rdtsc.h](#).

16.48 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_INLINE 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     ("        \n\t"
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     ("        \n\t"
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     ("        \n\t"
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

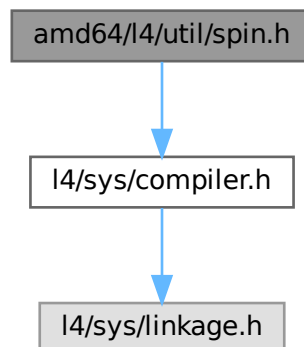
```

16.49 amd64/l4/util/spin.h File Reference

Spinning for amd64.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for spin.h:



16.49.1 Detailed Description

Spinning for amd64.

Definition in file [spin.h](#).

16.50 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

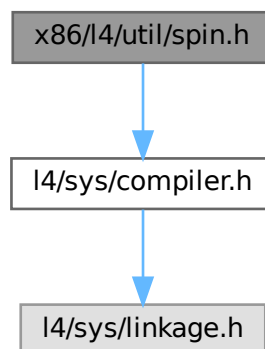
```

16.51 x86/l4/util/spin.h File Reference

Spinning for x86.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for spin.h:



16.51.1 Detailed Description

Spinning for x86.

Definition in file [spin.h](#).

16.52 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

```

16.53 amd64/l4/util/util.h File Reference

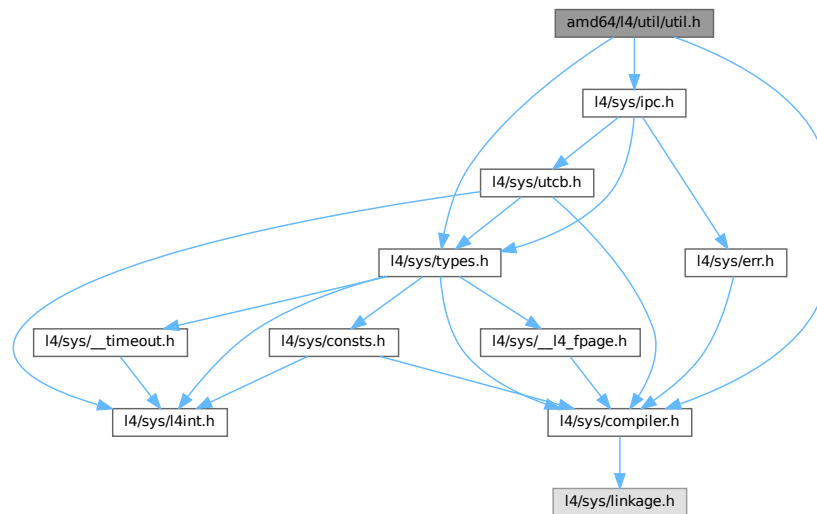
Utilities, amd64 version.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for util.h:



Functions

- [l4_timeout_s](#) [l4util_micros2l4to](#) (unsigned int mus) [L4_NOTHROW](#)
Calculate l4 timeouts.
- void [l4_sleep](#) (int ms) [L4_NOTHROW](#)
Suspend thread for a period of ms milliseconds.
- void [l4_sleep_forever](#) (void) [L4_NOTHROW](#)
Go sleep and never wake up.

16.53.1 Detailed Description

Utilities, amd64 version.

Definition in file [util.h](#).

16.53.2 Function Documentation

16.53.2.1 l4_sleep()

```
void l4_sleep (
    int ms )
```

Suspend thread for a period of *ms* milliseconds.

Parameters

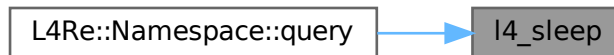
<i>ms</i>	Time in milliseconds
-----------	----------------------

Examples

[examples/libs/libirq/async_isr.c](#), [examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/](#)

Referenced by [L4Re::Namespace::query\(\)](#).

Here is the caller graph for this function:



16.53.2.2 l4util_micros2l4to()

```
l4_timeout_s l4util_micros2l4to (
    unsigned int mus )
```

Calculate l4 timeouts.

Parameters

<i>mus</i>	time in microseconds. Special cases: <ul style="list-style-type: none"> • 0 -> timeout 0 • ~0U -> timeout NEVER
------------	---

Returns

the corresponding l4_timeout value

Deprecated Use [l4_timeout_from_us\(\)](#).

16.54 util.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  * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00006  * economic rights: Technische Universität Dresden (Germany)
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 #ifndef __UTIL_H
```

```

00015 #define __UTIL_H
00016
00017 #include <l4/sys/types.h>
00018 #include <l4/sys/compiler.h>
00019 #include <l4/sys/ipc.h>
00020
00021 EXTERN_C_BEGIN
00022
00031 L4_CV l4_timeout_s l4util_micros2l4to(unsigned int mus) L4_NOTHROW;
00032
00036 L4_CV void l4_sleep(int ms) L4_NOTHROW;
00037
00038 /* Suspend thread for a period of \a us microseconds.
00039  * \param us Time in microseconds
00040  * WARNING: This function is mostly bogus since the timer resolution of
00041  * current L4 implementations is about lms! */
00042 L4_CV void l4_usleep(int us) L4_NOTHROW;
00043
00049 L4_INLINE void l4_sleep_forever(void) L4_NOTHROW __attribute__((noreturn));
00050
00051 L4_INLINE void
00052 l4_sleep_forever(void) L4_NOTHROW
00053 {
00054     for (;;)
00055         l4_ipc_sleep(L4_IPC_NEVER);
00056 }
00057
00059 static inline void
00060 l4_touch_ro(const void*addr, unsigned size) L4_NOTHROW
00061 {
00062     const char *bptr, *eptr;
00063
00064     bptr = (const char*)((l4_addr_t)addr) & L4_PAGEMASK;
00065     eptr = (const char*)((l4_addr_t)addr+size-1) & L4_PAGEMASK;
00066     for(;bptr<=eptr;bptr+=L4_PAGESIZE){
00067         asm volatile("or %0,%rax \n"
00068                     :
00069                     : "m" (*(const unsigned*)bptr)
00070                     : "rax" );
00071     }
00072 }
00073
00074
00076 static inline void
00077 l4_touch_rw(const void*addr, unsigned size) L4_NOTHROW
00078 {
00079     const char *bptr, *eptr;
00080
00081     bptr = (const char*)((l4_addr_t)addr) & L4_PAGEMASK;
00082     eptr = (const char*)((l4_addr_t)addr+size-1) & L4_PAGEMASK;
00083     for(;bptr<=eptr;bptr+=L4_PAGESIZE){
00084         asm volatile("orb $0,%0 \n"
00085                     :
00086                     : "m" (*(const unsigned*)bptr)
00087                     );
00088     }
00089 }
00090
00091 EXTERN_C_END
00092
00093 #endif
00094

```

16.55 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(unsigned int mus) L4_NOTHROW;
00037
00043 L4_CV void l4_sleep(int ms) L4_NOTHROW;
00044
00053 L4_CV void l4_usleep(int us) L4_NOTHROW;
00054
00060 L4_INLINE void l4_sleep_forever(void) L4_NOTHROW L4_NORETURN;
00061
00069 L4_INLINE void
00070 l4_touch_ro(const void *addr, unsigned size) L4_NOTHROW;
00071
00079 L4_INLINE void
00080 l4_touch_rw(const void *addr, unsigned size) L4_NOTHROW;
00081
00082
00083
00084 /*
00085  * Implementations
00086  */
00087
00088 L4_INLINE void
00089 l4_sleep_forever(void) L4_NOTHROW
00090 {
00091     for (;;)
00092         l4_ipc_sleep(L4_IPC_NEVER);
00093 }
00094
00095 L4_INLINE void
00096 l4_touch_ro(const void *addr, unsigned size) L4_NOTHROW
00097 {
00098     l4_addr_t b, e;
00099
00100     b = l4_trunc_page((l4_addr_t)addr);
00101     e = l4_trunc_page((l4_addr_t)addr + size - 1);
00102
00103     for (; b <= e; b += L4_PAGESIZE)
00104         (void)(*(volatile char *)b);
00105 }
00106
00107
00108 L4_INLINE void
00109 l4_touch_rw(const void *addr, unsigned size) L4_NOTHROW
00110 {
00111     l4_addr_t b, e;
00112
00113     b = l4_trunc_page((l4_addr_t)addr);
00114     e = l4_trunc_page((l4_addr_t)addr + size - 1);
00115
00116     for (; b <= e; b += L4_PAGESIZE)
00117     {
00118         char x = *(volatile char *)b;
00119         *(volatile char *)b = x;
00120     }
00121 }
00122
00123 EXTERN_C_END
00124
00125 #endif /* __L4UTIL__UTIL_H__ */

```

16.56 x86/I4/util/util.h File Reference

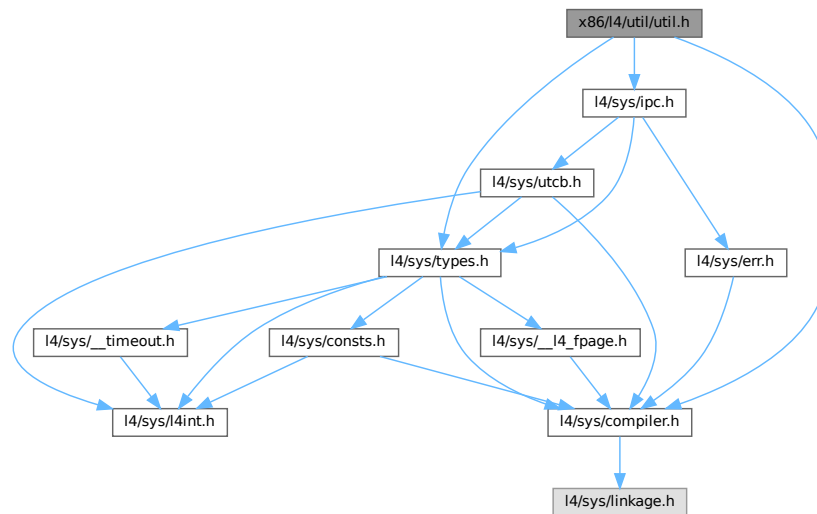
Utilities for x86.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for util.h:



Functions

- [l4_timeout_s l4util_micros2l4to](#) (unsigned int mus) [L4_NOTHROW](#)
Calculate l4 timeouts.
- void [l4_sleep](#) (int ms) [L4_NOTHROW](#)
Suspend thread for a period of ms milliseconds.
- void [l4_sleep_forever](#) (void) [L4_NOTHROW](#)
Go sleep and never wake up.

16.56.1 Detailed Description

Utilities for x86.

Definition in file [util.h](#).

16.56.2 Function Documentation

16.56.2.1 l4_sleep()

```
void l4_sleep (
    int ms )
```

Suspend thread for a period of ms milliseconds.

Parameters

<i>ms</i>	Time in milliseconds
-----------	----------------------

16.56.2.2 l4util_micros2l4to()

```
l4_timeout_s l4util_micros2l4to (
    unsigned int mus )
```

Calculate l4 timeouts.

Parameters

<i>mus</i>	time in microseconds. Special cases: <ul style="list-style-type: none"> • 0 -> timeout 0 • ~0U -> timeout NEVER
------------	---

Returns

the corresponding l4_timeout value

Deprecated Use l4_timeout_from_us().

16.57 util.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  *                Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00009  *                Jork Löser <jork@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 __UTIL_H
00016 #define __UTIL_H
00017
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/compiler.h>
00020 #include <l4/sys/ipc.h>
00021
00022 EXTERN_C_BEGIN
00023
00032 L4_CV l4_timeout_s l4util_micros2l4to(unsigned int mus) L4_NOTHROW;
00033
00037 L4_CV void l4_sleep(int ms) L4_NOTHROW;
00038
00039 /* Suspend thread for a period of \a us microseconds.
00040  * \param us Time in microseconds
00041  * WARNING: This function is mostly bogus since the timer resolution of
00042  *          current L4 implementations is about 1ms! */
00043 L4_CV void l4_usleep(int us) L4_NOTHROW;
00044
00050 L4_INLINE void l4_sleep_forever(void) L4_NOTHROW __attribute__((noreturn));
00051
00052 L4_INLINE void
00053 l4_sleep_forever(void) L4_NOTHROW
00054 {
00055     for (;;)
00056         l4_ipc_sleep(L4_IPC_NEVER);
00057 }
00058
00060 static inline void
00061 l4_touch_ro(const void*addr, unsigned size) L4_NOTHROW
00062 {
```

```

00063  const char *bptr, *eptr;
00064
00065  bptr = (const char*)((l4_addr_t)addr) & L4_PAGEMASK);
00066  eptr = (const char*)((l4_addr_t)addr+size-1) & L4_PAGEMASK);
00067  for(;bptr<=eptr;bptr+=L4_PAGESIZE){
00068      asm volatile("or  %0,%eax \n"
00069                  :
00070                  : "m" (*(const unsigned*)bptr)
00071                  : "eax" );
00072  }
00073 }
00074
00075
00077 static inline void
00078 l4_touch_rw(const void*addr, unsigned size) L4_NOTHROW
00079 {
00080     const char *bptr, *eptr;
00081
00082     bptr = (const char*)((l4_addr_t)addr) & L4_PAGEMASK);
00083     eptr = (const char*)((l4_addr_t)addr+size-1) & L4_PAGEMASK);
00084     for(;bptr<=eptr;bptr+=L4_PAGESIZE){
00085         asm volatile("orb $0,%0 \n"
00086                     :
00087                     : "m" (*(const unsigned*)bptr)
00088                     );
00089     }
00090 }
00091
00092 EXTERN_C_END
00093
00094 #endif
00095

```

16.58 amd64/l4/sys/segment.h File Reference

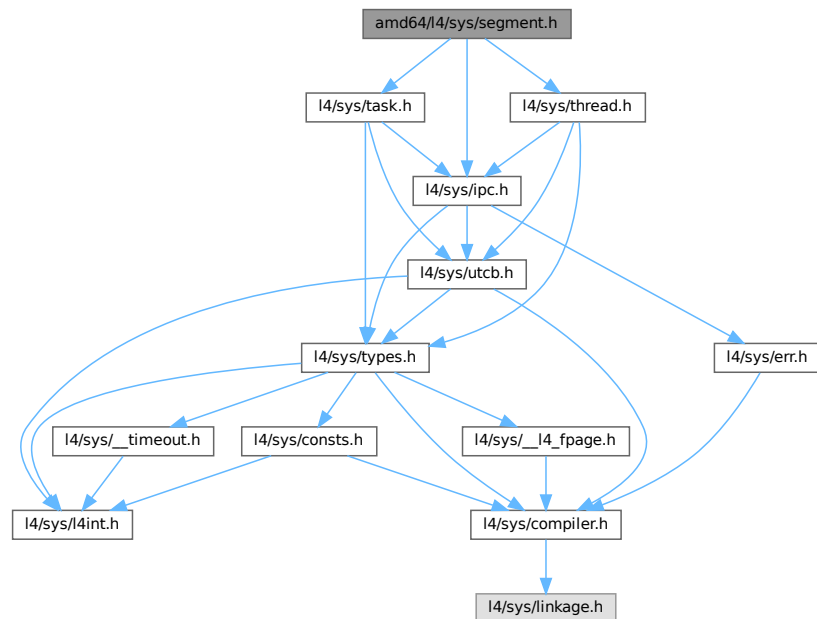
Segment handling.

```

#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.

16.58.1 Detailed Description

Segment handling.

Definition in file [segment.h](#).

16.58.2 Enumeration Type Documentation

16.58.2.1 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](#).

16.58.2.2 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](#).

16.58.3 Function Documentation

16.58.3.1 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 l4_utcb .

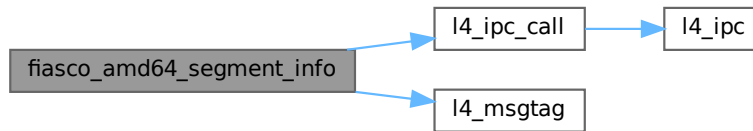
Returns

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:



16.58.3.2 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 l4_utcb .

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.

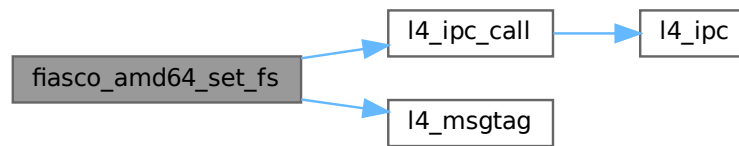
Note

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:



16.58.3.3 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 L4_sys_segment).
<i>base</i>	Base address.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb .

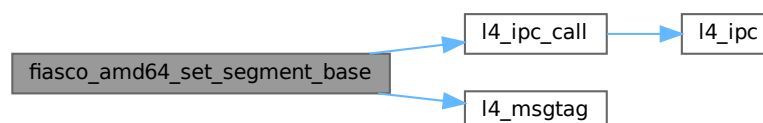
Return values

<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:



16.59 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);
00174 }
00175
00176 L4_INLINE unsigned
00177 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb)
00178 {
00179     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00180     if (l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER), utcb))
00181         return -1;
00182     return l4_utcb_mr_u(utcb)->mr[0];
00183 }
00184
00185 L4_INLINE long
00186 fiasco_amd64_segment_info(l4_cap_idx_t thread, unsigned *user_ds,
00187                           unsigned *user_cs, unsigned *user32_cs,
00188                           l4_utcb_t *utcb)
00189 {
00190     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00191     int r;
00192
00193     m->mr[0] = L4_THREAD_AMD64_GET_SEGMENT_INFO_OP;
00194
00195     r = l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0),
00196               L4_IPC_NEVER), utcb);
00197     if (r < 0)
00198         return r;
00199
00200     *user_ds = m->mr[0];
00201     *user_cs = m->mr[1];
00202     *user32_cs = m->mr[2];
00203
00204     return 0;
00205 }
00206
00207 #endif /* ! __L4_SYS__ARCH_X86__SEGMENT_H__ */

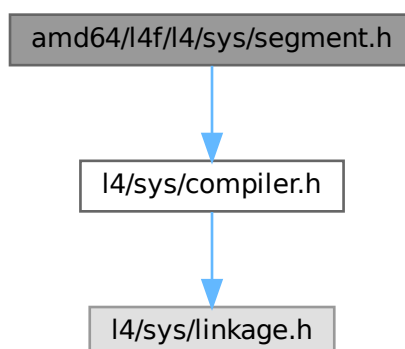
```

16.60 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.

16.60.1 Detailed Description

l4f specific fs/gs manipulation

Definition in file [segment.h](#).

16.60.2 Function Documentation

16.60.2.1 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 l4_utcb .

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.

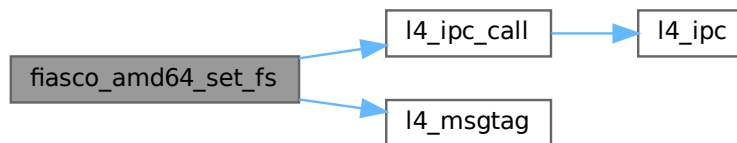
Note

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:



16.60.2.2 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 L4_sys_segment).
<i>base</i>	Base address.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb .

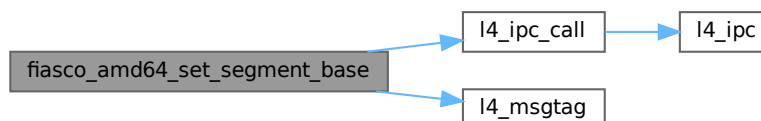
Return values

<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:



16.61 segment.h

[Go to the documentation of this file.](#)

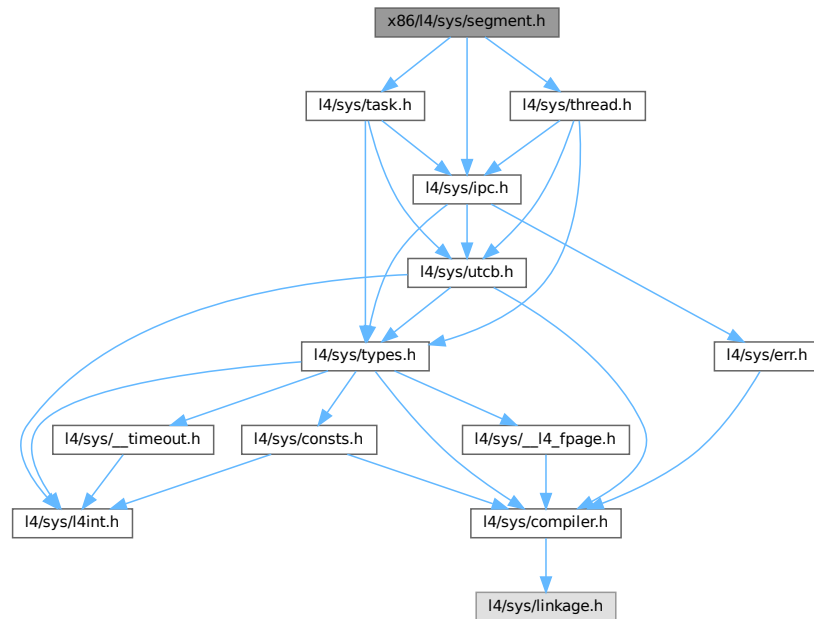
```
00001 #include_next <l4/sys/segment.h>
00002
00008 /*
00009  * (c) 2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public 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 /*****
00031  *** Implementation
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)
00047 {
00048     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_AMD64_SET_SEGMENT_BASE_OP | ((l4_umword_t)segr « 16);
00049     l4_utcb_mr_u(utcb)->mr[1] = base;
00050     return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER),
00051     utcb);
00052 }
00053
00054 L4_INLINE long
00055 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00056     unsigned int entry_number_start, l4_utcb_t *utcb)
00057 {
00058     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00059     l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00060     __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], desc, size);
00061     return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2 + (size / 8), 0, 0),
00062     L4_IPC_NEVER), utcb);
00063 }
00064
00065 #endif /* ! __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__ */
```

16.62 x86/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.

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.

16.62.1 Detailed Description

Segment handling.

Definition in file [segment.h](#).

16.62.2 Enumeration Type Documentation

16.62.2.1 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](#).

16.63 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
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);

```

```

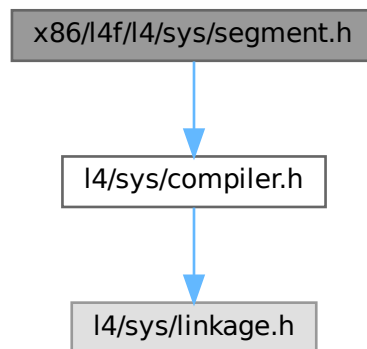
00113     return l4_error_u(l4_ipc_call(task, utcb, l4_msgtag(L4_PROTO_TASK, 2 + num_desc * 2, 0, 0),
00114                       L4_IPC_NEVER), utcb);
00114 }
00115
00116 L4_INLINE unsigned
00117 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb)
00118 {
00119     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00120     if (l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER), utcb))
00121         return -1;
00122     return l4_utcb_mr_u(utcb)->mr[0];
00123 }
00124
00125 #endif /* ! __L4_SYS__ARCH_X86__SEGMENT_H__ */

```

16.64 x86/l4f/l4/sys/segment.h File Reference

l4f 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.

16.64.1 Detailed Description

l4f specific segment manipulation

Definition in file [segment.h](#).

16.65 segment.h

[Go to the documentation of this file.](#)

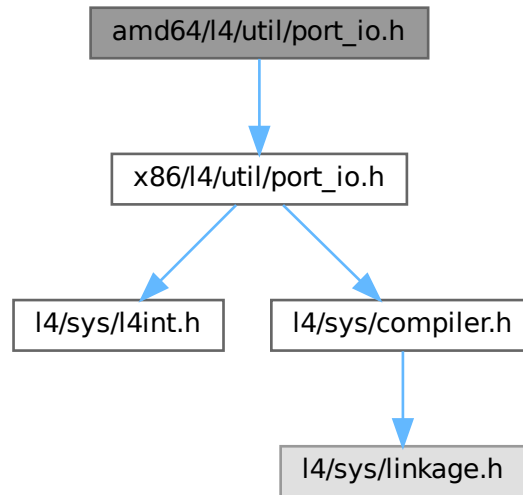
```
00001 #include_next <l4/sys/segment.h>
00002
00008 /*
00009  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General 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);
00042 }
00043
00044 #endif /* ! __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__ */
```

16.66 amd64/I4/util/port_io.h File Reference

Port I/O functions.


```
#include <x86/l4/util/port_io.h>
```

Include dependency graph for port_io.h:



16.66.1 Detailed Description

Port I/O functions.

Definition in file [port_io.h](#).

16.67 port_io.h

[Go to the documentation of this file.](#)

```

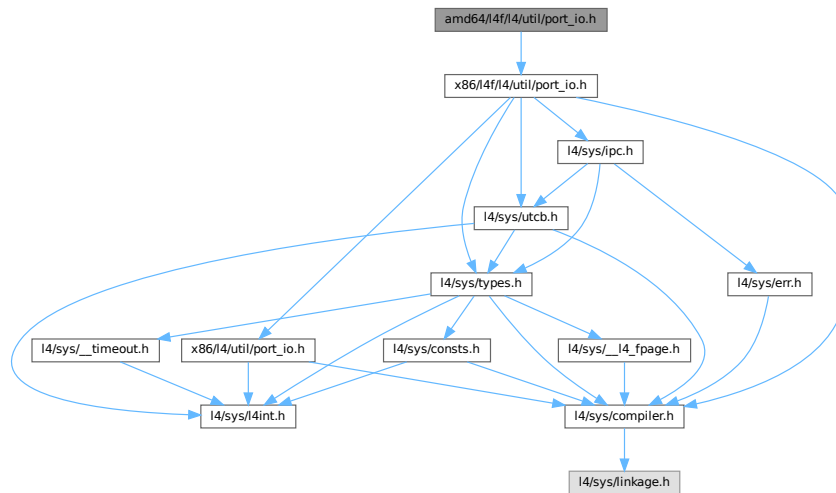
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012 #include <x86/l4/util/port_io.h>

```

16.68 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:
```



16.68.1 Detailed Description

Port I/O functions.

Definition in file [port_io.h](#).

16.69 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>

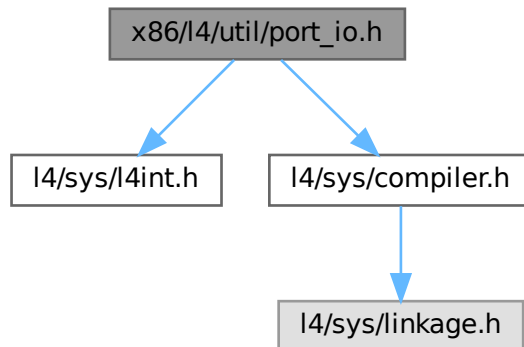
```

16.70 x86/l4/util/port_io.h File Reference

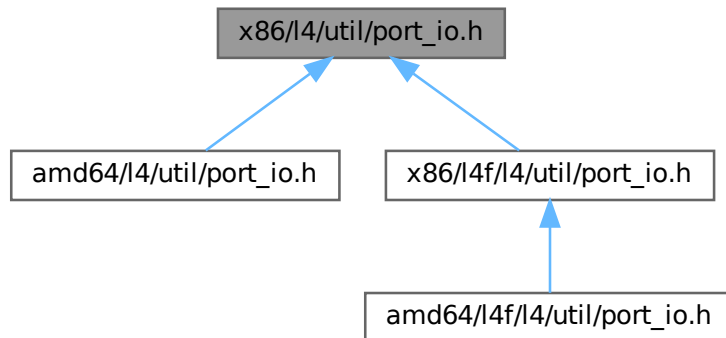
x86 port I/O

```
#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

16.70.1 Detailed Description

x86 port I/O

Date

06/2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [port_io.h](#).

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_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 %0, %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 %a1,$0x80");
00271 }
00272
00273 #endif

```

16.72 x86/I4f/I4/util/port_io.h File Reference

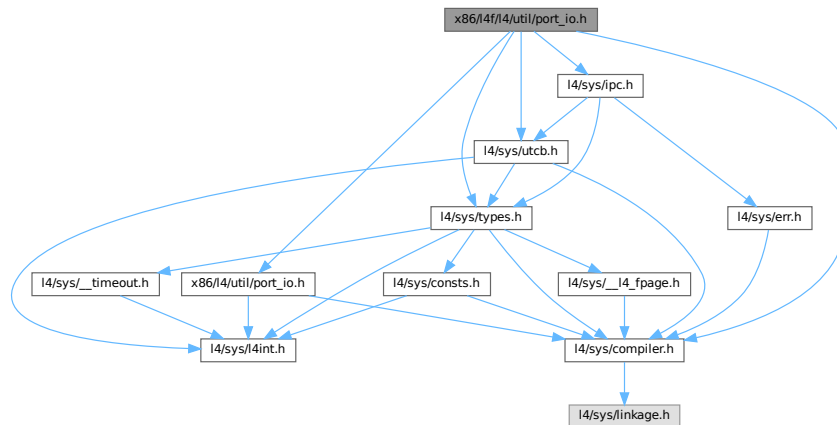
port I/O functions

```

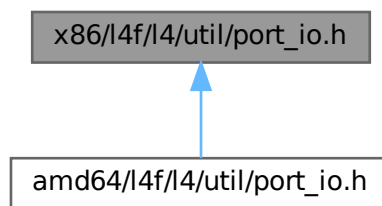
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <x86/l4/util/port_io.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for port_io.h:



This graph shows which files directly or indirectly include this file:



Functions

- int [i4util_ioport_map](#) ([i4_cap_idx_t](#) sigma0id, unsigned port_start, unsigned log2size)
Map a range of I/O ports.

16.72.1 Detailed Description

port I/O functions

Date

06/2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [port_io.h](#).

16.72.2 Function Documentation

16.72.2.1 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.

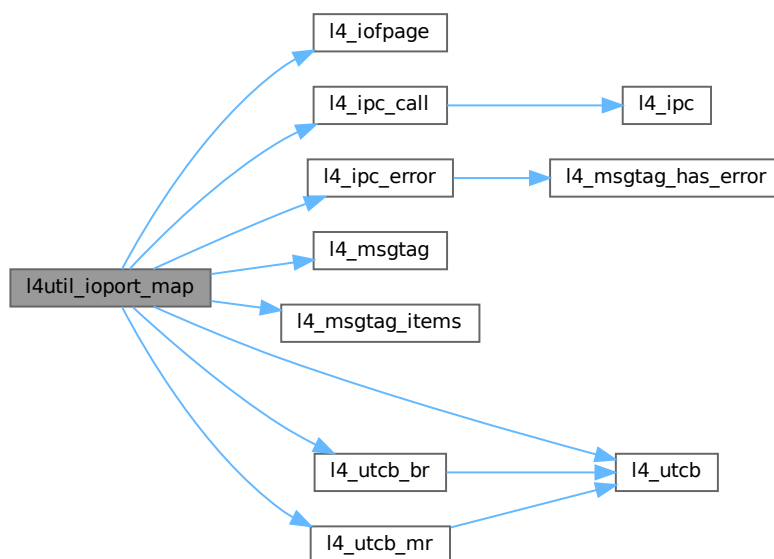
Returns

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:



16.73 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

```

16.74 __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 __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 };

```

16.76 __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 {};

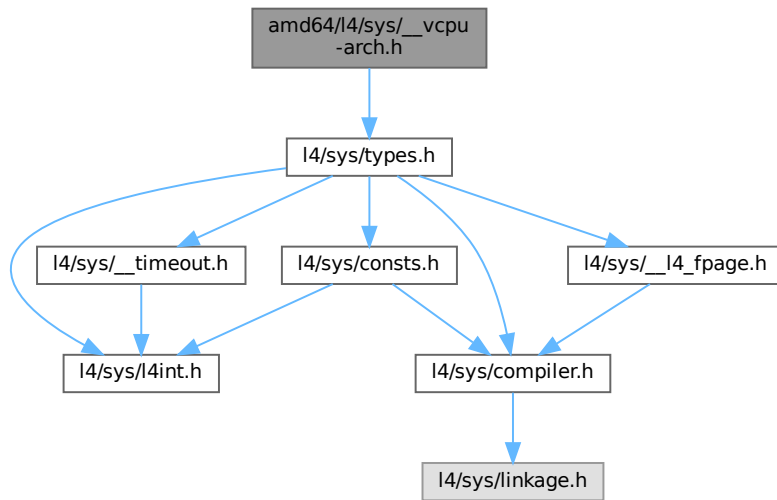
```

16.77 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](#) = 0x23 }

16.77.1 Detailed Description

AMD64-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 l4_vcpu_state_t structure after enabling vCPU mode or extended vCPU mode for a thread.
-----------------------	--

Definition at line 27 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
00027 enum
00028 {
00035     L4_VCPU_STATE_VERSION = 0x23
00036 };
00037
00041 typedef struct l4_vcpu_arch_state_t
00042 {
00043     l4_umword_t host_fs_base;
00044     l4_umword_t host_gs_base;
00045     l4_uint16_t host_ds, host_es, host_fs, host_gs;
00046
00047     l4_uint16_t const user_ds32;
00048     l4_uint16_t const user_cs64;
00049     l4_uint16_t const user_cs32;
00050 } l4_vcpu_arch_state_t;
00051
00052
00057 typedef struct l4_vcpu_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 di;
00068     l4_umword_t si;
00069     l4_umword_t bp;
00070     l4_umword_t pfa;
00071     l4_umword_t bx;
00072     l4_umword_t dx;

```

```

00074  l4_umword_t cx;
00075  l4_umword_t ax;
00077  l4_umword_t trapno;
00078  l4_umword_t err;
00080  l4_umword_t ip;
00081  l4_umword_t cs;
00082  l4_umword_t flags;
00083  l4_umword_t sp;
00084  l4_umword_t ss;
00085  l4_umword_t fs_base;
00086  l4_umword_t gs_base;
00087  l4_uint16_t ds, es, fs, gs;
00088
00089 } l4_vcpu_regs_t;
00090
00095 typedef struct l4_vcpu_ipc_regs_t
00096 {
00097     l4_umword_t _res[1];
00098     l4_umword_t label;
00099     l4_umword_t _res2[5];
00100     l4_msgtag_t tag;
00101 } l4_vcpu_ipc_regs_t;

```

16.79 arm/l4/sys/__vcpu-arch.h File Reference

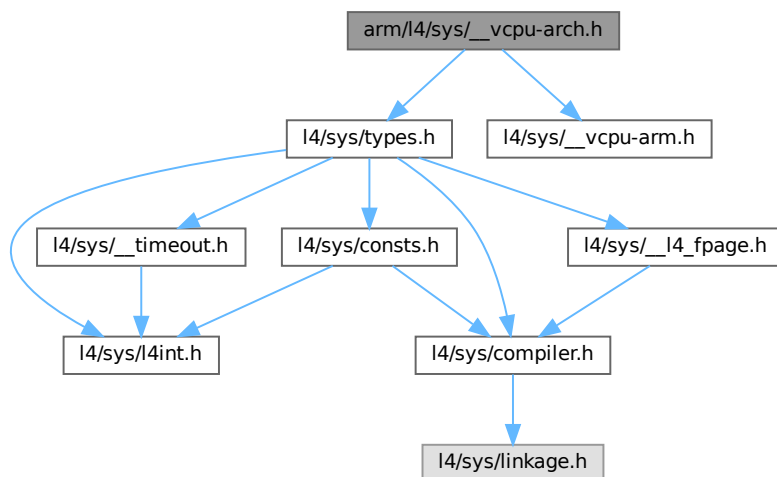
ARM-specific vCPU interface.

```

#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](#) = 0x35 }
- enum [L4_vcpu_e_field_ids](#)
IDs for extended vCPU state fields.

16.79.1 Detailed Description

ARM-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 l4_vcpu_state_t structure after enabling vCPU mode or extended vCPU mode for a thread.
---------------------------------------	--

Definition at line 28 of file [__vcpu-arch.h](#).

16.79.2.2 [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 87 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 #include <l4/sys/__vcpu-arm.h>
00027
00028 enum
00029 {
00036     L4_VCPU_STATE_VERSION = 0x35
00037 };
00038
00043 typedef struct l4_vcpu_regs_t
00044 {
00045     l4_umword_t pfa;
00046     l4_umword_t err;
00047
00048     l4_umword_t r[13];
00049
00050     l4_umword_t sp;
00051     l4_umword_t lr;
00052     l4_umword_t _dummy;
00053     l4_umword_t ip;
00054     l4_umword_t flags;
00055     l4_umword_t tpidrro;
00056     l4_umword_t tpidrurw;
00057 } l4_vcpu_regs_t;
00058
00062 typedef struct l4_vcpu_arch_state_t
00063 {
00064     l4_umword_t host_tpidrro;
00065 } l4_vcpu_arch_state_t;
00066
00071 typedef struct l4_vcpu_ipc_regs_t
00072 {
00073     l4_msgtag_t tag;
00074     l4_umword_t _d1[3];
00075     l4_umword_t label;
00076     l4_umword_t _d2[8];
00077 } l4_vcpu_ipc_regs_t;
00078
00087 enum L4_vcpu_e_field_ids
00088 {
00089     L4_VCPU_E_HCR           = 0x8008,
00090     L4_VCPU_E_TTBR0        = 0x8010,
00091     L4_VCPU_E_TTBR1        = 0x8018,
00092     L4_VCPU_E_TTBCR        = 0x0020,
00093     L4_VCPU_E_SCTLR        = 0x0024,
00094     L4_VCPU_E_DACR         = 0x0028,
00095     L4_VCPU_E_FCSEIDR      = 0x002c,
00096
00097     L4_VCPU_E_CNTVCTL       = 0x0030,
00098     L4_VCPU_E_CNTVOFF      = 0x8038,
00099
00100     L4_VCPU_E_VMPIDR       = 0x0040,
00101     L4_VCPU_E_VPIDR        = 0x0044,
00102
00103     L4_VCPU_E_GIC_HCR      = 0x0050,
00104     L4_VCPU_E_GIC_VTR      = 0x0054,
00105     L4_VCPU_E_GIC_VMCR     = 0x0058,
00106     L4_VCPU_E_GIC_MISR     = 0x005c,
00107     L4_VCPU_E_GIC_EISR     = 0x0060,
00108     L4_VCPU_E_GIC_ELSR     = 0x0064,
00109     L4_VCPU_E_GIC_V2_LR0   = 0x0068,
00110     L4_VCPU_E_GIC_V3_LR0   = 0x8068,
00111 };

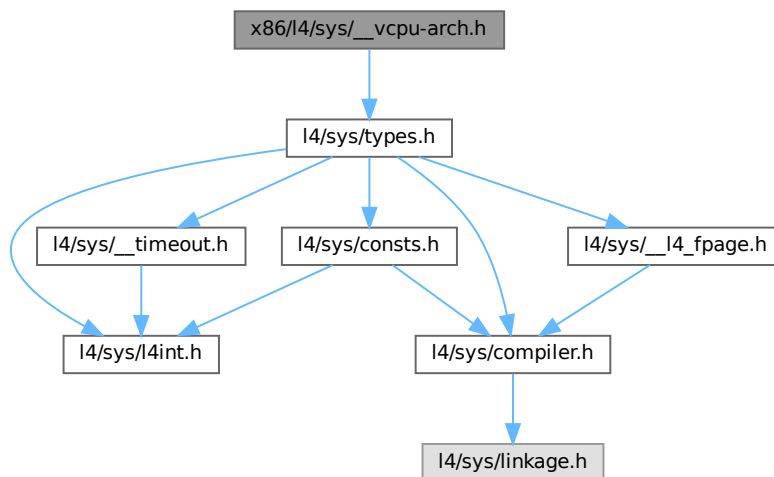
```

16.81 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` = 0x43 }

16.81.1 Detailed Description

x86-specific vCPU interface.

Definition in file [__vcpu-arch.h](#).

16.81.2 Enumeration Type Documentation

16.81.2.1 anonymous enum

anonymous enum

Enumerator

L4_VCPU_STATE_VERSION	Architecture-specific version ID. This ID must match the version field in the l4_vcpu_state_t structure after enabling vCPU mode or extended vCPU mode for a thread.
-----------------------	--

Definition at line 27 of file [__vcpu-arch.h](#).

16.82 __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 = 0x43
00036 };
00037
00042 typedef struct l4_vcpu_regs_t
00043 {
00044     l4_umword_t es;
00045     l4_umword_t ds;
00046     l4_umword_t gs;
00047     l4_umword_t fs;
00049     l4_umword_t di;
00050     l4_umword_t si;
00051     l4_umword_t bp;
00052     l4_umword_t pfa;
00053     l4_umword_t bx;
00054     l4_umword_t dx;
00055     l4_umword_t cx;
00056     l4_umword_t ax;
00058     l4_umword_t trapno;

```

```

00059     l4_umword_t err;
00061     l4_umword_t ip;
00062     l4_umword_t dummy1;
00063     l4_umword_t flags;
00064     l4_umword_t sp;
00065     l4_umword_t ss;
00066 } l4_vcpu_regs_t;
00067
00071 typedef struct l4_vcpu_arch_state_t {} l4_vcpu_arch_state_t;
00072
00077 typedef struct l4_vcpu_ipc_regs_t
00078 {
00079     l4_umword_t _res[2];
00080     l4_umword_t label;
00081     l4_umword_t _res2[3];
00082     l4_msgtag_t tag;
00083 } l4_vcpu_ipc_regs_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+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;

```

16.84 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;

```

16.85 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         struct __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 } exregs; /* 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;

```

16.86 amd64/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  *      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__ */

```

16.88 arm/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  *      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__ */

```

16.90 x86/l4/sys/linkage.h File Reference

Linkage.

Macros

- `#define L4_CV`

Define calling convention.

16.90.1 Detailed Description

Linkage.

Definition in file [linkage.h](#).

16.91 linkage.h

[Go to the documentation of this file.](#)

```

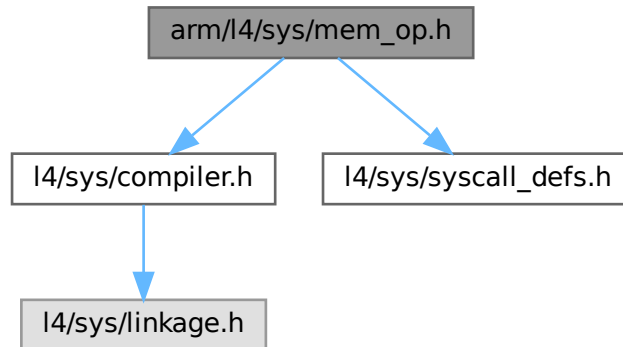
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  * Frank Mehnert <fm3@os.inf.tu-dresden.de>
00010  * economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS 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__ */

```

16.92 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.

16.92.1 Detailed Description

Memory access functions (ARM specific)

Date

2010-10

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de

Definition in file `mem_op.h`.

16.93 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    lr, pc          \n\t"
00120      "mov    pc, %[sc]       \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", "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__ */

```

16.94 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>

```

16.95 arm/l4/sys/vm.h File Reference

ARM virtualization interface.

Data Structures

- struct [l4_vm_tz_state](#)
state structure for TrustZone VMs

16.95.1 Detailed Description

ARM virtualization interface.

Definition in file [vm.h](#).

16.96 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 }

```

16.97 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>

```

16.98 amd64/l4/util/bitops_arch.h File Reference

amd64 bit manipulation functions

16.98.1 Detailed Description

amd64 bit manipulation functions

Definition in file [bitops_arch.h](#).

16.99 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_bsf(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

```

16.100 arm/l4/util/bitops_arch.h File Reference

ARM specific implementation of bitops functions.

16.100.1 Detailed Description

ARM specific implementation of bitops functions.

Definition in file [bitops_arch.h](#).

16.101 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__ */
```

16.102 x86/l4/util/bitops_arch.h File Reference

x86 bit manipulation functions

16.102.1 Detailed Description

x86 bit manipulation functions

Definition in file [bitops_arch.h](#).

16.103 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_btc(int b, volatile l4_umword_t * dest)
00140 {
00141     l4_int8_t bit;
00142
00143     __asm__ __volatile__
00144     (
00145         "lock; btc1 %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

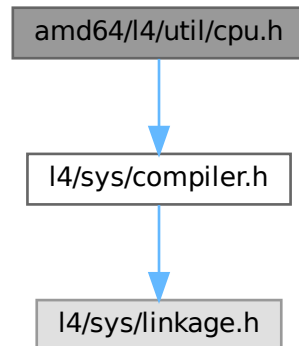
```

16.104 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.

16.104.1 Detailed Description

CPU related functions.

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [cpu.h](#).

16.105 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

```

16.106 arm/l4/util/cpu.h File Reference

CPU related functions.

16.106.1 Detailed Description

CPU related functions.

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de

Definition in file [cpu.h](#).

16.107 cpu.h

[Go to the documentation of this file.](#)

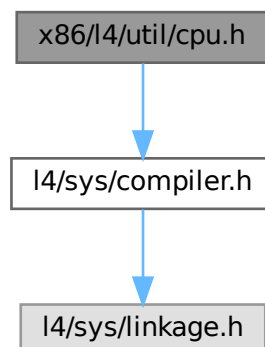
```
00001
00008 /*
00009  * (c) 2004-2009 Author(s)
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
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__ */
```

16.108 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_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.

16.108.1 Detailed Description

CPU related functions.

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [cpu.h](#).

16.109 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_nocheck(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

```

16.110 amd64/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

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_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 */
```

16.112 arm/l4/util/l4_macros.h File Reference

Main function.

16.112.1 Detailed Description

Main function.

Date

08/29/2000

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [l4_macros.h](#).

16.113 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 */
```


16.114 l4/util/l4_macros.h File Reference

Some useful generic macros, L4f version.

16.114.1 Detailed Description

Some useful generic macros, L4f version.

Date

11/12/2002

Author

Lars Reuther reuther@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) 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
```

16.116 x86/l4/util/l4_macros.h File Reference

Main function.

16.116.1 Detailed Description

Main function.

Date

08/29/2000

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [l4_macros.h](#).

16.117 l4_macros.h

[Go to the documentation of this file.](#)

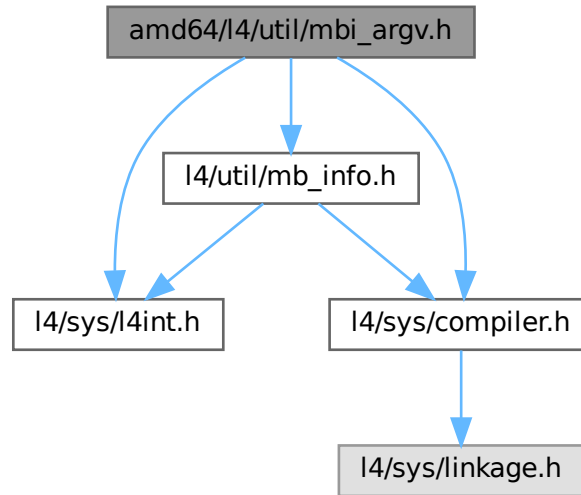
```
00001
00008 /*
00009  * (c) 2006-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
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 */
```

16.118 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:
```



16.118.1 Detailed Description

command line handling

Date

2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [mbi_argv.h](#).

16.119 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

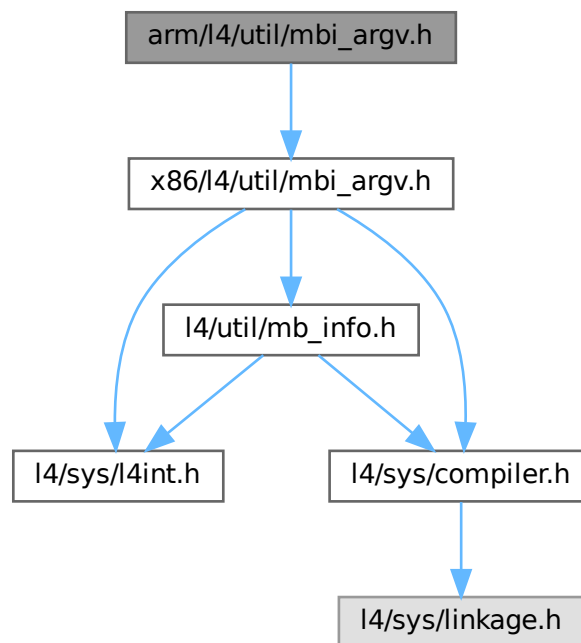
```

16.120 arm/l4/util/mbi_argv.h File Reference

Multiboot.

```
#include <x86/l4/util/mbi_argv.h>
```

Include dependency graph for mbi_argv.h:



16.120.1 Detailed Description

Multiboot.

Definition in file [mbi_argv.h](#).

16.121 mbi_argv.h

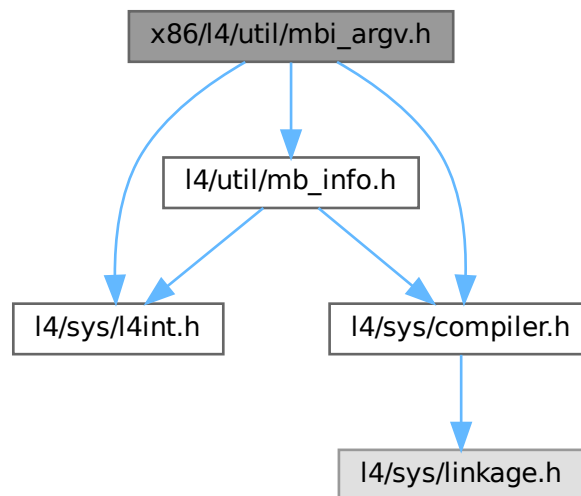
[Go to the documentation of this file.](#)

```
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>
```

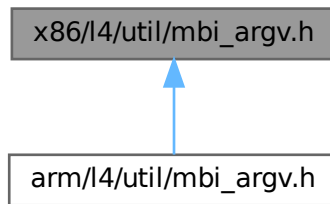
16.122 x86/l4/util/mbi_argv.h File Reference

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:



16.122.1 Detailed Description

command line handling

Date

2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [mbi_argv.h](#).

16.123 mbi_argv.h

[Go to the documentation of this file.](#)

```

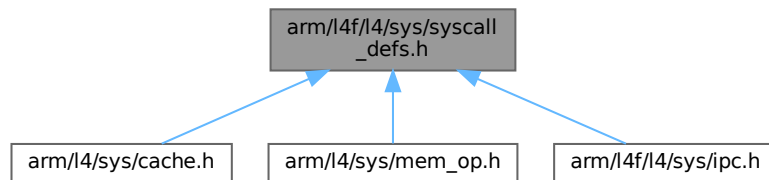
00001
00008 /*
00009  * (c) 2003-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
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

```

16.124 arm/l4f/l4/sys/syscall_defs.h File Reference

Syscall entry definitions.

This graph shows which files directly or indirectly include this file:



16.124.1 Detailed Description

Syscall entry definitions.

Definition in file [syscall_defs.h](#).

16.125 syscall_defs.h

[Go to the documentation of this file.](#)

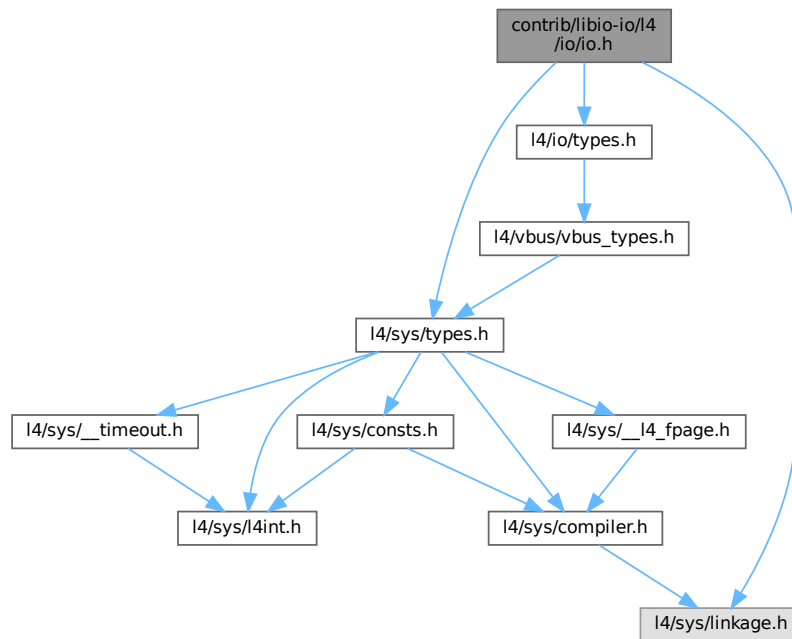
```

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 #ifndef L4_SYSCALL_MAGIC_OFFSET
00026 #   define L4_SYSCALL_MAGIC_OFFSET 8
00027 #endif
00028 #define L4_SYSCALL_INVOKE    (-0x00000004-L4_SYSCALL_MAGIC_OFFSET)
00029 #define L4_SYSCALL_MEM_OP    (-0x00000008-L4_SYSCALL_MAGIC_OFFSET)
00030
00031 #endif /* __L4SYS__ARCH_ARM__L4API_L4F__SYSCALL_DEFS_H__ */

```

16.126 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.

16.126.1 Function Documentation

16.126.1.1 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](#).

16.126.1.2 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

16.126.1.3 l4io_request_all_ioports()

```
void l4io_request_all_ioports (
    void(*) (l4vbus\_resource\_t const *res) res_cb )
```

Request all available IO-port resources.

Parameters

<code>res_cb</code>	Callback function called for every port resource found, give NULL for no callback.
---------------------	--

16.126.1.4 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.

16.127 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 /* ----- */
00255 /* Implementations */
00256
00257 L4_INLINE
00258 l4io_device_handle_t l4io_get_root_device(void)
00259 { return 0; }
00260
00261 EXTERN_C_END

```

16.128 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.

16.128.1 Detailed Description

Alloc list.

Definition in file [alloc.h](#).

16.129 alloc.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 namespace L4 {
00026
00031     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 }
```

16.130 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 }}

```

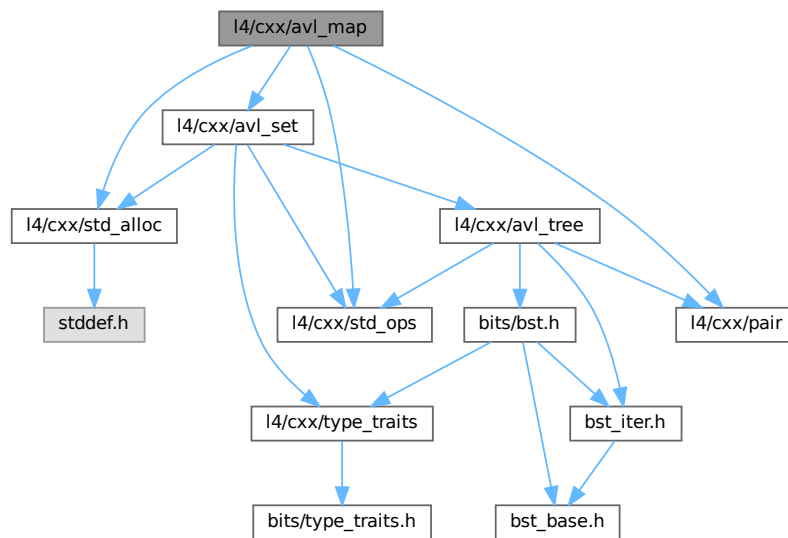
16.131 l4/cxx/avl_map File Reference

AVL map.

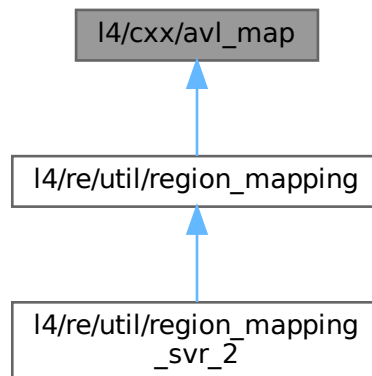
```

#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.

16.131.1 Detailed Description

AVL map.

Definition in file [avl_map](#).

16.132 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.

```

```

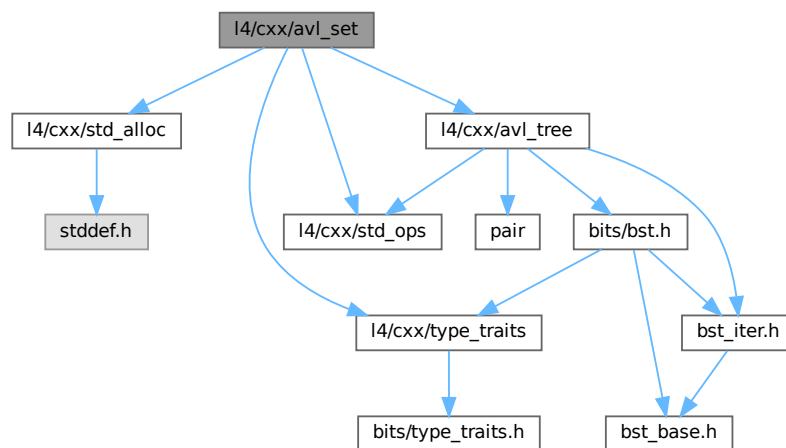
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files 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
00118     Data_type const &operator [] (Key_type const &key) const
00119     { return this->find_node(key)->second; }
00120
00130     Data_type &operator [] (Key_type const &key)
00131     {
00132         Node n = this->find_node(key);
00133         if (n)
00134             return const_cast<Data_type&>(n->second);
00135         else
00136             return insert(key, Data_type()).first->second;
00137     }
00138 };
00139
00140 }
00141

```

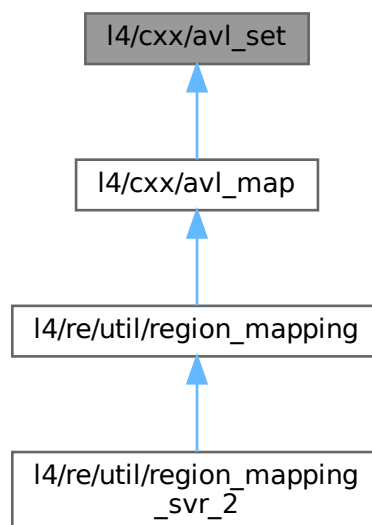
16.133 l4/cxx/avl_set File Reference

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.

16.133.1 Detailed Description

AVL set.

Definition in file `avl_set`.

16.134 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 namespace cxx {
00033 namespace Bits {
00042 template< typename Node, typename Key, typename Node_op >
00043 class Avl_set_iter : public __Bst_iter_b<Node, Node_op>
00044 {
00045 private:
00047     typedef __Bst_iter_b<Node, Node_op> Base;
00048
00050     typedef typename Type_traits<Key>::Non_const_type Non_const_key;
```

```

00051
00053     typedef Avl_set_iter<Node, Non_const_key, Node_op> Non_const_iter;
00054
00055     using Base::_n;
00056     using Base::_r;
00057     using Base::inc;
00058
00059 public:
00061     Avl_set_iter() = default;
00062
00067     Avl_set_iter(Node const *t) : Base(t) {}
00068
00073     Avl_set_iter(Base const &o) : Base(o) {}
00074
00076     Avl_set_iter(Non_const_iter const &o)
00077     : Base(o) {}
00078
00080     Avl_set_iter &operator = (Non_const_iter const &o)
00081     { Base::operator = (o); return *this; }
00082
00087     Key &operator * () const { return const_cast<Node*>(_n)->item; }
00092     Key *operator -> () const { return &const_cast<Node*>(_n)->item; }
00096     Avl_set_iter &operator ++ () { inc(); return *this; }
00100     Avl_set_iter operator ++ (int)
00101     { Avl_set_iter tmp = *this; inc(); return tmp; }
00102
00103 };
00104
00106 template<typename KEY_TYPE>
00107 struct Avl_set_get_key
00108 {
00109     typedef KEY_TYPE Key_type;
00110     template<typename NODE>
00111     static Key_type const &key_of(NODE const *n)
00112     { return n->item; }
00113 };
00114
00115
00128 template< typename ITEM_TYPE, class COMPARE,
00129           template<typename A> class ALLOC,
00130           typename GET_KEY>
00131 class Base_avl_set
00132 {
00133 public:
00140     enum
00141     {
00142         E_noent = 2,
00143         E_exist = 17,
00144         E_nomem = 12,
00145         E_inval = 22
00146     };
00148     typedef ITEM_TYPE Item_type;
00150     typedef GET_KEY Get_key;
00152     typedef typename GET_KEY::Key_type Key_type;
00154     typedef typename Type_traits<Item_type>::Const_type Const_item_type;
00156     typedef COMPARE Item_compare;
00157
00158 private:
00160     class _Node : public Avl_tree_node
00161     {
00162     public:
00164         Item_type item;
00165
00166         _Node() = default;
00167
00168         _Node(Item_type const &item) : Avl_tree_node(), item(item) {}
00169     };
00170
00171 public:
00175     class Node
00176     {
00177     private:
00178         struct No_type;
00179         friend class Base_avl_set<ITEM_TYPE, COMPARE, ALLOC, GET_KEY>;
00180         _Node const *_n;
00181         explicit Node(_Node const *n) : _n(n) {}
00182
00183     public:
00185         Node() : _n(0) {}
00186
00192         Item_type const &operator * () { return _n->item; }
00198         Item_type const *operator -> () { return &_n->item; }
00199
00204         bool valid() const { return _n; }
00205
00207         operator Item_type const * () { return _n ? &_n->item : 0; }
00208     };

```

```

00209
00211     typedef ALLOC<_Node> Node_allocator;
00212
00213 private:
00214     typedef Avl_tree<_Node, GET_KEY, COMPARE> Tree;
00215     Tree _tree;
00217     Node_allocator _alloc;
00218
00219     Base_avl_set &operator = (Base_avl_set const &) = delete;
00220
00221     typedef typename Tree::Fwd_iter_ops Fwd;
00222     typedef typename Tree::Rev_iter_ops Rev;
00223
00224 public:
00225     typedef typename Type_traits<Item_type>::Param_type Item_param_type;
00226
00228     typedef Avl_set_iter<_Node, Item_type, Fwd> Iterator;
00229     typedef Iterator iterator;
00231     typedef Avl_set_iter<_Node, Const_item_type, Fwd> Const_iterator;
00232     typedef Const_iterator const_iterator;
00234     typedef Avl_set_iter<_Node, Item_type, Rev> Rev_iterator;
00235     typedef Rev_iterator reverse_iterator;
00237     typedef Avl_set_iter<_Node, Const_item_type, Rev> Const_rev_iterator;
00238     typedef Const_rev_iterator const_reverse_iterator;
00239
00246     explicit Base_avl_set(Node_allocator const &alloc = Node_allocator())
00247     : _tree(), _alloc(alloc)
00248     {}
00249
00250     ~Base_avl_set()
00251     {
00252         _tree.remove_all([this](_Node *n)
00253             {
00254                 n->~_Node();
00255                 _alloc.free(n);
00256             });
00257     }
00258
00266     inline Base_avl_set(Base_avl_set const &o);
00267
00285     cxx::Pair<Iterator, int> insert(Item_type const &item);
00286
00295     int remove(Key_type const &item)
00296     {
00297         _Node *n = _tree.remove(item);
00298
00299         if (n)
00300         {
00301             n->~_Node();
00302             _alloc.free(n);
00303             return 0;
00304         }
00305
00306         return -E_noent;
00307     }
00308
00313     int erase(Key_type const &item)
00314     { return remove(item); }
00315
00324     Node find_node(Key_type const &item) const
00325     { return Node(_tree.find_node(item)); }
00326
00335     Node lower_bound_node(Key_type const &key) const
00336     { return Node(_tree.lower_bound_node(key)); }
00337
00338     Node lower_bound_node(Key_type &&key) const
00339     { return Node(_tree.lower_bound_node(key)); }
00340
00345     Const_iterator begin() const { return _tree.begin(); }
00350     Const_iterator end() const { return _tree.end(); }
00351
00356     Iterator begin() { return _tree.begin(); }
00361     Iterator end() { return _tree.end(); }
00362
00367     Const_rev_iterator rbegin() const { return _tree.rbegin(); }
00372     Const_rev_iterator rend() const { return _tree.rend(); }
00373
00378     Rev_iterator rbegin() { return _tree.rbegin(); }
00383     Rev_iterator rend() { return _tree.rend(); }
00384
00385     Const_iterator find(Key_type const &item) const
00386     { return _tree.find(item); }
00387
00388 #ifdef __DEBUG_L4_AVL
00389     bool rec_dump(bool print)
00390     {
00391         return _tree.rec_dump(print);

```

```

00392     }
00393 #endif
00394 };
00395
00396
00397 //-----
00398 /* Implementation of AVL Tree */
00399
00400 /* Create a copy */
00401 template< typename Item, class Compare, template<typename A> class Alloc, typename KEY_TYPE>
00402 Base_avl_set<Item,Compare,Alloc,KEY_TYPE>::Base_avl_set(Base_avl_set const &o)
00403 : _tree(), _alloc(o._alloc)
00404 {
00405     for (Const_iterator i = o.begin(); i != o.end(); ++i)
00406         insert(*i);
00407 }
00408
00409 /* Insert new _Node. */
00410 template< typename Item, class Compare, template< typename A > class Alloc, typename KEY_TYPE>
00411 Pair<typename Base_avl_set<Item,Compare,Alloc,KEY_TYPE>::Iterator, int>
00412 Base_avl_set<Item,Compare,Alloc,KEY_TYPE>::insert(Item const &item)
00413 {
00414     _Node *n = _alloc.alloc();
00415     if (!n)
00416         return cxx::pair(end(), -E_nomem);
00417     new (n, Nothrow()) _Node(item);
00418     Pair<_Node *, bool> err = _tree.insert(n);
00419     if (!err.second)
00420     {
00421         n->~_Node();
00422         _alloc.free(n);
00423     }
00424 }
00425
00426 return cxx::pair(Iterator(typename Tree::Iterator(err.first, err.first)), err.second ? 0 :
-E_exist);
00427 }
00428
00429 } // namespace Bits
00430
00442 template< typename ITEM_TYPE, class COMPARE = Lt_functor<ITEM_TYPE>,
00443           template<typename A> class ALLOC = New_allocator>
00444 class Avl_set :
00445     public Bits::Base_avl_set<ITEM_TYPE, COMPARE, ALLOC,
00446                             Bits::Avl_set_get_key<ITEM_TYPE> >
00447 {
00448 private:
00449     typedef Bits::Base_avl_set<ITEM_TYPE, COMPARE, ALLOC,
00450                             Bits::Avl_set_get_key<ITEM_TYPE> > Base;
00451
00452 public:
00453     typedef typename Base::Node_allocator Node_allocator;
00454     Avl_set() = default;
00455     Avl_set(Node_allocator const &alloc)
00456         : Base(alloc)
00457     {}
00458 };
00459
00460 } // namespace cxx

```

16.135 I4/cxx/avl_tree File Reference

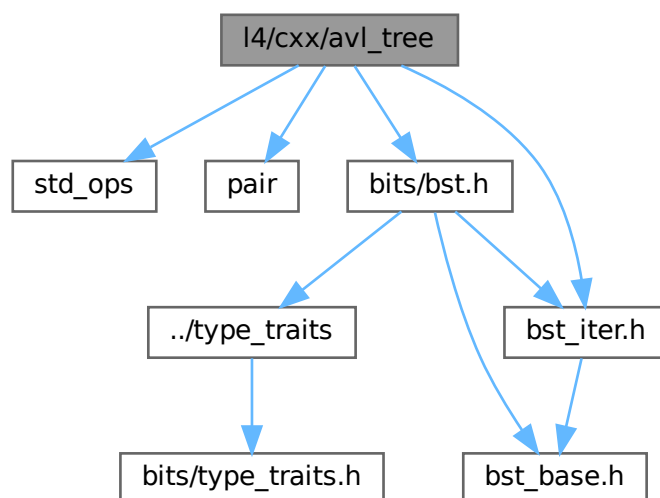
AVL tree.

```

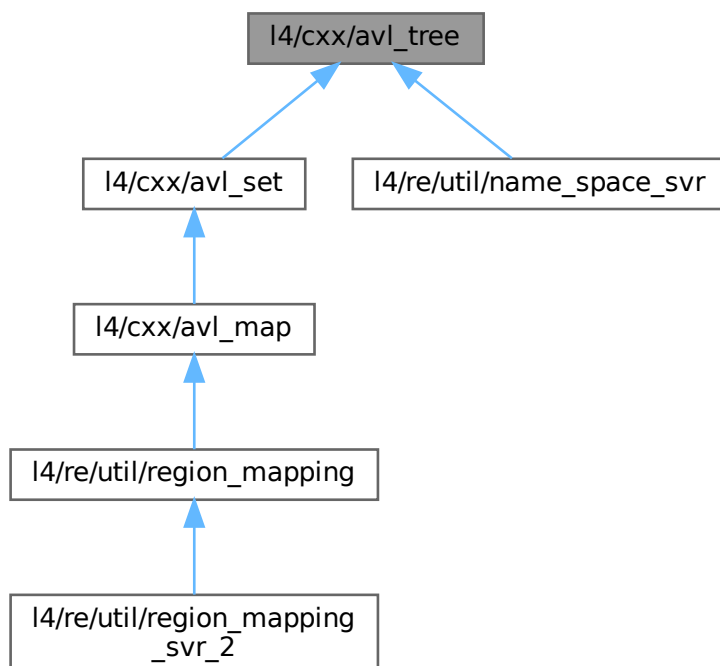
#include "std_ops"
#include "pair"
#include "bits/bst.h"
#include "bits/bst_iter.h"

```

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.

16.135.1 Detailed Description

AVL tree.

Definition in file [avl_tree](#).

16.136 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 namespace cxx {
00034
00038 class Avl_tree_node : public Bits::Bst_node
00039 {
00040 private:
00041     template< typename Node, typename Get_key, typename Compare >
00042     friend class Avl_tree;
00043
00045     typedef Bits::Direction Bal;
00047     typedef Bits::Direction Dir;
00048
00049     // We are a final BST node, hide interior.
00051     using Bits::Bst_node::next;
00052     using Bits::Bst_node::next_p;
00053     using Bits::Bst_node::rotate;
00057     Bal _balance;
00058
00059 protected:
```

```

00061     Avl_tree_node() = default;
00062
00063 private:
00064     Avl_tree_node(Avl_tree_node const &o) = delete;
00065     Avl_tree_node(Avl_tree_node &&o) = delete;
00066
00067     Avl_tree_node &operator = (Avl_tree_node const &o) = default;
00070     Avl_tree_node &operator = (Avl_tree_node &&o) = default;
00071
00073     explicit Avl_tree_node(bool) : Bits::Bst_node(true), _balance(Dir::N) {}
00074
00076     static Bits::Bst_node *rotate2(Bst_node **t, Bal idir, Bal pre);
00077
00079     bool balanced() const { return _balance == Bal::N; }
00080
00082     Bal balance() const { return _balance; }
00083
00085     void balance(Bal b) { _balance = b; }
00086 };
00087
00088
00105 template< typename Node, typename Get_key,
00106           typename Compare = Lt_functor<typename Get_key::Key_type> >
00107 class Avl_tree : public Bits::Bst<Node, Get_key, Compare>
00108 {
00109 private:
00110     typedef Bits::Bst<Node, Get_key, Compare> Bst;
00111
00113     using Bst::_head;
00114
00116     using Bst::k;
00117
00119     typedef typename Avl_tree_node::Bal Bal;
00121     typedef typename Avl_tree_node::Bal Dir;
00122
00123     Avl_tree(Avl_tree const &o) = delete;
00124     Avl_tree &operator = (Avl_tree const &o) = delete;
00125     Avl_tree(Avl_tree &&o) = delete;
00126     Avl_tree &operator = (Avl_tree &&o) = delete;
00127
00128 public:
00130     typedef typename Bst::Key_type Key_type;
00131     typedef typename Bst::Key_param_type Key_param_type;
00133
00134     // Grab iterator types from Bst
00137     typedef typename Bst::Iterator Iterator;
00139     typedef typename Bst::Const_iterator Const_iterator;
00141     typedef typename Bst::Rev_iterator Rev_iterator;
00143     typedef typename Bst::Const_rev_iterator Const_rev_iterator;
00145
00153     Pair<Node *, bool> insert(Node *new_node);
00154
00161     Node *remove(Key_param_type key);
00165     Node *erase(Key_param_type key) { return remove(key); }
00166
00168     Avl_tree() = default;
00169
00171     ~Avl_tree() noexcept
00172     {
00173         this->remove_all([](Node *){});
00174     }
00175
00176 #ifdef __DEBUG_L4_AVL
00177     bool rec_dump(Avl_tree_node *n, int depth, int *dp, bool print, char pfx);
00178     bool rec_dump(bool print)
00179     {
00180         int dp=0;
00181         return rec_dump(static_cast<Avl_tree_node *>(_head), 0, &dp, print, '+');
00182     }
00183 #endif
00184 };
00185
00186
00187 //-----
00188 /* IMPLEMENTATION: Bits::__Bst_iter_b */
00189
00190
00191 inline
00192 Bits::Bst_node *
00193 Avl_tree_node::rotate2(Bst_node **t, Bal idir, Bal pre)
00194 {
00195     typedef Bits::Bst_node N;
00196     typedef Avl_tree_node A;
00197     N *tmp[2] = { *t, N::next(*t, idir) };
00198     *t = N::next(tmp[1], !idir);
00199     A *n = static_cast<A*>(*t);
00200

```

```

00201 N::next(tmp[0], idir, N::next(n, !idir));
00202 N::next(tmp[1], !idir, N::next(n, idir));
00203 N::next(n, !idir, tmp[0]);
00204 N::next(n, idir, tmp[1]);
00205
00206 n->balance(Bal::N);
00207
00208 if (pre == Bal::N)
00209 {
00210     static_cast<A*>(tmp[0])->balance(Bal::N);
00211     static_cast<A*>(tmp[1])->balance(Bal::N);
00212     return 0;
00213 }
00214
00215 static_cast<A*>(tmp[pre != idir])->balance(!pre);
00216 static_cast<A*>(tmp[pre == idir])->balance(Bal::N);
00217
00218 return N::next(tmp[pre == idir], !pre);
00219 }
00220
00221 //-----
00222 /* Implementation of AVL Tree */
00223
00224 /* Insert new _Node. */
00225 template< typename Node, typename Get_key, class Compare>
00226 Pair<Node *, bool>
00227 Avl_tree<Node, Get_key, Compare>::insert(Node *new_node)
00228 {
00229     typedef Avl_tree_node A;
00230     typedef Bits::Bst_node N;
00231     N **t = &_head; /* search variable */
00232     N **s = &_head; /* node where rebalancing may occur */
00233     Key_param_type new_key = Get_key::key_of(new_node);
00234
00235     // search insertion point
00236     for (N *p; (p = *t);)
00237     {
00238         Dir b = this->dir(new_key, p);
00239         if (b == Dir::N)
00240             return pair(static_cast<Node*>(p), false);
00241
00242         if (!static_cast<A const *>(p)->balanced())
00243             s = t;
00244
00245         t = A::next_p(p, b);
00246     }
00247
00248     *static_cast<A*>(new_node) = A(true);
00249     *t = new_node;
00250
00251     N *n = *s;
00252     A *a = static_cast<A*>(n);
00253     if (!a->balanced())
00254     {
00255         A::Bal b(this->greater(new_key, n));
00256         if (a->balance() != b)
00257         {
00258             // ok we got in balance the shorter subtree go higher
00259             a->balance(Bal::N);
00260             // propagate the new balance down to the new node
00261             n = A::next(n, b);
00262         }
00263         else if (b == Bal(this->greater(new_key, A::next(n, b))))
00264         {
00265             // left-left or right-right case -> single rotation
00266             A::rotate(s, b);
00267             a->balance(Bal::N);
00268             static_cast<A*>(*s)->balance(Bal::N);
00269             n = A::next(*s, b);
00270         }
00271         else
00272         {
00273             // need a double rotation
00274             n = A::next(A::next(n, b), !b);
00275             n = A::rotate2(s, b, n == new_node ? Bal::N : Bal(this->greater(new_key, n)));
00276         }
00277     }
00278
00279     for (A::Bal b; n && n != new_node; static_cast<A*>(n)->balance(b), n = A::next(n, b))
00280         b = Bal(this->greater(new_key, n));
00281
00282     return pair(new_node, true);
00283 }
00284
00285
00286 /* remove an element */
00287 template< typename Node, typename Get_key, class Compare>

```



```

00288 inline
00289 Node *Avl_tree<Node, Get_key, Compare>::remove(Key_param_type key)
00290 {
00291     typedef Avl_tree_node A;
00292     typedef Bits::Bst_node N;
00293     N **q = &_head; /* search variable */
00294     N **s = &_head; /* last ('deepest') node on the search path to q
00295                      * with balance 0, at this place the rebalancing
00296                      * stops in any case */
00297     N **t = 0;
00298     Dir dir;
00299
00300     // find target node and rebalancing entry
00301     for (N *n; (n = *q); q = A::next_p(n, dir))
00302     {
00303         dir = Dir(this->greater(key, n));
00304         if (dir == Dir::L && !this->greater(k(n), key))
00305             /* found node */
00306             t = q;
00307
00308         if (!A::next(n, dir))
00309             break;
00310
00311         A const *a = static_cast<A const *>(n);
00312         if (a->balanced() || (a->balance() == !dir && A::next<A>(n, !dir)->balanced()))
00313             s = q;
00314     }
00315
00316     // nothing found
00317     if (!t)
00318         return 0;
00319
00320     A *i = static_cast<A*>(*t);
00321
00322     for (N *n; (n = *s); s = A::next_p(n, dir))
00323     {
00324         dir = Dir(this->greater(key, n));
00325
00326         if (!A::next(n, dir))
00327             break;
00328
00329         A *a = static_cast<A*>(n);
00330         // got one out of balance
00331         if (a->balanced())
00332             a->balance(!dir);
00333         else if (a->balance() == dir)
00334             a->balance(Bal::N);
00335         else
00336         {
00337             // we need rotations to get in balance
00338             Bal b = A::next<A>(n, !dir)->balance();
00339             if (b == dir)
00340                 A::rotate2(s, !dir, A::next<A>(A::next(n, !dir), dir)->balance());
00341             else
00342             {
00343                 A::rotate(s, !dir);
00344                 if (b != Bal::N)
00345                 {
00346                     a->balance(Bal::N);
00347                     static_cast<A*>(*s)->balance(Bal::N);
00348                 }
00349                 else
00350                 {
00351                     a->balance(!dir);
00352                     static_cast<A*>(*s)->balance(dir);
00353                 }
00354             }
00355             if (n == i)
00356                 t = A::next_p(*s, dir);
00357         }
00358     }
00359
00360     A *n = static_cast<A*>(*q);
00361     *t = n;
00362     *q = A::next(n, !dir);
00363     *n = *i;
00364
00365     return static_cast<Node*>(i);
00366 }
00367
00368 #ifdef __DEBUG_L4_AVL
00369 template< typename Node, typename Get_key, class Compare>
00370 bool Avl_tree<Node, Get_key, Compare>::rec_dump(Avl_tree_node *n, int depth, int *dp, bool print, char
00371 pfx)
00372 {
00373     typedef Avl_tree_node A;

```


Data Structures

- class [L4::IOModifier](#)
Modifier class for the IO stream.

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.

16.137.1 Detailed Description

Basic IO stream.

Definition in file [basic_ostream](#).

16.138 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((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((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((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((long long unsigned)u, -1);
00234     return s;
00235 }
00236
00237 inline
00238 L4::BasicOStream &
00239 operator « (L4::BasicOStream &s, unsigned u)

```

```

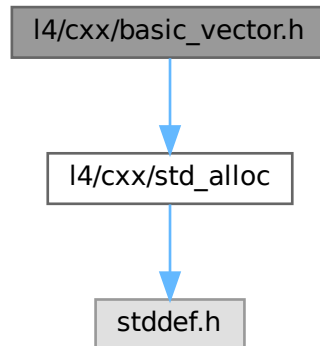
00240 {
00241     s.write((long long unsigned)u, -1);
00242     return s;
00243 }
00244
00245 inline
00246 L4::BasicOStream &
00247 operator « (L4::BasicOStream &s, unsigned long u)
00248 {
00249     s.write((long long unsigned)u, -1);
00250     return s;
00251 }
00252
00253 inline
00254 L4::BasicOStream &
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 = (long unsigned)u;
00266     L4::IOBackend::Mode mode = s.be_mode();
00267     s.write(L4::hex);
00268     s.write((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); }

```

16.139 I4/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.

16.139.1 Detailed Description

Basic vector.

Definition in file [basic_vector.h](#).

16.140 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 <l4/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 };

```

16.141 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
00024 namespace cxx {
00025
00026     template<typename T, unsigned LSB, unsigned MSB>
00027     class Bitfield
00028     {
00029     private:
00030         static_assert(MSB >= LSB, "boundary mismatch in bit-field definition");
00031         static_assert(MSB < sizeof(T) * 8, "MSB outside of bit-field type");
00032         static_assert(LSB < sizeof(T) * 8, "LSB outside of bit-field type");
00033
00034     public:
00035         template<unsigned BITS> struct Best_type
00036         {
00037             template< typename TY > struct Cmp { enum { value = (BITS <= sizeof(TY)*8) }; };
00038             typedef cxx::type_list<
00039                 unsigned char,
00040                 unsigned short,
00041                 unsigned int,
00042                 unsigned long,
00043                 unsigned long long
00044             > Unsigned_types;
00045             typedef cxx::find_type_t<Unsigned_types, Cmp> Type;
00046         };
00047
00048     public:
00049         enum
00050         {
00051             Bits = MSB + 1 - LSB,
00052             Lsb = LSB,
00053             Msb = MSB,
00054         };
00055
00056         enum Masks : T
00057         {
00058             Low_mask = ((T)~0ULL) >> (sizeof(T)*8 - Bits),
00059             Mask = Low_mask << Lsb,
00060         };
00061
00062         typedef typename Best_type<Bits>::Type Bits_type;
00063
00064         typedef typename Best_type<Bits + Lsb>::Type Shift_type;
00065
00066     };
00067
00068 };

```



```

00092
00093 private:
00094     static_assert(sizeof(Bits_type)*8 >= Bits, "error finding the type to store the bits");
00095     static_assert(sizeof(Shift_type)*8 >= Bits + Lsb, "error finding the type to keep the shifted
bits");
00096     static_assert(sizeof(Bits_type) <= sizeof(T), "size mismatch for Bits_type");
00097     static_assert(sizeof(Shift_type) <= sizeof(T), "size mismatch for Shift_type");
00098     static_assert(sizeof(Bits_type) <= sizeof(Shift_type), "size mismatch for Shift_type and
Bits_type");
00099
00100 public:
00108     static Bits_type get(Shift_type val)
00109     { return (val >> Lsb) & Low_mask; }
00110
00121     static T get_unshifted(Shift_type val)
00122     { return val & Mask; }
00123
00136     static T set_dirty(T dest, Shift_type val)
00137     {
00138         //assert (!(val & ~Low_mask));
00139         return (dest & ~Mask) | (val << Lsb);
00140     }
00141
00156     static T set_unshifted_dirty(T dest, Shift_type val)
00157     {
00158         //assert (!(val & ~Mask));
00159         return (dest & ~Mask) | val;
00160     }
00161
00170     static T set(T dest, Bits_type val)
00171     { return set_dirty(dest, val & Low_mask); }
00172
00182     static T set_unshifted(T dest, Shift_type val)
00183     { return set_unshifted_dirty(dest, val & Mask); }
00184
00196     static T val_dirty(Shift_type val) { return val << Lsb; }
00197
00205     static T val(Bits_type val) { return val_dirty(val & Low_mask); }
00206
00215     static T val_unshifted(Shift_type val) { return val & Mask; }
00216
00218 template< typename TT >
00219 class Value_base
00220 {
00221 private:
00222     TT v;
00223
00224 public:
00225     Value_base(TT t) : v(t) {}
00226     Bits_type get() const { return Bitfield::get(v); }
00227     T get_unshifted() const { return Bitfield::get_unshifted(v); }
00228
00229     void set(Bits_type val) { v = Bitfield::set(v, val); }
00230     void set_dirty(Bits_type val) { v = Bitfield::set_dirty(v, val); }
00231     void set_unshifted(Shift_type val) { v = Bitfield::set_unshifted(v, val); }
00232     void set_unshifted_dirty(Shift_type val) { v = Bitfield::set_unshifted_dirty(v, val); }
00233 };
00234
00236 template< typename TT >
00237 class Value : public Value_base<TT>
00238 {
00239 public:
00240     Value(TT t) : Value_base<TT>(t) {}
00241     operator Bits_type () const { return this->get(); }
00242     Value &operator = (Bits_type val) { this->set(val); return *this; }
00243     Value &operator = (Value const &o) { this->set(o.get()); return *this; }
00244     // This copy constructor initializes more bits than necessary but this is
00245     // usually not a problem.
00246     Value (Value const &o) = default;
00247 };
00248
00250 template< typename TT >
00251 class Value_unshifted : public Value_base<TT>
00252 {
00253 public:
00254     Value_unshifted(TT t) : Value_base<TT>(t) {}
00255     operator Shift_type () const { return this->get_unshifted(); }
00256     Value_unshifted &operator = (Shift_type val) { this->set_unshifted(val); return *this; }
00257     Value_unshifted &operator = (Value_unshifted const &o)
00258     {
00259         this->set_unshifted(o.get_unshifted());
00260         return *this;
00261     }
00262     // This copy constructor initializes more bits than necessary but this is
00263     // usually not a problem.
00264     Value_unshifted(Value_unshifted const &) = default;
00265 };

```

```

00266
00268     typedef Value<T&> Ref;
00270     typedef Value<T const> Val;
00271
00273     typedef Value_unshifted<T&> Ref_unshifted;
00275     typedef Value_unshifted<T 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     typename name ## _bfm_t::Val name() const { return data_member; } \
00284 \
00285     typename name ## _bfm_t::Ref name() { return data_member; } \
00286
00288 #define CXX_BITFIELD_MEMBER_RO(LSB, MSB, name, data_member) \
00289 \
00290 \
00291     typedef cxx::Bitfield<decltype(data_member), LSB, MSB> name ## _bfm_t; \
00292 \
00293     typename name ## _bfm_t::Val name() const { return data_member; } \
00294
00296 #define CXX_BITFIELD_MEMBER_UNSHIFTED(LSB, MSB, name, data_member) \
00297 \
00298 \
00299     typedef cxx::Bitfield<decltype(data_member), LSB, MSB> name ## _bfm_t; \
00300 \
00301     typename name ## _bfm_t::Val_unshifted name() const { return data_member; } \
00302 \
00303     typename name ## _bfm_t::Ref_unshifted name() { return data_member; } \
00304
00306 #define CXX_BITFIELD_MEMBER_UNSHIFTED_RO(LSB, MSB, name, data_member) \
00307 \
00308 \
00309     typedef cxx::Bitfield<decltype(data_member), LSB, MSB> name ## _bfm_t; \
00310 \
00311     typename name ## _bfm_t::Val_unshifted name() const { return data_member; } \
00312
00313 }

```

16.142 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
00030 class Bitmap_base
00031 {
00032 protected:
00036     typedef unsigned long word_type;
00037
00038     enum
00039     {
00040         W_bits = sizeof(word_type) * 8,
00041         C_bits = 8,
00042     };
00043
00047     word_type *_bits;
00048
00054     static unsigned word_index(unsigned bit) { return bit / W_bits; }
00055
00061     static unsigned bit_index(unsigned bit) { return bit % W_bits; }

```

```

00062
00066 class Bit
00067 {
00068     Bitmap_base *_bm;
00069     long _bit;
00070
00071 public:
00072     Bit(Bitmap_base *bm, long bit) : _bm(bm), _bit(bit) {}
00073     Bit &operator = (bool val) { _bm->bit(_bit, val); return *this; }
00074     operator bool () const { return _bm->bit(_bit); }
00075 };
00076
00077 public:
00078     explicit Bitmap_base(void *bits) noexcept : _bits((word_type *)bits) {}
00079
00081     static long words(long bits) noexcept { return (bits + W_bits - 1) / W_bits; }
00082     static long bit_buffer_bytes(long bits) noexcept
00083     { return words(bits) * W_bits / 8; }
00084
00086     template< long BITS >
00087     class Word
00088     {
00089     public:
00090         typedef unsigned long Type;
00091         enum
00092         {
00093             Size = (BITS + W_bits - 1) / W_bits
00094         };
00095     };
00096
00098     static long chars(long bits) throw ()
00099     { return (bits + C_bits - 1) / C_bits; }
00100
00102     template< long BITS >
00103     class Char
00104     {
00105     public:
00106         typedef unsigned char Type;
00107         enum
00108         {
00109             Size = (BITS + C_bits - 1) / C_bits
00110         };
00111     };
00112
00118     void bit(long bit, bool on) noexcept;
00119
00124     void clear_bit(long bit) noexcept;
00129     void set_bit(long bit) noexcept;
00130
00136     word_type bit(long bit) const noexcept;
00137
00143     word_type operator [] (long bit) const noexcept
00144     { return this->bit(bit); }
00145
00151     Bit operator [] (long bit) noexcept
00152     { return Bit(this, bit); }
00153
00165     long scan_zero(long max_bit, long start_bit = 0) const noexcept;
00166
00167     void *bit_buffer() const noexcept { return _bits; }
00168
00169 protected:
00170     static int _bzl(unsigned long w) noexcept;
00171 };
00172
00173
00179 template<int BITS>
00180 class Bitmap : public Bitmap_base
00181 {
00182 private:
00183     char _bits[Bitmap_base::Char<BITS>::Size];
00184
00185 public:
00187     Bitmap() noexcept : Bitmap_base(_bits) {}
00188     Bitmap(Bitmap<BITS> const &o) noexcept : Bitmap_base(_bits)
00189     { __builtin_memcpy(_bits, o._bits, sizeof(_bits)); }
00202     long scan_zero(long start_bit = 0) const noexcept;
00203
00204     void clear_all()
00205     { __builtin_memset(_bits, 0, sizeof(_bits)); }
00206 };
00207
00208
00209 inline
00210 void
00211 Bitmap_base::bit(long bit, bool on) noexcept
00212 {

```

```

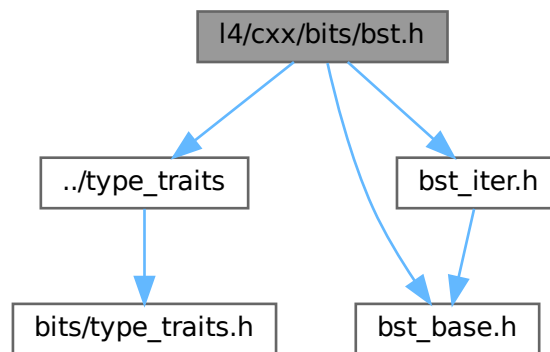
00213     long idx = word_index(bit);
00214     long b    = bit_index(bit);
00215     _bits[idx] = (_bits[idx] & ~(1UL << b)) | ((unsigned long)on << b);
00216 }
00217
00218 inline
00219 void
00220 Bitmap_base::clear_bit(long bit) noexcept
00221 {
00222     long idx = word_index(bit);
00223     long b    = bit_index(bit);
00224     _bits[idx] &= ~(1UL << b);
00225 }
00226
00227 inline
00228 void
00229 Bitmap_base::set_bit(long bit) noexcept
00230 {
00231     long idx = word_index(bit);
00232     long b    = bit_index(bit);
00233     _bits[idx] |= (1UL << b);
00234 }
00235
00236 inline
00237 Bitmap_base::word_type
00238 Bitmap_base::bit(long bit) const noexcept
00239 {
00240     long idx = word_index(bit);
00241     long b    = bit_index(bit);
00242     return _bits[idx] & (1UL << b);
00243 }
00244
00245 inline
00246 int
00247 Bitmap_base::_bzl(unsigned long w) noexcept
00248 {
00249     for (int i = 0; i < W_bits; ++i, w >>= 1)
00250     {
00251         if ((w & 1) == 0)
00252             return i;
00253     }
00254     return -1;
00255 }
00256
00257 inline
00258 long
00259 Bitmap_base::scan_zero(long max_bit, long start_bit) const noexcept
00260 {
00261     if (!(operator [] (start_bit)))
00262         return start_bit;
00263
00264     long idx = word_index(start_bit);
00265
00266     max_bit -= start_bit & ~(W_bits - 1);
00267
00268     for (; max_bit > 0; max_bit -= W_bits, ++idx)
00269     {
00270         if (_bits[idx] == 0)
00271             return idx * W_bits;
00272
00273         if (_bits[idx] != ~0UL)
00274         {
00275             long zbit = _bzl(_bits[idx]);
00276             return zbit < max_bit ? idx * W_bits + zbit : -1;
00277         }
00278     }
00279
00280     return -1;
00281 }
00282
00283 template<int BITS> inline
00284 long
00285 Bitmap<BITS>::scan_zero(long start_bit) const noexcept
00286 {
00287     return Bitmap_base::scan_zero(BITS, start_bit);
00288 }
00289
00290 };
00291

```

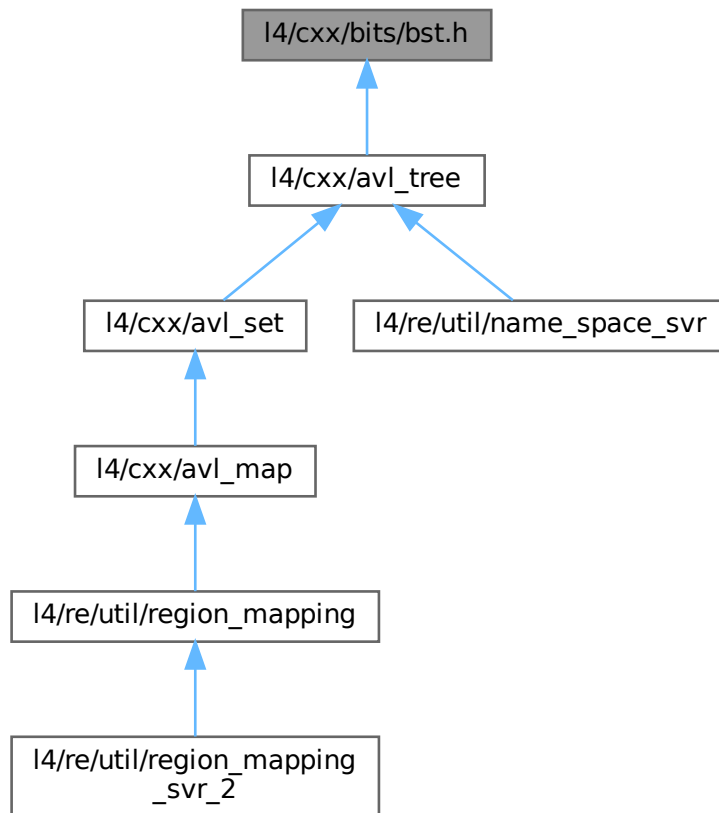
16.143 14/cxx/bits/bst.h File Reference

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.

16.143.1 Detailed Description

AVL tree.

Definition in file [bst.h](#).

16.144 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 namespace cxx { namespace Bits {
00031
00039 template< typename Node, typename Get_key, typename Compare >
00040 class Bst
00041 {
00042 private:
00043     typedef Direction Dir;
00044     struct Fwd
00045     {
00046     {
00047         static Node *child(Node const *n, Direction d)
00048         { return Bst_node::next<Node>(n, d); }
00049
00050         static bool cmp(Node const *l, Node const *r)
00051         { return Compare() (Get_key::key_of(l), Get_key::key_of(r)); }
00052     };
00053
00054     struct Rev
00055     {
00056     {
00057         static Node *child(Node const *n, Direction d)
00058         { return Bst_node::next<Node>(n, !d); }
00059
00060         static bool cmp(Node const *l, Node const *r)
00061         { return Compare() (Get_key::key_of(r), Get_key::key_of(l)); }
00062     };
00063
00064 public:
00066     typedef typename Get_key::Key_type Key_type;
00068     typedef typename Type_traits<Key_type>::Param_type Key_param_type;
00069
00071     typedef Fwd Fwd_iter_ops;
00073     typedef Rev Rev_iter_ops;
00074
00076
00078     typedef __Bst_iter<Node, Node, Fwd> Iterator;
00080     typedef __Bst_iter<Node, Node const, Fwd> Const_iterator;
00082     typedef __Bst_iter<Node, Node, Rev> Rev_iterator;
00084     typedef __Bst_iter<Node, Node const, Rev> Const_rev_iterator;
00087 protected:
00088     Bst_node *_head;
00089
00091     Bst() : _head(0) {}
00092
00094     Node *head() const { return static_cast<Node*>(_head); }
00095
00097     static Key_type k(Bst_node const *n)
00098     { return Get_key::key_of(static_cast<Node const *>(n)); }
00099
00101     static Dir dir(Key_param_type l, Key_param_type r)
00102     {
00103         Compare cmp;
00104         Dir d(cmp(r, l));
00105         if (d == Direction::L && !cmp(l, r))
00106             return Direction::N;
00107         return d;
00108     }
00109
00111     static Dir dir(Key_param_type l, Bst_node const *r)

```

```

00136 { return dir(l, k(r)); }
00137
00139 static bool greater(Key_param_type l, Key_param_type r)
00140 { return Compare()(r, l); }
00141
00143 static bool greater(Key_param_type l, Bst_node const *r)
00144 { return greater(l, k(r)); }
00145
00147 static bool greater(Bst_node const *l, Bst_node const *r)
00148 { return greater(k(l), k(r)); }
00157 template<typename FUNC>
00158 static void remove_tree(Bst_node *head, FUNC &&callback)
00159 {
00160     if (Bst_node *n = Bst_node::next(head, Dir::L))
00161         remove_tree(n, callback);
00162
00163     if (Bst_node *n = Bst_node::next(head, Dir::R))
00164         remove_tree(n, callback);
00165
00166     callback(static_cast<Node *>(head));
00167 }
00168
00169 public:
00170
00179 Const_iterator begin() const { return Const_iterator(head()); }
00184 Const_iterator end() const { return Const_iterator(); }
00185
00190 Iterator begin() { return Iterator(head()); }
00195 Iterator end() { return Iterator(); }
00196
00201 Const_rev_iterator rbegin() const { return Const_rev_iterator(head()); }
00206 Const_rev_iterator rend() const { return Const_rev_iterator(); }
00207
00212 Rev_iterator rbegin() { return Rev_iterator(head()); }
00217 Rev_iterator rend() { return Rev_iterator(); }
00225
00231 Node *find_node(Key_param_type key) const;
00232
00239 Node *lower_bound_node(Key_param_type key) const;
00240
00247 Const_iterator find(Key_param_type key) const;
00248
00257 template<typename FUNC>
00258 void remove_all(FUNC &&callback)
00259 {
00260     if (!_head)
00261         return;
00262
00263     Bst_node *head = _head;
00264     _head = 0;
00265     remove_tree(head, cxx::forward<FUNC>(callback));
00266 }
00267
00268
00270 };
00271
00272 /* find an element */
00273 template< typename Node, typename Get_key, class Compare>
00274 inline
00275 Node *
00276 Bst<Node, Get_key, Compare>::find_node(Key_param_type key) const
00277 {
00278     Dir d;
00279
00280     for (Bst_node *q = _head; q; q = Bst_node::next(q, d))
00281     {
00282         d = dir(key, q);
00283         if (d == Dir::N)
00284             return static_cast<Node*>(q);
00285     }
00286     return 0;
00287 }
00288
00289 template< typename Node, typename Get_key, class Compare>
00290 inline
00291 Node *
00292 Bst<Node, Get_key, Compare>::lower_bound_node(Key_param_type key) const
00293 {
00294     Dir d;
00295     Bst_node *r = 0;
00296
00297     for (Bst_node *q = _head; q; q = Bst_node::next(q, d))
00298     {
00299         d = dir(key, q);
00300         if (d == Dir::L)
00301             r = q; // found a node greater than key
00302         else if (d == Dir::N)

```

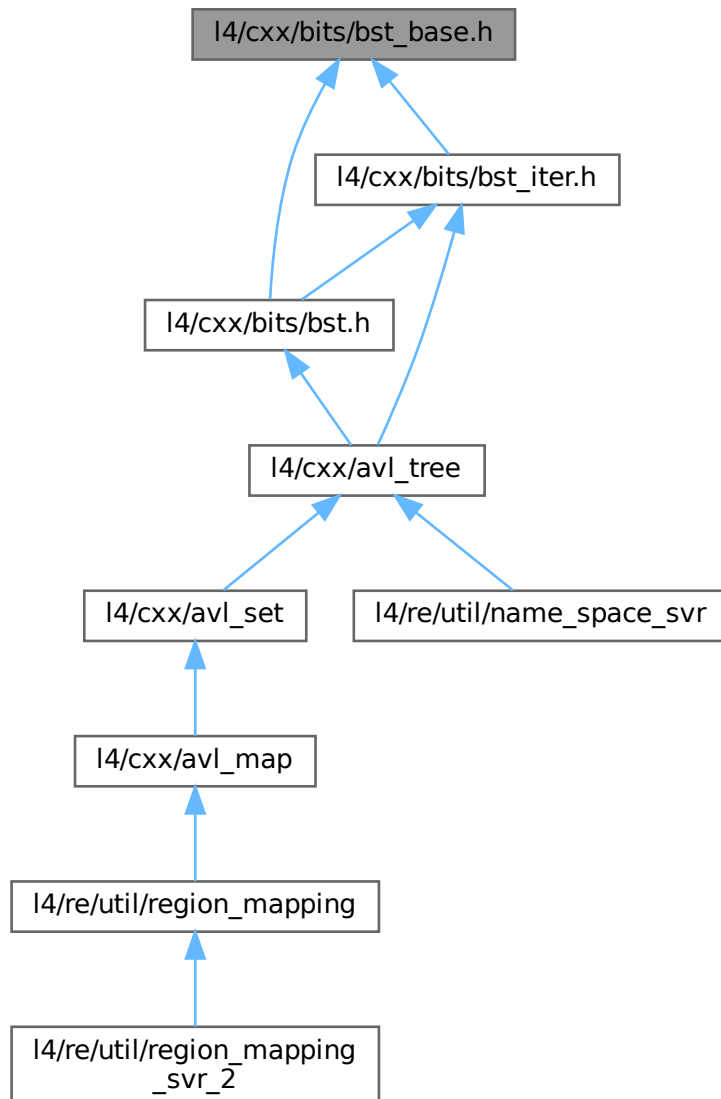


```
00303     return static_cast<Node*>(q);
00304 }
00305     return static_cast<Node*>(r);
00306 }
00307
00308 /* find an element */
00309 template< typename Node, typename Get_key, class Compare>
00310 inline
00311     typename Bst<Node, Get_key, Compare>::Const_iterator
00312 Bst<Node, Get_key, Compare>::find(Key_param_type key) const
00313 {
00314     Bst_node *q = _head;
00315     Bst_node *r = q;
00316
00317     for (Dir d; q; q = Bst_node::next(q, d))
00318     {
00319         d = dir(key, q);
00320         if (d == Dir::N)
00321             return Iterator(static_cast<Node*>(q), static_cast<Node *>(r));
00322
00323         if (d != Dir::L && q == r)
00324             r = Bst_node::next(q, d);
00325     }
00326     return Iterator();
00327 }
00328
00329 }
```

16.145 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.

16.145.1 Detailed Description

AVL tree.

Definition in file [bst_base.h](#).

16.146 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 }}

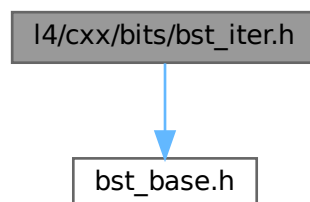
```

16.147 I4/cxx/bits/bst_iter.h File Reference

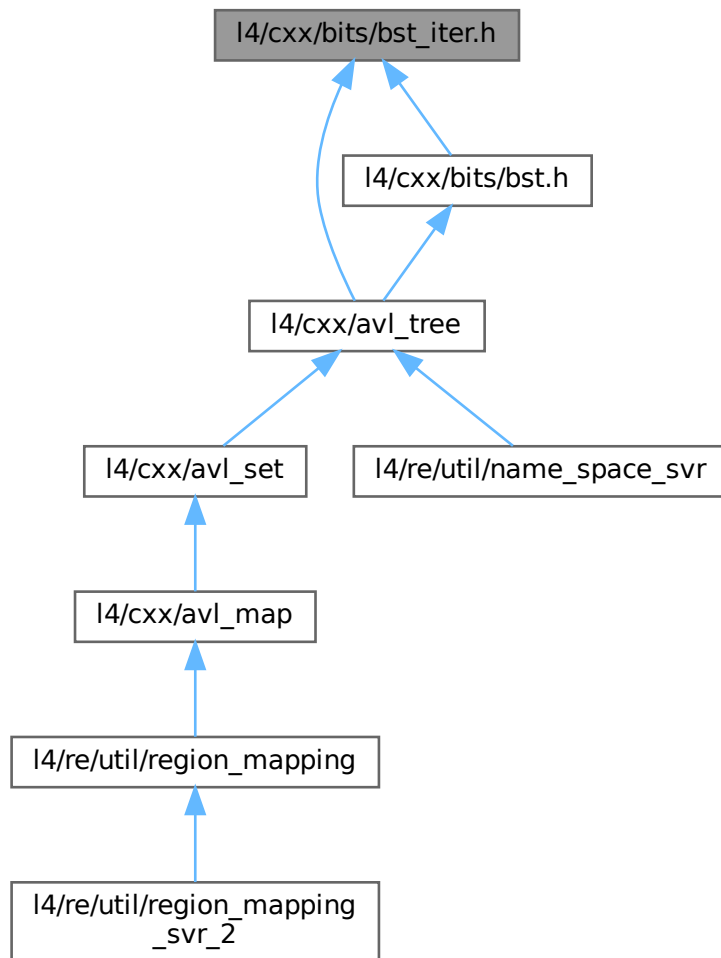
AVL tree.

```
#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.

16.147.1 Detailed Description

AVL tree.

Definition in file [bst_iter.h](#).

16.148 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 }}

```

16.149 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

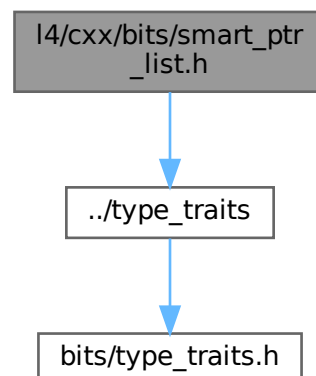
```

16.150 l4/cxx/bits/smart_ptr_list.h File Reference

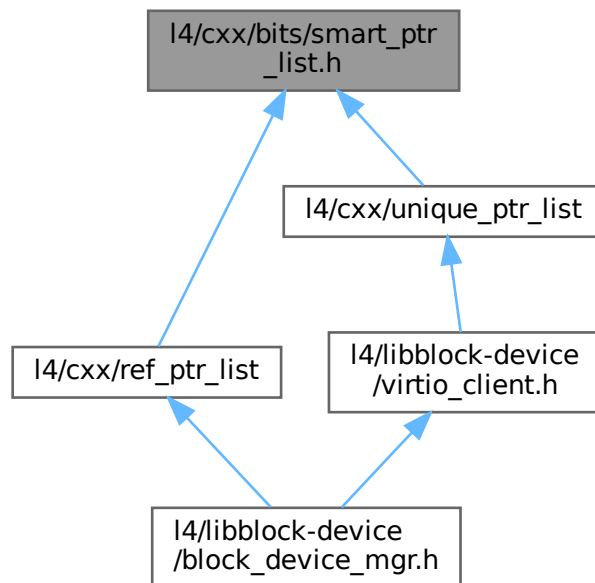
Implementation of a list of smart-pointer-managed objects.

```
#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.

16.150.1 Detailed Description

Implementation of a list of smart-pointer-managed objects.

Definition in file `smart_ptr_list.h`.

16.151 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 == &_amp;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 == &_amp;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 = &_amp;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 } }

```

16.152 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 { return (a + (unsigned long)alignment - 1UL)
00222   & ~((unsigned long)alignment - 1UL); }
00223 };
00224
00225
00226 };
00227
00228
00229

```

16.153 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

```

16.154 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>

```

[illegible]

```

graph TD
    libcxx_exceptions[libcxx/exceptions] --> libcxx_ipc_server[libcxx/ipc_server]
    libcxx_exceptions --> libcxx_htd_exc_io[libcxx/htd_exc_io]
    libcxx_exceptions --> lib4v_virtioqueue[lib4v/virtioqueue]
    libcxx_exceptions --> lib4v_error_helper[lib4v/error_helper]
    libcxx_exceptions --> lib4v_util_object_registry[lib4v/util/object_registry]
    
    lib4v_virtioqueue --> lib4v_virtio_server_virtio[lib4v/virtio/server/virtio]
    lib4v_virtioqueue --> lib4v_virtio_client_libvirtio[lib4v/virtio/client/libvirtio]
    lib4v_virtioqueue --> lib4v_virtio_client_block[lib4v/virtio/client/virtio-block]
    
    lib4v_error_helper --> lib4v_virtio_client_block
    
    lib4v_util_object_registry --> lib4v_virtio_client_block
    
    lib4v_virtio_server_virtio --> lib4v_virtio_server_block[lib4v/virtio/server/virtio-block]
    
    lib4v_virtio_client_block --> lib4v_virtio_client_net[lib4v/virtio/client/virtio-net]
    
    lib4v_virtio_client_block --> libblock_device_types_h[libblock-device/types.h]
    lib4v_virtio_client_block --> libblock_device_device_h[libblock-device/device.h]
    lib4v_virtio_client_block --> libblock_device_inout_memory_h[libblock-device/inout_memory.h]
    lib4v_virtio_client_block --> libblock_device_partition_h[libblock-device/partition.h]
    lib4v_virtio_client_block --> libblock_device_part_device_h[libblock-device/part_device.h]
    lib4v_virtio_client_block --> libblock_device_block_device_mgr_h[libblock-device/block_device_mgr.h]
    
    libblock_device_types_h --> libblock_device_device_h
    libblock_device_types_h --> libblock_device_inout_memory_h
    libblock_device_types_h --> libblock_device_partition_h
    libblock_device_types_h --> libblock_device_part_device_h
    libblock_device_types_h --> libblock_device_block_device_mgr_h
    
    libblock_device_device_h --> libblock_device_inout_memory_h
    libblock_device_device_h --> libblock_device_partition_h
    libblock_device_device_h --> libblock_device_part_device_h
    libblock_device_device_h --> libblock_device_block_device_mgr_h
    
    libblock_device_inout_memory_h --> libblock_device_partition_h
    libblock_device_inout_memory_h --> libblock_device_part_device_h
    libblock_device_inout_memory_h --> libblock_device_block_device_mgr_h
    
    libblock_device_partition_h --> libblock_device_part_device_h
    libblock_device_partition_h --> libblock_device_block_device_mgr_h
    
    libblock_device_part_device_h --> libblock_device_block_device_mgr_h

```

- 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.

Namespaces

- namespace [L4](#)
[L4](#) low-level kernel interface.

Macros

- `#define L4_CXX_EXCEPTION_BACKTRACE 20`
Number of instruction pointers in backtrace.

16.154.1 Detailed Description

Base exceptions.

Definition in file [exceptions](#).

16.155 exceptions

[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  *      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/l4types.h>
00022 #include <l4/cxx/basic_ostream>
00023 #include <l4/sys/err.h>
00024 #include <l4/sys/capability>
00025
00026 #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     #if defined(L4_CXX_EXCEPTION_BACKTRACE)
00052     private:
00053         void *_pc_array[L4_CXX_EXCEPTION_BACKTRACE];
00054         int _frame_cnt;
00055     protected:
00056     #if defined(__PIC__)
00057         Exception_tracer() noexcept : _frame_cnt(0) {}
00058     #else
00059         Exception_tracer() noexcept
00060         : _frame_cnt(l4util_backtrace(_pc_array, L4_CXX_EXCEPTION_BACKTRACE)) {}
00061     #endif
00062     public:
00063         void const *const *_pc_array() const noexcept { return _pc_array; }
00064         int frame_count() const noexcept { return _frame_cnt; }
00065     #else
00066     protected:
00067         Exception_tracer() noexcept {}
00068     public:
00069         void const *const *_pc_array() const noexcept { return 0; }
00070         int frame_count() const noexcept { return 0; }
00071     #endif
00072     };
00073
00074     class Base_exception : public Exception_tracer
00075     {
00076     protected:
00077         Base_exception() noexcept {}
00078     public:
00079         virtual char const *str() const throw () = 0;
00080         virtual ~Base_exception() throw () {}
00081     };
00082
00083     class Runtime_error : public Base_exception
00084     {
00085     private:
00086         long _errno;
00087         char _extra[80];
00088     public:
00089         explicit Runtime_error(long err_no, char const *extra = 0) throw ()
00090         : _errno(err_no)
00091         {
00092             if (!extra)
00093                 _extra[0] = 0;
00094             else
00095             {
00096                 unsigned i = 0;
00097                 for (; i < sizeof(_extra) && extra[i]; ++i)
00098                     _extra[i] = extra[i];
00099                 _extra[i < sizeof(_extra) ? i : sizeof(_extra) - 1] = 0;
00100             }
00101         }
00102         char const *str() const throw () override
00103         { return l4sys_errtostr(_errno); }
00104         char const *extra_str() const { return _extra; }
00105         ~Runtime_error() throw () {}
00106         long err_no() const noexcept { return _errno; }
00107     };
00108
00109     class Out_of_memory : public Runtime_error
00110     {
00111     public:
00112         explicit Out_of_memory(char const *extra = "") noexcept
00113         : Runtime_error(-L4_ENOMEM, extra) {}
00114         ~Out_of_memory() noexcept {}
00115     };
00116
00117     class Element_already_exists : public Runtime_error

```

```

00204 {
00205 public:
00206     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     return o;
00307 }
00308
00309 inline
00310 L4::BasicOStream &
00311 operator << (L4::BasicOStream &o, L4::Runtime_error const &e)
00312 {
00313     o << "Exception: " << e.str() << ": ";
00314     if (e.extra_str())
00315         o << e.extra_str() << ": ";
00316     o << "backtrace ...\n";
00317     for (int i = 0; i < e.frame_count(); ++i)
00318         o << L4::n_hex(l4_addr_t(e.pc_array()[i])) << '\n';
00319     return o;
00320 }
00321
00322 }

```

16.156 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 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 #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() : _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 };

```

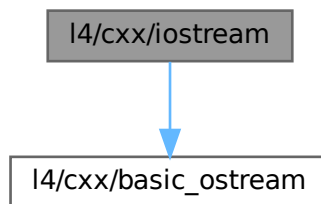


```
00280
00281
00282 }
```

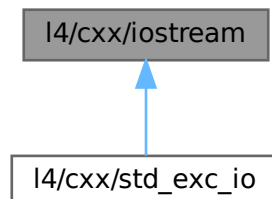
16.157 l4/cxx/iostream File Reference

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.

16.157.1 Detailed Description

IO Stream.

Definition in file [iostream](#).

16.158 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 };
```

16.159 l4/cxx/ipc_helper File Reference

IPC helper.

```
#include <l4/cxx/exceptions>
#include <l4/sys/utcb.h>
```



```

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
00029 namespace L4
00030 {
00031 #ifdef __EXCEPTIONS
00040 inline void
00041 throw_ipc_exception(L4::Cap<void> const &o, l4_msgtag_t const &err,
00042                    l4_utcb_t *utcb)
00043 {
00044     (void)o;
00045     if (err.has_error())
00046         throw (L4::Com_error(l4_error_u(err, utcb)));
00047 }
00048
00057 inline void
00058 throw_ipc_exception(void const *o, l4_msgtag_t const &err,
00059                    l4_utcb_t *utcb)
00060 { throw_ipc_exception(L4::Cap<void>(o), err, utcb); }
00061 #endif
00062
00063 }

```

16.161 l4/cxx/ipc_stream File Reference

IPC stream.

```

#include <l4/sys/ipc.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_varg>
#include <l4/cxx/type_traits>
#include <l4/cxx/minmax>

```


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::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 (Istream &s)`
Read a value out of a stream.

16.161.1 Detailed Description

IPC stream.

Definition in file [ipc_stream](#).

16.161.2 Function Documentation

16.161.2.1 operator<<() [1/4]

```
L4::Ipc::Ostream & operator<< (
    L4::Ipc::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 1202 of file [ipc_stream](#).

References [L4::Ipc::Ostream::put\(\)](#).

Here is the call graph for this function:



16.161.2.2 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 1274 of file [ipc_stream](#).

References [L4::Ipc::Ostream::put\(\)](#).

Here is the call graph for this function:

**16.161.2.3 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 1253 of file [ipc_stream](#).

References [L4::Ipc::Ostream::put\(\)](#).

Here is the call graph for this function:



16.161.2.4 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 L4 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 [1237](#) of file [ipc_stream](#).

References [L4::lpc::Ostream::tag\(\)](#).

Here is the call graph for this function:



16.161.2.5 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.

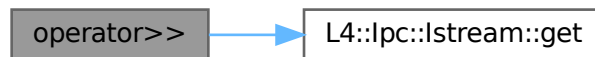
Returns

The stream *s*.

Definition at line 1049 of file `ipc_stream`.

References [L4::Ipc::Istream::get\(\)](#).

Here is the call graph for this function:

**16.161.2.6 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 (<code>Ipc::Buf_cp_out()</code>).

Returns

The stream *s*.

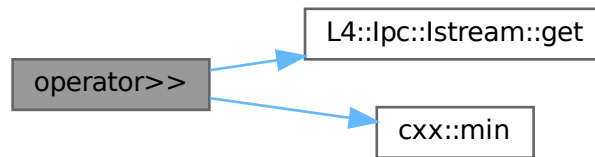
This operator does a copy out of the data into the given buffer.

See `Ipc::Buf_in`, `Ipc::Buf_cp_in`, and `Ipc::Buf_cp_out`.

Definition at line 1156 of file `ipc_stream`.

References [L4::Ipc::Istream::get\(\)](#), and [cxx::min\(\)](#).

Here is the call graph for this function:



16.161.2.7 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

	s	The stream to extract from.
out	v	Pointer to the extracted array (ipc_buf_in()).

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

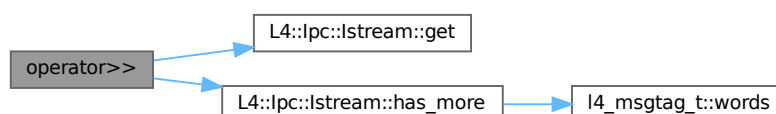
If array does not fit into transmitted words size will be set to zero. Client has to implement check against zero.

See `lpc::Buf_in`, `lpc::Buf_cp_in`, and `lpc::Buf_cp_out`.

Definition at line 1108 of file `ipc_stream`.

References `L4::lpc::Istream::get()`, and `L4::lpc::Istream::has_more()`.

Here is the call graph for this function:



16.161.2.8 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

	<code>s</code>	The stream to extract from.
<code>out</code>	<code>v</code>	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 1135 of file `ipc_stream`.

References `L4::Ipc::Istream::get()`.

Here is the call graph for this function:

**16.161.2.9 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

	<code>s</code>	The stream to extract from.
<code>out</code>	<code>v</code>	Buffer description to copy the array to (<code>Ipc::Str_cp_out()</code>).

Returns

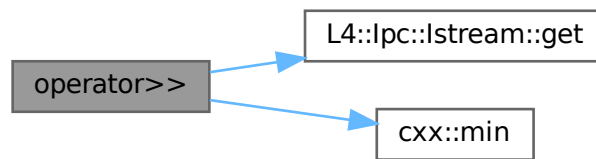
The stream *s*.

This operator does a copy out of the data into the given buffer.

Definition at line 1177 of file [ipc_stream](#).

References [L4::Ipc::Istream::get\(\)](#), and [cxx::min\(\)](#).

Here is the call graph for this function:

**16.161.2.10 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 1083 of file [ipc_stream](#).

References [L4::Ipc::Istream::tag\(\)](#).

Here is the call graph for this function:



16.162 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 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 Utcb_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(Buf_item 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 = (l4_utcb_t *)((l4_addr_t)&_mrs - (l4_addr_t)l4_utcb_mr_u((l4_utcb_t *)0));
00612         _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00613     }
00614
00615 };
00616
00632 class Ostream
00633 {
00634 public:
00638     Ostream(l4_utcb_t *utcb)
00639     : _tag(), _utcb(utcb),
00640       _current_msg(reinterpret_cast<char *>(l4_utcb_mr_u(_utcb)->mr)),
00641       _pos(0), _current_item(0)
00642     {}
00643
00647     void reset()
00648     {
00649         _pos = 0;
00650         _current_item = 0;
00651         _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00652     }
00653
00661
00668     template< typename T >
00669     bool put(T *buf, unsigned long size)
00670     {
00671         size *= sizeof(T);
00672         _pos = cxx::Type_traits<T>::align(_pos);
00673         if (Utc_stream_check::check_utcb_data_offset(_pos + size))
00674             return false;
00675
00676         __builtin_memcpy(_current_msg + _pos, buf, size);
00677         _pos += size;
00678         return true;
00679     }
00680
00686     template< typename T >
00687     bool put(T const &v)
00688     {

```

```

00689     _pos = cxx::Type_traits<T>::align(_pos);
00690     if (Utc_stream_check::check_utcb_data_offset(_pos + sizeof(T)))
00691         return false;
00692
00693     *(reinterpret_cast<T*>(_current_msg + _pos)) = v;
00694     _pos += sizeof(T);
00695     return true;
00696 }
00697
00698 int put(Varg const &va)
00699 {
00700     put(va.tag());
00701     put(va.data(), va.length());
00702
00703     return 0;
00704 }
00705
00706 template< typename T >
00707 int put(Varg_t<T> const &va)
00708 { return put(static_cast<Varg const &>(va)); }
00709
00715 l4_msgtag_t tag() const { return _tag; }
00716
00722 l4_msgtag_t &tag() { return _tag; }
00723
00725
00730 inline bool put_snd_item(Snd_item const &);
00731
00732
00737
00747 inline l4_msgtag_t send(l4_cap_idx_t dst, long proto = 0, unsigned flags = 0);
00748
00750
00754 inline l4_utcb_t *utcb() const { return _utcb; }
00755 #if 0
00759 unsigned long tell() const
00760 {
00761     unsigned w = (_pos + sizeof(l4_umword_t)-1) / sizeof(l4_umword_t);
00762     w -= _current_item * 2;
00763     _tag = l4_msgtag(0, w, _current_item, 0);
00764 }
00765 #endif
00766 public:
00767     l4_msgtag_t prepare_ipc(long proto = 0, unsigned flags = 0)
00768     {
00769         unsigned w = (_pos + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t);
00770         w -= _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_item const &v)
00922 {
00923     typedef Snd_item T;
00924     _pos = cxx::Type_traits<Snd_item>::align(_pos);
00925     if (Utc_b_stream_check::check_utc_b_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(Buf_item const &item)
00937 {
00938     if (_current_buf >= L4_UTCB_GENERIC_BUFFERS_SIZE - 3)
00939         return false;
00940
00941     l4_utcb_br_u(_utcb)->bdr &= ~L4_BDR_OFFSET_MASK;
00942
00943     reinterpret_cast<Buf_item&>(l4_utcb_br_u(_utcb)->br[_current_buf]) = item;
00944     _current_buf += 2;
00945     return true;
00946 }
00947
00948
00949 inline bool
00950 Istream::put(Small_buf const &item)
00951 {
00952     if (_current_buf >= L4_UTCB_GENERIC_BUFFERS_SIZE - 2)
00953         return false;
00954
00955     l4_utcb_br_u(_utcb)->bdr &= ~L4_BDR_OFFSET_MASK;
00956
00957     reinterpret_cast<Small_buf&>(l4_utcb_br_u(_utcb)->br[_current_buf]) = item;
00958     _current_buf += 1;
00959     return true;
00960 }
00961
00962
00963 inline l4_msgtag_t
00964 Ostream::send(l4_cap_idx_t dst, long proto, unsigned flags)
00965 {
00966     l4_msgtag_t tag = prepare_ipc(proto, L4_MSGTAG_FLAGS & flags);
00967     return l4_ipc_send(dst, _utcb, tag, L4_IPC_NEVER);
00968 }
00969
00970 inline l4_msgtag_t
00971 Iostream::call(l4_cap_idx_t dst, l4_timeout_t timeout, long label)
00972 {
00973     l4_msgtag_t tag = prepare_ipc(label);
00974     tag = l4_ipc_call(dst, Ostream::_utcb, tag, timeout);
00975     Istream::tag() = tag;
00976     Istream::_pos = 0;
00977     return tag;
00978 }
00979
00980 inline l4_msgtag_t

```

```

00981 Iostream::call(l4_cap_idx_t dst, long label)
00982 { return call(dst, L4_IPC_NEVER, label); }
00983
00984
00985 inline l4_msgtag_t
00986 Iostream::reply_and_wait(l4_umword_t *src_dst, l4_timeout_t timeout, long proto)
00987 {
00988     l4_msgtag_t tag = prepare_ipc(proto);
00989     tag = l4_ipc_reply_and_wait(Ostream::_utcb, tag, src_dst, timeout);
00990     Istream::tag() = tag;
00991     Istream::_pos = 0;
00992     return tag;
00993 }
00994
00995
00996 inline l4_msgtag_t
00997 Iostream::send_and_wait(l4_cap_idx_t dest, l4_umword_t *src,
00998                        l4_timeout_t timeout, long proto)
00999 {
01000     l4_msgtag_t tag = prepare_ipc(proto);
01001     tag = l4_ipc_send_and_wait(dest, Ostream::_utcb, tag, src, timeout);
01002     Istream::tag() = tag;
01003     Istream::_pos = 0;
01004     return tag;
01005 }
01006
01007 inline l4_msgtag_t
01008 Iostream::reply(l4_timeout_t timeout, long proto)
01009 {
01010     l4_msgtag_t tag = prepare_ipc(proto);
01011     tag = l4_ipc_send(L4_INVALID_CAP | L4_SYSF_REPLY, Ostream::_utcb, tag, timeout);
01012     Istream::tag() = tag;
01013     Istream::_pos = 0;
01014     return tag;
01015 }
01016
01017 inline l4_msgtag_t
01018 Istream::wait(l4_umword_t *src, l4_timeout_t timeout)
01019 {
01020     l4_msgtag_t res;
01021     res = l4_ipc_wait(_utcb, src, timeout);
01022     tag() = res;
01023     _pos = 0;
01024     return res;
01025 }
01026
01027
01028 inline l4_msgtag_t
01029 Istream::receive(l4_cap_idx_t src, l4_timeout_t timeout)
01030 {
01031     l4_msgtag_t res;
01032     res = l4_ipc_receive(src, _utcb, timeout);
01033     tag() = res;
01034     _pos = 0;
01035     return res;
01036 }
01037
01038 } // namespace Ipc
01039 } // namespace L4
01040
01049 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, bool &v) { s.get(v); return s; }
01050 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, int &v) { s.get(v); return s; }
01051 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, long int &v) { s.get(v); return s; }
01052 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, long long int &v) { s.get(v); return s; }
01053 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned int &v) { s.get(v); return s; }
01054 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned long int &v) { s.get(v); return s; }
01055 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned long long int &v) { s.get(v);
    return s; }
01056 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, short int &v) { s.get(v); return s; }
01057 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned short int &v) { s.get(v); return s; }
01058 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, char &v) { s.get(v); return s; }
01059 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned char &v) { s.get(v); return s; }
01060 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, signed char &v) { s.get(v); return s; }
01061 inline L4::Ipc::Istream &operator « (L4::Ipc::Istream &s, L4::Ipc::Buf_item const &v) { s.put(v);
    return s; }
01062 inline L4::Ipc::Istream &operator « (L4::Ipc::Istream &s, L4::Ipc::Small_buf const &v) { s.put(v);
    return s; }
01063 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, L4::Ipc::Snd_item &v)
01064 {
01065     l4_umword_t b, d;
01066     s » b » d;
01067     v = L4::Ipc::Snd_item(b, d);
01068     return s;
01069 }
01070 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, L4::Ipc::Varg &v)

```

```

01071 { s.get(&v); return s; }
01072
01073
01082 inline
01083 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, l4_msgtag_t &v)
01084 {
01085     v = s.tag();
01086     return s;
01087 }
01088
01106 template< typename T >
01107 inline
01108 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
01109                               L4::Ipc::Internal::Buf_in<T> const &v)
01110 {
01111     unsigned long si;
01112     if (s.get(si) && s.has_more<T>(si))
01113         v.set_size(s.get(L4::Ipc::Msg_ptr<T>(v.buf()), si));
01114     else
01115         v.set_size(0);
01116     return s;
01117 }
01118
01133 template< typename T >
01134 inline
01135 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
01136                               L4::Ipc::Msg_ptr<T> const &v)
01137 {
01138     s.get(v);
01139     return s;
01140 }
01141
01154 template< typename T >
01155 inline
01156 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
01157                               L4::Ipc::Internal::Buf_cp_in<T> const &v)
01158 {
01159     unsigned long sz;
01160     s.get(sz);
01161     v.size() = s.get(v.buf(), cxx::min(v.size(), sz));
01162     return s;
01163 }
01164
01175 template< typename T >
01176 inline
01177 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
01178                               L4::Ipc::Str_cp_in<T> const &v)
01179 {
01180     unsigned long sz;
01181     s.get(sz);
01182     unsigned long rsz = s.get(v.buf(), cxx::min(v.size(), sz));
01183     if (rsz < v.size() && v.buf()[rsz - 1])
01184         ++rsz; // add the zero termination behind the received data
01185
01186     if (rsz != 0)
01187         v.buf()[rsz - 1] = 0;
01188
01189     v.size() = rsz;
01190     return s;
01191 }
01192
01193
01202 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, bool v) { s.put(v); return s; }
01203 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, int v) { s.put(v); return s; }
01204 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, long int v) { s.put(v); return s; }
01205 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, long long int v) { s.put(v); return s; }
01206 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned int v) { s.put(v); return s; }
01207 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned long int v) { s.put(v); return s; }
01208 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned long long int v) { s.put(v); return
s; }
01209 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, short int v) { s.put(v); return s; }
01210 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned short int v) { s.put(v); return s;
}
01211 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, char v) { s.put(v); return s; }
01212 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned char v) { s.put(v); return s; }
01213 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, signed char v) { s.put(v); return s; }
01214 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Ipc::Snd_item const &v) {
s.put_snd_item(v); return s; }
01215 template< typename T >
01216 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Cap<T> const &v)
01217 { s « L4::Ipc::Snd_fpage(v.fpage()); return s; }
01218
01219 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Ipc::Varg const &v)
01220 { s.put(v); return s; }
01221 template< typename T >
01222 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Ipc::Varg_t<T> const &v)
01223 { s.put(v); return s; }

```

```

01224
01236 inline
01237 L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, l4_msgtag_t const &v)
01238 {
01239     s.tag() = v;
01240     return s;
01241 }
01242
01251 template< typename T >
01252 inline
01253 L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s,
01254                               L4::Ipc::Internal::Buf_cp_out<T> const &v)
01255 {
01256     s.put(v.size());
01257     s.put(v.buf(), v.size());
01258     return s;
01259 }
01260
01273 inline
01274 L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, char const *v)
01275 {
01276     unsigned long l = __builtin_strlen(v) + 1;
01277     s.put(l);
01278     s.put(v, l);
01279     return s;
01280 }
01281
01282 namespace L4 { namespace Ipc {
01292 template< typename T >
01293 inline
01294 T read(Istream &s) { T t; s » t; return t; }
01295
01296 } // namespace Ipc
01297 } // namespace L4

```

16.163 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 }}

```

16.164 l4/cxx/l4iostream File Reference

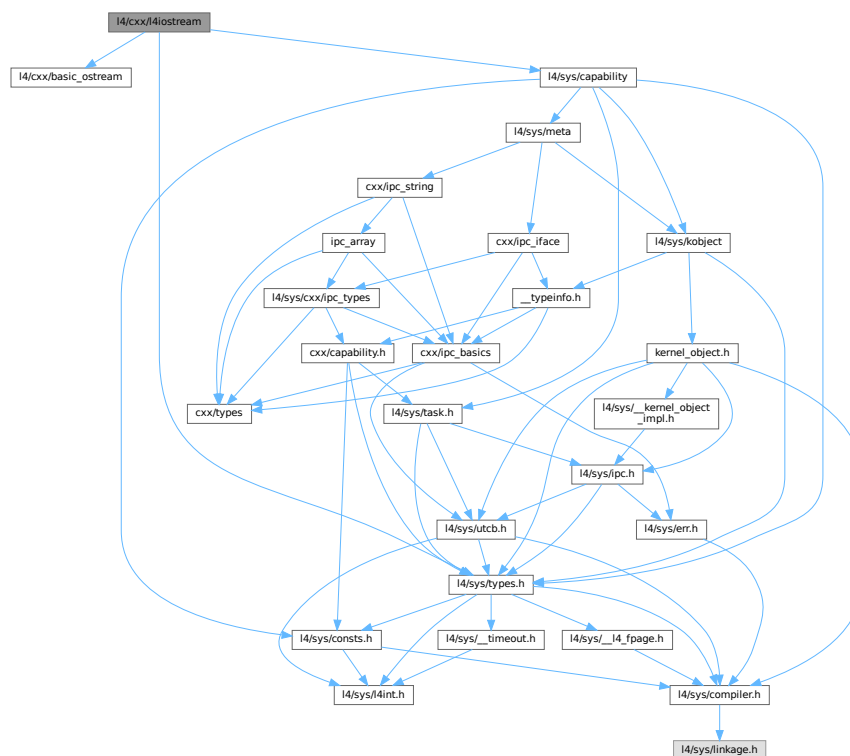
[L4](#) IO stream.

```

#include <l4/cxx/basic_ostream>
#include <l4/sys/types.h>
#include <l4/sys/capability>

```

Include dependency graph for l4iostream:



16.164.1 Detailed Description

[L4](#) IO stream.

Definition in file [l4iostream](#).

16.165 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 }
```

16.166 l4/cxx/l4types.h File Reference

[L4 Types.](#)

```
#include <l4/sys/types.h>
#include <l4/cxx/basic_ostream>
```



```

00001
00002 /*
00003  * (c) 2004-2009 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 <l4/sys/types.h>
00022 #include <l4/cxx/basic_ostream>

```

16.168 list

```

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 <l4/cxx/type_traits>
00022 #include <l4/cxx/std_alloc>
00023 #include <l4/cxx/std_ops>
00024
00025 namespace cxx {
00026 /*
00027  * Classes: List_item, List<D, Alloc>
00028  */
00029
00030 class List_item
00031 {
00032 public:
00033     class Iter
00034     {
00035     public:
00036         Iter(List_item *c, List_item *f) noexcept : _c(c), _f(f) {}
00037         Iter(List_item *f = 0) noexcept : _c(f), _f(f) {}
00038
00039         List_item *operator * () const noexcept { return _c; }
00040         List_item *operator -> () const noexcept { return _c; }
00041         Iter &operator ++ () noexcept
00042         {
00043             if (!_f)
00044                 _c = 0;
00045             else
00046                 _c = _c->get_next_item();
00047
00048             if (_c == _f)
00049                 _c = 0;
00050
00051             return *this;
00052         }
00053
00054         Iter operator ++ (int) noexcept
00055         { Iter o = *this; operator ++ (); return o; }
00056
00057         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
00085     List_item *remove_me() noexcept
00086     {
00087         if (!_c)
00088             return 0;
00089
00090         List_item *l = _c;
00091         operator ++ ();
00092         l->remove_me();
00093
00094         if (_f == l)
00095             _f = _c;
00096
00097         return l;
00098     }
00099
00100 private:
00101     List_item *_c, *_f;
00102 };
00103
00104 template< typename T, bool Poly = false>
00105 class T_iter : public Iter
00106 {
00107 private:
00108     static bool const P = !Conversion<const T*, const List_item *>::exists
00109         || Poly;
00110
00111     static List_item *cast_to_li(T *i, Int_to_type<true>) noexcept
00112     { return dynamic_cast<List_item*>(i); }
00113
00114     static List_item *cast_to_li(T *i, Int_to_type<false>) noexcept
00115     { return i; }
00116
00117     static T *cast_to_type(List_item *i, Int_to_type<true>) noexcept
00118     { return dynamic_cast<T*>(i); }
00119
00120     static T *cast_to_type(List_item *i, Int_to_type<false>) noexcept
00121     { return static_cast<T*>(i); }
00122
00123 public:
00124     template< typename O >
00125     explicit T_iter(T_iter<O> const &o) noexcept
00126     : Iter(o) { dynamic_cast<T*>(&o); }
00127
00128     //T_iter(CListItem *f) : Iter(f) {}
00129     T_iter(T *f = 0) noexcept : Iter(cast_to_li(f, Int_to_type<P>())) {}
00130     T_iter(T *c, T *f) noexcept
00131     : Iter(cast_to_li(c, Int_to_type<P>()),
00132         cast_to_li(f, Int_to_type<P>()))
00133     {}
00134
00135     inline T *operator * () const noexcept
00136     { return cast_to_type(Iter::operator * (), Int_to_type<P>()); }
00137     inline T *operator -> () const noexcept
00138     { return operator * (); }
00139
00140     T_iter<T, Poly> operator ++ (int) noexcept
00141     { T_iter<T, Poly> o = *this; Iter::operator ++ (); return o; }
00142     T_iter<T, Poly> operator -- (int) noexcept
00143     { T_iter<T, Poly> o = *this; Iter::operator -- (); return o; }
00144     T_iter<T, Poly> &operator ++ () noexcept
00145     { Iter::operator ++ (); return *this; }
00146     T_iter<T, Poly> &operator -- () noexcept
00147     { Iter::operator -- (); return *this; }
00148     inline T *remove_me() noexcept;
00149 };
00150
00151 List_item() noexcept : _n(this), _p(this) {}
00152
00153 protected:
00154     List_item(List_item const &) noexcept : _n(this), _p(this) {}
00155

```

```

00171 public:
00173     List_item *get_prev_item() const noexcept { return _p; }
00174
00176     List_item *get_next_item() const noexcept { return _n; }
00177
00179     void insert_prev_item(List_item *p) noexcept
00180     {
00181         p->_p->_n = this;
00182         List_item *pr = p->_p;
00183         p->_p = _p;
00184         _p->_n = p;
00185         _p = pr;
00186     }
00187
00189     void insert_next_item(List_item *p) noexcept
00190     {
00191         p->_p->_n = _n;
00192         p->_p = this;
00193         _n->_p = p;
00194         _n = p;
00195     }
00196
00198     void remove_me() noexcept
00199     {
00200         if (_p != this)
00201         {
00202             _p->_n = _n;
00203             _n->_p = _p;
00204         }
00205         _p = _n = this;
00206     }
00207
00216     template< typename C, typename N >
00217     static inline C *push_back(C *head, N *p) noexcept;
00218
00227     template< typename C, typename N >
00228     static inline C *push_front(C *head, N *p) noexcept;
00229
00238     template< typename C, typename N >
00239     static inline C *remove(C *head, N *p) noexcept;
00240
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     return h;
00284 }
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
00328 template< typename D, template<typename A> class Alloc = New_allocator >
00329 class List
00330 {
00331 private:
00332     class E : public List_item
00333     {
00334     public:
00335         E(D const &d) noexcept : data(d) {}
00336         D data;
00337     };
00338
00339 public:
00340     class Node : private E
00341     {};
00342
00343     typedef Alloc<Node> Node_alloc;
00344
00345     class Iter
00346     {
00347     private:
00348         List_item::T_iter<E> _i;
00349
00350     public:
00351         Iter(E *e) noexcept : _i(e) {}
00352
00353         D &operator * () const noexcept { return (*_i)->data; }
00354         D &operator -> () const noexcept { return (*_i)->data; }
00355
00356         Iter operator ++ (int) noexcept
00357         { Iter o = *this; operator ++ (); return o; }
00358         Iter operator -- (int) noexcept
00359         { Iter o = *this; operator -- (); return o; }
00360         Iter &operator ++ () noexcept { ++_i; return *this; }
00361         Iter &operator -- () noexcept { --_i; return *this; }
00362
00363         operator E* () const noexcept { return *_i; }
00364     };
00365
00366     List(Alloc<Node> const &a = Alloc<Node>()) noexcept : _h(0), _l(0), _a(a) {}
00367
00368     void push_back(D const &d) noexcept
00369     {
00370         void *n = _a.alloc();
00371         if (!n) return;
00372         _h = E::push_back(_h, new (n) E(d));
00373         ++_l;
00374     }

```

```

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

```

16.169 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
00026 namespace cxx {
00027
00031 class List_alloc
00032 {
00033 private:
00034     friend class List_alloc_sanity_guard;
00035
00036     struct Mem_block
00037     {
00038         Mem_block *next;
00039         unsigned long size;
00040     };
00041
00042     Mem_block *_first;
00043
00044     inline void check_overlap(void *, unsigned long );
00045     inline void sanity_check_list(char const *, char const *);
00046     inline void merge();
00047
00048 public:
00049
00056     List_alloc() : _first(0) {}
00057

```



```

00070 inline void free(void *block, unsigned long size, bool initial_free = false);
00071
00082 inline void *alloc(unsigned long size, unsigned long align);
00083
00099 inline void *alloc_max(unsigned long min, unsigned long *max,
00100                        unsigned long align, unsigned long granularity);
00101
00107 inline unsigned long avail();
00108
00109 template <typename DBG>
00110 void dump_free_list(DBG &out);
00111 };
00112
00113 #if !defined (CXX_LIST_ALLOC_SANITY)
00114 class List_alloc_sanity_guard
00115 {
00116 public:
00117     List_alloc_sanity_guard(List_alloc *, char const *)
00118     {}
00119
00120 };
00121
00122 void
00123 List_alloc::check_overlap(void *, unsigned long )
00124 {}
00125
00126 void
00127 List_alloc::sanity_check_list(char const *, char const *)
00128 {}
00129
00130 #else
00131
00132 class List_alloc_sanity_guard
00133 {
00134 private:
00135     List_alloc *a;
00136     char const *func;
00137
00138 public:
00139     List_alloc_sanity_guard(List_alloc *a, char const *func)
00140         : a(a), func(func)
00141     { a->sanity_check_list(func, "entry"); }
00142
00143     ~List_alloc_sanity_guard()
00144     { a->sanity_check_list(func, "exit"); }
00145 };
00146
00147 void
00148 List_alloc::check_overlap(void *b, unsigned long s)
00149 {
00150     unsigned long const mb_align = (1UL << arith::Ld<sizeof(Mem_block)>::value) - 1;
00151     if ((unsigned long)b & mb_align)
00152     {
00153         L4::cerr << "List_alloc(FATAL): trying to free unaligned memory: "
00154                 << b << " align=" << arith::Ld<sizeof(Mem_block)>::value << "\n";
00155     }
00156
00157     Mem_block *c = _first;
00158     for (; c ; c = c->next)
00159     {
00160         unsigned long x_s = (unsigned long)b;
00161         unsigned long x_e = x_s + s;
00162         unsigned long b_s = (unsigned long)c;
00163         unsigned long b_e = b_s + c->size;
00164
00165         if ((x_s >= b_s && x_s < b_e)
00166             || (x_e > b_s && x_e <= b_e)
00167             || (b_s >= x_s && b_s < x_e)
00168             || (b_e > x_s && b_e <= x_e))
00169         {
00170             L4::cerr << "List_alloc(FATAL): trying to free memory that "
00171                     << "is already free: \n ["
00172                     << (void*)x_s << '-' << (void*)x_e << " overlaps ["
00173                     << (void*)b_s << '-' << (void*)b_e << "]\n";
00174         }
00175     }
00176 }
00177
00178 void
00179 List_alloc::sanity_check_list(char const *func, char const *info)
00180 {
00181     Mem_block *c = _first;
00182     for (; c ; c = c->next)
00183     {
00184         if (c->next)
00185         {

```

```

00187         if (c >= c->next)
00188         {
00189             L4::cerr << "List_alloc(FATAL): " << func << ' (' << info
00190                 << "): list order violation\n";
00191         }
00192
00193         if (((unsigned long)c) + c->size > (unsigned long)c->next)
00194         {
00195             L4::cerr << "List_alloc(FATAL): " << func << ' (' << info
00196                 << "): list order violation\n";
00197         }
00198     }
00199 }
00200 }
00201
00202 #endif
00203
00204 void
00205 List_alloc::merge()
00206 {
00207     List_alloc_sanity_guard __attribute__((unused)) guard(this, __func__);
00208     Mem_block *c = _first;
00209     while (c && c->next)
00210     {
00211         unsigned long f_start = (unsigned long)c;
00212         unsigned long f_end   = f_start + c->size;
00213         unsigned long n_start = (unsigned long)c->next;
00214
00215         if (f_end == n_start)
00216         {
00217             c->size += c->next->size;
00218             c->next = c->next->next;
00219             continue;
00220         }
00221
00222         c = c->next;
00223     }
00224 }
00225
00226 void
00227 List_alloc::free(void *block, unsigned long size, bool initial_free)
00228 {
00229     List_alloc_sanity_guard __attribute__((unused)) guard(this, __func__);
00230
00231     unsigned long const mb_align = (1UL << arith::Ld<sizeof(Mem_block)>::value) - 1;
00232
00233     if (initial_free)
00234     {
00235         // enforce alignment constraint on initial memory
00236         unsigned long nblock = ((unsigned long)block + mb_align) & ~mb_align;
00237         size = (size - (nblock - (unsigned long)block)) & ~mb_align;
00238         block = (void*)nblock;
00239     }
00240     else
00241         // blow up size to the minimum aligned size
00242         size = (size + mb_align) & ~mb_align;
00243
00244     check_overlap(block, size);
00245
00246     Mem_block **c = &_first;
00247     Mem_block *next = 0;
00248
00249     if (*c)
00250     {
00251         while (*c && *c < block)
00252             c = &(*c)->next;
00253
00254         next = *c;
00255     }
00256
00257     *c = (Mem_block*)block;
00258
00259     (*c)->next = next;
00260     (*c)->size = size;
00261
00262     merge();
00263 }
00264
00265 void *
00266 List_alloc::alloc_max(unsigned long min, unsigned long *max, unsigned long align,
00267                     unsigned granularity)
00268 {
00269     List_alloc_sanity_guard __attribute__((unused)) guard(this, __func__);
00270
00271     unsigned char const mb_bits = arith::Ld<sizeof(Mem_block)>::value;
00272     unsigned long const mb_align = (1UL << mb_bits) - 1;
00273

```

```

00274 // blow minimum up to at least the minimum aligned size of a Mem_block
00275 min = l4_round_size(min, mb_bits);
00276 // truncate maximum to at least the size of a Mem_block
00277 *max = l4_trunc_size(*max, mb_bits);
00278 // truncate maximum size according to granularity
00279 *max = *max & ~(granularity - 1UL);
00280
00281 if (min > *max)
00282     return 0;
00283
00284 unsigned long almask = align ? (align - 1UL) : 0;
00285
00286 // minimum alignment is given by the size of a Mem_block
00287 if (almask < mb_align)
00288     almask = mb_align;
00289
00290 Mem_block **c = &_first;
00291 Mem_block **fit = 0;
00292 unsigned long max_fit = 0;
00293
00294 for (; *c; c = &(*c)->next)
00295 {
00296     // address of free memory block
00297     unsigned long n_start = (unsigned long)(*c);
00298
00299     // block too small, next
00300     // XXX: maybe we can skip this and just do the test below
00301     if ((*c)->size < min)
00302         continue;
00303
00304     // aligned start address within the free block
00305     unsigned long a_start = (n_start + almask) & ~almask;
00306
00307     // check if aligned start address is behind the block, next
00308     if (a_start - n_start >= (*c)->size)
00309         continue;
00310
00311     // remaining size after subtracting the padding for the alignment
00312     unsigned long r_size = (*c)->size - a_start + n_start;
00313     // round down according to granularity
00314     r_size &= ~(granularity - 1UL);
00315
00316     // block too small
00317     if (r_size < min)
00318         continue;
00319
00320     if (r_size >= *max)
00321     {
00322         fit = c;
00323         max_fit = *max;
00324         break;
00325     }
00326
00327     if (r_size > max_fit)
00328     {
00329         max_fit = r_size;
00330         fit = c;
00331     }
00332 }
00333
00334 if (fit)
00335 {
00336     unsigned long n_start = (unsigned long)(*fit);
00337     unsigned long a_start = (n_start + almask) & ~almask;
00338     unsigned long r_size = (*fit)->size - a_start + n_start;
00339
00340     if (a_start > n_start)
00341     {
00342         (*fit)->size -= r_size;
00343         fit = &(*fit)->next;
00344     }
00345     else
00346         *fit = (*fit)->next;
00347
00348     *max = max_fit;
00349     if (r_size == max_fit)
00350         return (void *)a_start;
00351
00352     Mem_block *m = (Mem_block*)(a_start + max_fit);
00353     m->next = *fit;
00354     m->size = r_size - max_fit;
00355     *fit = m;
00356     return (void *)a_start;
00357 }
00358
00359 return 0;
00360 }

```

```

00361
00362 void *
00363 List_alloc::alloc(unsigned long size, unsigned long align)
00364 {
00365     List_alloc_sanity_guard __attribute__((unused)) guard(this, __func__);
00366
00367     unsigned long const mb_align = (1UL « arith::Ld<sizeof(Mem_block)>::value) - 1;
00368
00369     // blow up size to the minimum aligned size
00370     size = (size + mb_align) & ~mb_align;
00371
00372     unsigned long almask = align ? (align - 1UL) : 0;
00373
00374     // minimum alignment is given by the size of a Mem_block
00375     if (almask < mb_align)
00376         almask = mb_align;
00377
00378     Mem_block **c = &_first;
00379
00380     for (; *c; c=&(*c)->next)
00381     {
00382         // address of free memory block
00383         unsigned long n_start = (unsigned long)(*c);
00384
00385         // block too small, next
00386         // XXX: maybe we can skip this and just do the test below
00387         if ((*c)->size < size)
00388             continue;
00389
00390         // aligned start address within the free block
00391         unsigned long a_start = (n_start + almask) & ~almask;
00392
00393         // block too small after alignment, next
00394         if (a_start - n_start >= (*c)->size)
00395             continue;
00396
00397         // remaining size after subtracting the padding
00398         // for the alignment
00399         unsigned long r_size = (*c)->size - a_start + n_start;
00400
00401         // block too small
00402         if (r_size < size)
00403             continue;
00404
00405         if (a_start > n_start)
00406         {
00407             // have free space before the allocated block
00408             // shrink the block and set c to the next pointer of that
00409             // block
00410             (*c)->size -= r_size;
00411             c = &(*c)->next;
00412         }
00413         else
00414             // drop the block, c remains the next pointer of the
00415             // previous block
00416             *c = (*c)->next;
00417
00418         // allocated the whole remaining space
00419         if (r_size == size)
00420             return (void*)a_start;
00421
00422         // add a new free block behind the allocated block
00423         Mem_block *m = (Mem_block*)(a_start + size);
00424         m->next = *c;
00425         m->size = r_size - size;
00426         *c = m;
00427         return (void *)a_start;
00428     }
00429
00430     return 0;
00431 }
00432
00433 unsigned long
00434 List_alloc::avail()
00435 {
00436     List_alloc_sanity_guard __attribute__((unused)) guard(this, __FUNCTION__);
00437     Mem_block *c = _first;
00438     unsigned long a = 0;
00439     while (c)
00440     {
00441         a += c->size;
00442         c = c->next;
00443     }
00444
00445     return a;
00446 }
00447

```

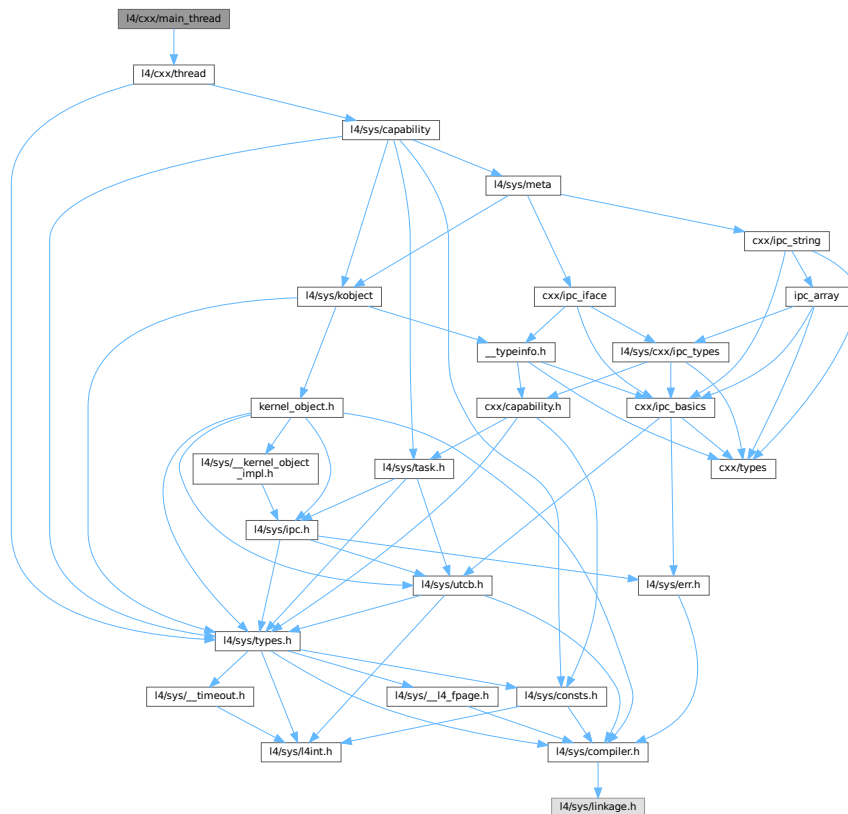
```
00448 template <typename DBG>
00449 void
00450 List_alloc::dump_free_list(DBG &out)
00451 {
00452     Mem_block *c = _first;
00453     while (c)
00454     {
00455         unsigned sz;
00456         const char *unit;
00457         if (c->size < 1024)
00458         {
00459             sz = c->size;
00460             unit = "Byte";
00461         }
00462         else if (c->size < 1 « 20)
00463         {
00464             sz = c->size » 10;
00465             unit = "kB";
00466         }
00467         else
00468         {
00469             sz = c->size » 20;
00470             unit = "MB";
00471         }
00472         out.printf("%12p - %12p (%u %s)\n", c, (char *) c + c->size - 1, sz, unit);
00473         c = c->next;
00474     }
00475 }
00476 }
00477 }
00478 }
00479 }
00480 }
```

16.170 I4/cxx/main_thread File Reference

Main thread.

```
#include <l4/cxx/thread>
```

Include dependency graph for `main_thread`:



Namespaces

- namespace `cxx`
Our C++ library.

16.170.1 Detailed Description

Main thread.

Definition in file [main_thread](#).

16.171 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__ */

```

16.172 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
00025 namespace cxx
00026 {
00033     template< typename T1 >
00034     inline
00035     T1 min(T1 a, T1 b)
00036     { return a < b ? a : b; }
00037
00044     template< typename T1 >
00045     inline
00046     T1 max(T1 a, T1 b)
00047     { return a > b ? a : b; }
00048
00056     template< typename T1 >
00057     inline
00058     T1 clamp(T1 v, T1 lo, T1 hi)
00059     { return min(hi, max(lo, v)); }
00060 };

```

16.173 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  */

```

```

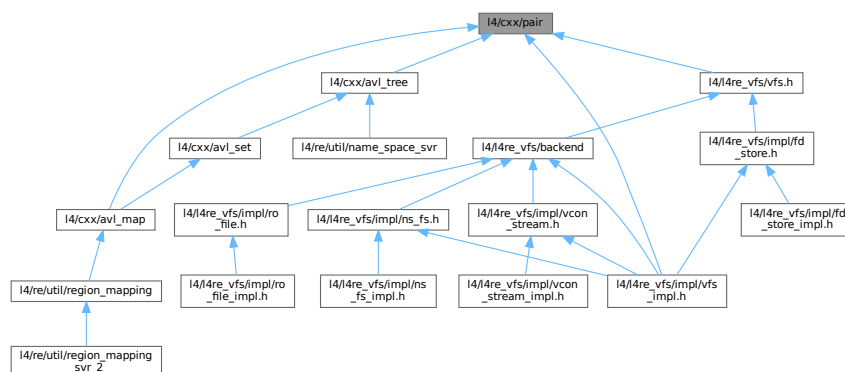
00010 #pragma once
00011
00012 #include <l4/cxx/hlist>
00013
00014 namespace cxx {
00015
00016 class Observer : public H_list_item
00017 {
00018 public:
00019     virtual void notify() = 0;
00020 };
00021
00022 class Notifier : public H_list<Observer>
00023 {
00024 public:
00025     void notify()
00026     {
00027         for (Iterator i = begin(); i != end(); ++i)
00028             i->notify();
00029     }
00030 };
00031
00032 }
00033
00034

```

16.174 I4/cxx/pair File Reference

Pair implementation.

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.

16.174.1 Detailed Description

Pair implementation.

Definition in file [pair](#).

16.175 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 namespace cxx {
00026
00035 template< typename First, typename Second >
00036 struct Pair
00037 {
00039     typedef First First_type;
00041     typedef Second Second_type;
00042
00044     First first;
00046     Second second;
00047
00053     template<typename A1, typename A2>
00054     Pair(A1 &&first, A2 &&second)
00055     : first(first), second(second) {}
00056
00058     Pair() = default;
00059 };
00060
00061 template< typename F, typename S >
00062 Pair<F,S> pair(F const &f, S const &s)
00063 { return cxx::Pair<F,S>(f,s); }
00064
00065
00074 template< typename Cmp, typename Typ >
00075 class Pair_first_compare
00076 {
00077 private:
00078     Cmp const &_cmp;
00079
00080 public:
00085     Pair_first_compare(Cmp const &cmp = Cmp()) : _cmp(cmp) {}
00086
00092     bool operator () (Typ const &l, Typ const &r) const
00093     { return _cmp(l.first,r.first); }
00094 };
00095
00096 }
00097
00098 template< typename OS, typename A, typename B >
00099 inline
00100 OS &operator << (OS &os, cxx::Pair<A,B> const &p)
00101 {
00102     os << p.first << ' ' << p.second;
00103     return os;
00104 }
00105
```

16.176 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
00077 template <
00078     typename T = void,
00079     template< typename X > class CNT = Default_ref_counter
00080 >
00081 class Ref_ptr : public Ref_ptr_base, private CNT<T>
00082 {
00083 private:
00084     typedef decltype(nullptr) Null_type;
00085     typedef Weak_ptr<T, CNT> Wp;
00086
00087 public:
00088     Ref_ptr() noexcept : _p(0) {}
00089
00090     Ref_ptr(Ref_ptr_base::Default_value v) : _p((T*)v) {}
00091
00092     Ref_ptr(Wp const &o) noexcept : _p(o.ptr())
00093     { __take_ref(); }
00094
00095     Ref_ptr(decltype(nullptr) n) noexcept : _p(n) {}
00096
00097     template<typename X>
00098     explicit Ref_ptr(X *o) noexcept : _p(o)
00099     { __take_ref(); }
00100
00101     Ref_ptr(T *o, bool d) noexcept : _p(o) { (void)d; }
00102
00103     T *get() const noexcept
00104     {
00105         return _p;
00106     }
00107
00108     T *ptr() const noexcept
00109     {
00110         return _p;
00111     }

```

```

00140     }
00141
00148     T *release() noexcept
00149     {
00150         T *p = _p;
00151         _p = 0;
00152         return p;
00153     }
00154
00155     ~Ref_ptr() noexcept
00156     { __drop_ref(); }
00157
00158     template<typename OT>
00159     Ref_ptr(Ref_ptr<OT, CNT> const &o) noexcept
00160     {
00161         _p = o.ptr();
00162         __take_ref();
00163     }
00164
00165     Ref_ptr(Ref_ptr<T> const &o) noexcept
00166     {
00167         _p = o._p;
00168         __take_ref();
00169     }
00170
00171     template< typename OT >
00172     void operator = (Ref_ptr<OT> const &o) noexcept
00173     {
00174         __drop_ref();
00175         _p = o.ptr();
00176         __take_ref();
00177     }
00178
00179     void operator = (Ref_ptr<T> const &o) noexcept
00180     {
00181         if (&o == this)
00182             return;
00183
00184         __drop_ref();
00185         _p = o._p;
00186         __take_ref();
00187     }
00188
00189     void operator = (Null_type) noexcept
00190     {
00191         __drop_ref();
00192         _p = 0;
00193     }
00194
00195     template<typename OT>
00196     Ref_ptr(Ref_ptr<OT, CNT> &&o) noexcept
00197     { _p = o.release(); }
00198
00199     Ref_ptr(Ref_ptr<T> &&o) noexcept
00200     { _p = o.release(); }
00201
00202     template< typename OT >
00203     void operator = (Ref_ptr<OT> &&o) noexcept
00204     {
00205         __drop_ref();
00206         _p = o.release();
00207     }
00208
00209     void operator = (Ref_ptr<T> &&o) noexcept
00210     {
00211         if (&o == this)
00212             return;
00213
00214         __drop_ref();
00215         _p = o.release();
00216     }
00217
00218     explicit operator bool () const noexcept { return _p; }
00219
00220     T *operator -> () const noexcept
00221     { return _p; }
00222
00223     bool operator == (Ref_ptr const &o) const noexcept
00224     { return _p == o._p; }
00225
00226     bool operator != (Ref_ptr const &o) const noexcept
00227     { return _p != o._p; }
00228
00229     bool operator < (Ref_ptr const &o) const noexcept
00230     { return _p < o._p; }
00231
00232     bool operator <= (Ref_ptr const &o) const noexcept

```

```

00233     { return _p <= o._p; }
00234
00235     bool operator > (Ref_ptr const &o) const noexcept
00236     { return _p > o._p; }
00237
00238     bool operator >= (Ref_ptr const &o) const noexcept
00239     { return _p >= o._p; }
00240
00241     bool operator == (T const *o) const noexcept
00242     { return _p == o; }
00243
00244     bool operator < (T const *o) const noexcept
00245     { return _p < o; }
00246
00247     bool operator <= (T const *o) const noexcept
00248     { return _p <= o; }
00249
00250     bool operator > (T const *o) const noexcept
00251     { return _p > o; }
00252
00253     bool operator >= (T const *o) const noexcept
00254     { return _p >= o; }
00255
00256 private:
00257     void __drop_ref() noexcept
00258     {
00259         if (_p)
00260             static_cast<CNT<T*>>(this)->h_drop_ref(_p);
00261     }
00262
00263     void __take_ref() noexcept
00264     {
00265         if (_p)
00266             static_cast<CNT<T*>>(this)->h_take_ref(_p);
00267     }
00268
00269     T *_p;
00270 };
00271
00272
00273 template<typename T, template< typename X > class CNT>
00274 class Weak_ptr
00275 {
00276 private:
00277     struct Null_type;
00278     typedef Ref_ptr<T, CNT> Rp;
00279
00280 public:
00281     Weak_ptr() = default;
00282     Weak_ptr(decltype(nullptr)) : _p(nullptr) {}
00283     // backwards 0 ctor
00284     explicit Weak_ptr(int x) noexcept
00285     L4_DEPRECATED("Use initialization from 'nullptr'")
00286     : _p(nullptr)
00287     { if (x != 0) __builtin_trap(); }
00288
00289     Weak_ptr(Rp const &o) noexcept : _p(o.ptr()) {}
00290     explicit Weak_ptr(T *o) noexcept : _p(o) {}
00291
00292     template<typename OT>
00293     Weak_ptr(Weak_ptr<OT, CNT> const &o) noexcept : _p(o.ptr()) {}
00294
00295     Weak_ptr(Weak_ptr<T, CNT> const &o) noexcept : _p(o._p) {}
00296
00297     Weak_ptr<T, CNT> &operator = (const Weak_ptr<T, CNT> &o) = default;
00298
00299     T *get() const noexcept { return _p; }
00300     T *ptr() const noexcept { return _p; }
00301
00302     T *operator -> () const noexcept { return _p; }
00303     operator Null_type const * () const noexcept
00304     { return reinterpret_cast<Null_type const*>(_p); }
00305
00306 private:
00307     T *_p;
00308 };
00309
00310 template<typename OT, typename T> inline
00311 Ref_ptr<OT> ref_ptr_static_cast(Ref_ptr<T> const &o)
00312 { return ref_ptr(static_cast<OT*>(o.ptr())); }
00313
00314 template< typename T >
00315 inline Ref_ptr<T> ref_ptr(T *t)
00316 { return Ref_ptr<T>(t); }
00317
00318 template< typename T >
00319 inline Weak_ptr<T> weak_ptr(T *t)

```

```

00320 { return Weak_ptr<T>(t); }
00321
00322
00323 class Ref_obj
00324 {
00325 private:
00326     mutable int _ref_cnt;
00327
00328 public:
00329     Ref_obj() : _ref_cnt(0) {}
00330     void add_ref() const noexcept { ++_ref_cnt; }
00331     int remove_ref() const noexcept { return --_ref_cnt; }
00332 };
00333
00334 template< typename T, typename... Args >
00335 Ref_ptr<T>
00336 make_ref_obj(Args &&... args)
00337 { return cxx::Ref_ptr<T>(new T(cxx::forward<Args>(args)...)); }
00338
00339 template<typename T, typename U>
00340 Ref_ptr<T>
00341 dynamic_pointer_cast(Ref_ptr<U> const &p) noexcept
00342 {
00343     // our constructor from a naked pointer increments the counter
00344     return Ref_ptr<T>(dynamic_cast<T *>(p.get()));
00345 }
00346
00347 template<typename T, typename U>
00348 Ref_ptr<T>
00349 static_pointer_cast(Ref_ptr<U> const &p) noexcept
00350 {
00351     // our constructor from a naked pointer increments the counter
00352     return Ref_ptr<T>(static_cast<T *>(p.get()));
00353 }
00354
00355 }

```

16.177 l4/cxx/ref_ptr_list File Reference

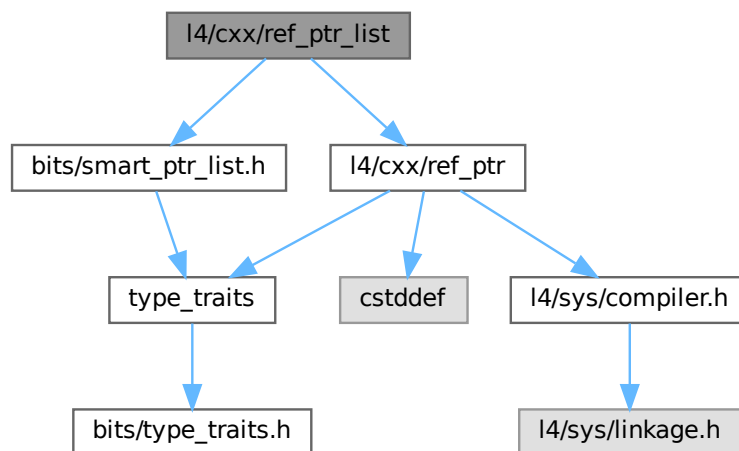
Implementation of a list of ref-ptr-managed objects.

```

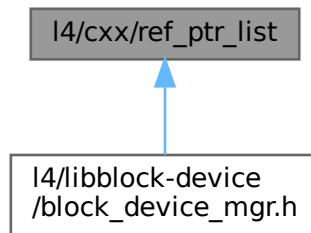
#include <l4/cxx/ref_ptr>
#include "bits/smart_ptr_list.h"

```

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`.

16.177.1 Detailed Description

Implementation of a list of ref-ptr-managed objects.

Definition in file `ref_ptr_list`.

16.178 ref_ptr_list

[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  */
00009 #pragma once
00010
00011 #include <l4/cxx/ref_ptr>
00012
00013 #include "bits/smart_ptr_list.h"
00014
00015 namespace cxx {
00016
00017 template <typename T>
00018 using Ref_ptr_list_item = Bits::Smart_ptr_list_item<T, cxx::Ref_ptr<T> >;
00019
00020 template <typename T>
00021 struct Ref_obj_list_item : public Ref_ptr_list_item<T>, public cxx::Ref_obj {};
00022
00023 template <typename T>
00024 using Ref_ptr_list = Bits::Smart_ptr_list<Ref_ptr_list_item<T> >;
00025
00026 }
00027
```

16.179 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 #pragma once
00021
00022 #include <l4/cxx/std_alloc>
00023 #include <l4/cxx/hlist>
00024 #include <l4/sys/consts.h>
00025
00026 namespace cxx {
00027
00028 template< int Obj_size, int Slab_size = L4_PAGESIZE,
00029         int Max_free = 2, template<typename A> class Alloc = New_allocator >
00030 class Base_slab
00031 {
00032 private:
00033     struct Free_o
00034     {
00035         Free_o *next;
00036     };
00037
00038 protected:
00039     struct Slab_i;
00040
00041 private:
00042     struct Slab_head : public H_list_item
00043     {
00044         unsigned num_free;
00045         Free_o *free;
00046         Base_slab<Obj_size, Slab_size, Max_free, Alloc> *cache;
00047
00048         inline Slab_head() noexcept : num_free(0), free(0), cache(0)
00049         {}
00050     };
00051
00052 }
00053
```

```

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     {
00078         object_size      = Obj_size,
00079         slab_size        = Slab_size,
00080         objects_per_slab = (Slab_size - sizeof(Slab_head)) / object_size,
00081         max_free_slabs   = Max_free,
00082     };
00083
00084 protected:
00085     struct Slab_store
00086     {
00087         char _o[slab_size - sizeof(Slab_head)];
00088         Free_o *object(unsigned obj) noexcept
00089         { return reinterpret_cast<Free_o*>(_o + object_size * obj); }
00090     };
00091
00092     struct Slab_i : public Slab_store, public Slab_head
00093     {};
00094
00095 public:
00096     typedef Alloc<Slab_i> Slab_alloc;
00097
00098     typedef void Obj_type;
00099
00100 private:
00101     Slab_alloc _alloc;
00102     unsigned _num_free;
00103     unsigned _num_slabs;
00104     H_list<Slab_i> _full_slabs;
00105     H_list<Slab_i> _partial_slabs;
00106     H_list<Slab_i> _empty_slabs;
00107
00108     void add_slab(Slab_i *s) noexcept
00109     {
00110         s->num_free = objects_per_slab;
00111         s->cache = this;
00112
00113         //L4::cerr << "Slab: " << this << "->add_slab(" << s << ", size="
00114         // << slab_size << "):" << " f=" << s->object(0) << '\n';
00115
00116         // initialize free list
00117         Free_o *f = s->free = s->object(0);
00118         for (unsigned i = 1; i < objects_per_slab; ++i)
00119         {
00120             f->next = s->object(i);
00121             f = f->next;
00122         }
00123         f->next = 0;
00124
00125         // insert slab into cache's list
00126         _empty_slabs.push_front(s);
00127         ++_num_slabs;
00128         ++_num_free;
00129     }
00130
00131     bool grow() noexcept
00132     {
00133         Slab_i *s = _alloc.alloc();
00134         if (!s)
00135             return false;
00136
00137         new (s, cxx::Nothrow()) Slab_i();
00138
00139         add_slab(s);
00140         return true;
00141     }
00142
00143     void shrink() noexcept
00144     {
00145         if (!_alloc.can_free)
00146             return;
00147
00148         while (!_empty_slabs.empty() && _num_free > max_free_slabs)
00149         {
00150             Slab_i *s = _empty_slabs.front();

```



```

00174     _empty_slabs.remove(s);
00175     --_num_free;
00176     --_num_slabs;
00177     _alloc.free(s);
00178     }
00179     }
00180
00181 public:
00182     Base_slab(Slab_alloc const &alloc = Slab_alloc()) noexcept
00183         : _alloc(alloc), _num_free(0), _num_slabs(0), _full_slabs(),
00184           _partial_slabs(), _empty_slabs()
00185     {}
00186
00187     ~Base_slab() noexcept
00188     {
00189         while (!_empty_slabs.empty())
00190         {
00191             Slab_i *o = _empty_slabs.front();
00192             _empty_slabs.remove(o);
00193             _alloc.free(o);
00194         }
00195         while (!_partial_slabs.empty())
00196         {
00197             Slab_i *o = _partial_slabs.front();
00198             _partial_slabs.remove(o);
00199             _alloc.free(o);
00200         }
00201         while (!_full_slabs.empty())
00202         {
00203             Slab_i *o = _full_slabs.front();
00204             _full_slabs.remove(o);
00205             _alloc.free(o);
00206         }
00207     }
00208
00218 void *alloc() noexcept
00219 {
00220     H_list<Slab_i> *free = &_partial_slabs;
00221     if (free->empty())
00222         free = &_empty_slabs;
00223
00224     if (free->empty() && !grow())
00225         return 0;
00226
00227     Slab_i *s = free->front();
00228     Free_o *o = s->free;
00229     s->free = o->next;
00230
00231     if (free == &_empty_slabs)
00232     {
00233         _empty_slabs.remove(s);
00234         --_num_free;
00235     }
00236
00237     --(s->num_free);
00238
00239     if (!s->free)
00240     {
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 = (unsigned long)_o;
00263
00264     // find out the slab the object is in
00265     addr = (addr / slab_size) * slab_size;
00266     Slab_i *s = (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         _empty_slabs.push_front(s);
00291         ++_num_free;
00292         if (_num_free > max_free_slabs)
00293             shrink();
00294     }
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 }

```

16.180 slist

```

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 #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     explicit S_list_item(bool) {}
00031
00032 private:
00033     template<typename T, typename P> friend class S_list;
00034     template<typename T, typename P> friend class S_list_tail;
00035     template<typename T, typename X> friend struct Bits::Basic_list_policy;
00036
00037     S_list_item(S_list_item const &);
00038     void operator = (S_list_item const &);
00039
00040     S_list_item *_n;
00041 };
00042
00049 template< typename T, typename POLICY = Bits::Basic_list_policy< T, S_list_item > >
00050 class S_list : public Bits::Basic_list<POLICY>
00051 {
00052     S_list(S_list const &) = delete;
00053     void operator = (S_list const &) = delete;

```

```

00054
00055 private:
00056     typedef typename Bits::Basic_list<POLICY> Base;
00057
00058 public:
00059     typedef typename Base::Iterator Iterator;
00060
00061     S_list(S_list &&o) : Base(static_cast<Base&&>(o)) {}
00062
00063     S_list &operator = (S_list &&o)
00064     {
00065         Base::operator = (static_cast<Base&&>(o));
00066         return *this;
00067     }
00068
00069     // BSS allocation
00070     explicit S_list(bool x) : Base(x) {}
00071
00072     S_list() : Base() {}
00073
00075 void add(T *e)
00076 {
00077     e->_n = this->_f;
00078     this->_f = e;
00079 }
00080
00081 template< typename CAS >
00082 void add(T *e, CAS const &c)
00083 {
00084     do
00085     {
00086         e->_n = this->_f;
00087     }
00088     while (!c(&this->_f, e->_n, e));
00089 }
00090
00092 void push_front(T *e) { add(e); }
00093
00099 T *pop_front()
00100 {
00101     T *r = this->front();
00102     if (this->_f)
00103         this->_f = this->_f->_n;
00104     return r;
00105 }
00106
00107 void insert(T *e, Iterator const &pred)
00108 {
00109     S_list_item *p = *pred;
00110     e->_n = p->_n;
00111     p->_n = e;
00112 }
00113
00114 static void insert_before(T *e, Iterator const &succ)
00115 {
00116     S_list_item **x = Base::__get_internal(succ);
00117
00118     e->_n = *x;
00119     *x = e;
00120 }
00121
00122 static void replace(Iterator const &p, T*e)
00123 {
00124     S_list_item **x = Base::__get_internal(p);
00125     e->_n = (*x)->_n;
00126     *x = e;
00127 }
00128
00129 static Iterator erase(Iterator const &e)
00130 {
00131     S_list_item **x = Base::__get_internal(e);
00132     *x = (*x)->_n;
00133     return e;
00134 }
00135
00136 };
00137
00138
00139 template< typename T >
00140 class S_list_bss : public S_list<T>
00141 {
00142 public:
00143     S_list_bss() : S_list<T>(true) {}
00144 };
00145
00146 template< typename T, typename POLICY = Bits::Basic_list_policy< T, S_list_item > >
00147 class S_list_tail : public S_list<T, POLICY>

```

```

00148 {
00149 private:
00150     typedef S_list<T, POLICY> Base;
00151     void add(T *e) = delete;
00152
00153 public:
00154     using Iterator = typename Base::Iterator;
00155     S_list_tail() : Base(), _tail(&this->_f) {}
00156
00157     S_list_tail(S_list_tail &t)
00158     : Base(static_cast<Base&&>(t)), _tail(t.empty() ? &this->_f : t._tail)
00159     {
00160         t._tail = &t._f;
00161     }
00162
00163     S_list_tail &operator = (S_list_tail &t)
00164     {
00165         if (&t == this)
00166             return *this;
00167
00168         Base::operator = (static_cast<Base &&>(t));
00169         _tail = t.empty() ? &this->_f : t._tail;
00170         t._tail = &t._f;
00171         return *this;
00172     }
00173
00174     void push_front(T *e)
00175     {
00176         if (Base::empty())
00177             _tail = &e->_n;
00178
00179         Base::push_front(e);
00180     }
00181
00182     void push_back(T *e)
00183     {
00184         e->_n = 0;
00185         *_tail = e;
00186         _tail = &e->_n;
00187     }
00188
00189     void clear()
00190     {
00191         Base::clear();
00192         _tail = &this->_f;
00193     }
00194
00195     void append(S_list_tail &o)
00196     {
00197         T *x = o.front();
00198         *_tail = x;
00199         if (x)
00200             _tail = o._tail;
00201         o.clear();
00202     }
00203
00204     T *pop_front()
00205     {
00206         T *t = Base::pop_front();
00207         if (t && Base::empty())
00208             _tail = &this->_f;
00209         return t;
00210     }
00211
00212 private:
00213     static void insert(T *e, Iterator const &pred);
00214     static void insert_before(T *e, Iterator const &succ);
00215     static void replace(Iterator const &p, T*e);
00216     static Iterator erase(Iterator const &e);
00217
00218 private:
00219     S_list_item **_tail;
00220 };
00221
00222 }

```

16.181 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

```

16.182 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 }

```

16.183 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 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

```



```

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/exceptions>
00028 #include <iostream>
00029
00030 inline
00031 std::ostream &
00032 operator << (std::ostream &o, L4::Base_exception const &e)
00033 {
00034     o << "Exception: " << e.str() << ", backtrace ...\n";
00035     for (int i = 0; i < e.frame_count(); ++i)
00036         o << (void *) (e.pc_array()[i]) << '\n';
00037
00038     return o;
00039 }
00040
00041 inline
00042 std::ostream &
00043 operator << (std::ostream &o, L4::Runtime_error const &e)
00044 {
00045     o << "Exception: " << e.str() << ": ";
00046     if (e.extra_str())
00047         o << e.extra_str() << ": ";
00048     o << "backtrace ...\n";
00049     for (int i = 0; i < e.frame_count(); ++i)
00050         o << (void *) (e.pc_array()[i]) << '\n';
00051
00052     return o;
00053 }

```

16.186 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
00021 #pragma once
00022
00023 namespace cxx {
00024
00025 template< typename Obj >
00026 struct Lt_functor
00027 {
00028     bool operator () (Obj const &l, Obj const &r) const
00029     { return l < r; }
00030 };
00031
00032 };
00033
00034 };
00035
00036

```

16.187 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(), 0UL);
00115     }
00116
00118     template< typename F >
00119     char const *find_match(F &&match) const
00120     {
00121         String::Index s = _start;
00122         while (1)
00123         {
00124             if (eof(s))
00125                 return s;
00126
00127             if (match(*s))
00128                 return s;

```

```

00129
00130     ++s;
00131     }
00132 }
00133
00135 char const *find(char const *c) const
00136 { return find(c, start()); }
00137
00139 char const *find(int c) const
00140 { return find(c, start()); }
00141
00143 char const *rfind(char const *c) const
00144 {
00145     if (!_len)
00146         return end();
00147
00148     char const *p = end();
00149     --p;
00150     while (p >= _start)
00151     {
00152         if (*p == *c)
00153             return p;
00154         --p;
00155     }
00156     return end();
00157 }
00158
00159
00166 Index starts_with(cxx::String const &c) const
00167 {
00168     unsigned long i;
00169     for (i = 0; i < c._len && i < _len; ++i)
00170         if (_start[i] != c[i])
00171             return 0;
00172     return i == c._len ? start() + i : 0;
00173 }
00174
00176 char const *find(int c, char const *s) const
00177 {
00178     if (s < _start)
00179         return end();
00180
00181     while (1)
00182     {
00183         if (eof(s))
00184             return s;
00185
00186         if (*s == c)
00187             return s;
00188
00189         ++s;
00190     }
00191 }
00192
00202 char const *find(char const *c, char const *s) const
00203 {
00204     if (s < _start)
00205         return end();
00206
00207     while (1)
00208     {
00209         if (eof(s))
00210             return s;
00211
00212         for (char const *x = c; *x; ++x)
00213             if (*s == *x)
00214                 return s;
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 }

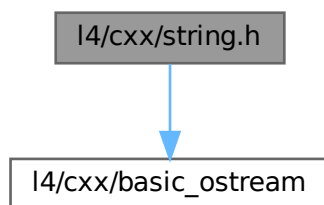
```

16.188 I4/cxx/string.h File Reference

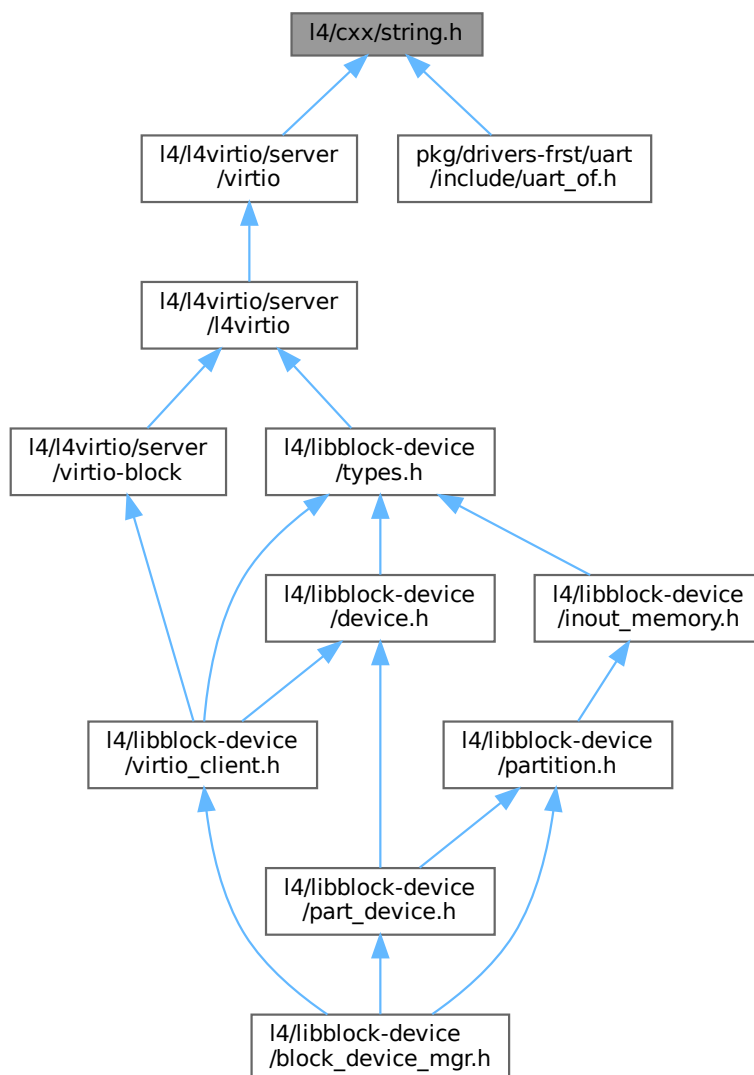
String.

```
#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.

16.188.1 Detailed Description

String.

Definition in file [string.h](#).

16.189 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 }

```

16.190 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
00023 #include "type_traits"
00024
00025 namespace cxx {
00026
00027 template< typename ...T >
00028 struct type_list;
00029
00030 template<>
00031 struct type_list<>
00032 {
00033     typedef false_type head;
00034     typedef false_type tail;
00035 };
00036
00037 template<typename HEAD, typename ...TAIL>
00038 struct type_list<HEAD, TAIL...>
00039 {
00040     typedef HEAD head;
00041     typedef type_list<TAIL...> tail;
00042 };
00043
00044 template<typename TYPELIST, template <typename T> class PREDICATE>
00045 struct find_type;
00046
00047 template<template <typename T> class PREDICATE>
00048 struct find_type<type_list<>, PREDICATE>
00049 {
00050     typedef false_type type;
00051 };
00052
00053 template<typename TYPELIST, template <typename T> class PREDICATE>
00054 struct find_type
00055 {
00056     typedef typename conditional<PREDICATE<typename TYPELIST::head>::value,
00057                                 typename TYPELIST::head,
00058                                 typename find_type<typename TYPELIST::tail, PREDICATE>::type>::type
00059     type;
00060 };
00061
00062 template<typename TYPELIST, template <typename T> class PREDICATE>
00063 using find_type_t = typename find_type<TYPELIST, PREDICATE>::type;
00064 }
00065

```

16.191 type_traits

```

00001 // vi:set ft=cpp: -*- 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 "bits/type_traits.h"
00028
00029
00030 #define CXX_GCC_VERSION (__GNUC__ * 100 + __GNUC_MINOR__)
00031
00032
00033 namespace cxx {
00034
00035 template< typename T, T V >
00036 struct integral_constant
00037 {
00038     static T const value = V;
00039     typedef T value_type;
00040     typedef integral_constant<T, V> type;
00041 };
00042
00043 typedef integral_constant<bool, true> true_type;
00044 typedef integral_constant<bool, false> false_type;
00045
00046 template< typename T > struct remove_reference;
00047
00048 template< typename T > struct identity { typedef T type; };
00049 template< typename T > using identity_t = typename identity<T>::type;
00050
00051 template< typename T1, typename T2 > struct is_same;
00052
00053 template< typename T > struct remove_const;
00054
00055 template< typename T > struct remove_volatile;
00056
00057 template< typename T > struct remove_cv;
00058
00059 template< typename T > struct remove_pointer;
00060
00061 template< typename T > struct remove_extent;
00062
00063 template< typename T > struct remove_all_extents;
00064
00065
00066
00067 template< typename, typename >
00068 struct is_same : false_type {};
00069
00070 template< typename T >
00071 struct is_same<T, T> : true_type {};
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 #if __cplusplus >= 201103L
00080 template< typename T >
00081 struct remove_reference<T &&> { typedef T type; };
00082 #endif
00083
00084 template< typename T >
00085 using remove_reference_t = typename remove_reference<T>::type;
00086
00087 template< typename T > struct remove_const { typedef T type; };
00088 template< typename T > struct remove_const<T const> { typedef T type; };
00089 template< typename T > using remove_const_t = typename remove_const<T>::type;
00090
00091 template< typename T > struct remove_volatile { typedef T type; };
00092 template< typename T > struct remove_volatile<T volatile> { typedef T type; };
00093 template< typename T > using remove_volatile_t = typename remove_volatile<T>::type;
00094
00095 template< typename T >
00096 struct remove_cv { typedef remove_const_t<remove_volatile_t<T> type; };
00097
00098 template< typename T >
00099 using remove_cv_t = typename remove_cv<T>::type;
00100
00101
00102 template< typename T, typename >
00103 struct __remove_pointer_h { typedef T type; };
00104
00105 template< typename T, typename I >
00106 struct __remove_pointer_h<T, I*> { typedef I type; };
00107
00108 template< typename T >
00109 struct remove_pointer : __remove_pointer_h<T, remove_cv_t<T> {};
00110
00111 template< typename T >

```

```

00112 using remove_pointer_t = typename remove_pointer<T>::type;
00113
00114
00115 template< typename T >
00116 struct remove_extent { typedef T type; };
00117
00118 template< typename T >
00119 struct remove_extent<T[]> { typedef T type; };
00120
00121 template< typename T, unsigned long N >
00122 struct remove_extent<T[N]> { typedef T type; };
00123
00124 template< typename T >
00125 using remove_extent_t = typename remove_extent<T>::type;
00126
00127
00128 template< typename T >
00129 struct remove_all_extents { typedef T type; };
00130
00131 template< typename T >
00132 struct remove_all_extents<T[]> { typedef typename remove_all_extents<T>::type type; };
00133
00134 template< typename T, unsigned long N >
00135 struct remove_all_extents<T[N]> { typedef typename remove_all_extents<T>::type type; };
00136
00137 template< typename T >
00138 using remove_all_extents_t = typename remove_all_extents<T>::type;
00139
00140 #if __cplusplus >= 201103L
00141
00142 template< typename T >
00143 inline T &&
00144 forward(cxx::remove_reference_t<T> &t)
00145 { return static_cast<T &&>(t); }
00146
00147 template< typename T >
00148 inline T &&
00149 forward(cxx::remove_reference_t<T> &&t)
00150 { return static_cast<T &&>(t); }
00151
00152 template< typename T >
00153 inline cxx::remove_reference_t<T> &&
00154 move(T &&t) { return static_cast<cxx::remove_reference_t<T> &&>(t); }
00155 #else
00156 template< typename T >
00157 inline T move(T t) { return t; }
00158 #endif
00159
00160 template< bool, typename T = void >
00161 struct enable_if {};
00162
00163 template< typename T >
00164 struct enable_if<true, T> { typedef T type; };
00165
00166 template< bool C, typename T = void >
00167 using enable_if_t = typename enable_if<C, T>::type;
00168
00169 template< typename T >
00170 struct is_const : false_type {};
00171
00172 template< typename T >
00173 struct is_const<T const> : true_type {};
00174
00175 template< bool, typename, typename >
00176 struct conditional;
00177
00178 template< bool C, typename T_TRUE, typename T_FALSE >
00179 struct conditional { typedef T_TRUE type; };
00180
00181 template< typename T_TRUE, typename T_FALSE >
00182 struct conditional< false, T_TRUE, T_FALSE > { typedef T_FALSE type; };
00183
00184 template< bool C, typename T_TRUE, typename T_FALSE >
00185 using conditional_t = typename conditional<C, T_TRUE, T_FALSE>::type;
00186
00187 template<typename T>
00188 struct is_enum : integral_constant<bool, __is_enum(T)> {};
00189
00190 template<typename T>
00191 struct is_polymorphic : cxx::integral_constant<bool, __is_polymorphic(T)> {};
00192
00193 template< typename T > struct is_integral : false_type {};
00194
00195 template<> struct is_integral<bool> : true_type {};
00196
00197 template<> struct is_integral<char> : true_type {};
00198 template<> struct is_integral<signed char> : true_type {};

```

```

00199 template<> struct is_integral<unsigned char> : true_type {};
00200 template<> struct is_integral<short> : true_type {};
00201 template<> struct is_integral<unsigned short> : true_type {};
00202 template<> struct is_integral<int> : true_type {};
00203 template<> struct is_integral<unsigned int> : true_type {};
00204 template<> struct is_integral<long> : true_type {};
00205 template<> struct is_integral<unsigned long> : true_type {};
00206 template<> struct is_integral<long long> : true_type {};
00207 template<> struct is_integral<unsigned long long> : true_type {};
00208
00209 template< typename T, bool = is_integral<T>::value || is_enum<T>::value >
00210 struct __is_signed_helper : integral_constant<bool, static_cast<bool>(T(-1) < T(0))> {};
```

```

00211
00212 template< typename T >
00213 struct __is_signed_helper<T, false> : integral_constant<bool, false> {};
00214
00215 template< typename T >
00216 struct is_signed : __is_signed_helper<T> {};
00217
00218
00219 template< typename >
00220 struct is_array : false_type {};
00221
00222 template< typename T >
00223 struct is_array<T[]> : true_type {};
00224
00225 template< typename T, unsigned long N >
00226 struct is_array<T[N]> : true_type {};
00227
00228 template< typename T, unsigned N >
00229 constexpr unsigned array_size(T const (&)[N]) { return N; }
00230
00231 template< int SIZE, bool SIGN = false, bool = true > struct int_type_for_size;
00232
00233 template<> struct int_type_for_size<sizeof(char), true, true>
00234 { typedef signed char type; };
00235
00236 template<> struct int_type_for_size<sizeof(char), false, true>
00237 { typedef unsigned char type; };
00238
00239 template<> struct int_type_for_size<sizeof(short), true, (sizeof(short) > sizeof(char))>
00240 { typedef short type; };
00241
00242 template<> struct int_type_for_size<sizeof(short), false, (sizeof(short) > sizeof(char))>
00243 { typedef unsigned short type; };
00244
00245 template<> struct int_type_for_size<sizeof(int), true, (sizeof(int) > sizeof(short))>
00246 { typedef int type; };
00247
00248 template<> struct int_type_for_size<sizeof(int), false, (sizeof(int) > sizeof(short))>
00249 { typedef unsigned int type; };
00250
00251 template<> struct int_type_for_size<sizeof(long), true, (sizeof(long) > sizeof(int))>
00252 { typedef long type; };
00253
00254 template<> struct int_type_for_size<sizeof(long), false, (sizeof(long) > sizeof(int))>
00255 { typedef unsigned long type; };
00256
00257 template<> struct int_type_for_size<sizeof(long long), true, (sizeof(long long) > sizeof(long))>
00258 { typedef long long type; };
00259
00260 template<> struct int_type_for_size<sizeof(long long), false, (sizeof(long long) > sizeof(long))>
00261 { typedef unsigned long long type; };
00262
00263 template< int SIZE, bool SIGN = false>
00264 using int_type_for_size_t = typename int_type_for_size<SIZE, SIGN>::type;
00265
00266 template< typename T, class Enable = void > struct underlying_type {};
00267
00268 template< typename T >
00269 struct underlying_type<T, typename enable_if<is_enum<T>::value>::type >
00270 {
00271     typedef typename int_type_for_size<sizeof(T), is_signed<T>::value>::type type;
00272 };
00273
00274 template< typename T >
00275 using underlying_type_t = typename underlying_type<T>::type;
00276
00277 template< typename T > struct make_signed;
00278 template<> struct make_signed<char> { typedef signed char type; };
00279 template<> struct make_signed<unsigned char> { typedef signed char type; };
00280 template<> struct make_signed<signed char> { typedef signed char type; };
00281 template<> struct make_signed<unsigned int> { typedef signed int type; };
00282 template<> struct make_signed<signed int> { typedef signed int type; };
00283 template<> struct make_signed<unsigned long int> { typedef signed long int type; };
00284 template<> struct make_signed<signed long int> { typedef signed long int type; };
00285 template<> struct make_signed<unsigned long long int> { typedef signed long long int type; };

```

```

00286 template<> struct make_signed<signed long long int> { typedef signed long long int type; };
00287 template< typename T > using make_signed_t = typename make_signed<T>::type;
00288
00289 template< typename T > struct make_unsigned;
00290 template<> struct make_unsigned<char> { typedef unsigned char type; };
00291 template<> struct make_unsigned<unsigned char> { typedef unsigned char type; };
00292 template<> struct make_unsigned<signed char> { typedef unsigned char type; };
00293 template<> struct make_unsigned<unsigned int> { typedef unsigned int type; };
00294 template<> struct make_unsigned<signed int> { typedef unsigned int type; };
00295 template<> struct make_unsigned<unsigned long int> { typedef unsigned long int type; };
00296 template<> struct make_unsigned<signed long int> { typedef unsigned long int type; };
00297 template<> struct make_unsigned<unsigned long long int> { typedef unsigned long long int type; };
00298 template<> struct make_unsigned<signed long long int> { typedef unsigned long long int type; };
00299 template< typename T > using make_unsigned_t = typename make_unsigned<T>::type;
00300
00301
00302 template<typename From, typename To>
00303 struct is_convertible
00304 {
00305 private:
00306     struct _true { char x[2]; };
00307     struct _false {};
00308
00309     static _true _helper(To const *);
00310     static _false _helper(...);
00311 public:
00312     enum
00313     {
00314         value = sizeof(_true) == sizeof(_helper(static_cast<From*>(0)))
00315             ? true : false
00316     };
00317     typedef bool value_type;
00318 };
00319
00320
00321 }
00322

```

16.192 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;

```

```

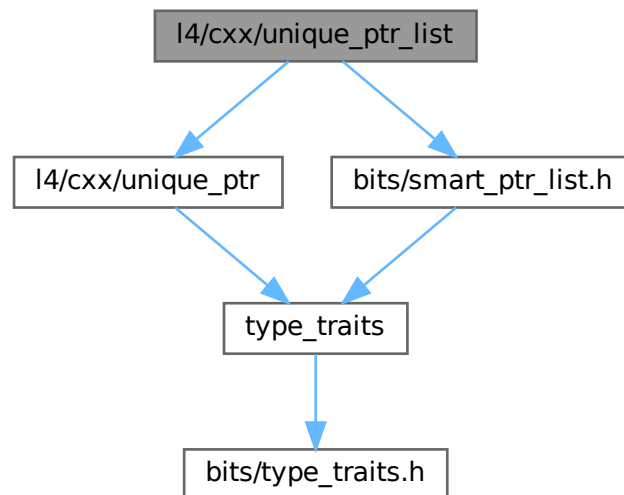
00044 reference operator [] (int idx) const
00045 { return static_cast<C const *>(this)->get()[idx]; }
00046 };
00047
00048 template< typename T, typename T_Del = default_delete<T> >
00049 class unique_ptr : public unique_ptr_index_op<T, unique_ptr<T, T_Del> >
00050 {
00051 private:
00052     struct _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 }

```

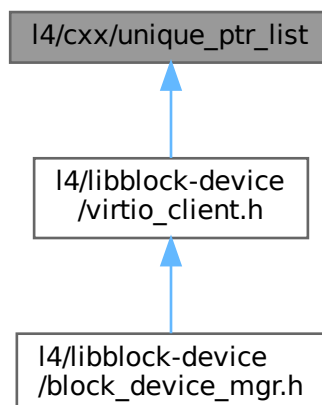
16.193 l4/cxx/unique_ptr_list File Reference

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.

16.193.1 Detailed Description

Implementation of a list of unique-ptr-managed objects.

Definition in file `unique_ptr_list`.

16.194 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 <l4/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 }
```

16.195 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
00051 template< typename T, typename VAL > inline
00052 void write_now(T *a, VAL &&val)
00053 {
00054     __asm__ __volatile__ ( " : "=m"(*a);
00055     *a = val;
00056     __asm__ __volatile__ ( " : : "=m"(*a);
00057 }
00058 }
00059 }
00060 }

```

16.196 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
00016 class Weak_ref_base : public H_list_item_t<Weak_ref_base>
00017 {
00018 protected:
00019     Weak_ref_base(void const *ptr = nullptr) : _obj(ptr) {}
00020     void reset_hard() { _obj = nullptr; }
00021     void const *_obj;
00022
00023 public:
00024     struct List : H_list_t<Weak_ref_base>
00025     {
00026         void reset()
00027         {
00028             while (!empty())
00029                 pop_front()->reset_hard();
00030         }
00031
00032         ~List()
00033         { reset(); }
00034     };
00035
00036     explicit operator bool () const
00037     { return _obj ? true : false; }
00038 };
00039
00040 template <typename T>
00041 class Weak_ref : public Weak_ref_base
00042 {
00043 public:
00044     T *get() const
00045     { return reinterpret_cast<T*>(const_cast<void *>(_obj)); }
00046
00047     T *reset(T *n)
00048     {
00049         T *r = get();
00050         if (r)
00051             r->remove_weak_ref(this);
00052
00053         _obj = n;
00054         if (n)
00055             n->add_weak_ref(this);
00056
00057         return r;
00058     }
00059 }

```



```

00085
00086 Weak_ref(T *s = nullptr) : Weak_ref_base(s)
00087 {
00088     if (s)
00089         s->add_weak_ref(this);
00090 }
00091
00092 ~Weak_ref() { reset(0); }
00093
00094 void operator = (T *n)
00095 { reset(n); }
00096
00097 Weak_ref(Weak_ref const &o) : Weak_ref_base(o._obj)
00098 {
00099     if (T *x = get())
00100         x->add_weak_ref(this);
00101 }
00102
00103 Weak_ref &operator = (Weak_ref const &o)
00104 {
00105     if (&o == this)
00106         return *this;
00107     reset(o.get());
00108     return *this;
00109 }
00110
00111 T &operator * () const { return get(); }
00112 T *operator -> () const { return get(); }
00113 };
00114
00115 class Weak_ref_obj
00116 {
00117 {
00118 protected:
00119     template <typename T> friend class Weak_ref;
00120     mutable Weak_ref_base::List weak_references;
00121
00122     void add_weak_ref(Weak_ref_base *ref) const
00123     { weak_references.push_front(ref); }
00124
00125     void remove_weak_ref(Weak_ref_base *ref) const
00126     { weak_references.remove(ref); }
00127 };
00128
00129 }

```

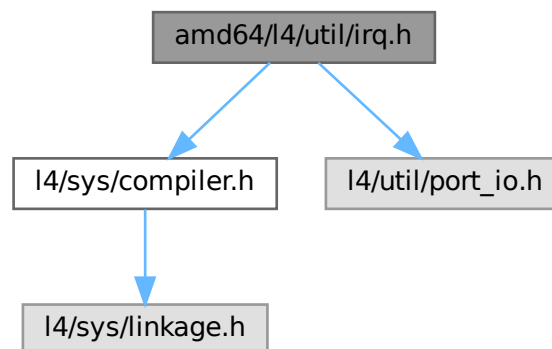
16.197 amd64/l4/util/irq.h File Reference

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.

16.197.1 Detailed Description

some PIC and hardware interrupt related functions

Date

2003

Author

Jork Loeser jork.loeser@inf.tu-dresden.de Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [irq.h](#).

16.197.2 Function Documentation

16.197.2.1 l4util_irq_acknowledge()

```
void l4util_irq_acknowledge (  
    unsigned int irq ) [inline]
```

Acknowledge IRQ at PIC in fully special nested mode.

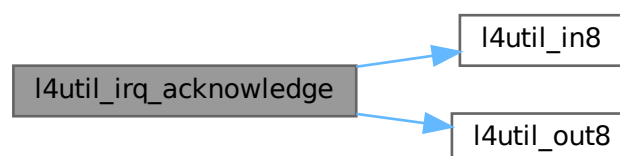
Parameters

<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:



16.198 irq.h

[Go to the documentation of this file.](#)

```

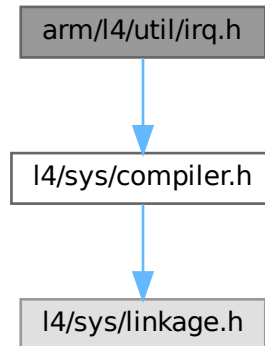
00001
00010 /*
00011  * (c) 2003-2009 Author(s)
00012  *     economic rights: Technische Universität Dresden (Germany)
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU Lesser General Public License 2.1.
00015  * Please see the COPYING-LGPL-2.1 file for details.
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

```

16.199 arm/l4/util/irq.h File Reference

ARM specific implementation of irq functions.

```
#include <l4/sys/compiler.h>
Include dependency graph for irq.h:
```



16.199.1 Detailed Description

ARM specific implementation of irq functions.

Do not use.

Definition in file [irq.h](#).

16.200 irq.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  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>
00006  *      economic rights: Technische Universität Dresden (Germany)
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 #ifndef __L4UTIL__ARCH_ARCH__IRQ_H__
00012 #define __L4UTIL__ARCH_ARCH__IRQ_H__
00013
00014 #ifdef __GNUC__
00015
00016 #include <l4/sys/compiler.h>
00017
00018 EXTERN_C_BEGIN
00019
00020 L4_INLINE void l4util_cli (void);
00021 L4_INLINE void l4util_sti (void);
00022 L4_INLINE void l4util_flags_save(l4_umword_t *flags);
00023 L4_INLINE void l4util_flags_restore(l4_umword_t *flags);
00024
00025 L4_INLINE
00026 void
00027 l4util_cli(void)
00028 {
00029     extern void __do_not_use_l4util_cli(void);
00030     __do_not_use_l4util_cli();
00031 }
00032

```

```

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__ */

```

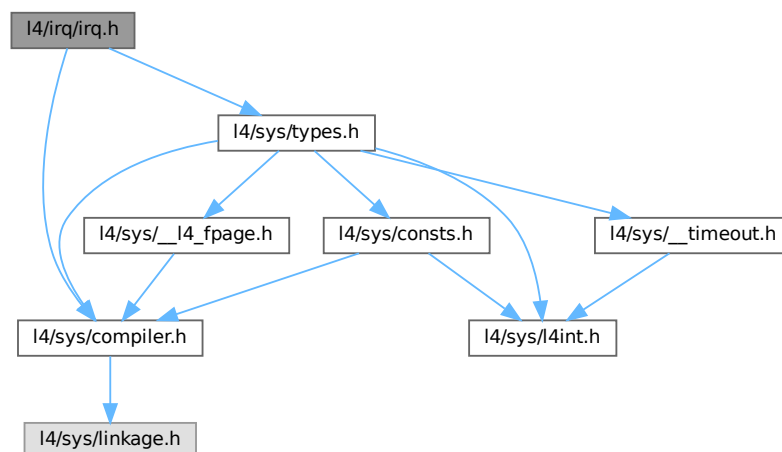
16.201 l4/irq/irq.h File Reference

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.*

16.201.1 Detailed Description

IRQ handling routines.

Definition in file [irq.h](#).

16.202 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
00021  struct l4irq_t;
00022  typedef struct l4irq_t l4irq_t;
00023
00024  L4_CV l4irq_t *
00025  l4irq_attach(int irqnum);
00026
00027  L4_CV l4irq_t *
00028  l4irq_attach_ft(int irqnum, unsigned mode);
00029
00030  L4_CV l4irq_t *
00031  l4irq_attach_thread(int irqnum, l4_cap_idx_t to_thread);
00032
00033  L4_CV l4irq_t *
00034  l4irq_attach_thread_ft(int irqnum, l4_cap_idx_t to_thread,
00035                          unsigned mode);
00036
00037  L4_CV long
00038  l4irq_wait(l4irq_t *irq);
00039
00040  L4_CV long
00041  l4irq_unmask_and_wait_any(l4irq_t *unmask_irq, l4irq_t **ret_irq);
00042
00043  L4_CV long
00044  l4irq_wait_any(l4irq_t **irq);
00045
00046  L4_CV long
00047  l4irq_unmask(l4irq_t *irq);
00048
00049  L4_CV long
00050  l4irq_detach(l4irq_t *irq);
00051
00052  /*****
00053  L4_CV l4irq_t *
00054  l4irq_request(int irqnum, void (*isr_handler)(void *), void *isr_data,
00055               int irq_thread_prio, unsigned mode);
00056
00057  L4_CV long
00058  l4irq_release(l4irq_t *irq);
00059
00060  *****/
00061
00062  /*****
00063  L4_CV l4irq_t *
00064  l4irq_attach_cap(l4_cap_idx_t irqcap);
00065
00066  L4_CV l4irq_t *
00067  l4irq_attach_cap_ft(l4_cap_idx_t irqcap, unsigned mode);
00068
00069  L4_CV l4irq_t *
00070  l4irq_attach_thread_cap(l4_cap_idx_t irqcap, l4_cap_idx_t to_thread);
00071
00072  L4_CV l4irq_t *
00073  l4irq_attach_thread_cap_ft(l4_cap_idx_t irqcap, l4_cap_idx_t to_thread,
00074                             unsigned mode);
00075
00076  *****/
00077
00078  L4_CV l4irq_t *
00079  l4irq_request_cap(l4_cap_idx_t irqcap,
00080                   void (*isr_handler)(void *), void *isr_data,
00081                   int irq_thread_prio, unsigned mode);
00082
00083  __END_DECLS

```

16.203 l4/sys/irq.h File Reference

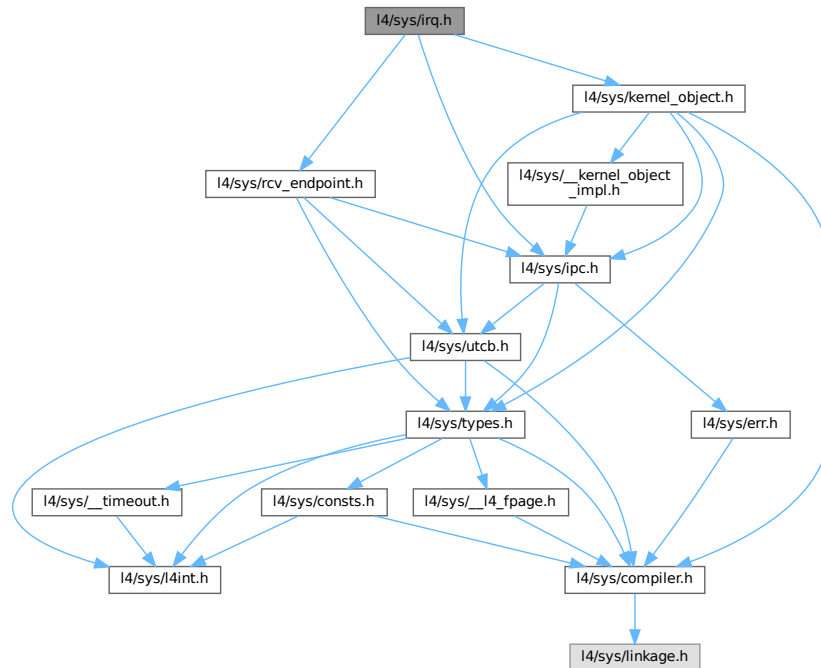
C Irq interface.

```

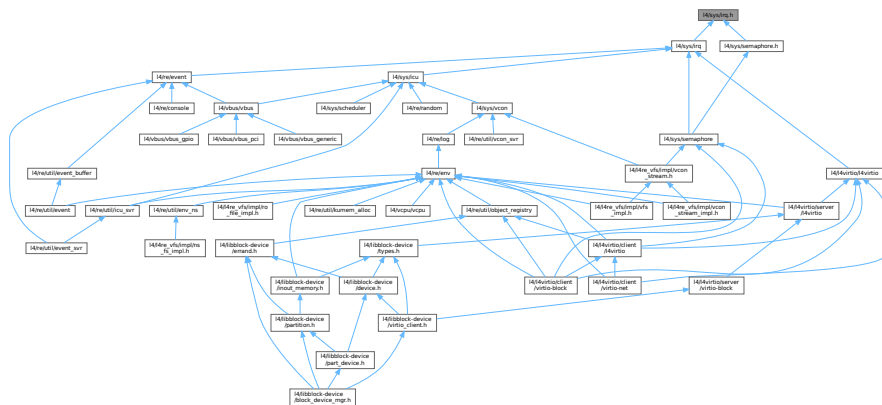
#include <l4/sys/kernel_object.h>
#include <l4/sys/ipc.h>

```

```
#include <linux/sys/rcv_endpoint.h>
```



This graph shows which files directly or indirectly include this file:



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 IRQ.

16.203.1 Detailed Description

C Irq interface.

Definition in file [irq.h](#).

16.204 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_utcb());
00327 }
00328

```

16.205 x86/I4/util/irq.h File Reference

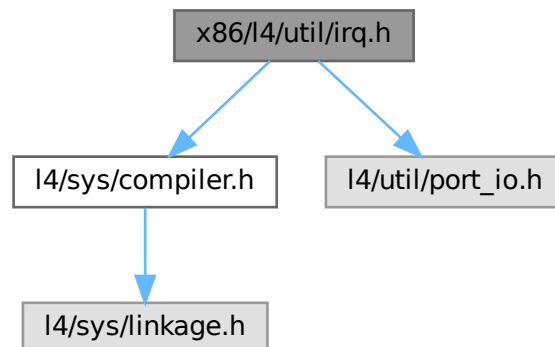
some PIC and hardware interrupt related functions

```

#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.

16.205.1 Detailed Description

some PIC and hardware interrupt related functions

Date

2003

Author

Jork Loeser jork.loeser@inf.tu-dresden.de Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file `irq.h`.

16.205.2 Function Documentation

16.205.2.1 `l4util_irq_acknowledge()`

```
void l4util_irq_acknowledge (
    unsigned int irq ) [inline]
```

Acknowledge IRQ at PIC in fully special nested mode.

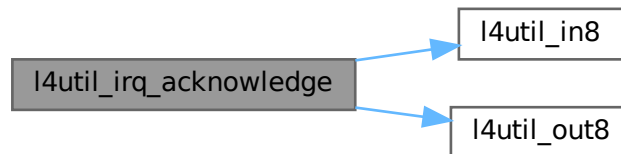
Parameters

<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:



16.206 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");

```



```

00074
00076 off64_t lseek64(off64_t, int) noexcept override
00077 { return -ESPIPE; }
00078
00080 int ftruncate64(off64_t) noexcept override
00081 { return -EINVAL; }
00082
00084 int fsync() const noexcept override
00085 { return -EINVAL; }
00086
00088 int fdatsync() const noexcept override
00089 { return -EINVAL; }
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 private:
00221 int get_entry(const char *, int, mode_t, cxx::Ref_ptr<File> *) noexcept override
00222 { return -ENOTDIR; }
00223
00224 protected:
00225 const char *get_mount(const char *path, cxx::Ref_ptr<File> *dir) noexcept;
00226 };
00227
00228 inline
00229 Be_file::~Be_file() noexcept {}
00230
00231 class Be_file_pos : public Be_file
00232 {
00233 public:
00234     Be_file_pos() noexcept : Be_file(), _pos(0) {}
00235
00236     virtual off64_t size() const noexcept = 0;
00237
00238     ssize_t readv(const struct iovec *v, int iovcnt) noexcept override
00239     {
00240         ssize_t r = preadv(v, iovcnt, _pos);
00241         if (r > 0)
00242             _pos += r;
00243         return r;
00244     }
00245
00246     ssize_t writev(const struct iovec *v, int iovcnt) noexcept override
00247     {
00248         ssize_t r = pwritev(v, iovcnt, _pos);
00249         if (r > 0)
00250             _pos += r;
00251         return r;
00252     }
00253
00254     ssize_t preadv(const struct iovec *v, int iovcnt, off64_t offset) noexcept override = 0;
00255     ssize_t pwritev(const struct iovec *v, int iovcnt, off64_t offset) noexcept override = 0;
00256
00257     off64_t lseek64(off64_t offset, int whence) noexcept override
00258     {
00259         off64_t r;
00260         switch (whence)
00261         {
00262             case SEEK_SET: r = offset; break;
00263             case SEEK_CUR: r = _pos + offset; break;
00264             case SEEK_END: r = size() + offset; break;
00265             default: return -EINVAL;
00266         };
00267
00268         if (r < 0)
00269             return -EINVAL;
00270     }

```



```

00271
00272     _pos = r;
00273     return _pos;
00274 }
00275
00276 ~Be_file_pos() noexcept = 0;
00277
00278 protected:
00279     off64_t pos() const noexcept { return _pos; }
00280
00281 private:
00282     off64_t _pos;
00283 };
00284
00285 inline Be_file_pos::~Be_file_pos() noexcept {}
00286
00287 class Be_file_stream : public Be_file
00288 {
00289 public:
00290     ssize_t preadv(const struct iovec *v, int iovcnt, off64_t) noexcept override
00291     { return readv(v, iovcnt); }
00292
00293     ssize_t pwritev(const struct iovec *v, int iovcnt, off64_t) noexcept override
00294     { return writev(v, iovcnt); }
00295
00296     ~Be_file_stream() throw () = 0;
00297
00298 };
00299
00300 inline Be_file_stream::~Be_file_stream() noexcept {}
00301
00302 class Be_file_system : public File_system
00303 {
00304 private:
00305     char const *_fstype;
00306
00307 public:
00308     explicit Be_file_system(char const *_fstype) noexcept
00309     : File_system(), _fstype(_fstype)
00310     {
00311         vfs_ops->register_file_system(this);
00312     }
00313
00314     ~Be_file_system() noexcept
00315     {
00316         vfs_ops->unregister_file_system(this);
00317     }
00318
00319     char const *_type() const noexcept override { return _fstype; }
00320 };
00321
00322 /* Make sure filesystems can register before the constructor of libmount
00323  * runs */
00324 #define L4RE_VFS_FILE_SYSTEM_ATTRIBUTE \
00325     __attribute__((init_priority(INIT_PRIO_LATE)))
00326
00327 }

```

16.208 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<L4::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 };

```

16.209 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
00058     return Ref_ptr<>::Nil;
00059 }
00060
00061 inline
00062 void
00063 Fd_store::set(int fd, Ref_ptr<L4Re::Vfs::File> const &f) noexcept
00064 {
00065     _files[fd] = f;
00066 }
00067
00068 }}

```

16.210 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
00034         return -1;
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

```

16.211 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 }

```

16.212 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         if (!file.is_valid())
00063             return -ENOMEM;
00064         int err = _ns->query(path, file.get());
00065         if (err < 0)
00066             return -ENOENT;
00067         *ds = cxx::move(file);
00068         return err;
00069     }
00070
00071     int
00072     Ns_dir::get_entry(const char *path, int flags, mode_t mode,
00073                      Ref_ptr<L4Re::Vfs::File> *f) noexcept
00074     {
00075         (void)mode; (void)flags;
00076         if (!*path)
00077         {
00078             *f = cxx::ref_ptr(this);
00079             return 0;
00080         }
00081         L4Re::Unique_cap<Dataspace> file;
00082         int err = get_ds(path, &file);
00083         if (err < 0)
00084             return -ENOENT;
00085         cxx::Ref_ptr<L4Re::Vfs::File> fi = cap_to_vfs_object(file.get(), &err);
00086         if (!fi)

```

```

00095     return err;
00096
00097     file.release();
00098     *f = cxx::move(fi);
00099     return 0;
00100 }
00101
00102 int
00103 Ns_dir::faccessat(const char *path, int mode, int flags) noexcept
00104 {
00105     (void)flags;
00106     auto tmpcap = L4Re::make_unique_cap<void>(L4Re::virt_cap_alloc);
00107
00108     if (!tmpcap.is_valid())
00109         return -ENOMEM;
00110
00111     if (_ns->query(path, tmpcap.get()))
00112         return -ENOENT;
00113
00114     if (mode & W_OK)
00115         return -EACCES;
00116
00117     return 0;
00118 }
00119
00120 int
00121 Ns_dir::fstat64(struct stat64 *b) const noexcept
00122 {
00123     b->st_dev = 1;
00124     b->st_ino = 1;
00125     b->st_mode = S_IRWXU | S_IFDIR;
00126     b->st_nlink = 0;
00127     b->st_uid = 0;
00128     b->st_gid = 0;
00129     b->st_rdev = 0;
00130     b->st_size = 0;
00131     b->st_blksize = 0;
00132     b->st_blocks = 0;
00133     b->st_atime = 0;
00134     b->st_mtime = 0;
00135     b->st_ctime = 0;
00136     return 0;
00137 }
00138
00139 ssize_t
00140 Ns_dir::getdents(char *buf, size_t dest_sz) noexcept
00141 {
00142     struct dirent64 *dest = (struct dirent64 *)buf;
00143     ssize_t ret = 0;
00144     L4_addr_t infoaddr;
00145     size_t infosz;
00146
00147     L4Re::Unique_cap<Dataspace> dirinfofile;
00148     int err = get_ds(".dirinfo", &dirinfofile);
00149     if (err)
00150         return 0;
00151
00152     infosz = dirinfofile->size();
00153     if (infosz <= 0)
00154         return 0;
00155
00156     infoaddr = L4_PAGESIZE;
00157     err = L4Re::Env::env()->rm()->attach(&infoaddr, infosz,
00158                                         Rm::F::Search_addr | Rm::F::R,
00159                                         dirinfofile.get(), 0);
00160     if (err < 0)
00161         return 0;
00162
00163     char *p = (char *)infoaddr + _current_dir_pos;
00164     char *end = (char *)infoaddr + infosz;
00165
00166     char *current_dirinfo_entry = p;
00167     while (dest && p < end)
00168     {
00169         // parse lines of dirinfofile
00170         long len = 0;
00171         for (; p < end && *p >= '0' && *p <= '9'; ++p)
00172         {
00173             len += 10;
00174             len += *p - '0';
00175         }
00176
00177         if (len == 0)
00178             break;
00179
00180         if (p == end)
00181             break;

```

```

00182
00183     if (*p != ':')
00184         break;
00185     p++; // skip colon
00186
00187     if (p + len >= end)
00188         break;
00189
00190     unsigned l = len + 1;
00191     if (l > sizeof(dest->d_name))
00192         l = sizeof(dest->d_name);
00193
00194     unsigned n = offsetof(struct dirent64, d_name) + l;
00195     n = (n + sizeof(long) - 1) & ~(sizeof(long) - 1);
00196
00197     if (n > dest_sz)
00198         break;
00199
00200     dest->d_ino = 1;
00201     dest->d_off = 0;
00202     memcpy(dest->d_name, p, l - 1);
00203     dest->d_name[l - 1] = 0;
00204     dest->d_reclen = n;
00205     dest->d_type = DT_REG;
00206     ret += n;
00207     dest_sz -= n;
00208
00209     // next entry
00210     dest = (struct dirent64 *)((unsigned long)dest + n);
00211
00212     // next infodirfile line
00213     p += len;
00214     while (p < end && *p && (*p == '\n' || *p == '\r'))
00215         p++;
00216     current_dirinfo_entry = p;
00217 }
00218
00219 _current_dir_pos = current_dirinfo_entry - (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

```

```

00269     (void)mode; (void)flags;
00270     if (!*path)
00271     {
00272         *f = cxx::ref_ptr(this);
00273         return 0;
00274     }
00275
00276     L4Re::Unique_cap<Dataspace> file;
00277     int err = get_ds(path, &file);
00278
00279     if (err < 0)
00280         return -ENOENT;
00281
00282     cxx::Ref_ptr<L4Re::Vfs::File> fi = cap_to_vfs_object(file.get(), &err);
00283     if (!fi)
00284         return err;
00285
00286     file.release();
00287     *f = cxx::move(fi);
00288     return 0;
00289 }
00290
00291 int
00292 Env_dir::faccessat(const char *path, int mode, int flags) noexcept
00293 {
00294     (void)flags;
00295     Vfs::Path p(path);
00296     Vfs::Path first = p.strip_first();
00297
00298     if (first.empty())
00299         return -ENOENT;
00300
00301     L4::Cap<L4Re::Namespace>
00302     c = _env->get_cap<L4Re::Namespace>(first.path(), first.length());
00303
00304     if (!c.is_valid())
00305         return -ENOENT;
00306
00307     if (p.empty())
00308     {
00309         if (mode & W_OK)
00310             return -EACCES;
00311
00312         return 0;
00313     }
00314
00315     auto tmpcap = L4Re::make_unique_cap<void>(L4Re::virt_cap_alloc);
00316
00317     if (!tmpcap.is_valid())
00318         return -ENOMEM;
00319
00320     if (c->query(p.path(), p.length(), tmpcap.get()))
00321         return -ENOENT;
00322
00323     if (mode & W_OK)
00324         return -EACCES;
00325
00326     return 0;
00327 }
00328
00329 bool
00330 Env_dir::check_type(Env::Cap_entry const *e, long protocol) noexcept
00331 {
00332     L4::Cap<L4::Meta> m(e->cap);
00333     return m->supports(protocol).label();
00334 }
00335
00336 int
00337 Env_dir::fstat64(struct stat64 *b) const noexcept
00338 {
00339     b->st_dev = 1;
00340     b->st_ino = 1;
00341     b->st_mode = S_IRWXU | S_IFDIR;
00342     b->st_nlink = 0;
00343     b->st_uid = 0;
00344     b->st_gid = 0;
00345     b->st_rdev = 0;
00346     b->st_size = 0;
00347     b->st_blksize = 0;
00348     b->st_blocks = 0;
00349     b->st_atime = 0;
00350     b->st_mtime = 0;
00351     b->st_ctime = 0;
00352     return 0;
00353 }
00354
00355 ssize_t

```



```

00356 Env_dir::getdents(char *buf, size_t sz) noexcept
00357 {
00358     struct dirent64 *d = (struct dirent64 *)buf;
00359     ssize_t ret = 0;
00360
00361     while (d
00362         && _current_cap_entry
00363         && _current_cap_entry->flags != ~0UL)
00364     {
00365         unsigned l = strlen(_current_cap_entry->name) + 1;
00366         if (l > sizeof(d->d_name))
00367             l = sizeof(d->d_name);
00368
00369         unsigned n = offsetof (struct dirent64, d_name) + 1;
00370         n = (n + sizeof(long) - 1) & ~(sizeof(long) - 1);
00371
00372         if (n <= sz)
00373         {
00374             d->d_ino = 1;
00375             d->d_off = 0;
00376             memcpy(d->d_name, _current_cap_entry->name, l);
00377             d->d_name[l - 1] = 0;
00378             d->d_reclen = n;
00379             if (check_type(_current_cap_entry, L4Re::Namespace::Protocol))
00380                 d->d_type = DT_DIR;
00381             else if (check_type(_current_cap_entry, L4Re::Dataspace::Protocol))
00382                 d->d_type = DT_REG;
00383             else
00384                 d->d_type = DT_UNKNOWN;
00385             ret += n;
00386             sz -= n;
00387             d = (struct dirent64 *)((unsigned long)d + n);
00388             _current_cap_entry++;
00389         }
00390         else
00391             return ret;
00392     }
00393
00394     // bit of a hack because we should only (re)set this when opening the dir
00395     if (!ret)
00396         _current_cap_entry = _env->initial_caps();
00397
00398     return ret;
00399 }
00400
00401 }}

```

16.213 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() const noexcept override { return _ds; }
00040
00041 int fstat64(struct stat64 *buf) const noexcept override;
00042
00043 int ioctl(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 }

```

16.214 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
00033         L4Re::virt_cap_alloc->release(_ds);
00034     }
00035
00036     int
00037     Ro_file::fstat64(struct stat64 *buf) const noexcept
00038     {
00039         static int fake = 0;
00040
00041         memset(buf, 0, sizeof(*buf));
00042         buf->st_size = _size;
00043         buf->st_mode = S_IFREG | 0644;
00044         buf->st_dev = _ds.cap();
00045         buf->st_ino = ++fake;
00046         buf->st_blksize = L4_PAGESIZE;
00047         buf->st_blocks = l4_round_page(_size);
00048         return 0;
00049     }
00050
00051     ssize_t
00052     Ro_file::read_single(const struct iovec *vec, off64_t pos) noexcept

```

```

00053 {
00054     off64_t l = vec->iov_len;
00055     if (_size - pos < l)
00056         l = _size - pos;
00057
00058     if (l > 0)
00059     {
00060         Vfs_config::memcpy(vec->iov_base, _addr + pos, l);
00061         return 1;
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 = (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 = (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 ((size_t)r < vec->iov_len)
00092             return 1;
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 -EINVAL;
00117 }
00118
00119 }

```

16.215 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 ioctl1(unsigned long request, va_list args) noexcept override;
00044
00045      ~Vcon_stream() noexcept {}
00046      void operator delete (void *) {}
00047  };
00048
00049  }}

```

16.216 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
00022  #include "vcon_stream.h"
00023
00024  #include <termios.h>
00025  #include <unistd.h>
00026  #include <sys/ioctl.h>
00027  #include <sys/ttydefaults.h>
00028
00029  namespace L4Re { namespace Core {
00030  Vcon_stream::Vcon_stream(L4::Cap<L4::Vcon> s) noexcept
00031  : Be_file_stream(),
00032    _s(s), _irq(L4Re::virt_cap_alloc->alloc<L4::Semaphore>()), _irq_bound(false)
00033  {
00034      int res = l4_error(L4Re::Env::env()->factory()->create(_irq));
00035      (void)res; // handle errors!
00036  }
00037
00038  ssize_t
00039  Vcon_stream::readv(const struct iovec *iovec, int iovcnt) noexcept
00040  {
00041      if (!_irq_bound)
00042      {
00043          bool was_bound = __atomic_exchange_n(&_irq_bound, true, __ATOMIC_SEQ_CST);
00044          if (!was_bound)

```

```

00045         if (l4_error(_s->bind(0, _irq)) < 0)
00046             return -EIO;
00047     }
00048
00049     ssize_t bytes = 0;
00050     for (; iocnt > 0; --iocnt, ++iovec)
00051     {
00052         if (iovec->iov_len == 0)
00053             continue;
00054
00055         char *buf = (char *)iovec->iov_base;
00056         size_t len = iovec->iov_len;
00057
00058         while (1)
00059         {
00060             int ret = _s->read(buf, len);
00061
00062             // BS: what is this ??
00063             if (ret > (int)len)
00064                 ret = len;
00065
00066             if (ret < 0)
00067                 return ret;
00068             else if (ret == 0)
00069             {
00070                 if (bytes)
00071                     return bytes;
00072
00073                 ret = _s->read(buf, len);
00074                 if (ret < 0)
00075                     return ret;
00076                 else if (ret == 0)
00077                 {
00078                     _irq->down();
00079                     continue;
00080                 }
00081             }
00082
00083             bytes += ret;
00084             len -= ret;
00085             buf += ret;
00086
00087             if (len == 0)
00088                 break;
00089         }
00090     }
00091     return bytes;
00092 }
00093
00094
00095 ssize_t
00096 Vcon_stream::writev(const struct iovec *iovec, int iocnt) noexcept
00097 {
00098     l4_msg_regs_t store;
00099     l4_msg_regs_t *mr = l4_utcb_mr();
00100
00101     Vfs_config::memcpy(&store, mr, sizeof(store));
00102
00103     ssize_t written = 0;
00104     while (iocnt)
00105     {
00106         size_t sl = iovec->iov_len;
00107         char const *b = (char const *)iovec->iov_base;
00108
00109         for (; sl > L4_VCON_WRITE_SIZE
00110             ; sl -= L4_VCON_WRITE_SIZE, b += L4_VCON_WRITE_SIZE)
00111             _s->send(b, L4_VCON_WRITE_SIZE);
00112
00113         _s->send(b, sl);
00114
00115         written += iovec->iov_len;
00116
00117         ++iovec;
00118         --iocnt;
00119     }
00120     Vfs_config::memcpy(mr, &store, sizeof(store));
00121     return written;
00122 }
00123
00124 int
00125 Vcon_stream::fstat64(struct stat64 *buf) const noexcept
00126 {
00127     buf->st_size = 0;
00128     buf->st_mode = 0666;
00129     buf->st_dev = _s.cap();
00130     buf->st_ino = 0;
00131     return 0;

```

```

00132 }
00133
00134 int
00135 Vcon_stream::ioctl(unsigned long request, va_list args) noexcept
00136 {
00137     switch (request) {
00138         case TCGETS:
00139         {
00140             //vt100_tcgetattr(term, (struct termios *)argp);
00141
00142             struct termios *t = va_arg(args, struct termios *);
00143
00144             l4_vcon_attr_t l4a;
00145             if (!l4_error(_s->get_attr(&l4a)))
00146             {
00147                 t->c_iflag = l4a.i_flags;
00148                 t->c_oflag = l4a.o_flags; // output flags
00149                 t->c_cflag = 0; // control flags
00150                 t->c_lflag = l4a.l_flags; // local flags
00151             }
00152             else
00153                 t->c_iflag = t->c_oflag = t->c_cflag = t->c_lflag = 0;
00154             #if 0
00155                 //t->c_lflag |= ECHO; // if term->echo
00156                 t->c_lflag |= ICANON; // if term->term_mode == VT100MODE_COOKED
00157             #endif
00158
00159             t->c_cc[VEOF] = CEOF;
00160             t->c_cc[VEOL] = _POSIX_VDISABLE;
00161             t->c_cc[VEOL2] = _POSIX_VDISABLE;
00162             t->c_cc[VERASE] = CERASE;
00163             t->c_cc[VWERASE] = CWERASE;
00164             t->c_cc[VKILL] = CKILL;
00165             t->c_cc[VREPRINT] = CREPRINT;
00166             t->c_cc[VINTR] = CINTR;
00167             t->c_cc[VQUIT] = _POSIX_VDISABLE;
00168             t->c_cc[VSUSP] = CSUSP;
00169             t->c_cc[VSTART] = CSTART;
00170             t->c_cc[VSTOP] = CSTOP;
00171             t->c_cc[VLNEXT] = CLNEXT;
00172             t->c_cc[VDISCARD] = CDISCARD;
00173             t->c_cc[VMIN] = CMIN;
00174             t->c_cc[VTIME] = 0;
00175
00176         }
00177
00178         return 0;
00179
00180         case TCSETS:
00181         case TCSETSW:
00182         case TCSETSF:
00183         {
00184             //vt100_tcsetattr(term, (struct termios *)argp);
00185             struct termios const *t = va_arg(args, struct termios const *);
00186
00187             // XXX: well, we're cheating, get this from the other side!
00188
00189             l4_vcon_attr_t l4a;
00190             l4a.i_flags = t->c_iflag;
00191             l4a.o_flags = t->c_oflag; // output flags
00192             l4a.l_flags = t->c_lflag; // local flags
00193             _s->set_attr(&l4a);
00194         }
00195         return 0;
00196
00197         default:
00198             break;
00199     };
00200     return -EINVAL;
00201 }
00202
00203 }}

```

16.217 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/re/env>
00026 #include <l4/re/rm>
00027 #include <l4/re/dataspace>
00028 #include <l4/cxx/hlist>
00029 #include <l4/cxx/pair>
00030 #include <l4/cxx/std_alloc>
00031
00032 #include <l4/l4re_vfs/backend>
00033 #include <l4/re/shared_cap>
00034
00035 #include <unistd.h>
00036 #include <cstdarg>
00037 #include <errno.h>
00038 #include <sys/uio.h>
00039
00040 #if 0
00041 #include <l4/sys/kdebug.h>
00042 static int debug_mmap = 1;
00043 #define DEBUG_LOG(level, dbg...) do { if (level) dbg } while (0)
00044 #else
00045 #define DEBUG_LOG(level, dbg...) do { } while (0)
00046 #endif
00047
00053 #define USE_BIG_ANON_DS
00054
00055 using L4Re::Rm;
00056
00057 namespace {
00058
00059 using cxx::Ref_ptr;
00060
00061 class Fd_store : public L4Re::Core::Fd_store
00062 {
00063 public:
00064     Fd_store() noexcept;
00065 };
00066
00067 // for internal Vcon_streams we want to have a placement new operator, so
00068 // inherit and add one
00069 class Std_stream : public L4Re::Core::Vcon_stream
00070 {
00071 public:
00072     Std_stream(L4::Cap<L4::Vcon> c) : L4Re::Core::Vcon_stream(c) {}
00073 };
00074
00075 Fd_store::Fd_store() noexcept
00076 {
00077     // use this strange way to prevent deletion of the stdio object
00078     // this depends on Fd_store to being a singleton !!!
00079     static char m[sizeof(Std_stream)] __attribute__((aligned(sizeof(long))));
00080     Std_stream *s = new (m) Std_stream(L4Re::Env::env()->log());
00081     // make sure that we never delete the static io stream thing
00082     s->add_ref();
00083     set(0, cxx::ref_ptr(s)); // stdin
00084     set(1, cxx::ref_ptr(s)); // stdout
00085     set(2, cxx::ref_ptr(s)); // stderr
00086 }
00087
00088 class Root_mount_tree : public L4Re::Vfs::Mount_tree
00089 {
00090 public:
00091     Root_mount_tree() : L4Re::Vfs::Mount_tree(0) {}
00092     void operator delete (void *) {}
00093 };
00094
00095 class Vfs : public L4Re::Vfs::Ops
00096 {
00097 private:
00098     bool _early_oom;
00099
00100 public:

```

```

00101 Vfs()
00102 : _early_oom(true), _root_mount(), _root(L4Re::Env::env())
00103 {
00104     _root_mount.add_ref();
00105     _root.add_ref();
00106     _root_mount.mount(cxx::ref_ptr(&_root));
00107     _cwd = cxx::ref_ptr(&_root);
00108
00109 #if 0
00110     Ref_ptr<L4Re::Vfs::File> rom;
00111     _root.openat("rom", 0, 0, &rom);
00112
00113     _root_mount.create_tree("lib/foo", rom);
00114
00115     _root.openat("lib", 0, 0, &_cwd);
00116
00117 #endif
00118 }
00119
00120 int alloc_fd(Ref_ptr<L4Re::Vfs::File> const &f) noexcept override;
00121 Ref_ptr<L4Re::Vfs::File> free_fd(int fd) noexcept override;
00122 Ref_ptr<L4Re::Vfs::File> get_root() noexcept override;
00123 Ref_ptr<L4Re::Vfs::File> get_cwd() noexcept override;
00124 void set_cwd(Ref_ptr<L4Re::Vfs::File> const &dir) noexcept override;
00125 Ref_ptr<L4Re::Vfs::File> get_file(int fd) noexcept override;
00126 cxx::Pair<Ref_ptr<L4Re::Vfs::File>, int>
00127     set_fd(int fd, Ref_ptr<L4Re::Vfs::File> const &f = Ref_ptr<>::Nil) noexcept
00128     override;
00129
00130 int mmap2(void *start, size_t len, int prot, int flags, int fd,
00131     off_t offset, void **ptr) noexcept override;
00132
00133 int munmap(void *start, size_t len) noexcept override;
00134 int mremap(void *old, size_t old_sz, size_t new_sz, int flags,
00135     void **new_addr) noexcept override;
00136 int mprotect(const void *a, size_t sz, int prot) noexcept override;
00137 int msync(void *addr, size_t len, int flags) noexcept override;
00138 int madvise(void *addr, size_t len, int advice) noexcept override;
00139
00140 int register_file_system(L4Re::Vfs::File_system *f) noexcept override;
00141 int unregister_file_system(L4Re::Vfs::File_system *f) noexcept override;
00142 L4Re::Vfs::File_system *get_file_system(char const *fstype) noexcept override;
00143
00144 int register_file_factory(cxx::Ref_ptr<L4Re::Vfs::File_factory> f) noexcept override;
00145 int unregister_file_factory(cxx::Ref_ptr<L4Re::Vfs::File_factory> f) noexcept override;
00146 Ref_ptr<L4Re::Vfs::File_factory> get_file_factory(int proto) noexcept override;
00147 Ref_ptr<L4Re::Vfs::File_factory> get_file_factory(char const *proto_name) noexcept override;
00148 int mount(char const *path, cxx::Ref_ptr<L4Re::Vfs::File> const &dir) noexcept override;
00149
00150 void operator delete (void *) {}
00151
00152 void *malloc(size_t size) noexcept override { return Vfs_config::malloc(size); }
00153 void free(void *m) noexcept override { Vfs_config::free(m); }
00154
00155 private:
00156     Root_mount_tree _root_mount;
00157     L4Re::Core::Env_dir _root;
00158     Ref_ptr<L4Re::Vfs::File> _cwd;
00159     Fd_store fds;
00160
00161     L4Re::Vfs::File_system *_fs_registry;
00162
00163     struct File_factory_item : cxx::H_list_item_t<File_factory_item>
00164     {
00165         cxx::Ref_ptr<L4Re::Vfs::File_factory> f;
00166         explicit File_factory_item(cxx::Ref_ptr<L4Re::Vfs::File_factory> const &f)
00167             : f(f) {};
00168
00169         File_factory_item() = default;
00170         File_factory_item(File_factory_item const &) = delete;
00171         File_factory_item &operator = (File_factory_item const &) = delete;
00172     };
00173
00174     cxx::H_list_t<File_factory_item> _file_factories;
00175
00176     l4_addr_t _anon_offset;
00177     L4Re::Shared_cap<L4Re::Dataspace> _anon_ds;
00178
00179     int alloc_ds(unsigned long size, L4Re::Shared_cap<L4Re::Dataspace> *ds);
00180     int alloc_anon_mem(l4_umword_t size, L4Re::Shared_cap<L4Re::Dataspace> *ds,
00181         l4_addr_t *offset);
00182
00183     void align_mmap_start_and_length(void **start, size_t *length);
00184 };
00185
00186 static inline bool strequal(char const *a, char const *b)
00187 {

```



```

00188     for (;*a && *a == *b; ++a, ++b)
00189         ;
00190     return *a == *b;
00191 }
00192
00193 int
00194 Vfs::register_file_system(L4Re::Vfs::File_system *f) noexcept
00195 {
00196     using L4Re::Vfs::File_system;
00197
00198     if (!f)
00199         return -EINVAL;
00200
00201     for (File_system *c = _fs_registry; c; c = c->next())
00202         if (strequal(c->type(), f->type()))
00203             return -EEXIST;
00204
00205     f->next(_fs_registry);
00206     _fs_registry = f;
00207
00208     return 0;
00209 }
00210
00211 int
00212 Vfs::unregister_file_system(L4Re::Vfs::File_system *f) noexcept
00213 {
00214     using L4Re::Vfs::File_system;
00215
00216     if (!f)
00217         return -EINVAL;
00218
00219     File_system **p = &_fs_registry;
00220
00221     for (; *p; p = &(*p)->next())
00222         if (*p == f)
00223         {
00224             *p = f->next();
00225             f->next() = 0;
00226             return 0;
00227         }
00228
00229     return -ENOENT;
00230 }
00231
00232 L4Re::Vfs::File_system *
00233 Vfs::get_file_system(char const *fstype) noexcept
00234 {
00235     bool try_dynamic = true;
00236     for (;;)
00237     {
00238         using L4Re::Vfs::File_system;
00239         for (File_system *c = _fs_registry; c; c = c->next())
00240             if (strequal(c->type(), fstype))
00241                 return c;
00242
00243         if (!try_dynamic)
00244             return 0;
00245
00246         // try to load a file system module dynamically
00247         int res = Vfs_config::load_module(fstype);
00248
00249         if (res < 0)
00250             return 0;
00251
00252         try_dynamic = false;
00253     }
00254 }
00255
00256 int
00257 Vfs::register_file_factory(cxx::Ref_ptr<L4Re::Vfs::File_factory> f) noexcept
00258 {
00259     if (!f)
00260         return -EINVAL;
00261
00262     void *x = this->malloc(sizeof(File_factory_item));
00263     if (!x)
00264         return -ENOMEM;
00265
00266     auto ff = new (x, cxx::Nothrow()) File_factory_item(f);
00267     _file_factories.push_front(ff);
00268     return 0;
00269 }
00270
00271 int
00272 Vfs::unregister_file_factory(cxx::Ref_ptr<L4Re::Vfs::File_factory> f) noexcept
00273 {
00274     for (auto p: _file_factories)

```

```

00275     {
00276         if (p->f == f)
00277         {
00278             _file_factories.remove(p);
00279             p->~File_factory_item();
00280             this->free(p);
00281             return 0;
00282         }
00283     }
00284     return -ENOENT;
00285 }
00286
00287 Ref_ptr<L4Re::Vfs::File_factory>
00288 Vfs::get_file_factory(int proto) noexcept
00289 {
00290     for (auto p: _file_factories)
00291         if (p->f->proto() == proto)
00292             return p->f;
00293
00294     return Ref_ptr<L4Re::Vfs::File_factory>();
00295 }
00296
00297 Ref_ptr<L4Re::Vfs::File_factory>
00298 Vfs::get_file_factory(char const *proto_name) noexcept
00299 {
00300     for (auto p: _file_factories)
00301     {
00302         auto n = p->f->proto_name();
00303         if (n)
00304         {
00305             char const *a = n;
00306             char const *b = proto_name;
00307             for (; *a && *b && *a == *b; ++a, ++b)
00308                 ;
00309
00310             if ((*a == 0) && (*b == 0))
00311                 return p->f;
00312         }
00313     }
00314
00315     return Ref_ptr<L4Re::Vfs::File_factory>();
00316 }
00317
00318 int
00319 Vfs::alloc_fd(Ref_ptr<L4Re::Vfs::File> const &f) noexcept
00320 {
00321     int fd = fds.alloc();
00322     if (fd < 0)
00323         return -EMFILE;
00324
00325     if (f)
00326         fds.set(fd, f);
00327
00328     return fd;
00329 }
00330
00331 Ref_ptr<L4Re::Vfs::File>
00332 Vfs::free_fd(int fd) noexcept
00333 {
00334     Ref_ptr<L4Re::Vfs::File> f = fds.get(fd);
00335
00336     if (!f)
00337         return Ref_ptr<>::Nil;
00338
00339     fds.free(fd);
00340     return f;
00341 }
00342
00343
00344 Ref_ptr<L4Re::Vfs::File>
00345 Vfs::get_root() noexcept
00346 {
00347     return cxx::ref_ptr(&_root);
00348 }
00349
00350 Ref_ptr<L4Re::Vfs::File>
00351 Vfs::get_cwd() noexcept
00352 {
00353     return _cwd;
00354 }
00355
00356 void
00357 Vfs::set_cwd(Ref_ptr<L4Re::Vfs::File> const &dir) noexcept
00358 {
00359     // FIXME: check for is dir
00360     if (dir)
00361         _cwd = dir;

```

```

00362 }
00363
00364 Ref_ptr<L4Re::Vfs::File>
00365 Vfs::get_file(int fd) noexcept
00366 {
00367     return fds.get(fd);
00368 }
00369
00370 cxx::Pair<Ref_ptr<L4Re::Vfs::File>, int>
00371 Vfs::set_fd(int fd, Ref_ptr<L4Re::Vfs::File> const &f) noexcept
00372 {
00373     if (!fds.check_fd(fd))
00374         return cxx::pair(Ref_ptr<L4Re::Vfs::File>(Ref_ptr<>::Nil), EBADF);
00375
00376     Ref_ptr<L4Re::Vfs::File> old = fds.get(fd);
00377     fds.set(fd, f);
00378     return cxx::pair(old, 0);
00379 }
00380
00381
00382 #define GET_FILE_DBG(fd, err) \
00383     Ref_ptr<L4Re::Vfs::File> fi = fds.get(fd); \
00384     if (!fi) \
00385     { \
00386         return -err; \
00387     }
00388
00389 #define GET_FILE(fd, err) \
00390     Ref_ptr<L4Re::Vfs::File> fi = fds.get(fd); \
00391     if (!fi) \
00392         return -err;
00393
00394
00395 void
00396 Vfs::align_mmap_start_and_length(void **start, size_t *length)
00397 {
00398     l4_addr_t s = l4_addr_t(*start);
00399
00400     *length += s & (L4_PAGESIZE - 1); // Add rounding down delta to length
00401     *start = (void *)l4_trunc_page(s); // Make start page aligned
00402     *length = l4_round_page(*length); // Round length up to page size
00403 }
00404
00405 int
00406 Vfs::munmap(void *start, size_t len) L4_NOTHROW
00407 {
00408     using namespace L4;
00409     using namespace L4Re;
00410
00411     int err;
00412     Cap<Dataspace> ds;
00413     Cap<Rm> r = Env::env()->rm();
00414
00415     if (l4_addr_t(start) & (L4_PAGESIZE - 1))
00416         return -EINVAL;
00417
00418     align_mmap_start_and_length(&start, &len);
00419
00420     while (1)
00421     {
00422         DEBUG_LOG(debug_mmap, {
00423             outstring("DETACH: ");
00424             outhex32(l4_addr_t(start));
00425             outstring(" ");
00426             outhex32(len);
00427             outstring("\n");
00428         });
00429         err = r->detach(l4_addr_t(start), len, &ds, This_task);
00430         if (err < 0)
00431             return err;
00432
00433         switch (err & Rm::Detach_result_mask)
00434         {
00435             case Rm::Split_ds:
00436                 if (ds.is_valid())
00437                     L4Re::virt_cap_alloc->take(ds);
00438                 return 0;
00439             case Rm::Detached_ds:
00440                 if (ds.is_valid())
00441                     L4Re::virt_cap_alloc->release(ds);
00442                 break;
00443             default:
00444                 break;
00445         }
00446
00447         if (!(err & Rm::Detach_again))
00448             return 0;

```

```

00449     }
00450 }
00451
00452 int
00453 Vfs::alloc_ds(unsigned long size, L4Re::Shared_cap<L4Re::Dataspace> *ds)
00454 {
00455     *ds = L4Re::make_shared_cap<L4Re::Dataspace>(L4Re::virt_cap_alloc);
00456
00457     if (!ds->is_valid())
00458         return -ENOMEM;
00459
00460     int err;
00461     if ((err = Vfs_config::allocator()->alloc(size, ds->get())) < 0)
00462         return err;
00463
00464     DEBUG_LOG(debug_mmap, {
00465         outstring("ANON DS ALLOCATED: size=");
00466         outhex32(size);
00467         outstring(" cap=");
00468         outhex32(ds->cap());
00469         outstring("\n");
00470     });
00471
00472     return 0;
00473 }
00474
00475 int
00476 Vfs::alloc_anon_mem(l4_umword_t size, L4Re::Shared_cap<L4Re::Dataspace> *ds,
00477                     l4_addr_t *offset)
00478 {
00479     #ifdef USE_BIG_ANON_DS
00480         enum
00481         {
00482             ANON_MEM_DS_POOL_SIZE = 256UL « 20, // size of a pool dataspace used for anon memory
00483             ANON_MEM_MAX_SIZE      = 32UL « 20,  // chunk size that will be allocate a dataspace
00484         };
00485     #else
00486         enum
00487         {
00488             ANON_MEM_DS_POOL_SIZE = 256UL « 20, // size of a pool dataspace used for anon memory
00489             ANON_MEM_MAX_SIZE      = 0UL « 20,   // chunk size that will be allocate a dataspace
00490         };
00491     #endif
00492
00493     if (size >= ANON_MEM_MAX_SIZE)
00494     {
00495         int err;
00496         if ((err = alloc_ds(size, ds)) < 0)
00497             return err;
00498
00499         *offset = 0;
00500
00501         if (!_early_oom)
00502             return err;
00503
00504         return (*ds)->allocate(0, size);
00505     }
00506
00507     if (!_anon_ds.is_valid() || _anon_offset + size >= ANON_MEM_DS_POOL_SIZE)
00508     {
00509         int err;
00510         if ((err = alloc_ds(ANON_MEM_DS_POOL_SIZE, ds)) < 0)
00511             return err;
00512
00513         _anon_offset = 0;
00514         _anon_ds = *ds;
00515     }
00516     else
00517         *ds = _anon_ds;
00518
00519     if (_early_oom)
00520     {
00521         if (int err = (*ds)->allocate(_anon_offset, size))
00522             return err;
00523     }
00524
00525     *offset = _anon_offset;
00526     _anon_offset += size;
00527     return 0;
00528 }
00529
00530 int
00531 Vfs::mmap2(void *start, size_t len, int prot, int flags, int fd, off_t page4k_offset,
00532            void **resptr) L4_NOTHROW
00533 {
00534     using namespace L4Re;
00535     off64_t offset = l4_trunc_page(page4k_offset « 12);

```

```

00536
00537     if (flags & MAP_FIXED)
00538         if (l4_addr_t(start) & (L4_PAGESIZE - 1))
00539             return -EINVAL;
00540
00541     align_mmap_start_and_length(&start, &len);
00542
00543     // special code to just reserve an area of the virtual address space
00544     if (flags & 0x1000000)
00545     {
00546         int err;
00547         L4::Cap<Rm> r = Env::env()->rm();
00548         l4_addr_t area = (l4_addr_t)start;
00549         err = r->reserve_area(&area, len, L4Re::Rm::F::Search_addr);
00550         if (err < 0)
00551             return err;
00552         *resptr = (void*)area;
00553         DEBUG_LOG(debug_mmap, {
00554             outstring("MMAP reserved area: ");
00555             outhex32(area);
00556             outstring(" length=");
00557             outhex32(len);
00558             outstring("\n");
00559         });
00560         return 0;
00561     }
00562
00563     L4Re::Shared_cap<L4Re::Dataspace> ds;
00564     l4_addr_t anon_offset = 0;
00565     L4Re::Rm::Flags rm_flags(0);
00566
00567     if (flags & (MAP_ANONYMOUS | MAP_PRIVATE))
00568     {
00569         rm_flags |= L4Re::Rm::F::Detach_free;
00570
00571         int err = alloc_anon_mem(len, &ds, &anon_offset);
00572         if (err)
00573             return err;
00574
00575         DEBUG_LOG(debug_mmap, {
00576             outstring("USE ANON MEM: ");
00577             outhex32(ds.cap());
00578             outstring(" ofs=");
00579             outhex32(anon_offset);
00580             outstring("\n");
00581         });
00582     }
00583
00584     if (!(flags & MAP_ANONYMOUS))
00585     {
00586         Ref_ptr<L4Re::Vfs::File> fi = fds.get(fd);
00587         if (!fi)
00588         {
00589             return -EBADF;
00590         }
00591
00592         L4::Cap<L4Re::Dataspace> fds = fi->data_space();
00593
00594         if (!fds.is_valid())
00595         {
00596             return -EINVAL;
00597         }
00598
00599         if (len + offset > l4_round_page(fds->size()))
00600         {
00601             return -EINVAL;
00602         }
00603
00604         if (flags & MAP_PRIVATE)
00605         {
00606             DEBUG_LOG(debug_mmap, outstring("COW\n"));
00607             int err = ds->copy_in(anon_offset, fds, offset, len);
00608             if (err < 0)
00609                 return err;
00610
00611             offset = anon_offset;
00612         }
00613         else
00614         {
00615             L4Re::virt_cap_alloc->take(fds);
00616             ds = L4Re::Shared_cap<L4Re::Dataspace>(fds, L4Re::virt_cap_alloc);
00617         }
00618     }
00619     else
00620         offset = anon_offset;
00621
00622

```

```

00623     if (!(flags & MAP_FIXED) && start == 0)
00624         start = (void*)L4_PAGESIZE;
00625
00626     char *data = (char *)start;
00627     L4::Cap<Rm> r = Env::env()->rm();
00628     l4_addr_t overmap_area = L4_INVALID_ADDR;
00629
00630     int err;
00631     if (flags & MAP_FIXED)
00632     {
00633         overmap_area = l4_addr_t(start);
00634
00635         err = r->reserve_area(&overmap_area, len);
00636         if (err < 0)
00637             overmap_area = L4_INVALID_ADDR;
00638
00639         rm_flags |= Rm::F::In_area;
00640
00641         err = munmap(start, len);
00642         if (err && err != -ENOENT)
00643             return err;
00644     }
00645
00646     if (!(flags & MAP_FIXED))    rm_flags |= Rm::F::Search_addr;
00647     if (prot & PROT_READ)       rm_flags |= Rm::F::R;
00648     if (prot & PROT_WRITE)      rm_flags |= Rm::F::W;
00649     if (prot & PROT_EXEC)       rm_flags |= Rm::F::X;
00650
00651     err = r->attach(&data, len, rm_flags,
00652                    L4::Ipc::make_cap(ds.get(), (prot & PROT_WRITE)
00653                                     ? L4_CAP_FPAGE_RW
00654                                     : L4_CAP_FPAGE_RO),
00655                    offset);
00656
00657     DEBUG_LOG(debug_mmap, {
00658         outstring("  MAPPED: ");
00659         outhex32(ds.cap());
00660         outstring("  addr: ");
00661         outhex32(l4_addr_t(data));
00662         outstring("  bytes: ");
00663         outhex32(len);
00664         outstring("  offset: ");
00665         outhex32(offset);
00666         outstring("  err=");
00667         outdec(err);
00668         outstring("\n");
00669     });
00670
00671
00672     if (overmap_area != L4_INVALID_ADDR)
00673         r->free_area(overmap_area);
00674
00675     if (err < 0)
00676         return err;
00677
00678     l4_assert (!(start && !data));
00679
00680     // release ownership of the attached DS
00681     ds.release();
00682     *resptr = data;
00683
00684     return 0;
00685 }
00686
00687 namespace {
00688     class Auto_area
00689     {
00690     public:
00691         L4::Cap<L4Re::Rm> r;
00692         l4_addr_t a;
00693
00694         explicit Auto_area(L4::Cap<L4Re::Rm> r, l4_addr_t a = L4_INVALID_ADDR)
00695             : r(r), a(a) {}
00696
00697         int reserve(l4_addr_t _a, l4_size_t sz, L4Re::Rm::Flags flags)
00698         {
00699             free();
00700             a = _a;
00701             int e = r->reserve_area(&a, sz, flags);
00702             if (e)
00703                 a = L4_INVALID_ADDR;
00704             return e;
00705         }
00706
00707         void free()
00708         {
00709             if (is_valid())

```

```

00710     {
00711         r->free_area(a);
00712         a = L4_INVALID_ADDR;
00713     }
00714 }
00715
00716 bool is_valid() const { return a != L4_INVALID_ADDR; }
00717
00718 ~Auto_area() { free(); }
00719 };
00720 }
00721
00722 int
00723 Vfs::mremap(void *old_addr, size_t old_size, size_t new_size, int flags,
00724             void **new_addr) L4_NOTHROW
00725 {
00726     using namespace L4Re;
00727
00728     DEBUG_LOG(debug_mmap, {
00729         outstring("Mremap: addr=");
00730         outhex32((l4_umword_t)old_addr);
00731         outstring(" old_size=");
00732         outhex32(old_size);
00733         outstring(" new_size=");
00734         outhex32(new_size);
00735         outstring("\n");
00736     });
00737
00738     if (flags & MREMAP_FIXED && !(flags & MREMAP_MAYMOVE))
00739         return -EINVAL;
00740
00741     l4_addr_t oa = l4_trunc_page((l4_addr_t)old_addr);
00742     if (oa != (l4_addr_t)old_addr)
00743         return -EINVAL;
00744
00745     bool const fixed = flags & MREMAP_FIXED;
00746     bool const maymove = flags & MREMAP_MAYMOVE;
00747
00748     L4::Cap<Rm> r = Env::env()->rm();
00749
00750     // sanitize input parameters to multiples of pages
00751     old_size = l4_round_page(old_size);
00752     new_size = l4_round_page(new_size);
00753
00754     if (!fixed)
00755     {
00756         if (new_size < old_size)
00757         {
00758             *new_addr = old_addr;
00759             return munmap((void*)(oa + new_size), old_size - new_size);
00760         }
00761
00762         if (new_size == old_size)
00763         {
00764             *new_addr = old_addr;
00765             return 0;
00766         }
00767     }
00768
00769     Auto_area old_area(r);
00770     int err = old_area.reserve(oa, old_size, L4Re::Rm::Flags(0));
00771     if (err < 0)
00772         return -EINVAL;
00773
00774     l4_addr_t pad_addr;
00775     Auto_area new_area(r);
00776     if (fixed)
00777     {
00778         l4_addr_t na = l4_trunc_page((l4_addr_t)*new_addr);
00779         if (na != (l4_addr_t)*new_addr)
00780             return -EINVAL;
00781
00782         // check if the current virtual memory area can be expanded
00783         int err = new_area.reserve(na, new_size, L4Re::Rm::Flags(0));
00784         if (err < 0)
00785             return err;
00786
00787         pad_addr = na;
00788         // unmap all stuff and remap ours ....
00789     }
00790     else
00791     {
00792         l4_addr_t ta = oa + old_size;
00793         unsigned long ts = new_size - old_size;
00794         // check if the current virtual memory area can be expanded
00795         long err = new_area.reserve(ta, ts, L4Re::Rm::Flags(0));
00796         if (!maymove && err)

```

```

00797         return -ENOMEM;
00798
00799     L4Re::Rm::Offset toffs;
00800     L4Re::Rm::Flags tflags;
00801     L4::Cap<L4Re::Dataspace> tds;
00802
00803     err = r->find(&ta, &ts, &toffs, &tflags, &tds);
00804
00805     // there is enough space to expand the mapping in place
00806     if (err == -ENOENT || (err == 0 && (tflags & Rm::F::In_area)))
00807     {
00808         old_area.free(); // pad at the original address
00809         pad_addr = oa + old_size;
00810         *new_addr = old_addr;
00811     }
00812     else if (!maymove)
00813         return -ENOMEM;
00814     else
00815     {
00816         // search for a new area to remap
00817         err = new_area.reserve(0, new_size, Rm::F::Search_addr);
00818         if (err < 0)
00819             return -ENOMEM;
00820
00821         pad_addr = new_area.a + old_size;
00822         *new_addr = (void *)new_area.a;
00823     }
00824 }
00825
00826 if (old_area.is_valid())
00827 {
00828     l4_addr_t a = old_area.a;
00829     unsigned long s = old_size;
00830     L4Re::Rm::Offset o;
00831     L4Re::Rm::Flags f;
00832     L4::Cap<L4Re::Dataspace> ds;
00833
00834     for (; r->find(&a, &s, &o, &f, &ds) >= 0 && !(f & Rm::F::In_area));
00835     {
00836         if (a < old_area.a)
00837         {
00838             auto d = old_area.a - a;
00839             a = old_area.a;
00840             s -= d;
00841             o += d;
00842         }
00843
00844         if (a + s > old_area.a + old_size)
00845             s = old_area.a + old_size - a;
00846
00847         l4_addr_t x = a - old_area.a + new_area.a;
00848
00849         int err = r->attach(&x, s, Rm::F::In_area | f,
00850                          L4::Ipc::make_cap(ds, f.cap_rights()),
00851                          o);
00852         if (err < 0)
00853             return err;
00854
00855         // cout the new attached ds reference
00856         L4Re::virt_cap_alloc->take(ds);
00857
00858         err = r->detach(a, s, &ds, This_task,
00859                       Rm::Detach_exact | Rm::Detach_keep);
00860         if (err < 0)
00861             return err;
00862
00863         switch (err & Rm::Detach_result_mask)
00864         {
00865             case Rm::Split_ds:
00866                 // add a reference as we split up a mapping
00867                 if (ds.is_valid())
00868                     L4Re::virt_cap_alloc->take(ds);
00869                 break;
00870             case Rm::Detached_ds:
00871                 if (ds.is_valid())
00872                     L4Re::virt_cap_alloc->release(ds);
00873                 break;
00874             default:
00875                 break;
00876         }
00877     }
00878     old_area.free();
00879 }
00880
00881 if (old_size < new_size)
00882 {
00883     l4_addr_t const pad_sz = new_size - old_size;

```



```

00884     L4_addr_t toffs;
00885     L4Re::Shared_cap<L4Re::Dataspace> tds;
00886     int err = alloc_anon_mem(pad_sz, &tds, &toffs);
00887     if (err)
00888         return err;
00889
00890     // FIXME: must get the protection rights from the old
00891     // mapping and use the same here, for now just use RWX
00892     err = r->attach(&pad_addr, pad_sz,
00893                   Rm::F::In_area | Rm::F::Detach_free | Rm::F::RWX,
00894                   L4::Ipc::make_cap_rw(tds.get()), toffs);
00895     if (err < 0)
00896         return err;
00897
00898     // release ownership of tds, the region map is now the new owner
00899     tds.release();
00900 }
00901
00902 return 0;
00903 }
00904
00905 int
00906 Vfs::mprotect(const void *a, size_t sz, int prot) L4_NOTHROW
00907 {
00908     (void)a;
00909     (void)sz;
00910     return (prot & PROT_WRITE) ? -1 : 0;
00911 }
00912
00913 int
00914 Vfs::msync(void *, size_t, int) L4_NOTHROW
00915 { return 0; }
00916
00917 int
00918 Vfs::madvice(void *, size_t, int) L4_NOTHROW
00919 { return 0; }
00920
00921 }
00922
00923 L4Re::Vfs::Ops *__rtld_l4re_env_posix_vfs_ops;
00924 extern void *l4re_env_posix_vfs_ops __attribute__((alias("__rtld_l4re_env_posix_vfs_ops"),
00925 visibility("default"))));
00926
00927 namespace {
00928     class Real_mount_tree : public L4Re::Vfs::Mount_tree
00929     {
00930     public:
00931         explicit Real_mount_tree(char *n) : Mount_tree(n) {}
00932
00933         void *operator new (size_t size)
00934         { return __rtld_l4re_env_posix_vfs_ops->malloc(size); }
00935
00936         void operator delete (void *mem)
00937         { __rtld_l4re_env_posix_vfs_ops->free(mem); }
00938     };
00939
00940 int
00941 Vfs::mount(char const *path, cxx::Ref_ptr<L4Re::Vfs::File> const &dir) noexcept
00942 {
00943     using L4Re::Vfs::File;
00944     using L4Re::Vfs::Mount_tree;
00945     using L4Re::Vfs::Path;
00946
00947     cxx::Ref_ptr<Mount_tree> root = get_root()->mount_tree();
00948     if (!root)
00949         return -EINVAL;
00950
00951     cxx::Ref_ptr<Mount_tree> base;
00952     Path p = root->lookup(Path(path), &base);
00953
00954     while (!p.empty())
00955     {
00956         Path f = p.strip_first();
00957
00958         if (f.empty())
00959             return -EEXIST;
00960
00961         char *name = __rtld_l4re_env_posix_vfs_ops->strndup(f.path(), f.length());
00962         if (!name)
00963             return -ENOMEM;
00964
00965         cxx::Ref_ptr<Mount_tree> nt(new Real_mount_tree(name));
00966         if (!nt)
00967         {
00968             __rtld_l4re_env_posix_vfs_ops->free(name);
00969             return -ENOMEM;

```

```

00970     }
00971
00972     base->add_child_node(nt);
00973     base = nt;
00974
00975     if (p.empty())
00976     {
00977         nt->mount(dir);
00978         return 0;
00979     }
00980 }
00981
00982 return -EINVAL;
00983 }
00984
00985
00986 #undef DEBUG_LOG
00987 #undef GET_FILE_DBG
00988 #undef GET_FILE
00989

```

16.218 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() const 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 private:
00470     int _ref_cnt;
00471     cxx::Ref_ptr<Mount_tree> _mount_tree;
00472 };
00473
00474
00475 inline
00476 File::~File() noexcept
00477 {}
00478
00479 class Path
00480 {
00481 private:
00482     char const *_p;
00483     unsigned _l;
00484
00485 public:
00486     Path() noexcept : _p(0), _l(0) {}
00487
00488     explicit Path(char const *p) noexcept : _p(p)
00489     { for (_l = 0; *p; ++p, ++_l) ; }
00490

```

```

00491 Path(char const *p, unsigned l) noexcept : _p(p), _l(l)
00492 {}
00493
00494 static bool __is_sep(char s) noexcept;
00495
00496 Path cmp_path(char const *prefix) const noexcept;
00497
00498 struct Invalid_ptr;
00499 operator Invalid_ptr const * () const
00500 { return reinterpret_cast<Invalid_ptr const *>(_p); }
00501
00502 unsigned length() const { return _l; }
00503 char const *path() const { return _p; }
00504
00505 bool empty() const { return _l == 0; }
00506
00507 bool is_sep(unsigned offset) const { return __is_sep(_p[offset]); }
00508
00509 bool strip_sep()
00510 {
00511     bool s = false;
00512     for (; __is_sep(*_p) && _l; ++_p, --_l)
00513         s = true;
00514     return s;
00515 }
00516
00517 Path first() const
00518 {
00519     unsigned i;
00520     for (i = 0; i < _l && !is_sep(i); ++i)
00521         ;
00522     return Path(_p, i);
00523 }
00524
00525 Path strip_first()
00526 {
00527     Path r = first();
00528     _p += r.length();
00529     _l -= r.length();
00530     strip_sep();
00531     return r;
00532 }
00533
00534 };
00535
00536
00537 class Mount_tree
00538 {
00539 public:
00540     explicit Mount_tree(char *n) noexcept;
00541
00542     Path lookup(Path const &path, cxx::Ref_ptr<Mount_tree> *mt,
00543                 cxx::Ref_ptr<Mount_tree> *mp = 0) noexcept;
00544
00545     Path find(Path const &p, cxx::Ref_ptr<Mount_tree> *t) noexcept;
00546
00547     cxx::Ref_ptr<File> mount() const
00548     { return _mount; }
00549
00550     void mount(cxx::Ref_ptr<File> const &m)
00551     {
00552         m->_mount_tree = cxx::ref_ptr(this);
00553         _mount = m;
00554     }
00555
00556     static int create_tree(cxx::Ref_ptr<Mount_tree> const &root,
00557                           char const *path,
00558                           cxx::Ref_ptr<File> const &dir) noexcept;
00559
00560     void add_child_node(cxx::Ref_ptr<Mount_tree> const &cld);
00561
00562     virtual ~Mount_tree() noexcept = 0;
00563
00564     void add_ref() noexcept { ++_ref_cnt; }
00565     int remove_ref() noexcept { return --_ref_cnt; }
00566
00567 private:
00568     friend class Real_mount_tree;
00569
00570     int _ref_cnt;
00571     char *_name;
00572     cxx::Ref_ptr<Mount_tree> _cld;
00573     cxx::Ref_ptr<Mount_tree> _sib;
00574     cxx::Ref_ptr<File> _mount;
00575 };

```

```

00584
00585 inline
00586 Mount_tree::~Mount_tree() noexcept
00587 {}
00588
00589 inline bool
00590 Path::__is_sep(char s) noexcept
00591 { return s == '/'; }
00592
00593 inline Path
00594 Path::cmp_path(char const *n) const noexcept
00595 {
00596     char const *p = _p;
00597     for (; *p && !__is_sep(*p) && *n; ++p, ++n)
00598         if (*p != *n)
00599             return Path();
00600     if (*n || (*p && !__is_sep(*p)))
00601         return Path();
00602     return Path(p, _l - (p - _p));
00603 }
00604
00605 inline
00606 Mount_tree::Mount_tree(char *n) noexcept
00607 : _ref_cnt(0), _name(n)
00608 {}
00609
00610 inline Path
00611 Mount_tree::find(Path const &p, cxx::Ref_ptr<Mount_tree> *t) noexcept
00612 {
00613     if (!_cld)
00614         return Path();
00615     for (cxx::Ref_ptr<Mount_tree> x = _cld; x; x = x->_sib)
00616     {
00617         Path const r = p.cmp_path(x->_name);
00618         if (r)
00619         {
00620             *t = x;
00621             return r;
00622         }
00623     }
00624     return Path();
00625 }
00626
00627 inline Path
00628 Mount_tree::lookup(Path const &path, cxx::Ref_ptr<Mount_tree> *mt,
00629 cxx::Ref_ptr<Mount_tree> *mp) noexcept
00630 {
00631     cxx::Ref_ptr<Mount_tree> x(this);
00632     Path p = path;
00633     if (p.first().cmp_path("."))
00634         p.strip_first();
00635     Path last_mp = p;
00636     if (mp)
00637         *mp = x;
00638     while (1)
00639     {
00640         Path r = x->find(p, &x);
00641         if (!r)
00642         {
00643             if (mp)
00644                 return last_mp;
00645             if (mt)
00646                 *mt = x;
00647             return p;
00648         }
00649         r.strip_sep();
00650         if (mp && x->_mount)
00651         {
00652             last_mp = r;
00653             *mp = x;
00654         }
00655         if (r.empty())
00656         {
00657

```

```

00671     if (mt)
00672         *mt = x;
00673
00674     if (mp)
00675         return last_mp;
00676     else
00677         return r;
00678 }
00679
00680     p = r;
00681 }
00682 }
00683
00684 inline
00685 void
00686 Mount_tree::add_child_node(cxx::Ref_ptr<Mount_tree> const &cld)
00687 {
00688     cld->_sib = _cld;
00689     _cld = cld;
00690 }
00691
00692 inline
00693 const char *
00694 File::get_mount(const char *path, cxx::Ref_ptr<File> *dir,
00695                 cxx::Ref_ptr<Mount_tree> *mt) noexcept
00696 {
00697     if (!_mount_tree)
00698     {
00699         *dir = cxx::ref_ptr(this);
00700         return path;
00701     }
00702
00703     cxx::Ref_ptr<Mount_tree> mp;
00704     Path p = _mount_tree->lookup(Path(path), mt, &mp);
00705     if (mp->mount())
00706     {
00707         *dir = mp->mount();
00708         return p.path();
00709     }
00710     else
00711     {
00712         *dir = cxx::ref_ptr(this);
00713         return path;
00714     }
00715 }
00716
00717 inline int
00718 File::openat(const char *path, int flags, mode_t mode,
00719              cxx::Ref_ptr<File> *f) noexcept
00720 {
00721     cxx::Ref_ptr<File> dir;
00722     cxx::Ref_ptr<Mount_tree> mt;
00723     path = get_mount(path, &dir, &mt);
00724
00725     int res = dir->get_entry(path, flags, mode, f);
00726
00727     if (res < 0)
00728         return res;
00729
00730     if (!(*f)->_mount_tree && mt)
00731         (*f)->_mount_tree = mt;
00732
00733     return res;
00734 }
00735
00744 class Mman
00745 {
00746 public:
00748     virtual int mmap2(void *start, size_t len, int prot, int flags, int fd,
00749                      off_t offset, void **ptr) noexcept = 0;
00750
00752     virtual int munmap(void *start, size_t len) noexcept = 0;
00753
00755     virtual int mremap(void *old, size_t old_sz, size_t new_sz, int flags,
00756                       void **new_addr) noexcept = 0;
00757
00759     virtual int mprotect(const void *a, size_t sz, int prot) noexcept = 0;
00760
00762     virtual int msync(void *addr, size_t len, int flags) noexcept = 0;
00763
00765     virtual int madvise(void *addr, size_t len, int advice) noexcept = 0;
00766
00767     virtual ~Mman() noexcept = 0;
00768 };
00769
00770 inline
00771 Mman::~Mman() noexcept {}

```

```

00772
00773 class File_factory
00774 {
00775 private:
00776     int _ref_cnt = 0;
00777     int _proto = 0;
00778     char const *_proto_name = 0;
00779
00780     template<typename T> friend struct cxx::Default_ref_counter;
00781     void add_ref() noexcept { ++_ref_cnt; }
00782     int remove_ref() noexcept { return --_ref_cnt; }
00783
00784 public:
00785     explicit File_factory(int proto) : _proto(proto) {}
00786     explicit File_factory(char const *proto_name) : _proto_name(proto_name) {}
00787     File_factory(int proto, char const *proto_name)
00788     : _proto(proto), _proto_name(proto_name)
00789     {}
00790
00791     File_factory(File_factory const &) = delete;
00792     File_factory &operator = (File_factory const &) = delete;
00793
00794     char const *proto_name() const { return _proto_name; }
00795     int proto() const { return _proto; }
00796
00797     virtual ~File_factory() noexcept = 0;
00798     virtual cxx::Ref_ptr<File> create(L4::Cap<void> file) = 0;
00799 };
00800
00801 inline File_factory::~File_factory() noexcept {}
00802
00803 template<typename IFACE, typename IMPL>
00804 class File_factory_t : public File_factory
00805 {
00806 public:
00807     File_factory_t()
00808     : File_factory(IFACE::Protocol, L4::kobject_typeid<IFACE>()->name())
00809     {}
00810
00811     cxx::Ref_ptr<File> create(L4::Cap<void> file) override
00812     { return cxx::ref_ptr(new IMPL(L4::cap_cast<IFACE>(file))); }
00813 };
00814
00828 class File_system
00829 {
00830 protected:
00831     File_system *_next;
00832
00833 public:
00834     File_system() noexcept : _next(0) {}
00840     virtual char const *type() const noexcept = 0;
00841
00858     virtual int mount(char const *source, unsigned long mountflags,
00859                     void const *data, cxx::Ref_ptr<File> *dir) noexcept = 0;
00860
00861     virtual ~File_system() noexcept = 0;
00862
00867     File_system *next() const noexcept { return _next; }
00868     File_system *&next() noexcept { return _next; }
00869     void next(File_system *n) noexcept { _next = n; }
00870 };
00871
00872 inline
00873 File_system::~File_system() noexcept
00874 {}
00875
00881 class Fs
00882 {
00883 public:
00889     virtual cxx::Ref_ptr<File> get_file(int fd) noexcept = 0;
00890
00892     virtual cxx::Ref_ptr<File> get_root() noexcept = 0;
00893
00895     virtual cxx::Ref_ptr<File> get_cwd() noexcept { return get_root(); }
00896
00898     virtual void set_cwd(cxx::Ref_ptr<File> const &) noexcept {}
00899
00905     virtual int alloc_fd(cxx::Ref_ptr<File> const &f = cxx::Ref_ptr<>::Nil) noexcept = 0;
00906
00917     virtual cxx::Pair<cxx::Ref_ptr<File>, int>
00918     set_fd(int fd, cxx::Ref_ptr<File> const &f = cxx::Ref_ptr<>::Nil) noexcept = 0;
00919
00925     virtual cxx::Ref_ptr<File> free_fd(int fd) noexcept = 0;
00926
00934     virtual int mount(char const *path, cxx::Ref_ptr<File> const &dir) noexcept = 0;
00935
00943     virtual int register_file_system(File_system *f) noexcept = 0;

```



```

00944
00952     virtual int unregister_file_system(File_system *f) noexcept = 0;
00953
00961     virtual File_system *get_file_system(char const *fstype) noexcept = 0;
00962
00966     int mount(char const *source, char const *target,
00967               char const *fstype, unsigned long mountflags,
00968               void const *data) noexcept;
00969
00970     virtual int register_file_factory(cxx::Ref_ptr<File_factory> f) noexcept = 0;
00971     virtual int unregister_file_factory(cxx::Ref_ptr<File_factory> f) noexcept = 0;
00972     virtual cxx::Ref_ptr<File_factory> get_file_factory(int proto) noexcept = 0;
00973     virtual cxx::Ref_ptr<File_factory> get_file_factory(char const *proto_name) noexcept = 0;
00974
00975     virtual ~Fs() = 0;
00976 };
00977
00978 inline int
00979 Fs::mount(char const *source, char const *target,
00980           char const *fstype, unsigned long mountflags,
00981           void const *data) noexcept
00982 {
00983     File_system *fs = get_file_system(fstype);
00984
00985     if (!fs)
00986         return -ENODEV;
00987
00988     cxx::Ref_ptr<File> dir;
00989     int res = fs->mount(source, mountflags, data, &dir);
00990
00991     if (res < 0)
00992         return res;
00993
00994     return mount(target, dir);
00995 }
00996
00997 inline
00998 Fs::~Fs()
00999 {}
01000
01007 class Ops : public Mman, public Fs
01008 {
01009 public:
01010     virtual void *malloc(size_t bytes) noexcept = 0;
01011     virtual void free(void *mem) noexcept = 0;
01012     virtual ~Ops() noexcept = 0;
01013
01014     char *strndup(char const *str, unsigned l) noexcept
01015     {
01016         unsigned len;
01017         for (len = 0; str[len] && len < l; ++len)
01018             ;
01019
01020         if (len == 0)
01021             return nullptr;
01022
01023         ++len;
01024
01025         char *b = (char *)this->malloc(len);
01026         if (b == nullptr)
01027             return nullptr;
01028
01029         char *r = b;
01030         for (; len - 1 > 0 && *str; --len, ++b, ++str)
01031             *b = *str;
01032
01033         *b = 0;
01034         return r;
01035     }
01036
01037 };
01038
01039 inline
01040 Ops::~Ops() noexcept
01041 {}
01042
01043 }}
01044
01045 #endif
01046

```

16.219 virtio-net

```
00001 // vi:ft=cpp
```

```

00002  /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
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::chkcap(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 } }

```

16.220 l4virtio

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
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::Eoq) // 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     {
00331         ret = _next_devaddr;
00332         if (l4_umword_t(~0) - ret < size)
00333             L4Re::chksys(-L4_ENOMEM, "Out of device address space.");
00334     }
00335     while (!l4util_cmpxchg(&_next_devaddr, ret, ret + size));
00336
00337     return ret;
00338 }
00339
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 } }

```

16.221 l4virtio

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00003 /*
00004  * Copyright (C) 2013-2022 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/utills>
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
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 send_cmd(unsigned command, L4::Cap<L4::Triggerable> out_notify,
00088                 L4::Cap<L4::Triggerable> in_notify,
00089                 l4_timeout_s to = L4_IPC_TIMEOUT_NEVER)
00090     {
00091         cxx::write_now(&cmd, command);
00092
00093         if (out_notify)
00094             out_notify->trigger();
00095
00096         auto utcb = l4_utcb();
00097         auto ipc_to = l4_timeout(L4_IPC_TIMEOUT_0, to);
00098
00099         do
00100         {
00101             if (in_notify)
00102                 if (l4_ipc_error(l4_ipc_receive(in_notify.cap(), utcb, ipc_to),
00103                                utcb) == L4_IPC_RETIMEOUT)
00104                     break;
00105         }
00106         while (cxx::access_once(&cmd));
00107
00108         return cxx::access_once(&cmd) ? -L4_EBUSY : L4_EOK;
00109     }
00110 };
00111
00121 L4_INLINE_RPC_OP(L4VIRTIO_OP_SET_STATUS, long,
00122                 set_status, (unsigned status));

```

```

00123
00139     L4_INLINE_RPC_OP(L4VIRTIO_OP_CONFIG_QUEUE, long,
00140                     config_queue, (unsigned queue));
00141
00165     L4_INLINE_RPC_OP(L4VIRTIO_OP_REGISTER_DS, long,
00166                     register_ds, (L4::Ipc::Cap<L4Re::Dataspace> ds_cap,
00167                                   l4_uint64_t base, l4_umword_t offset,
00168                                   l4_umword_t size));
00169
00178     L4_INLINE_RPC_OP(L4VIRTIO_OP_DEVICE_CONFIG, long, device_config,
00179                     (L4::Ipc::Out<L4::Cap<L4Re::Dataspace> > config_ds,
00180                      l4_addr_t *ds_offset));
00181
00200     L4_INLINE_RPC_OP(L4VIRTIO_OP_GET_DEVICE_IRQ, long, device_notification_irq,
00201                     (unsigned index, L4::Ipc::Out<L4::Cap<L4::Triggerable> > irq));
00202
00203
00204     typedef L4::Typeid::Rpc<set_status_t, config_queue_t, register_ds_t,
00205                             device_config_t, device_notification_irq_t>
00206         Rpc;
00207 };
00208
00209 }

```

16.222 l4virtio

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00003 /*
00004  * Copyright (C) 2014-2022 Kernkonzept GmbH.
00005  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00006  *            Manuel von Oltersdorff-Kalettka <manuel.kalettka@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::chkcap;
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 = chkcap(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]; }

```

```

00180
00192  l4_uint32_t guest_features(unsigned idx) const
00193  { return _config->driver_features_map[idx]; }
00194
00206  l4_uint32_t negotiated_features(unsigned idx) const
00207  { return _config->driver_features_map[idx] & _host_features[idx]; }
00208
00216  Status status() const { return _status; }
00217
00224  l4_uint32_t get_cmd() const
00225  {
00226      return hdr()->cmd;
00227  }
00228
00235  void reset_cmd()
00236  {
00237      const_cast<l4_uint32_t volatile &>(hdr()->cmd) = 0;
00238  }
00239
00247  void set_status(Status status)
00248  {
00249      _status = status;
00250      const_cast<l4_uint32_t volatile &>(hdr()->status) = status.raw;
00251  }
00252
00259  void set_device_needs_reset()
00260  {
00261      _status.device_needs_reset() = 1;
00262      const_cast<l4_uint32_t volatile &>(hdr()->status) = _status.raw;
00263  }
00264
00269  bool change_queue_config(l4_uint32_t num_queues)
00270  {
00271      if (sizeof(l4virtio_config_queue_t) * num_queues + _qoffset > L4_PAGESIZE)
00272          // too many queues does not fit into our page
00273          return false;
00274
00275      _nqueues = num_queues;
00276      reset_hdr(true);
00277      return true;
00278  }
00279
00286  l4virtio_config_queue_t volatile const *qconfig(unsigned index) const
00287  {
00288      if (L4_UNLIKELY(_qoffset < sizeof (l4virtio_config_hdr_t)))
00289          return 0;
00290
00291      if (L4_UNLIKELY(index >= _nqueues))
00292          return 0;
00293
00294      return reinterpret_cast<l4virtio_config_queue_t const *>
00295          (reinterpret_cast<char *>(_config.get()) + _qoffset) + index;
00296  }
00297
00301  void reset_hdr(bool inc_generation = false) const
00302  {
00303      _config->magic = L4VIRTIO_MAGIC;
00304      _config->version = 2;
00305      _config->device = _device;
00306      _config->vendor = _vendor;
00307      _config->status = 0;
00308      _config->irq_status = 0;
00309      _config->num_queues = _nqueues;
00310      _config->queues_offset = _qoffset;
00311
00312      memcpy(_config->dev_features_map, _host_features,
00313             sizeof(_config->dev_features_map));
00314      wmb();
00315      if (inc_generation)
00316          ++_config->generation;
00317  }
00318
00319
00328  bool reset_queue(unsigned index, unsigned num_max,
00329                  bool inc_generation = false) const
00330  {
00331      l4virtio_config_queue_t volatile *qc;
00332      // this function is allowed to write to the device config
00333      qc = const_cast<l4virtio_config_queue_t volatile *>(qconfig(index));
00334      if (L4_UNLIKELY(qc == 0))
00335          return false;
00336
00337      qc->num_max = num_max;
00338      qc->num = 0;
00339      qc->ready = 0;
00340      wmb();
00341      if (inc_generation)

```

```

00342         ++_config->generation;
00343
00344         return true;
00345     }
00346
00351     l4virtio_config_hdr_t const volatile *hdr() const
00352     { return _config.get(); }
00353
00358     L4::Cap<L4Re::Dataspace> ds() const { return _ds.get(); }
00359
00364     l4_addr_t ds_offset() const
00365     { return _ds_offset; }
00366 };
00367
00368
00369 template<typename PRIV_CONFIG>
00370 class Dev_config_t : public Dev_config
00371 {
00372 public:
00374     typedef PRIV_CONFIG Priv_config;
00375
00387     Dev_config_t(l4_uint32_t vendor, l4_uint32_t device,
00388                 l4_uint32_t num_queues = 0)
00389     : Dev_config(vendor, device, sizeof(PRIV_CONFIG), num_queues)
00390     {}
00391
00402     Dev_config_t(L4Re::Util::Shared_cap<L4Re::Dataspace> const &cfg,
00403                 l4_addr_t cfg_offset, l4_uint32_t vendor, l4_uint32_t device,
00404                 l4_uint32_t num_queues = 0)
00405     : Dev_config(cfg, cfg_offset, vendor, device, sizeof(PRIV_CONFIG),
00406                 num_queues)
00407     {}
00408
00418     Priv_config volatile *priv_config() const
00419     {
00420         return static_cast<Priv_config volatile *>(get_priv_config());
00421     }
00422
00423 };
00424
00425 struct No_custom_data {};
00426
00432 template <typename DATA>
00433 class Driver_mem_region_t : public DATA
00434 {
00435 public:
00436     struct Flags
00437     {
00438         l4_uint32_t raw;
00439         CXX_BITFIELD_MEMBER(0, 0, rw, raw);
00440     };
00441
00442 private:
00444     typedef L4Re::Util::Unique_cap<L4Re::Dataspace> Ds_cap;
00445
00446     l4_uint64_t _drv_base;
00447     l4_uint64_t _trans_offset;
00448     l4_umword_t _size;
00449     Flags      _flags;
00450
00451     Ds_cap      _ds;
00452     l4_addr_t    _ds_offset;
00453
00455     L4Re::Rm::Unique_region<l4_addr_t> _local_base;
00456
00457     template<typename T>
00458     T _local(l4_uint64_t addr) const
00459     {
00460         return (T)(addr - _trans_offset);
00461     }
00462
00463 public:
00465     Driver_mem_region_t() : _size(0) {}
00466
00478     Driver_mem_region_t(l4_uint64_t drv_base, l4_umword_t size,
00479                         l4_addr_t offset, Ds_cap &&ds)
00480     : _drv_base(l4_trunc_page(drv_base)), _size(0),
00481       _ds_offset(l4_trunc_page(offset))
00482     {
00483         using L4Re::chksys;
00484         using L4Re::Env;
00485
00486         L4Re::Dataspace::Stats ds_info = L4Re::Dataspace::Stats();
00487         // Sometimes we work with dataspace that do not implement all dataspace
00488         // methods and return an error instead. An example of such a dataspace is
00489         // io's Vi::System_bus. We detect this case when the info method returns
00490         // -L4_ENOSYS and simply assume the dataspace is good for us.

```

```

00491     long err = ds->info(&ds_info);
00492     if (err >= 0)
00493     {
00494         l4_addr_t ds_size = l4_round_page(ds_info.size);
00495
00496         if (ds_size < L4_PAGESIZE)
00497             chksys(-L4_EINVAL, "DS too small");
00498
00499         if (_ds_offset >= ds_size)
00500             chksys(-L4_ERANGE, "offset larger than DS size");
00501
00502         size = l4_round_page(size);
00503         if (size > ds_size)
00504             chksys(-L4_EINVAL, "size larger than DS size");
00505
00506         if (_ds_offset > ds_size - size)
00507             chksys(-L4_EINVAL, "invalid offset or size");
00508
00509         // overflow check
00510         if ((ULONG_MAX - size) < _drv_base)
00511             chksys(-L4_EINVAL, "invalid size");
00512
00513         _flags.rw() = (ds_info.flags & L4Re::Dataspace::F::W).raw != 0;
00514     }
00515     else if (err == -L4_ENOSYS)
00516     {
00517         _flags.rw() = true;
00518     }
00519     else
00520     {
00521         chksys(err, "getting data-space infos");
00522     }
00523
00524     auto f = L4Re::Rm::F::Search_addr | L4Re::Rm::F::R;
00525     if (_flags.rw())
00526         f |= L4Re::Rm::F::W;
00527
00528     // use a big alignment to save PT/TLB entries and kernel memory resources!
00529     chksys(Env::env()->rm()->attach(&_local_base, size, f,
00530                                     L4::Ipc::make_cap(ds.get(), _flags.rw()
00531                                                         ? L4_CAP_FPAGE_RW
00532                                                         : L4_CAP_FPAGE_RO),
00533                                     _ds_offset, L4_SUPERPAGESHIFT));
00534
00535     _size = size;
00536     _ds = cxx::move(ds);
00537     _trans_offset = _drv_base - _local_base.get();
00538 }
00539
00541 bool is_writable() const { return _flags.rw(); }
00542
00544 Flags flags() const { return _flags; }
00545
00547 bool empty() const
00548 { return _size == 0; }
00549
00551 l4_uint64_t drv_base() const { return _drv_base; }
00552
00554 void *local_base() const { return (void*)_local_base.get(); }
00555
00557 l4_umword_t size() const { return _size; }
00558
00560 l4_addr_t ds_offset() const { return _ds_offset; }
00561
00563 L4::Cap<L4Re::Dataspace> ds() const { return _ds.get(); }
00564
00572 bool contains(l4_uint64_t base, l4_umword_t size) const
00573 {
00574     if (base < _drv_base)
00575         return false;
00576
00577     if (base > _drv_base + _size - 1)
00578         return false;
00579
00580     if (size > _size)
00581         return false;
00582
00583     if (base - _drv_base > _size - size)
00584         return false;
00585
00586     return true;
00587 }
00588
00595 template<typename T>
00596 T *local(Ptr<T> p) const
00597 { return _local<T*>(p.get()); }
00598 };

```

```

00599
00600 typedef Driver_mem_region_t<No_custom_data> Driver_mem_region;
00601
00602 template <typename DATA>
00603 class Driver_mem_list_t
00604 {
00605 public:
00606     typedef Driver_mem_region_t<DATA> Mem_region;
00607
00608 private:
00609     cxx::unique_ptr<Mem_region[]> _l;
00610     unsigned _max;
00611     unsigned _free;
00612
00613 public:
00614     typedef L4Re::Util::Unique_cap<L4Re::Dataspace> Ds_cap;
00615
00616     Driver_mem_list_t() : _max(0), _free(0) {}
00617
00618     void init(unsigned max)
00619     {
00620         _l = cxx::make_unique<Driver_mem_region_t<DATA>[]>(max);
00621         _max = max;
00622         _free = 0;
00623     }
00624
00625     bool full() const
00626     { return _free == _max; }
00627
00628     Mem_region const *add(l4_uint64_t drv_base, l4_umword_t size,
00629                          l4_addr_t offset, Ds_cap &&ds)
00630     {
00631         if (full())
00632             L4Re::chksys(-L4_ENOMEM);
00633
00634         _l[_free++] = Mem_region(drv_base, size, offset, cxx::move(ds));
00635         return &_l[_free - 1];
00636     }
00637
00638     void remove(Mem_region const *r)
00639     {
00640         if (r < &_l[0] || r >= &_l[_free])
00641             L4Re::chksys(-L4_ERANGE);
00642
00643         unsigned idx = r - &_l[0];
00644
00645         for (unsigned i = idx + 1; i < _free - 1; ++i)
00646             _l[i] = cxx::move(_l[i + 1]);
00647
00648         _l[--_free] = Mem_region();
00649     }
00650
00651     Mem_region *find(l4_uint64_t base, l4_umword_t size) const
00652     {
00653         return _find(base, size);
00654     }
00655
00656     void load_desc(Virtqueue::Desc const &desc, Request_processor const *p,
00657                   Virtqueue::Desc const **table) const
00658     {
00659         Mem_region const *r = find(desc.addr.get(), desc.len);
00660         if (L4_UNLIKELY(!r))
00661             throw Bad_descriptor(p, Bad_descriptor::Bad_address);
00662
00663         *table = static_cast<Virtqueue::Desc const *>(r->local(desc.addr));
00664     }
00665
00666     void load_desc(Virtqueue::Desc const &desc, Request_processor const *p,
00667                   Mem_region const **data) const
00668     {
00669         Mem_region const *r = find(desc.addr.get(), desc.len);
00670         if (L4_UNLIKELY(!r))
00671             throw Bad_descriptor(p, Bad_descriptor::Bad_address);
00672
00673         *data = r;
00674     }
00675
00676     template<typename ARG>
00677     void load_desc(Virtqueue::Desc const &desc, Request_processor const *p,
00678                   ARG *data) const
00679     {
00680         Mem_region *r = find(desc.addr.get(), desc.len);
00681         if (L4_UNLIKELY(!r))
00682             throw Bad_descriptor(p, Bad_descriptor::Bad_address);
00683
00684         *data = ARG(r, desc, p);
00685     }
00686
00687 }

```

```

00753
00754 private:
00755     Mem_region *_find(l4_uint64_t base, l4_umword_t size) const
00756     {
00757         for (unsigned i = 0; i < _free; ++i)
00758             if (_l[i].contains(base, size))
00759                 return &_l[i];
00760         return 0;
00761     }
00762
00763 };
00764
00765 typedef Driver_mem_list_t<No_custom_data> Driver_mem_list;
00766
00767 template<typename DATA>
00775 class Device_t
00776 {
00777 public:
00778     typedef Driver_mem_list_t<DATA> Mem_list;
00779
00780 protected:
00781     Mem_list _mem_info;
00782
00783 private:
00784     Dev_config *_device_config;
00785
00786     using Ds_vector = std::vector<L4::Cap<L4Re::Dataspace>>;
00787     std::shared_ptr<Ds_vector const> _trusted_ds_caps;
00788
00789     bool _trusted_ds_validation_enabled = false;
00790
00791 public:
00792     L4_RPC_LEGACY_DISPATCH(L4virtio::Device);
00793     template<typename IOS> int virtio_dispatch(unsigned r, IOS &ios)
00794     { return dispatch(r, ios); }
00795
00796     virtual void reset() = 0;
00797
00798     virtual bool check_features()
00799     { return true; }
00800
00801     virtual bool check_queues() = 0;
00802
00803     virtual int reconfig_queue(unsigned idx) = 0;
00804
00805     virtual void register_single_driver_irq()
00806     { L4Re::chksys(-L4_ENOSYS, "Legacy single IRQ interface not implemented."); }
00807
00808     virtual void trigger_driver_config_irq() const = 0;
00809
00810     virtual L4::Cap<L4::Irq> device_notify_irq() const
00811     {
00812         L4Re::chksys(-L4_ENOSYS, "Legacy single IRQ interface not implemented.");
00813         return L4::Cap<L4::Irq>();
00814     }
00815
00816     virtual void register_driver_irq(unsigned idx)
00817     {
00818         if (idx != 0)
00819             L4Re::chksys(-L4_ENOSYS, "Multi IRQ interface not implemented.");
00820
00821         register_single_driver_irq();
00822     }
00823
00824     virtual L4::Cap<L4::Irq> device_notify_irq(unsigned idx)
00825     {
00826         if (idx != 0)
00827             L4Re::chksys(-L4_ENOSYS, "Multi IRQ interface not implemented.");
00828
00829         return device_notify_irq();
00830     }
00831
00832     virtual unsigned num_events_supported() const
00833     { return 1; }
00834
00835     virtual L4::Ipc_svr::Server_iface *server_iface() const = 0;
00836
00837     Device_t(Dev_config *dev_config)
00838     : _device_config(dev_config)
00839     {}
00840
00841     Mem_list const *mem_info() const
00842     { return &_mem_info; }
00843
00844     long op_set_status(L4virtio::Device::Rights, unsigned status)
00845     { return _set_status(status); }

```

```

00874
00875 long op_config_queue(L4virtio::Device::Rights, unsigned queue)
00876 {
00877     Dev_config::Status status = _device_config->status();
00878     if (status.fail_state() || !status.acked() || !status.driver())
00879         return -L4_EIO;
00880
00881     return reconfig_queue(queue);
00882 }
00883
00884 long op_register_ds(L4virtio::Device::Rights,
00885                    L4::Ipc::Snd_fpage ds_cap_fp, l4_uint64_t ds_base,
00886                    l4_umword_t offset, l4_umword_t sz)
00887 {
00888     L4Re::Util::Dbg()
00889         .printf("Registering dataspace from 0x%llx with %lu KiB, offset 0x%lx\n",
00890                ds_base, sz » 10, offset);
00891
00892     _check_n_init_shm(ds_cap_fp, ds_base, sz, offset);
00893
00894     return 0;
00895 }
00896
00897 long op_device_config(L4virtio::Device::Rights,
00898                      L4::Ipc::Cap<L4Re::Dataspace> &config_ds,
00899                      l4_addr_t &ds_offset)
00900 {
00901     L4Re::Util::Dbg()
00902         .printf("register client: host IRQ: %lx config DS: %lx\n",
00903                device_notify_irq().cap(), _device_config->ds().cap());
00904
00905     config_ds = L4::Ipc::make_cap(_device_config->ds(), L4_CAP_FPAGE_RW);
00906     ds_offset = _device_config->ds_offset();
00907     return 0;
00908 }
00909
00910 long op_device_notification_irq(L4virtio::Device::Rights,
00911                                unsigned idx,
00912                                L4::Ipc::Cap<L4::Triggerable> &irq)
00913 {
00914     auto cap = device_notify_irq(idx);
00915
00916     if (!cap.is_valid())
00917         return -L4_EINVAL;
00918
00919     irq = L4::Ipc::make_cap(cap, L4_CAP_FPAGE_RO);
00920     return L4_EOK;
00921 }
00922
00923 int op_bind(L4::Icu::Rights, l4_umword_t idx, L4::Ipc::Snd_fpage irq_cap_fp)
00924 {
00925     if (idx >= num_events_supported())
00926         return -L4_ERANGE;
00927
00928     if (!irq_cap_fp.cap_received())
00929         return -L4_EINVAL;
00930
00931     register_driver_irq(idx);
00932
00933     return L4_EOK;
00934 }
00935
00936 int op_unbind(L4::Icu::Rights, l4_umword_t, L4::Ipc::Snd_fpage)
00937 {
00938     return -L4_ENOSYS;
00939 }
00940
00941 int op_info(L4::Icu::Rights, L4::Icu::_Info &info)
00942 {
00943     info.features = 0;
00944     info.nr_irqs = num_events_supported();
00945     info.nr_msis = 0;
00946
00947     return L4_EOK;
00948 }
00949
00950 int op_msi_info(L4::Icu::Rights, l4_umword_t, l4_uint64_t, l4_icu_msi_info_t &)
00951 { return -L4_ENOSYS; }
00952
00953 int op_mask(L4::Icu::Rights, l4_umword_t)
00954 { return -L4_ENOSYS; }
00955
00956 int op_unmask(L4::Icu::Rights, l4_umword_t)
00957 { return -L4_ENOREPLY; }
00958
00959 int op_set_mode(L4::Icu::Rights, l4_umword_t, l4_umword_t)
00960 { return -L4_ENOSYS; }

```

```

00961
00973 void reset_queue_config(unsigned idx, unsigned num_max,
00974                          bool inc_generation = false)
00975 {
00976     _device_config->reset_queue(idx, num_max, inc_generation);
00977 }
00978
00983 void init_mem_info(unsigned num)
00984 {
00985     _mem_info.init(num);
00986 }
00987
00995 void device_error()
00996 {
00997     reset();
00998     _device_config->set_device_needs_reset();
00999
01000     // the device MUST NOT notify the driver before DRIVER_OK.
01001     if (_device_config->status().driver_ok())
01002     {
01003         // we do not care about this anywhere, so skip
01004         // _dev_config->irq_status |= 1;
01005         trigger_driver_config_irq();
01006     }
01007 }
01008
01022 bool setup_queue(Virtqueue *q, unsigned qn, unsigned num_max)
01023 {
01024     l4virtio_config_queue_t volatile const *qc;
01025     qc = _device_config->qconfig(qn);
01026     if (L4_UNLIKELY(qc == 0))
01027         return false;
01028
01029     if (!qc->ready)
01030     {
01031         q->disable();
01032         return true;
01033     }
01034
01035     // read to local variables before check
01036     l4_uint32_t num = qc->num;
01037     l4_uint64_t desc = qc->desc_addr;
01038     l4_uint64_t avail = qc->avail_addr;
01039     l4_uint64_t used = qc->used_addr;
01040
01041     if (0)
01042         printf("%p: setup queue: num=0x%x max_num=0x%x desc=0x%llx avail=0x%llx used=0x%llx\n",
01043               this, num, num_max, desc, avail, used);
01044
01045     if (!num || num > num_max)
01046         return false;
01047
01048     // num must be power of two
01049     if (num & (num - 1))
01050         return false;
01051
01052     if (desc & 0xf)
01053         return false;
01054
01055     if (avail & 0x1)
01056         return false;
01057
01058     if (used & 0x3)
01059         return false;
01060
01061     auto const *desc_info = _mem_info.find(desc, Virtqueue::desc_size(num));
01062     if (L4_UNLIKELY(!desc_info))
01063         return false;
01064
01065     auto const *avail_info = _mem_info.find(avail, Virtqueue::avail_size(num));
01066     if (L4_UNLIKELY(!avail_info))
01067         return false;
01068
01069     auto const *used_info = _mem_info.find(used, Virtqueue::used_size(num));
01070     if (L4_UNLIKELY(!used_info || !used_info->is_writable()))
01071         return false;
01072
01073     L4Re::Util::Dbg()
01074         .printf("shm=[%llx-%llx] local=[%lx-%lx] desc=[%llx-%llx] (%p-%p)\n",
01075               desc_info->drv_base(), desc_info->drv_base() + desc_info->size() - 1,
01076               (unsigned long)desc_info->local_base(),
01077               (unsigned long)desc_info->local_base() + desc_info->size() - 1,
01078               desc, desc + Virtqueue::desc_size(num),
01079               desc_info->local(Ptr<char>(desc)),
01080               desc_info->local(Ptr<char>(desc)) + Virtqueue::desc_size(num));
01081
01082     L4Re::Util::Dbg()

```



```

01083     .printf("shm=[%llx-%llx] local=[%lx-%lx] avail=[%llx-%llx] (%p-%p)\n",
01084             avail_info->drv_base(), avail_info->drv_base() + avail_info->size() - 1,
01085             (unsigned long)avail_info->local_base(),
01086             (unsigned long)avail_info->local_base() + avail_info->size() - 1,
01087             avail, avail + Virtqueue::avail_size(num),
01088             avail_info->local(Ptr<char>(avail)),
01089             avail_info->local(Ptr<char>(avail)) + Virtqueue::avail_size(num));
01090
01091     L4Re::Util::Dbg()
01092     .printf("shm=[%llx-%llx] local=[%lx-%lx] used=[%llx-%llx] (%p-%p)\n",
01093            used_info->drv_base(), used_info->drv_base() + used_info->size() - 1,
01094            (unsigned long)used_info->local_base(),
01095            (unsigned long)used_info->local_base() + used_info->size() - 1,
01096            used, used + Virtqueue::used_size(num),
01097            used_info->local(Ptr<char>(used)),
01098            used_info->local(Ptr<char>(used)) + Virtqueue::used_size(num));
01099
01100     q->setup(num, desc_info->local(Ptr<void>(desc)),
01101             avail_info->local(Ptr<void>(avail)),
01102             used_info->local(Ptr<void>(used)));
01103     return true;
01104 }
01105
01106 void check_n_init_shm(L4Re::Util::Unique_cap<L4Re::Dataspace> &&shm,
01107                      l4_uint64_t base, l4_umword_t size, l4_addr_t offset)
01108 {
01109     if (_mem_info.full())
01110         L4Re::chksys(-L4_ENOMEM);
01111
01112     auto const *i = _mem_info.add(base, size, offset, cxx::move(shm));
01113     L4Re::Util::Dbg()
01114     .printf("PORT[%p]: DMA guest [%llx-%llx] local [%lx-%lx] offset %lx\n",
01115            this, i->drv_base(), i->drv_base() + i->size() - 1,
01116            (unsigned long)i->local_base(),
01117            (unsigned long)i->local_base() + i->size() - 1,
01118            i->ds_offset());
01119 }
01120
01129 bool handle_mem_cmd_write()
01130 {
01131     l4_uint32_t cmd = _device_config->get_cmd();
01132     if (L4_LIKELY(!(cmd & L4VIRTIO_CMD_MASK)))
01133         return false;
01134
01135     switch (cmd & L4VIRTIO_CMD_MASK)
01136     {
01137     case L4VIRTIO_CMD_SET_STATUS:
01138         _set_status(cmd & ~L4VIRTIO_CMD_MASK);
01139         break;
01140
01141     case L4VIRTIO_CMD_CFG_QUEUE:
01142         reconfig_queue(cmd & ~L4VIRTIO_CMD_MASK);
01143         break;
01144
01145     default:
01146         // unknown command
01147         break;
01148     }
01149
01150     _device_config->reset_cmd();
01151
01152     return true;
01153 }
01154
01158 void enable_trusted_ds_validation()
01159 {
01160     _trusted_ds_validation_enabled = true;
01161 }
01162
01168 void
01169 add_trusted_dataspaces(std::shared_ptr<Ds_vector const> ds)
01170 {
01171     _trusted_ds_caps = ds;
01172 }
01173
01174 private:
01187 long validate_ds(L4::Cap<L4Re::Dataspace> ds)
01188 {
01189     if (!_trusted_ds_caps)
01190         return -L4_EINVAL;
01191     if (std::any_of(_trusted_ds_caps->cbegin(), _trusted_ds_caps->cend(),
01192                    [&ds](L4::Cap<L4Re::Dataspace> cap)
01193                    {
01194                        return L4Re::Env::env()->task()
01195                               ->cap_equal(ds, cap).label() == 1;
01196                    }

```

```

01197     ))
01198     {
01199         return L4_EOK;
01200     }
01201     return -L4_EINVAL;
01202 }
01203
01204 void _check_n_init_shm(L4::Ipc::Snd_fpage shm_cap_fp,
01205                      l4_uint64_t base, l4_umword_t size, l4_addr_t offset)
01206 {
01207     if (!shm_cap_fp.cap_received())
01208         L4Re::chksys(-L4_EINVAL);
01209
01210     L4Re::Util::Unique_cap<L4Re::Dataspace> ds(
01211         L4Re::chkcap(server_iface()->template_rcv_cap<L4Re::Dataspace>(0)));
01212     L4Re::chksys(server_iface()->realloc_rcv_cap(0));
01213
01214     if (_trusted_ds_validation_enabled)
01215         L4Re::chksys(validate_ds(ds.get()), "Validating the dataspace.");
01216
01217     check_n_init_shm(cxx::move(ds), base, size, offset);
01218 }
01219
01220 bool check_features_internal()
01221 {
01222     static_assert(sizeof(l4virtio_config_hdr_t::driver_features_map)
01223                 == sizeof(l4virtio_config_hdr_t::dev_features_map),
01224                 "Driver and device feature maps must be of the same size");
01225
01226     // From the Virtio 1.0 specification 6.1 Driver Requirements and 6.2 Device
01227     // Requirements: A driver MUST accept VIRTIO_F_VERSION_1 if it is offered.
01228     // A device MUST offer VIRTIO_F_VERSION_1. A device MAY fail to operate
01229     // further if VIRTIO_F_VERSION_1 is not accepted.
01230     //
01231     // The L4virtio implementation does not support legacy interfaces so we
01232     // fail here if the Virtio 1.0 feature was not accepted.
01233     if (!_device_config->get_guest_feature(L4VIRTIO_FEATURE_VERSION_1))
01234         return false;
01235
01236     for (auto i = 0u;
01237          i < sizeof(l4virtio_config_hdr_t::driver_features_map)
01238              / sizeof(l4virtio_config_hdr_t::driver_features_map[0]);
01239          i++)
01240     {
01241         // Driver must not accept features that were not offered by device
01242         if (_device_config->guest_features(i)
01243             & ~_device_config->host_features(i))
01244             return false;
01245     }
01246     return check_features();
01247 }
01248
01270 void check_and_update_status(Dev_config::Status status)
01271 {
01272     // snapshot of current status
01273     Dev_config::Status current_status = _device_config->status();
01274
01275     // handle reset
01276     if (!status.raw)
01277     {
01278         _device_config->set_status(status);
01279         return;
01280     }
01281
01282     // Do no further processing in case of driver or device failure. If FAILED
01283     // or DEVICE_NEEDS_RESET are set only these fail_state bits will be set in
01284     // addition to the current status bits already set.
01285     if (current_status.fail_state() || status.fail_state())
01286     {
01287         if (current_status.fail_state() != status.fail_state())
01288         {
01289             current_status.fail_state() =
01290                 current_status.fail_state() | status.fail_state();
01291             _device_config->set_status(current_status);
01292         }
01293         return;
01294     }
01295
01296     // Enforce init sequence ACKNOWLEDGE, DRIVER, FEATURES_OK, DRIVER_OK.
01297     // We do not enforce that only one additional new bit is set per call.
01298     if ((!status.acked() && status.driver())
01299         || (!status.driver() && status.features_ok())
01300         || (!status.features_ok() && status.driver_ok()))
01301     {
01302         current_status.device_needs_reset() = 1;
01303         _device_config->set_status(current_status);
01304         return;

```

```

01305     }
01306
01307     // only check feature compatibility before DRIVER_OK is set
01308     if (status.features_ok() && !status.driver_ok()
01309         && !check_features_internal())
01310         status.features_ok() = 0;
01311
01312     // Note that if FEATURES_OK and DRIVER_OK are both updated to being set
01313     // at the same time the above check_features_internal() is skipped; this is
01314     // considered undefined behaviour but it is not prevented.
01315     if (status.running() && !check_queues())
01316     {
01317         current_status.device_needs_reset() = 1;
01318         _device_config->set_status(current_status);
01319         return;
01320     }
01321
01322     _device_config->set_status(status);
01323 }
01324
01337 int _set_status(unsigned new_status)
01338 {
01339     if (new_status == 0)
01340     {
01341         L4Re::Util::Dbg().printf("Resetting device\n");
01342         reset();
01343         _device_config->reset_hdr(true);
01344     }
01345
01346     Dev_config::Status status(new_status);
01347     check_and_update_status(status);
01348
01349     return 0;
01350 }
01351
01352 };
01353
01354 typedef Device_t<No_custom_data> Device;
01355
01356 } // namespace Svr
01357
01358 }
01359

```

16.223 virtio

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
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;
00121         Request() = default;
00122     private:
00123         friend class Virtqueue;
00124         Request(Virtqueue *r, unsigned i) : Head_desc(r, i), ring(r) {}
00125     };
00126
00127
00137     Request next_avail()
00138     {
00139         if (L4_LIKELY(_current_avail != _avail->idx))
00140         {
00141             rmb();
00142             unsigned head = _current_avail & _idx_mask;
00143             ++_current_avail;
00144             return Request(this, _avail->ring[head]);
00145         }
00146         return Request();
00147     }
00148
00154     bool desc_avail() const
00155     {
00156         return _current_avail != _avail->idx;
00157     }
00158
00166     void consumed(Head_desc const &r, l4_uint32_t len = 0)
00167     {
00168         l4_uint16_t i = _used->idx & _idx_mask;
00169         _used->ring[i] = Used_elem(r._d - _desc, len);
00170         wmb();
00171         ++_used->idx;
00172     }
00173
00174     template<typename ITER>
00175     void consumed(ITER const &begin, ITER const &end)
00176     {
00177         l4_uint16_t added = 0;
00178         l4_uint16_t idx = _used->idx;
00179
00180         for (auto elem = begin ; elem != end; ++elem, ++added)
00181             _used->ring[(idx + added) & _idx_mask]
00182                 = Used_elem(elem->first._d - _desc, elem->second);

```

```

00183
00184     wmb();
00185     _used->idx += added;
00186 }
00187
00188 template<typename QUEUE_OBSERVER>
00189 void finish(Head_desc &d, QUEUE_OBSERVER *o, l4_uint32_t len = 0)
00190 {
00191     consumed(d, len);
00192     o->notify_queue(this);
00193     d._d = 0;
00194 }
00195
00196 template<typename ITER, typename QUEUE_OBSERVER>
00197 void finish(ITER const &begin, ITER const &end, QUEUE_OBSERVER *o)
00198 {
00199     consumed(begin, end);
00200     o->notify_queue(this);
00201 }
00202
00208 void disable_notify()
00209 {
00210     if (L4_LIKELY(ready()))
00211         _used->flags.no_notify() = 1;
00212 }
00213
00219 void enable_notify()
00220 {
00221     if (L4_LIKELY(ready()))
00222         _used->flags.no_notify() = 0;
00223 }
00224
00231 Desc const *desc(unsigned idx) const
00232 { return _desc + idx; }
00233 };
00234 };
00235
00239 struct Data_buffer
00240 {
00241     char *pos;
00242     l4_uint32_t left;
00243
00244     Data_buffer() = default;
00245
00254     template<typename T>
00255     explicit Data_buffer(T *p)
00256     : pos(reinterpret_cast<char *>(p)), left(sizeof(T))
00257     {}
00258
00267     template<typename T>
00268     void set(T *p)
00269     {
00270         pos = reinterpret_cast<char *>(p);
00271         left = sizeof(T);
00272     }
00273
00284     l4_uint32_t copy_to(Data_buffer *dst, l4_uint32_t max = UINT_MAX)
00285     {
00286         unsigned long bytes = cxx::min(cxx::min(left, dst->left), max);
00287         memcpy(dst->pos, pos, bytes);
00288         left -= bytes;
00289         pos += bytes;
00290         dst->left -= bytes;
00291         dst->pos += bytes;
00292         return bytes;
00293     }
00294
00304     l4_uint32_t skip(l4_uint32_t bytes)
00305     {
00306         unsigned long b = cxx::min(left, bytes);
00307         left -= b;
00308         pos += b;
00309         return b;
00310     }
00311
00316     bool done() const
00317     { return left == 0; }
00318 };
00319
00320 class Request_processor;
00321
00325 struct Bad_descriptor
00326 {
00328     enum Error
00329     {
00330         Bad_address,
00331         Bad_rights,

```

```

00332     Bad_flags,
00333     Bad_next,
00334     Bad_size
00335 };
00336
00337 Request_processor const *proc;
00338
00339 // The error code
00340 Error error;
00341
00342 Bad_descriptor(Request_processor const *proc, Error e)
00343 : proc(proc), error(e)
00344 {}
00345
00346 char const *message() const
00347 {
00348     static char const *const err[] =
00349     {
00350         [Bad_address] = "Descriptor address cannot be translated",
00351         [Bad_rights]   = "Insufficient memory access rights",
00352         [Bad_flags]    = "Invalid descriptor flags",
00353         [Bad_next]     = "The descriptor's `next` index is invalid",
00354         [Bad_size]     = "Invalid size of the memory block"
00355     };
00356
00357     if (error >= (sizeof(err) / sizeof(err[0])) || !err[error])
00358         return "Unknown error";
00359
00360     return err[error];
00361 }
00362 };
00363
00364 class Request_processor
00365 {
00366 private:
00367     Virtqueue::Desc const *_table;
00368
00369     Virtqueue::Desc _current;
00370
00371     l4_uint16_t _num;
00372
00373 public:
00374     template<typename DESC_MAN, typename ...ARGS>
00375     void start(DESC_MAN *dm, Virtqueue *ring, Virtqueue::Head_desc const &request, ARGS... args)
00376     {
00377         _current = cxx::access_once(request.desc());
00378
00379         if (_current.flags.indirect())
00380         {
00381             dm->load_desc(_current, this, &_table);
00382             _num = _current.len / sizeof(Virtqueue::Desc);
00383             if (L4_UNLIKELY(!_num))
00384                 throw Bad_descriptor(this, Bad_descriptor::Bad_size);
00385
00386             _current = cxx::access_once(_table);
00387         }
00388         else
00389         {
00390             _table = ring->desc(0);
00391             _num = ring->num();
00392         }
00393
00394         dm->load_desc(_current, this, cxx::forward<ARGS>(args)...);
00395     }
00396
00397     template<typename DESC_MAN, typename ...ARGS>
00398     Virtqueue::Request const &start(DESC_MAN *dm, Virtqueue::Request const &request, ARGS... args)
00399     {
00400         start(dm, request.ring, request, cxx::forward<ARGS>(args)...);
00401         return request;
00402     }
00403
00404     Virtqueue::Desc::Flags current_flags() const
00405     { return _current.flags; }
00406
00407     bool has_more() const
00408     { return _current.flags.next(); }
00409
00410     template<typename DESC_MAN, typename ...ARGS>
00411     bool next(DESC_MAN *dm, ARGS... args)
00412     {
00413         if (!_current.flags.next())
00414             return false;
00415
00416         if (L4_UNLIKELY(_current.next >= _num))
00417             throw Bad_descriptor(this, Bad_descriptor::Bad_next);
00418     }

```

```

00500
00501     _current = cxx::access_once(_table + _current.next);
00502
00503     if (0) // we ignore this for performance reasons
00504         if (L4_UNLIKELY(_current.flags.indirect()))
00505             throw Bad_descriptor(this, Bad_descriptor::Bad_flags);
00506
00507     // must throw an exception in case of a bad descriptor
00508     dm->load_desc(_current, this, cxx::forward<ARGS>(args)...);
00509     return true;
00510 }
00511 };
00512
00513 }
00514 }

```

16.224 virtio-block

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00003 /*
00004  * Copyright (C) 2015-2020, 2022 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *           Manuel von Oltersdorff-Kalettkka <manuel.kalettkka@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     {
00059         friend Block_device;
00060         l4_uint16_t head;
00061
00062         explicit Handle(l4_uint16_t descno) : head(descno) {}
00063
00064     public:
00065         Handle() : head(Virtqueue::Eq) {}
00066         bool valid() const { return head != Virtqueue::Eq; }
00067     };
00091 void setup_device(L4::Cap<L4virtio::Device> srvcap, l4_size_t usermem,
00092                 void **userdata, Ptr<void> &user_devaddr,
00093                 L4::Cap<L4Re::Dataspace> qds = L4::Cap<L4Re::Dataspace>(),
00094                 l4_uint32_t fmask0 = -1U, l4_uint32_t fmask1 = -1U)

```

```

00095 {
00096     // Contact device.
00097     driver_connect(srvcap);
00098
00099     if (_config->device != L4VIRTIO_ID_BLOCK)
00100         L4Re::chksys(-L4_ENODEV, "Device is not a block device.");
00101
00102     if (_config->num_queues != 1)
00103         L4Re::chksys(-L4_EINVAL, "Invalid number of queues reported.");
00104
00105     // Memory is shared in one large dataspace which contains queues,
00106     // space for header/status and additional user-defined memory.
00107     unsigned queuesz = max_queue_size(0);
00108     l4_size_t totalsz = l4_round_page(usermem);
00109
00110     l4_uint64_t const header_offset =
00111         l4_round_size(_queue.total_size(queuesz),
00112             l4util_bsr(alignedof(l4virtio_block_header_t)));
00113     l4_uint64_t const status_offset = header_offset + queuesz * Header_size;
00114     l4_uint64_t const usermem_offset = l4_round_page(status_offset + queuesz);
00115
00116     // reserve space for one header/status per descriptor
00117     // TODO Should be reduced to 1/3 but this way no freelist is needed.
00118     totalsz += usermem_offset;
00119
00120     auto *e = L4Re::Env::env();
00121     if (!qds.is_valid())
00122     {
00123         _ds = L4Re::chkcap(L4Re::Util::make_unique_cap<L4Re::Dataspace>(),
00124             "Allocate queue dataspace capability");
00125         L4Re::chksys(e->mem_alloc()->alloc(totalsz, _ds.get(),
00126             L4Re::Mem_alloc::Continuous
00127             | L4Re::Mem_alloc::Pinned),
00128             "Allocate memory for virtio structures");
00129         _queue_ds = _ds.get();
00130     }
00131     else
00132     {
00133         if (qds->size() < totalsz)
00134             L4Re::chksys(-L4_EINVAL, "External queue dataspace too small.");
00135         _queue_ds = qds;
00136     }
00137
00138     // Now sort out which region goes where in the dataspace.
00139     L4Re::chksys(e->rm()->attach(&_queue_region, totalsz,
00140         L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00141         L4::Ipc::make_cap_rw(_queue_ds), 0,
00142         L4_PAGESHIFT),
00143         "Attach dataspace for virtio structures");
00144
00145     l4_uint64_t devaddr;
00146     L4Re::chksys(register_ds(_queue_ds, 0, totalsz, &devaddr),
00147         "Register queue dataspace with device");
00148
00149     _queue.init_queue(queuesz, _queue_region.get());
00150
00151     config_queue(0, queuesz, devaddr, devaddr + _queue.avail_offset(),
00152         devaddr + _queue.used_offset());
00153
00154     _header_addr = devaddr + header_offset;
00155     _headers = reinterpret_cast<l4virtio_block_header_t *>(_queue_region.get()
00156         + header_offset);
00157
00158     _status_addr = devaddr + status_offset;
00159     _status = _queue_region.get() + status_offset;
00160
00161     user_devaddr = Ptr<void>(devaddr + usermem_offset);
00162     if (userdata)
00163         *userdata = _queue_region.get() + usermem_offset;
00164
00165     // setup the callback mechanism
00166     _pending.assign(queuesz, Request());
00167
00168     // Finish handshake with device.
00169     _config->driver_features_map[0] = fmask0;
00170     _config->driver_features_map[1] = fmask1;
00171     driver_acknowledge();
00172 }
00173
00177 l4virtio_block_config_t const &device_config() const
00178 {
00179     return *_config->device_config<l4virtio_block_config_t>();
00180 }
00181
00190 Handle start_request(l4_uint64_t sector, l4_uint32_t type,
00191     Callback callback)
00192 {

```



```

00193     l4_uint16_t descno = _queue.alloc_descriptor();
00194     if (descno == Virtqueue::Eoq)
00195         return Handle(Virtqueue::Eoq);
00196
00197     L4virtio::Virtqueue::Desc &desc = _queue.desc(descno);
00198     Request &req = _pending[descno];
00199
00200     // setup the header
00201     l4virtio_block_header_t &head = _headers[descno];
00202     head.type = type;
00203     head.ioprio = 0;
00204     head.sector = sector;
00205
00206     // and put it in the descriptor
00207     desc.addr = Ptr<void>(_header_addr + descno * Header_size);
00208     desc.len = Header_size;
00209     desc.flags.raw = 0; // no write, no indirect
00210
00211     req.tail = descno;
00212     req.callback = callback;
00213
00214     return Handle(descno);
00215 }
00216
00228 int add_block(Handle handle, Ptr<void> addr, l4_uint32_t size)
00229 {
00230     l4_uint16_t descno = _queue.alloc_descriptor();
00231     if (descno == Virtqueue::Eoq)
00232         return -L4_EAGAIN;
00233
00234     Request &req = _pending[handle.head];
00235     L4virtio::Virtqueue::Desc &desc = _queue.desc(descno);
00236     L4virtio::Virtqueue::Desc &prev = _queue.desc(req.tail);
00237
00238     prev.next = descno;
00239     prev.flags.next() = true;
00240
00241     desc.addr = addr;
00242     desc.len = size;
00243     desc.flags.raw = 0;
00244     if (_headers[handle.head].type > 0) // write or flush request
00245         desc.flags.write() = true;
00246
00247     req.tail = descno;
00248
00249     return L4_EOK;
00250 }
00251
00264 int send_request(Handle handle)
00265 {
00266     // add the status bit
00267     auto descno = _queue.alloc_descriptor();
00268     if (descno == Virtqueue::Eoq)
00269         return -L4_EAGAIN;
00270
00271     Request &req = _pending[handle.head];
00272     L4virtio::Virtqueue::Desc &desc = _queue.desc(descno);
00273     L4virtio::Virtqueue::Desc &prev = _queue.desc(req.tail);
00274
00275     prev.next = descno;
00276     prev.flags.next() = true;
00277
00278     desc.addr = Ptr<void>(_status_addr + descno);
00279     desc.len = 1;
00280     desc.flags.raw = 0;
00281     desc.flags.write() = true;
00282
00283     req.tail = descno;
00284
00285     send(_queue, handle.head);
00286
00287     return L4_EOK;
00288 }
00289
00299 int process_request(Handle handle)
00300 {
00301     // add the status bit
00302     auto descno = _queue.alloc_descriptor();
00303     if (descno == Virtqueue::Eoq)
00304         return descno;
00305
00306     L4virtio::Virtqueue::Desc &desc = _queue.desc(descno);
00307     L4virtio::Virtqueue::Desc &prev = _queue.desc(_pending[handle.head].tail);
00308
00309     prev.next = descno;
00310     prev.flags.next() = true;
00311

```

```

00312     desc.addr = Ptr<void>(_status_addr + descno);
00313     desc.len = 1;
00314     desc.flags.raw = 0;
00315     desc.flags.write() = true;
00316
00317     _pending[handle.head].tail = descno;
00318
00319     int ret = send_and_wait(_queue, handle.head);
00320     unsigned char status = _status[descno];
00321     free_request(handle);
00322
00323     if (ret < 0)
00324         return ret;
00325
00326     switch (status)
00327     {
00328     case L4VIRTIO_BLOCK_S_OK: return L4_EOK;
00329     case L4VIRTIO_BLOCK_S_IOERR: return -L4_EIO;
00330     case L4VIRTIO_BLOCK_S_UNSUPP: return -L4_ENOSYS;
00331     }
00332
00333     return -L4_EINVAL;
00334 }
00335
00336 void free_request(Handle handle)
00337 {
00338     if (handle.head != Virtqueue::Eoq
00339         && _pending[handle.head].tail != Virtqueue::Eoq)
00340         _queue.free_descriptor(handle.head, _pending[handle.head].tail);
00341     _pending[handle.head].tail = Virtqueue::Eoq;
00342 }
00343
00350 void process_used_queue()
00351 {
00352     for (l4_uint16_t descno = _queue.find_next_used();
00353          descno != Virtqueue::Eoq;
00354          descno = _queue.find_next_used()
00355          )
00356     {
00357         if (descno >= _queue.num() || _pending[descno].tail == Virtqueue::Eoq)
00358             L4Re::chksys(-L4_ENOSYS, "Bad descriptor number");
00359
00360         unsigned char status = _status[descno];
00361         free_request(Handle(descno));
00362
00363         if (_pending[descno].callback)
00364             _pending[descno].callback(status);
00365     }
00366 }
00367
00368 protected:
00369     L4Re::Util::Unique_cap<L4Re::Dataspace> _ds;
00370     L4::Cap<L4Re::Dataspace> _queue_ds;
00371
00372 private:
00373     L4Re::Rm::Unique_region<unsigned char *> _queue_region;
00374     l4virtio_block_header_t *_headers;
00375     unsigned char *_status;
00376     l4_uint64_t _header_addr;
00377     l4_uint64_t _status_addr;
00378     Virtqueue _queue;
00379     std::vector<Request> _pending;
00380 };
00381
00382 } }

```

16.225 virtio-block

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00003 /*
00004  * Copyright (C) 2017-2021 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *
00007  */
00008 #pragma once
00009
00010 #include <l4/cxx/unique_ptr>
00011 #include <l4/re/util/unique_cap>
00012
00013 #include <climits>
00014
00015 #include <l4/l4virtio/virtio.h>

```

```

00016 #include <l4/l4virtio/virtio_block.h>
00017 #include <l4/l4virtio/server/l4virtio>
00018 #include <l4/sys/cxx/ipc_epifance>
00019
00020 namespace L4virtio { namespace Svr {
00021
00022 template <typename Ds_data> class Block_dev_base;
00023
00027 template<typename Ds_data>
00028 class Block_request
00029 {
00030     friend class Block_dev_base<Ds_data>;
00031     enum { Header_size = sizeof(l4virtio_block_header_t) };
00032
00033 public:
00034     struct Data_block
00035     {
00037         Driver_mem_region_t<Ds_data> *mem;
00039         void *addr;
00041         l4_uint32_t len;
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
00207 Driver_mem_list_t<Ds_data> *_mem_list;
00209 l4virtio_block_header_t _header;
00211 Request_processor _rp;
00213 Data_block _data;
00214
00216 Virtqueue::Request _request;
00218 unsigned _todo_blocks;
00220 l4_uint32_t _max_block_size;
00221 };
00222
00223 struct Block_features : public Dev_config::Features
00224 {
00225     Block_features() = default;
00226     Block_features(l4_uint32_t raw) : Dev_config::Features(raw) {}
00227
00229     CXX_BITFIELD_MEMBER( 1, 1, size_max, raw);

```

```

00231 CXX_BITFIELD_MEMBER( 2, 2, seg_max, raw);
00233 CXX_BITFIELD_MEMBER( 4, 4, geometry, raw);
00235 CXX_BITFIELD_MEMBER( 5, 5, ro, raw);
00237 CXX_BITFIELD_MEMBER( 6, 6, blk_size, raw);
00239 CXX_BITFIELD_MEMBER( 9, 9, flush, raw);
00241 CXX_BITFIELD_MEMBER(10, 10, topology, raw);
00243 CXX_BITFIELD_MEMBER(11, 11, config_wce, raw);
00245 CXX_BITFIELD_MEMBER(13, 13, discard, raw);
00247 CXX_BITFIELD_MEMBER(14, 14, write_zeroes, raw);
00248 };
00249
00250
00256 template <typename Ds_data>
00257 class Block_dev_base : public L4virtio::Svr::Device_t<Ds_data>
00258 {
00259 private:
00260     class Irq_object : public L4::Irqep_t<Irq_object>
00261     {
00262     public:
00263         Irq_object(Block_dev_base<Ds_data> *parent) : _parent(parent) {}
00264
00265         void handle_irq()
00266         {
00267             _parent->kick();
00268         }
00269
00270     private:
00271         Block_dev_base<Ds_data> *_parent;
00272     };
00273     Irq_object _irq_handler;
00274
00275 protected:
00276     L4::Epiface *irq_iface()
00277     { return &_irq_handler; }
00278
00279 private:
00280     L4Re::Util::Unique_cap<L4::Irq> _kick_guest_irq;
00281     Virtqueue _queue;
00282     unsigned _vq_max;
00283     l4_uint32_t _max_block_size = UINT_MAX;
00284     Dev_config_t<l4virtio_block_config_t> _dev_config;
00285
00286 public:
00287     typedef Block_request<Ds_data> Request;
00288
00289 protected:
00290     Block_features negotiated_features() const
00291     { return _dev_config.negotiated_features(0); }
00292
00293     Block_features device_features() const
00294     { return _dev_config.host_features(0); }
00295
00296     void set_device_features(Block_features df)
00297     { _dev_config.host_features(0) = df.raw; }
00298
00308     void set_size_max(l4_uint32_t sz)
00309     {
00310         _dev_config.priv_config()->size_max = sz;
00311         Block_features df = device_features();
00312         df.size_max() = true;
00313         set_device_features(df);
00314
00315         _max_block_size = sz;
00316     }
00317
00322     void set_seg_max(l4_uint32_t sz)
00323     {
00324         _dev_config.priv_config()->seg_max = sz;
00325         Block_features df = device_features();
00326         df.seg_max() = true;
00327         set_device_features(df);
00328     }
00329
00333     void set_geometry(l4_uint16_t cylinders, l4_uint8_t heads, l4_uint8_t sectors)
00334     {
00335         l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00336         pc->geometry.cylinders = cylinders;
00337         pc->geometry.heads = heads;
00338         pc->geometry.sectors = sectors;
00339         Block_features df = device_features();
00340         df.geometry() = true;
00341         set_device_features(df);
00342     }
00343
00350     void set_blk_size(l4_uint32_t sz)
00351     {
00352         _dev_config.priv_config()->blk_size = sz;

```

```

00353     Block_features df = device_features();
00354     df.blk_size() = true;
00355     set_device_features(df);
00356 }
00357
00366 void set_topology(l4_uint8_t physical_block_exp,
00367                  l4_uint8_t alignment_offset,
00368                  l4_uint32_t min_io_size,
00369                  l4_uint32_t opt_io_size)
00370 {
00371     l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00372     pc->topology.physical_block_exp = physical_block_exp;
00373     pc->topology.alignment_offset = alignment_offset;
00374     pc->topology.min_io_size = min_io_size;
00375     pc->topology.opt_io_size = opt_io_size;
00376     Block_features df = device_features();
00377     df.topology() = true;
00378     set_device_features(df);
00379 }
00380
00382 void set_flush()
00383 {
00384     Block_features df = device_features();
00385     df.flush() = true;
00386     set_device_features(df);
00387 }
00388
00393 void set_config_wce(l4_uint8_t writeback)
00394 {
00395     l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00396     pc->writeback = writeback;
00397     Block_features df = device_features();
00398     df.config_wce() = true;
00399     set_device_features(df);
00400 }
00401
00406 l4_uint8_t get_writeback()
00407 {
00408     l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00409     return pc->writeback;
00410 }
00411
00420 void set_discard(l4_uint32_t max_discard_sectors, l4_uint32_t max_discard_seg,
00421                 l4_uint32_t discard_sector_alignment)
00422 {
00423     l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00424     pc->max_discard_sectors = max_discard_sectors;
00425     pc->max_discard_seg = max_discard_seg;
00426     pc->discard_sector_alignment = discard_sector_alignment;
00427     Block_features df = device_features();
00428     df.discard() = true;
00429     set_device_features(df);
00430 }
00431
00440 void set_write_zeroes(l4_uint32_t max_write_zeroes_sectors,
00441                      l4_uint32_t max_write_zeroes_seg,
00442                      l4_uint8_t write_zeroes_may_unmap)
00443 {
00444     l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00445     pc->max_write_zeroes_sectors = max_write_zeroes_sectors;
00446     pc->max_write_zeroes_seg = max_write_zeroes_seg;
00447     pc->write_zeroes_may_unmap = write_zeroes_may_unmap;
00448     Block_features df = device_features();
00449     df.write_zeroes() = true;
00450     set_device_features(df);
00451 }
00452
00453 public:
00462 Block_dev_base(l4_uint32_t vendor, unsigned queue_size, l4_uint64_t capacity,
00463               bool read_only)
00464 : L4virtio::Svr::Device_t<Ds_data>(&_dev_config), _irq_handler(this),
00465   _vq_max(queue_size), _dev_config(vendor, L4VIRTIO_ID_BLOCK, 1)
00466 {
00467     this->reset_queue_config(0, queue_size);
00468
00469     Block_features df(0);
00470     df.ring_indirect_desc() = true;
00471     df.ro() = read_only;
00472     set_device_features(df);
00473
00474     _dev_config.priv_config()->capacity = capacity;
00475 }
00476
00477
00492 virtual bool process_request(cxx::unique_ptr<Request> &&req) = 0;
00493
00497 virtual void reset_device() = 0;

```

```

00498
00502 virtual bool queue_stopped() = 0;
00503
00515 void finalize_request(cxx::unique_ptr<Request> req, unsigned sz,
00516                      l4_uint8_t status = L4VIRTIO_BLOCK_S_OK)
00517 {
00518     if (_dev_config.status().fail_state() || !_queue.ready())
00519         return;
00520
00521     if (req->release_request(&_queue, status, sz) < 0)
00522         this->device_error();
00523
00524     // XXX not implemented
00525     // _dev_config->irq_status |= 1;
00526     _kick_guest_irq->trigger();
00527
00528     // Request can be dropped here.
00529 }
00530
00531 int reconfig_queue(unsigned idx) override
00532 {
00533     if (idx == 0 && this->setup_queue(&_queue, 0, _vq_max))
00534         return 0;
00535
00536     return -L4_EINVAL;
00537 }
00538
00539 void reset() override
00540 {
00541     _queue.disable();
00542     _dev_config.reset_queue(0, _vq_max);
00543     _dev_config.reset_hdr();
00544     reset_device();
00545 }
00546
00547 protected:
00548 void kick()
00549 {
00550     if (!_queue.ready() || queue_stopped())
00551         return;
00552
00553     while (!_dev_config.status().fail_state())
00554     {
00555         auto r = _queue.next_avail();
00556         if (!r)
00557             return;
00558
00559         try
00560         {
00561             cxx::unique_ptr<Request>
00562                 cur{new Request(r, &(this->mem_info), _vq_max, _max_block_size)};
00563
00564             if (!process_request(cxx::move(cur)))
00565                 return;
00566         }
00567         catch (Bad_descriptor const &e)
00568         {
00569             this->device_error();
00570             return;
00571         }
00572     }
00573 }
00574
00575 private:
00576 L4::Cap<L4::Irq> device_notify_irq() const override
00577 {
00578     return L4::cap_cast<L4::Irq>(_irq_handler.obj_cap());
00579 }
00580
00581 void register_single_driver_irq() override
00582 {
00583     _kick_guest_irq = L4Re::Util::Unique_cap<L4::Irq>(
00584         L4Re::chkcapi(this->server_iface()->template rcv_cap<L4::Irq>(0)));
00585
00586     L4Re::chksys(this->server_iface()->realloc_rcv_cap(0));
00587 }
00588
00589 void trigger_driver_config_irq() const override
00590 {
00591     _kick_guest_irq->trigger();
00592 }
00593
00594 bool check_queues() override
00595 {
00596     if (!_queue.ready())
00597     {
00598         reset();
00599     }
00598

```

```

00599         return false;
00600     }
00601
00602     return true;
00603 }
00604 };
00605
00606 template <typename Ds_data>
00607 struct Block_dev
00608 : Block_dev_base<Ds_data>,
00609   L4::Epiface_t<Block_dev<Ds_data>, L4virtio::Device>
00610 {
00611     using Block_dev_base<Ds_data>::Block_dev_base;
00612
00623     L4::Cap<void> register_obj(L4::Registry_iface *registry,
00624                               char const *service = 0)
00625     {
00626         L4Re::chkcap(registry->register_irq_obj(this->irq_iface()));
00627         L4::Cap<void> ret;
00628         if (service)
00629             ret = registry->register_obj(this, service);
00630         else
00631             ret = registry->register_obj(this);
00632         L4Re::chkcap(ret);
00633
00634         return ret;
00635     }
00636
00637     L4::Cap<void> register_obj(L4::Registry_iface *registry,
00638                               L4::Cap<L4::Rcv_endpoint> ep)
00639     {
00640         L4Re::chkcap(registry->register_irq_obj(this->irq_iface()));
00641
00642         return L4Re::chkcap(registry->register_obj(this, ep));
00643     }
00644
00645 protected:
00646     L4::Ipc_svr::Server_iface *server_iface() const override
00647     {
00648         return this->L4::Epiface::server_iface();
00649     }
00650 };
00651
00652 } }

```

16.226 virtio.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
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,
00078     L4VIRTIO_ID_SOCK         = 0x9999,
00079 };
00080
00082 enum L4virtio_device_status
00083 {
00084     L4VIRTIO_STATUS_ACKNOWLEDGE = 1,
00085     L4VIRTIO_STATUS_DRIVER      = 2,
00086     L4VIRTIO_STATUS_DRIVER_OK   = 4,
00087     L4VIRTIO_STATUS_FEATURES_OK = 8,
00088     L4VIRTIO_STATUS_DEVICE_NEEDS_RESET = 0x40,
00089     L4VIRTIO_STATUS_FAILED      = 0x80
00090 };
00091
00093 enum L4virtio_feature_bits
00094 {
00096     L4VIRTIO_FEATURE_VERSION_1 = 32,
00098     L4VIRTIO_FEATURE_CMD_CONFIG = 224
00099 };
00100
00105 enum L4_virtio_irq_status
00106 {
00107     L4VIRTIO_IRQ_STATUS_VRING = 1,
00108     L4VIRTIO_IRQ_STATUS_CONFIG = 2,
00109 };
00110
00114 enum L4_virtio_cmd
00115 {
00116     L4VIRTIO_CMD_NONE          = 0x00000000,
00117     L4VIRTIO_CMD_SET_STATUS    = 0x01000000,
00118     L4VIRTIO_CMD_CFG_QUEUE     = 0x02000000,
00119     L4VIRTIO_CMD_MASK          = 0xff000000,
00120 };
00121
00125 typedef struct l4virtio_config_hdr_t
00126 {
00127     l4_uint32_t magic;
00128     l4_uint32_t version;
00129     l4_uint32_t device;
00130     l4_uint32_t vendor;
00132     l4_uint32_t dev_features;
00133     l4_uint32_t dev_features_sel;
00134     l4_uint32_t _res1[2];
00135
00136     l4_uint32_t driver_features;
00137     l4_uint32_t driver_features_sel;
00138
00139     /* some L4virtio specific members ... */
00140     l4_uint32_t num_queues;
00141     l4_uint32_t queues_offset;
00143     /* must start at 0x30 (per virtio-mmio layout) */
00144     l4_uint32_t queue_sel;
00145     l4_uint32_t queue_num_max;
00146     l4_uint32_t queue_num;
00147     l4_uint32_t _res3[2];
00148     l4_uint32_t queue_ready;
00149     l4_uint32_t _res4[2];
00150
00151     l4_uint32_t queue_notify;
00152     l4_uint32_t _res5[3];
00153
00154     l4_uint32_t irq_status;
00155     l4_uint32_t irq_ack;
00156     l4_uint32_t _res6[2];
00157
00164     l4_uint32_t status;
00165     l4_uint32_t _res7[3];
00166
00167     l4_uint64_t queue_desc;
00168     l4_uint32_t _res8[2];
00169     l4_uint64_t queue_avail;

```

```

00170     l4_uint32_t _res9[2];
00171     l4_uint64_t queue_used;
00172
00173     /* use the unused space here for device and driver feature bitmaps */
00174     l4_uint32_t dev_features_map[8];
00175     l4_uint32_t driver_features_map[8];
00176
00177     l4_uint32_t _res10[2];
00178
00180     l4_uint32_t cfg_driver_notify_index;
00182     l4_uint32_t cfg_device_notify_index;
00183
00185     l4_uint32_t cmd;
00186     l4_uint32_t generation;
00187 } l4virtio_config_hdr_t;
00188
00206 typedef struct l4virtio_config_queue_t
00207 {
00209     l4_uint16_t num_max;
00211     l4_uint16_t num;
00212
00214     l4_uint16_t ready;
00215
00217     l4_uint16_t driver_notify_index;
00218
00219     l4_uint64_t desc_addr;
00220     l4_uint64_t avail_addr;
00221     l4_uint64_t used_addr;
00224     l4_uint16_t device_notify_index;
00225 } l4virtio_config_queue_t;
00226
00227 EXTERN_C_BEGIN
00228
00234 L4_INLINE l4virtio_config_queue_t *
00235 l4virtio_config_queues(l4virtio_config_hdr_t const *cfg)
00236 {
00237     return (l4virtio_config_queue_t *)(((l4_addr_t)cfg) + cfg->queues_offset);
00238 }
00239
00245 L4_INLINE void *
00246 l4virtio_device_config(l4virtio_config_hdr_t const *cfg)
00247 {
00248     return (void *)(((l4_addr_t)cfg) + 0x100);
00249 }
00250
00254 L4_INLINE void
00255 l4virtio_set_feature(l4_uint32_t *feature_map, unsigned feat)
00256 {
00257     unsigned idx = feat / 32;
00258
00259     if (idx < 8)
00260         feature_map[idx] |= 1UL << (feat % 32);
00261 }
00262
00266 L4_INLINE void
00267 l4virtio_clear_feature(l4_uint32_t *feature_map, unsigned feat)
00268 {
00269     unsigned idx = feat / 32;
00270
00271     if (idx < 8)
00272         feature_map[idx] &= ~(1UL << (feat % 32));
00273 }
00274
00278 L4_INLINE unsigned
00279 l4virtio_get_feature(l4_uint32_t *feature_map, unsigned feat)
00280 {
00281     unsigned idx = feat / 32;
00282
00283     if (idx >= 8)
00284         return 0;
00285
00286     return feature_map[idx] & (1UL << (feat % 32));
00287 }
00288
00294 L4_CV int
00295 l4virtio_set_status(l4_cap_idx_t cap, unsigned status) L4_NOTHROW;
00296
00302 L4_CV int
00303 l4virtio_config_queue(l4_cap_idx_t cap, unsigned queue) L4_NOTHROW;
00304
00310 L4_CV int
00311 l4virtio_register_ds(l4_cap_idx_t cap, l4_cap_idx_t ds_cap,
00312                     l4_uint64_t base, l4_umword_t offset,
00313                     l4_umword_t size) L4_NOTHROW;
00314
00320 L4_CV int
00321 l4virtio_device_config_ds(l4_cap_idx_t cap, l4_cap_idx_t config_ds,

```

```

00322             l4_addr_t *ds_offset) L4_NOTHROW;
00323
00329 L4_CV int
00330 l4virtio_device_notification_irq(l4_cap_idx_t cap, unsigned index,
00331             l4_cap_idx_t irq) L4_NOTHROW;
00332
00333 EXTERN_C_END
00334

```

16.227 virtio_block.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  * 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
00018 #pragma once
00019
00026 #include <l4/sys/types.h>
00027
00031 enum L4virtio_block_operations
00032 {
00033     L4VIRTIO_BLOCK_T_IN      = 0,
00034     L4VIRTIO_BLOCK_T_OUT     = 1,
00035     L4VIRTIO_BLOCK_T_FLUSH   = 4,
00036     L4VIRTIO_BLOCK_T_GET_ID  = 8,
00037     L4VIRTIO_BLOCK_T_DISCARD = 11,
00038     L4VIRTIO_BLOCK_T_WRITE_ZEROES = 13,
00039 };
00040
00044 enum L4virtio_block_status
00045 {
00046     L4VIRTIO_BLOCK_S_OK      = 0,
00047     L4VIRTIO_BLOCK_S_IOERR   = 1,
00048     L4VIRTIO_BLOCK_S_UNSUPP  = 2
00049 };
00050
00054 typedef struct l4virtio_block_header_t
00055 {
00056     l4_uint32_t type;
00057     l4_uint32_t ioprio;
00058     l4_uint64_t sector;
00059 } l4virtio_block_header_t;
00060
00061 enum L4virtio_block_discard_flags_t
00062 {
00063     L4VIRTIO_BLOCK_DISCARD_F_UNMAP    = 0x00000001UL,
00064     L4VIRTIO_BLOCK_DISCARD_F_RESERVED = 0xFFFFFFFFEUL,
00065 };
00066
00070 typedef struct l4virtio_block_discard_t
00071 {
00072     l4_uint64_t sector;
00073     l4_uint32_t num_sectors;
00074     l4_uint32_t flags;
00075 } l4virtio_block_discard_t;
00076
00080 typedef struct l4virtio_block_config_t
00081 {
00082     l4_uint64_t capacity;
00083     l4_uint32_t size_max;
00084     l4_uint32_t seg_max;
00085     struct l4virtio_block_config_geometry_t
00086     {
00087         l4_uint16_t cylinders;
00088         l4_uint8_t heads;
00089         l4_uint8_t sectors;
00090     } geometry;
00091     l4_uint32_t blk_size;
00092     struct l4virtio_block_config_topology_t
00093     {

```

```

00095     l4_uint8_t physical_block_exp;
00097     l4_uint8_t alignment_offset;
00099     l4_uint16_t min_io_size;
00101     l4_uint32_t opt_io_size;
00102 } topology;
00103 l4_uint8_t writeback;
00104 l4_uint8_t unused0[3];
00105 l4_uint32_t max_discard_sectors;
00106 l4_uint32_t max_discard_seg;
00107 l4_uint32_t discard_sector_alignment;
00108 l4_uint32_t max_write_zeroes_sectors;
00109 l4_uint32_t max_write_zeroes_seg;
00110 l4_uint8_t write_zeroes_may_unmap;
00111 l4_uint8_t unused1[3];
00112 } l4virtio_block_config_t;
00113

```

16.228 virtio_input.h

```

00001 /*
00002  * Copyright (C) 2019, 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  * 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
00025 #include <l4/sys/types.h>
00026
00030 enum L4virtio_input_config_select
00031 {
00032     L4VIRTIO_INPUT_CFG_UNSET = 0,
00033     L4VIRTIO_INPUT_CFG_ID_NAME = 1,
00034     L4VIRTIO_INPUT_CFG_ID_SERIAL = 2,
00035     L4VIRTIO_INPUT_CFG_ID_DEVIDS = 3,
00036     L4VIRTIO_INPUT_CFG_PROP_BITS = 0x10,
00037     L4VIRTIO_INPUT_CFG_EV_BITS = 0x11,
00038     L4VIRTIO_INPUT_CFG_ABS_INFO = 0x12
00039 };
00040
00044 typedef struct l4virtio_input_absinfo_t
00045 {
00046     l4_uint32_t min;
00047     l4_uint32_t max;
00048     l4_uint32_t fuzz;
00049     l4_uint32_t flat;
00050     l4_uint32_t res;
00051 } l4virtio_absinfo_t;
00052
00056 typedef struct l4virtio_input_devids_t
00057 {
00058     l4_uint16_t bustype;
00059     l4_uint16_t vendor;
00060     l4_uint16_t product;
00061     l4_uint16_t version;
00062 } l4virtio_input_devids_t;
00063
00067 typedef struct l4virtio_input_config_t
00068 {
00069     l4_uint8_t select;
00070     l4_uint8_t subsl;
00071     l4_uint8_t size;
00072     l4_uint8_t reserved[5];
00073     union
00074     {
00075         char string[128];
00076         l4_uint8_t bitmap[128];
00077         struct l4virtio_input_absinfo_t abs;
00078         struct l4virtio_input_devids_t ids;
00079     } u;
00080 } l4virtio_input_config_t;
00081
00085 typedef struct l4virtio_input_event_t

```

```

00086 {
00087     l4_uint16_t type;
00088     l4_uint16_t code;
00089     l4_uint32_t value;
00090 } l4virtio_input_event_t;
00091

```

16.229 virtio_net.h

```

00001 /* SPDX-License-Identifier: GPL-2.0-only OR License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2022 Kernkonzept GmbH.
00004  * Author(s): Stephan Gerhold <stephan.gerhold@kernkonzept.com>
00005  */
00006
00007 #pragma once
00008
00015 #include <l4/sys/types.h>
00016
00020 typedef struct l4virtio_net_header_t
00021 {
00022     l4_uint8_t flags;
00023     l4_uint8_t gso_type;
00024     l4_uint16_t hdr_len;
00025     l4_uint16_t gso_size;
00026     l4_uint16_t csum_start;
00027     l4_uint16_t csum_offset;
00028     l4_uint16_t num_buffers;
00029 } l4virtio_net_header_t;
00030
00034 typedef struct l4virtio_net_config_t
00035 {
00036     l4_uint8_t mac[6];
00037     l4_uint16_t status;
00038     l4_uint16_t max_virtqueue_pairs;
00039     l4_uint16_t mtu;
00040     l4_uint32_t speed;
00041     l4_uint8_t duplex;
00042 } l4virtio_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

```

16.230 virtqueue

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <warg@os.inf.tu-dresden.de>
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/util/debug>
00020 #include <l4/sys/types.h>
00021 #include <l4/sys/err.h>
00022 #include <l4/cxx/bitfield>
00023 #include <l4/cxx/exceptions>
00024 #include <stdint>
00025
00026 #pragma once
00027
00028 namespace L4virtio {
00029
00030 // __ARM_ARCH_7A__ not defined by Clang
00031 #if defined(__ARM_ARCH_7A__) \
00032     || ( defined(__ARM_ARCH) && __ARM_ARCH == 7 \
00033         && defined(__ARM_ARCH_PROFILE) && __ARM_ARCH_PROFILE >= 'A')
00034 static inline void wmb() { asm volatile ("dmb" : : : "memory"); }
00035 static inline void rmb() { asm volatile ("dmb" : : : "memory"); }
00036 // __ARM_ARCH_8A__ not defined by Clang
00037 #elif defined(__ARM_ARCH_8A__) \
00038     || ( defined(__ARM_ARCH) && __ARM_ARCH == 8 \
00039         && defined(__ARM_ARCH_PROFILE) && __ARM_ARCH_PROFILE >= 'A') \
00040     || (defined(__ARM_ARCH) && __ARM_ARCH > 8)
00041 static inline void wmb() { asm volatile ("dsb ishst" : : : "memory"); }
00042 static inline void rmb() { asm volatile ("dsb ishld" : : : "memory"); }
00043 #elif defined(__mips__)
00044 static inline void wmb() { asm volatile ("sync" : : : "memory"); }
00045 static inline void rmb() { asm volatile ("sync" : : : "memory"); }
00046 #elif defined(__amd64__) || defined(__i386__) || defined(__i686__)
00047 static inline void wmb() { asm volatile ("sfence" : : : "memory"); }
00048 static inline void rmb() { asm volatile ("lfence" : : : "memory"); }
00049 #else
00050 #warning Missing proper memory write barrier
00051 static inline void wmb() { asm volatile ("": : : : "memory"); }
00052 static inline void rmb() { asm volatile ("": : : : "memory"); }
00053 #endif
00054
00055
00062 template< typename T >
00063 class Ptr
00064 {
00065 public:
00066     enum Invalid_type { Invalid };
00067
00068     Ptr() = default;
00069
00072     Ptr(Invalid_type) : _p(~0ULL) {}
00073
00075     explicit Ptr(l4_uint64_t vm_addr) : _p(vm_addr) {}
00076
00078     l4_uint64_t get() const { return _p; }
00079
00081     bool is_valid() const { return _p != ~0ULL; }
00082
00083 private:
00084     l4_uint64_t _p;
00085 };
00086
00087
00096 class Virtqueue
00097 {
00098 public:
00102     class Desc
00103     {
00104     public:
00108         struct Flags
00109         {
00110             l4_uint16_t raw;
00111             Flags() = default;
00112
00114             explicit Flags(l4_uint16_t v) : raw(v) {}
00115
00117             CXX_BITFIELD_MEMBER( 0, 0, next, raw);
00119             CXX_BITFIELD_MEMBER( 1, 1, write, raw);
00121             CXX_BITFIELD_MEMBER( 2, 2, indirect, raw);
00122         };
00123
00124         Ptr<void> addr;
00125         l4_uint32_t len;
00126         Flags flags;
00127         l4_uint16_t next;
00128

```

```

00132     void dump(unsigned idx) const
00133     {
00134         L4Re::Util::Dbg().printf("D[%04x]: %08llx (%x) f=%04x n=%04x\n",
00135                                 idx, addr.get(),
00136                                 len, (unsigned)flags.raw, (unsigned)next);
00137     }
00138 };
00139
00143 class Avail
00144 {
00145 public:
00149     struct Flags
00150     {
00151         l4_uint16_t raw;
00152         Flags() = default;
00153
00155         explicit Flags(l4_uint16_t v) : raw(v) {}
00156
00158         CXX_BITFIELD_MEMBER( 0, 0, no_irq, raw);
00159     };
00160
00161     Flags flags;
00162     l4_uint16_t idx;
00163     l4_uint16_t ring[];
00164 };
00165
00169 struct Used_elem
00170 {
00171     Used_elem() = default;
00172
00180     Used_elem(l4_uint16_t id, l4_uint32_t len) : id(id), len(len) {}
00181     l4_uint32_t id;
00182     l4_uint32_t len;
00183 };
00184
00188 class Used
00189 {
00190 public:
00194     struct Flags
00195     {
00196         l4_uint16_t raw;
00197         Flags() = default;
00198
00200         explicit Flags(l4_uint16_t v) : raw(v) {}
00201
00203         CXX_BITFIELD_MEMBER( 0, 0, no_notify, raw);
00204     };
00205
00206     Flags flags;
00207     l4_uint16_t idx;
00208     Used_elem ring[];
00209 };
00210
00211 protected:
00212     Desc *_desc;
00213     Avail *_avail;
00214     Used *_used;
00215
00217     l4_uint16_t _current_avail;
00218
00223     l4_uint16_t _idx_mask;
00224
00228     Virtqueue() : _desc(0), _idx_mask(0) {}
00229     Virtqueue(Virtqueue const &) = delete;
00230
00231 public:
00237     void disable()
00238     { _desc = 0; }
00239
00243     enum
00244     {
00245         Desc_align = 4, ///< Alignment of the descriptor table.
00246         Avail_align = 1, ///< Alignment of the available ring.
00247         Used_align = 2, ///< Alignment of the used ring.
00248     };
00249
00258     static unsigned long total_size(unsigned num)
00259     {
00260         static_assert(Desc_align >= Avail_align,
00261             "virtqueue alignment assumptions broken");
00262         return l4_round_size(desc_size(num) + avail_size(num), Used_align)
00263             + used_size(num);
00264     }
00265
00274     static unsigned long desc_size(unsigned num)
00275     { return num * 16; }
00276

```

```

00282 static unsigned long desc_align()
00283 { return Desc_align; }
00284
00292 static unsigned long avail_size(unsigned num)
00293 { return 2 * num + 6; }
00294
00300 static unsigned long avail_align()
00301 { return Avail_align; }
00302
00311 static unsigned long used_size(unsigned num)
00312 { return 8 * num + 6; }
00313
00319 static unsigned long used_align()
00320 { return Used_align; }
00321
00327 unsigned long total_size() const
00328 {
00329     return ((char *) _used - (char *) _desc)
00330         + used_size(num());
00331 }
00332
00336 unsigned long avail_offset() const
00337 { return (char const *) _avail - (char const *) _desc; }
00338
00342 unsigned long used_offset() const
00343 { return (char const *) _used - (char const *) _desc; }
00344
00362 void setup(unsigned num, void *desc, void *avail, void *used)
00363 {
00364     if (num > 0x10000)
00365         throw L4::Runtime_error(-L4_EINVAL, "Queue too large.");
00366
00367     _idx_mask = num - 1;
00368     _desc = (Desc*)desc;
00369     _avail = (Avail*)avail;
00370     _used = (Used*)used;
00371
00372     _current_avail = 0;
00373
00374     L4Re::Util::Dbg().printf("VQ[%p]: num=%d d:%p a:%p u:%p\n",
00375                             this, num, _desc, _avail, _used);
00376 }
00377
00391 void setup_simple(unsigned num, void *ring)
00392 {
00393     l4_addr_t desc = reinterpret_cast<l4_addr_t>(ring);
00394     l4_addr_t avail = l4_round_size(desc + desc_size(num), Avail_align);
00395     l4_addr_t used = l4_round_size(avail + avail_size(num), Used_align);
00396     setup(num, (void *)desc, (void *)avail, (void *)used);
00397 }
00398
00404 void dump(Desc const *d) const
00405 { d->dump(d - _desc); }
00406
00412 bool ready() const
00413 { return L4_LIKELY(_desc != 0); }
00414
00416 unsigned num() const
00417 { return _idx_mask + 1; }
00418
00426 bool no_notify_guest() const
00427 {
00428     return _avail->flags.no_irq();
00429 }
00430
00438 bool no_notify_host() const
00439 {
00440     return _used->flags.no_notify();
00441 }
00442
00448 void no_notify_host(bool value)
00449 {
00450     _used->flags.no_notify() = value;
00451 }
00452
00461 l4_uint16_t get_avail_idx() const { return _avail->idx; }
00462
00468 l4_uint16_t get_tail_avail_idx() const { return _current_avail; }
00469
00470 };
00471
00472 namespace Driver {
00473
00482 class Virtqueue : public L4virtio::Virtqueue
00483 {
00484 private:
00486     l4_uint16_t _next_free;

```



```

00487
00488 public:
00489     enum End_of_queue
00490     {
00491         // Indicates the end of the queue.
00492         Eoq = 0xFFFF
00493     };
00494
00495     Virtqueue() : _next_free(Eoq) {}
00496
00506     void initialize_rings(unsigned num)
00507     {
00508         _used->idx = 0;
00509         _avail->idx = 0;
00510
00511         // setup the freelist
00512         for (l4_uint16_t d = 0; d < num - 1; ++d)
00513             _desc[d].next = d + 1;
00514         _desc[num - 1].next = Eoq;
00515         _next_free = 0;
00516     }
00517
00534     void init_queue(unsigned num, void *desc, void *avail, void *used)
00535     {
00536         setup(num, desc, avail, used);
00537         initialize_rings(num);
00538     }
00539
00549     void init_queue(unsigned num, void *base)
00550     {
00551         setup_simple(num, base);
00552         initialize_rings(num);
00553     }
00554
00555
00570     l4_uint16_t alloc_descriptor()
00571     {
00572         l4_uint16_t idx = _next_free;
00573         if (idx == Eoq)
00574             return Eoq;
00575
00576         _next_free = _desc[idx].next;
00577
00578         return idx;
00579     }
00580
00586     void enqueue_descriptor(l4_uint16_t descno)
00587     {
00588         if (descno > _idx_mask)
00589             throw L4::Bounds_error();
00590
00591         _avail->ring[_avail->idx & _idx_mask] = descno; // _avail->idx expected to wrap
00592         wmb();
00593         ++_avail->idx;
00594     }
00595
00602     Desc &desc(l4_uint16_t descno)
00603     {
00604         if (descno > _idx_mask)
00605             throw L4::Bounds_error();
00606
00607         return _desc[descno];
00608     }
00609
00621     l4_uint16_t find_next_used(l4_uint32_t *len = nullptr)
00622     {
00623         if (_current_avail == _used->idx)
00624             return Eoq;
00625
00626         auto elem = _used->ring[_current_avail++ & _idx_mask];
00627
00628         if (len)
00629             *len = elem.len;
00630
00631         return elem.id;
00632     }
00633
00643     void free_descriptor(l4_uint16_t head, l4_uint16_t tail)
00644     {
00645         if (head > _idx_mask || tail > _idx_mask)
00646             throw L4::Bounds_error();
00647
00648         _desc[tail].next = _next_free;
00649         _next_free = head;
00650     }
00651 };
00652

```

```
00653 }
00654 } // namespace L4virtio
```

16.231 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
00030 namespace Block_device {
00031
00032 template <typename DEV>
00033 struct Simple_factory
00034 {
00035     using Device_type = DEV;
00036     using Client_type = Virtio_client<Device_type>;
00037
00038     static cxx::unique_ptr<Client_type>
00039     create_client(cxx::Ref_ptr<Device_type> const &dev,
00040                 unsigned numds, bool readonly)
00041     { return cxx::make_unique<Client_type>(dev, numds, readonly); }
00042 };
00043
00044 template <typename BASE_DEV>
00045 struct Partitionable_factory
00046 {
00047     using Device_type = BASE_DEV;
00048     using Client_type = Virtio_client<Device_type>;
00049
00050     static cxx::unique_ptr<Client_type>
00051     create_client(cxx::Ref_ptr<Device_type> const &dev,
00052                 unsigned numds, bool readonly)
00053     {
00054         return cxx::make_unique<Client_type>(dev, numds, readonly);
00055     }
00056
00057     static cxx::Ref_ptr<Device_type>
00058     create_partition(cxx::Ref_ptr<Device_type> const &dev, unsigned partition_id,
00059                    Partition_info const &pi)
00060     {
00061         return cxx::Ref_ptr<Device_type>{
00062             new Partitioned_device<Device_type>(dev, partition_id, pi)};
00063     }
00064 };
00065
00066
00067 template <typename DEV, typename FACTORY = Simple_factory<DEV>
00068 class Device_mgr
00069 {
00070     using Device_factory_type = FACTORY;
00071     using Client_type = typename Device_factory_type::Client_type;
00072     using Device_type = typename Device_factory_type::Device_type;
00073
00074     using Ds_vector = std::vector<L4::Cap<L4Re::Dataspace>>;
00075
00076     using Pairing_callback = std::function<void(Device_type *)>;
00077
00078     struct Pending_client
```

```

00090 {
00092     std::string device_id;
00094     L4::Cap<L4::Rcv_endpoint> gate;
00096     int num_ds;
00098     bool readonly;
00099
00100     bool enable_trusted_ds_validation;
00101
00102     std::shared_ptr<Ds_vector const> trusted_dataspaces;
00103
00105     Pairing_callback pairing_cb;
00106
00107     Pending_client() = default;
00108
00109     Pending_client(L4::Cap<L4::Rcv_endpoint> g, std::string const &dev, int ds,
00110                   bool ro, bool enable_trusted_ds_validation,
00111                   std::shared_ptr<Ds_vector const> trusted_dataspaces,
00112                   Pairing_callback cb)
00113     : device_id(dev), gate(g), num_ds(ds), readonly(ro),
00114       enable_trusted_ds_validation(enable_trusted_ds_validation),
00115       trusted_dataspaces(trusted_dataspaces), pairing_cb(cb)
00116     {}
00117 };
00118
00119 class Connection : public cxx::Ref_obj_list_item<Connection>
00120 {
00121 public:
00122     explicit Connection(cxx::Ref_ptr<Device_type> &&dev)
00123     : _shutdown_state(Shutdown_type::Running),
00124       _device(cxx::move(dev))
00125     {}
00126
00127     L4::Cap<void> cap() const
00128     { return _interface ? _interface->obj_cap() : L4::Cap<void>(); }
00129
00130     void start_disk_scan(Errand::Callback const &callback)
00131     {
00132         _device->start_device_scan(
00133             [=] ()
00134             {
00135                 scan_disk_partitions(callback, 0);
00136             });
00137     }
00138
00139     void unregister_interfaces(L4::Registry_iface *registry) const
00140     {
00141         if (_interface)
00142             registry->unregister_obj(_interface.get());
00143
00144         for (auto *sub : _subs)
00145             sub->unregister_interfaces(registry);
00146     }
00147
00148     int create_interface_for(Pending_client *c, L4::Registry_iface *registry)
00149     {
00150         if (_shutdown_state != Shutdown_type::Running)
00151             return -L4_EIO;
00152
00153         if (_interface)
00154             return contains_device(c->device_id) ? -L4_EBUSY : -L4_ENODEV;
00155
00156         // check for match in partitions
00157
00158         bool busy = false;
00159         for (auto *sub : _subs)
00160         {
00161             if (sub->_interface)
00162                 busy = true;
00163
00164             int ret = sub->create_interface_for(c, registry);
00165
00166             if (ret != -L4_ENODEV) // includes L4_EOK
00167                 return ret;
00168         }
00169
00170         if (!match_hid(c->device_id))
00171             return -L4_ENODEV;
00172
00173         if (busy)
00174             return -L4_EBUSY;
00175
00176         auto clt = Device_factory_type::create_client(_device, c->num_ds,
00177                                                       c->readonly);
00178
00179         clt->add_trusted_dataspaces(c->trusted_dataspaces);
00180         if (c->enable_trusted_ds_validation)
00181             clt->enable_trusted_ds_validation();

```

```

00182
00183     if (c->gate.is_valid())
00184     {
00185         if (!clt->register_obj(registry, c->gate).is_valid())
00186             return -L4_ENOMEM;
00187     }
00188     else
00189     {
00190         c->gate = L4::cap_reinterpret_cast<L4::Rcv_endpoint>(
00191             clt->register_obj(registry));
00192         if (!c->gate.is_valid())
00193             return -L4_ENOMEM;
00194     }
00195
00196     _interface.reset(clt.release());
00197
00198     // Let it be known that the client and the device paired
00199     if (c->pairing_cb)
00200         c->pairing_cb(_device.get());
00201     return L4_EOK;
00202 }
00203
00204 void check_clients(L4::Registry_iface *registry)
00205 {
00206     if (_interface)
00207     {
00208         if (_interface->obj_cap() && !_interface->obj_cap().validate().label())
00209             remove_client(registry);
00210
00211         return;
00212     }
00213
00214     // Sub-devices only need to be checked when the parent device was free.
00215     for (auto *sub : _subs)
00216         sub->check_clients(registry);
00217 }
00218
00220 void shutdown_event(Shutdown_type type)
00221 {
00222     // Set new shutdown state
00223     _shutdown_state = type;
00224     for (auto const &sub: _subs)
00225         sub->shutdown_event(type);
00226     if (_interface)
00227         _interface->shutdown_event(type);
00228 }
00229
00230 private:
00242 template <typename T = Device_factory_type>
00243 auto scan_disk_partitions(Errand::Callback const &callback, int)
00244     -> decltype((T::create_partition)(cxx::Ref_ptr<Device_type>(), 0, Partition_info()), void())
00245 {
00246     auto reader = cxx::make_ref_obj<Partition_reader<Device_type>>(_device.get());
00247     // The reference to reader will be captured in the lambda passed to
00248     // reader's own read() method. At the same time, reader will store
00249     // the reference to the lambda.
00250     reader->read(
00251         [=] ()
00252         {
00253             l4_size_t sz = reader->table_size();
00254
00255             for (l4_size_t i = 1; i <= sz; ++i)
00256             {
00257                 Partition_info info;
00258                 if (reader->get_partition(i, &info) < 0)
00259                     continue;
00260
00261                 auto conn = cxx::make_ref_obj<Connection>(
00262                     Device_factory_type::create_partition(_device, i, info));
00263                 _subs.push_front(std::move(conn));
00264             }
00265
00266             callback();
00267
00268             // Prolong the life-span of reader until we are sure the reader is
00269             // not currently invoked (i.e. capture the last reference to it in
00270             // an independent timeout callback).
00271             Errand::schedule([reader] () {}, 0);
00272         });
00273 }
00274
00282 template <typename T = Device_factory_type>
00283 void scan_disk_partitions(Errand::Callback const &callback, long)
00284 { callback(); }
00285
00289 void remove_client(L4::Registry_iface *registry)
00290 {

```

```

00291     assert(_interface);
00292
00293     // This operation is idempotent.
00294     _interface->shutdown_event(Shutdown_type::Client_gone);
00295
00296     if (_interface->busy())
00297     {
00298         Dbg::trace().printf("Deferring dead client removal.\n");
00299
00300         // Cannot remove the client while it still has active I/O requests.
00301         // This means that the device did not abort its inflight requests in
00302         // its reset() callback. It is still desirable though to wait for
00303         // those requests to finish and defer the dead client removal until
00304         // later.
00305         Errand::schedule([this, registry]() { remove_client(registry); },
00306                         10000);
00307         return;
00308     }
00309
00310     _interface->unregister_obj(registry);
00311     _interface.reset();
00312 }
00313
00314 bool contains_device(std::string const &name) const
00315 {
00316     if (match_hid(name))
00317         return true;
00318
00319     for (auto *sub : _subs)
00320         if (sub->contains_device(name))
00321             return true;
00322
00323     return false;
00324 }
00325
00326 bool match_hid(std::string const &name) const
00327 { return _device->match_hid(cxx::String(name.c_str(), name.length())); }
00328
00329 Shutdown_type _shutdown_state;
00330 cxx::Ref_ptr<Device_type> _device;
00331 cxx::unique_ptr<Client_type> _interface;
00332 cxx::Ref_ptr_list<Connection> _subs;
00333 };
00334
00335 public:
00336 Device_mgr(L4::Registry_iface *registry)
00337 : _registry(registry)
00338 {}
00339
00340 virtual ~Device_mgr()
00341 {
00342     for (auto *c : _connpts)
00343         c->unregister_interfaces(_registry);
00344 }
00345
00346 int add_static_client(L4::Cap<L4::Rcv_endpoint> client, const char *device,
00347                     int partno, int num_ds, bool readonly = false,
00348                     Pairing_callback cb = nullptr,
00349                     bool enable_trusted_ds_validation = false,
00350                     std::shared_ptr<Ds_vector const> trusted_dataspaces
00351                     = nullptr)
00352 {
00353     char _buf[30];
00354     const char *buf;
00355
00356     if (partno == 0)
00357     {
00358         Err().printf("Invalid partition number 0.\n");
00359         return -L4_ENODEV;
00360     }
00361
00362     if (partno != -1)
00363     {
00364         /* Could we avoid to make a string here and parsing this again
00365          * deeper in the stack? */
00366         snprintf(_buf, sizeof(_buf), "%s:%d", device, partno);
00367         buf = _buf;
00368     }
00369     else
00370         buf = device;
00371
00372     _pending_clients.emplace_back(client, buf, num_ds, readonly,
00373                                   enable_trusted_ds_validation,
00374                                   trusted_dataspaces, cb);
00375
00376     return L4_EOK;
00377 }
00378
00379 }

```

```

00382
00383 int create_dynamic_client(std::string const &device, int partno, int num_ds,
00384                          L4::Cap<void> *cap, bool readonly = false,
00385                          Pairing_callback cb = nullptr,
00386                          bool enable_trusted_ds_validation = false,
00387                          std::shared_ptr<Ds_vector const> trusted_dataspaces
00388                          = nullptr)
00389 {
00390     Pending_client clt;
00391
00392     // Maximum number of dataspaces that can be registered.
00393     clt.num_ds = num_ds;
00394
00395     clt.readonly = readonly;
00396
00397     clt.device_id = device;
00398
00399     clt.pairing_cb = cb;
00400
00401     clt.trusted_dataspaces = trusted_dataspaces;
00402
00403     clt.enable_trusted_ds_validation = enable_trusted_ds_validation;
00404
00405     if (partno > 0)
00406     {
00407         clt.device_id += ':';
00408         clt.device_id += std::to_string(partno);
00409     }
00410
00411     for (auto *c : _connpts)
00412     {
00413         int ret = c->create_interface_for(&clt, _registry);
00414
00415         if (ret == -L4_ENODEV)
00416             continue;
00417
00418         if (ret < 0)
00419             return ret;
00420
00421         // found the requested device
00422         *cap = clt.gate;
00423         return L4_EOK;
00424     }
00425
00426     return -L4_ENODEV;
00427 }
00428
00432 void check_clients()
00433 {
00434     for (auto *c : _connpts)
00435         c->check_clients(_registry);
00436 }
00437
00438 void add_disk(cxx::Ref_ptr<Device_type> &&device, Errand::Callback const &callback)
00439 {
00440     auto conn = cxx::make_ref_obj<Connection>(std::move(device));
00441
00442     conn->start_disk_scan(
00443         [=]() {
00444             {
00445                 _connpts.push_front(conn);
00446                 connect_static_clients(conn.get());
00447                 callback();
00448             }
00449         }
00450     );
00451
00452 void shutdown_event(Shutdown_type type)
00453 {
00454     l4_assert(type != Client_gone);
00455     l4_assert(type != Client_shutdown);
00456
00457     for (auto const &con : _connpts)
00458         con->shutdown_event(type);
00459 }
00460
00461 private:
00462 void connect_static_clients(Connection *con)
00463 {
00464     for (auto &c : _pending_clients)
00465     {
00466         Dbg::trace().printf("Checking existing client %s\n", c.device_id.c_str());
00467         if (!c.gate.is_valid())
00468             continue;
00469
00470         int ret = con->create_interface_for(&c, _registry);
00471
00472         if (ret == L4_EOK)

```

```

00473         {
00474             c.gate = L4::Cap<L4::Rcv_endpoint>();
00475             // There might be other clients waiting for other partitions.
00476             // Continue search.
00477             continue;
00478         }
00479
00480         if (ret != -L4_ENODEV)
00481             break;
00482     }
00483 }
00484
00485
00486 L4::Registry_iface *_registry;
00487 cxx::Ref_ptr_list<Connection> _connpts;
00491 std::vector<Pending_client> _pending_clients;
00492 };
00493
00494 } // name space

```

16.232 debug.h

```

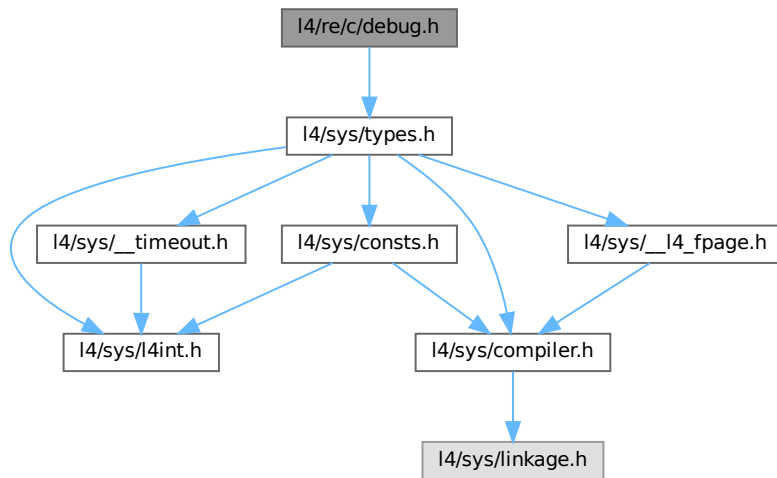
00001 /*
00002  * Copyright (C) 2018 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * This file is distributed under the terms of the GNU General Public
00006  * License, version 2. Please see the COPYING-GPL-2 file for details.
00007  */
00008 #pragma 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     public:
00024         enum Level
00025         {
00026             Blk_warn = 1,
00027             Blk_info = 2,
00028             Blk_trace = 4,
00029             Blk_steptrace = 8
00030         };
00031
00032     public:
00033         Dbg(unsigned long l = Blk_info, char const *subsys = "")
00034             : L4Re::Util::Dbg(l, "libblock", subsys) {}
00035
00036         static Dbg warn(char const *subsys = "")
00037         { return Dbg(Blk_warn, subsys); }
00038
00039         static Dbg info(char const *subsys = "")
00040         { return Dbg(Blk_info, subsys); }
00041
00042         static Dbg trace(char const *subsys = "")
00043         { return Dbg(Blk_trace, subsys); }
00044
00045         static Dbg steptrace(char const *subsys = "")
00046         { return Dbg(Blk_steptrace, subsys); }
00047     };
00048 } // name space
00049

```

16.233 l4/re/c/debug.h File Reference

Debug C interface.

```
#include <l4/sys/types.h>
Include dependency graph for debug.h:
```



Functions

- long `l4re_debug_obj_debug` (`l4_cap_idx_t` srv, unsigned long function) `L4_NOTHROW`
Call debug function of `L4Re` service.

16.233.1 Detailed Description

Debug C interface.

Definition in file `debug.h`.

16.234 debug.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
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

```

16.235 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 #include <l4/libblock-device/request_queue.h>
00018
00019 namespace Block_device {
00020
00021 struct Device : public cxx::Ref_obj
00022 {
00023     virtual ~Device() = 0;
00024
00026     virtual bool is_read_only() const = 0;
00028     virtual bool match_hid(cxx::String const &hid) const = 0;
00030     virtual l4_uint64_t capacity() const = 0;
00032     virtual l4_size_t sector_size() const = 0;
00034     virtual l4_size_t max_size() const = 0;
00036     virtual unsigned max_segments() const = 0;
00037
00046     virtual Request_queue *request_queue() = 0;
00047
00049     virtual void reset() = 0;
00050
00052     virtual int dma_map(Block_device::Mem_region *region, l4_addr_t offset,
00053                        l4_size_t num_sectors, L4Re::Dma_space::Direction dir,
00054                        L4Re::Dma_space::Dma_addr *phys) = 0;
00055
00057     virtual int dma_unmap(L4Re::Dma_space::Dma_addr phys, l4_size_t num_sectors,
00058                          L4Re::Dma_space::Direction dir) = 0;
00059
00074     virtual int inout_data(l4_uint64_t sector,
00075                           Block_device::Inout_block const &blocks,
00076                           Block_device::Inout_callback const &cb,
00077                           L4Re::Dma_space::Direction dir) = 0;
00078
00089     virtual int flush(Block_device::Inout_callback const &cb) = 0;
00090
00092     virtual void start_device_scan(Block_device::Errand::Callback const &callback) = 0;
00093 };
00094
00095 inline Device::~Device() = default;
00096
00097 template <typename DEV>
00098 class Device_with_request_queue : public DEV
00099 {
00100 public:
00101     Request_queue *request_queue() override
00102     { return &_request_queue; }
00103
00104 private:
00105     Simple_request_queue _request_queue;
00106 };
00107
00111 struct Device_discard_feature
00112 {
00113     struct Discard_info
00114     {
00115         unsigned max_discard_sectors = 0;
00116         unsigned max_discard_seg = 0;
00117         unsigned discard_sector_alignment = 1;
00118         unsigned max_write_zeroes_sectors = 0;
00119     };
00120 };

```

```

00119     unsigned max_write_zeroes_seg = 0;
00120     bool write_zeroes_may_unmap = false;
00121 };
00122
00123     virtual Discard_info discard_info() const = 0;
00124
00126     virtual int discard(l4_uint64_t offset,
00127                        Block_device::Inout_block const &blocks,
00128                        Block_device::Inout_callback const &cb, bool discard) = 0;
00129
00130 protected:
00131     ~Device_discard_feature() = default;
00132 };
00133
00134 } // name space

```

16.236 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
00074     _sif->add_timeout(p.release(), l4_kip_clock(l4re_kip()) + _interval);
00075 }
00076
00077 // Class can only be instantiated as a reference counting object.
00078 template< typename T, typename... Args >
00079 friend
00080 cxx::Ref_ptr<T> cxx::make_ref_obj(Args &&... args);
00081
00082 private:
00083 Poll_errand(int retries, int interval,
00084             std::function<bool()> const &poll_func,
00085             std::function<void(bool)> const &callback)
00086 : _retries(retries),
00087   _interval(interval),
00088   _poll(poll_func),
00089   _callback(callback)
00090 {}
00091
00092 int _retries;
00093 int _interval;
00094 std::function<bool()> _poll;
00095 std::function<void(bool)> _callback;
00096 };
00097
00098 class Errand
00099 : public L4::Ipc_svr::Timeout_queue::Timeout,
00100   public cxx::Ref_obj
00101 {
00102 public:
00103     void expired() final
00104     {
00105         // Recapture the reference pointer from the timeout queue.
00106         cxx::Ref_ptr<Errand> p(this, false);
00107
00108         if (_callback)
00109         {
00110             try
00111             {
00112                 _callback();
00113             }
00114             catch (L4::Runtime_error const &e)
00115             {
00116                 Err().printf("Asynchronous task failed: %s\n", e.str());
00117             }
00118         }
00119     }
00120
00121 void reschedule(unsigned interval = 0)
00122 {
00123     // create a placeholder reference pointer for the timeout queue
00124     cxx::Ref_ptr<Errand> p(this);
00125
00126     _sif->add_timeout(p.release(), l4_kip_clock(l4re_kip()) + interval);
00127 }
00128
00129 // Class can only be instantiated as a reference counting object.
00130 template< typename T, typename... Args >
00131 friend
00132 cxx::Ref_ptr<T> cxx::make_ref_obj(Args &&... args);
00133
00134 private:
00135     Errand(Callback const &callback) : _callback(callback) {}
00136
00137     Callback _callback;
00138 };
00139
00140 struct Loop_hooks
00141 : L4::Ipc_svr::Timeout_queue_hooks<Loop_hooks, L4Re::Util::Br_manager>,
00142   L4::Ipc_svr::Ignore_errors
00143 {
00144     l4_kernel_clock_t now() { return l4_kip_clock(l4re_kip()); }
00145 };
00146
00147 using Errand_server = L4Re::Util::Registry_server<Loop_hooks>;
00148
00149 inline void set_server_iface(L4::Ipc_svr::Server_iface *sif) { _sif = sif; }
00150
00151 inline void schedule(Callback const &callback, int interval)
00152 {
00153     cxx::make_ref_obj<Errand>(callback)->reschedule(interval);
00154 }
00155
00156 inline void poll(int retries, int interval,
00157                 std::function<bool()> const &poll_func,
00158                 std::function<void(bool)> const &callback)
00159 {

```

```

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

```

16.237 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 <linux/types.h>
00011
00012 namespace Block_device {
00013 namespace Gpt {
00014
00015 struct Header
00016 {
00017     char signature[8];
00018     uint32_t version;
00019     uint32_t header_size;
00020     uint32_t crc;
00021     uint32_t _reserved;
00022     uint64_t current_lba;
00023     uint64_t backup_lba;
00024     uint64_t first_lba;
00025     uint64_t last_lba;
00026     char disk_guid[16];
00027     uint64_t partition_array_lba;
00028     uint32_t partition_array_size;
00029     uint32_t entry_size;
00030     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     uint64_t first;
00038     uint64_t last;
00039     uint64_t flags;
00040     uint16_t name[36];
00041 };
00042
00043 } // namespace
00044
00045 namespace Pci_partition_table {
00046
00047 struct Part_table {
00048     uint8_t bootable;
00049     uint8_t first_sector_chs[3];
00050     uint8_t type;
00051     uint8_t last_sector_chs[3];
00052     uint32_t start_sector_lba;
00053     uint32_t num_sector_lba;
00054 } __attribute__((packed));
00055
00056 } // namespace
00057 } // namespace

```

16.238 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 =
00049                 cxx::make_unique<Block_device::Mem_region>(0, sz, 0, cxx::move(lcap));
00050             L4Re::chksys(_device->dma_map(_mem_region.get(), 0, _num_sectors, dir,
00051                                         &_paddr),
00052                         "Lock memory region for DMA.");
00053         }
00054
00055         Inout_memory(Inout_memory const &) = delete;
00056         Inout_memory(Inout_memory &&) = delete;
00057
00058         Inout_memory &operator=(Inout_memory &&rhs)
00059         {
00060             if (this != &rhs)
00061             {
00062                 _device = rhs._device;
00063                 _mem_region = cxx::move(rhs._mem_region);
00064                 _region = cxx::move(rhs._region);
00065                 _paddr = rhs._paddr;
00066                 _dir = rhs._dir;
00067                 _num_sectors = rhs._num_sectors;
00068                 rhs._paddr = 0;
00069             }
00070
00071             return *this;
00072         }
00073
00074         ~Inout_memory()
00075         {
00076             if (_paddr)
00077                 unmap();
00078         }
00079
00080         void unmap()
00081         {
00082             L4Re::chksys(_device->dma_unmap(_paddr, _num_sectors, _dir));
00083             _paddr = 0;
00084         }
00085
00086         Inout_block inout_block() const
00087         {
00088             Inout_block blk;
00089
00090             blk.dma_addr = _paddr;
00091             blk.virt_addr = _region.get();
00092             blk.num_sectors = _num_sectors;
00093             blk.next.reset();
00094         }
00095     };
00096 }

```

```

00099
00100     return blk;
00101 }
00102
00103 template <class T>
00104 T *get(unsigned offset) const
00105 { return reinterpret_cast<T *>(_region.get() + offset); }
00106
00107 private:
00108     Device_type *_device;
00109     cxx::unique_ptr<Block_device::Mem_region> _mem_region;
00110     L4Re::Rm::Unique_region<char *> _region;
00111     L4Re::Dma_space::Dma_addr _paddr;
00112     L4Re::Dma_space::Direction _dir;
00113     l4_uint32_t _num_sectors;
00114 };
00115
00116 } // name space

```

16.239 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     bool is_read_only() const override
00120     { return _parent->is_read_only(); }
00121
00122     bool match_hid(cxx::String const &hid) const override
00123     {
00124         if (hid == cxx::String(_guid, 36))
00125             return true;
00126
00127         std::u16string whid =
00128             std::wstring_convert<std::codecvt_utf8_utf16<char16_t>, char16_t>{}
00129                 .from_bytes(std::string(hid.start(), hid.len()));
00130         if (whid == _name)
00131             return true;
00132
00133         // check for identifier of form: <device_name>:<partition id>
00134         char const *delim = ":";
00135         char const *pos = hid.rfind(delim);
00136
00137         if (pos == hid.end() || !_parent->match_hid(cxx::String(hid.start(), pos)))
00138             return false;
00139
00140         return cxx::String(pos + 1, hid.end()) == cxx::String(_partition_id);
00141     }
00142
00143     l4_uint64_t capacity() const override
00144     { return _size * _parent->sector_size(); }
00145
00146     l4_size_t sector_size() const override
00147     { return _parent->sector_size(); }
00148
00149     l4_size_t max_size() const override
00150     { return _parent->max_size(); }
00151
00152     unsigned max_segments() const override
00153     { return _parent->max_segments(); }
00154
00155     Request_queue *request_queue() override
00156     { return _parent->request_queue(); }
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

```

16.240 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::ul6string((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%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 }
```

16.241 request_queue.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 #include <functional>
00011 #include <deque>
00012
00013 #include <l4/cxx/unique_ptr>
00014
00015 namespace Block_device {
00016
00020 struct Pending_request
00021 {
00029     struct Owner {};
00030
00031     virtual ~Pending_request() = 0;
00032
00043     virtual int handle_request() = 0;
00044
00051     virtual void fail_request() = 0;
00052
00060     virtual bool is_owner(Owner *owner) = 0;
00061 };
00062
00063 inline Pending_request::~Pending_request() = default;
00064
00065 struct Request_queue
00066 {
00067     virtual ~Request_queue() = 0;
00068
00075     virtual void add_to_queue(cxx::unique_ptr<Pending_request> &&request) = 0;
00076 }
```

```

00088     virtual void drain_queue_for(Pending_request::Owner *owner, bool finalize) = 0;
00089
00093     virtual void process_pending() = 0;
00094
00096     virtual bool empty() const = 0;
00097 };
00098
00099 inline Request_queue::~Request_queue() = default;
00100
00104 class Simple_request_queue : public Request_queue
00105 {
00106 public:
00107     bool empty() const override { return _queue.empty(); }
00108
00109     void add_to_queue(cxx::unique_ptr<Pending_request> &&request) override
00110     { _queue.emplace_back(std::move(request)); }
00111
00112     void drain_queue_for(Pending_request::Owner *owner, bool finalize) override
00113     {
00114         for (auto it = _queue.begin(); it != _queue.end(); )
00115         {
00116             if ((*it)->is_owner(owner))
00117             {
00118                 if (finalize)
00119                     (*it)->fail_request();
00120                 it = _queue.erase(it);
00121             }
00122             else
00123                 ++it;
00124         }
00125     }
00126
00127     void process_pending() override
00128     {
00129         while (!_queue.empty())
00130         {
00131             auto &front = _queue.front();
00132             int ret = front->handle_request();
00133
00134             if (ret == -L4_EBUSY)
00135                 // still no processing unit available, keep element in queue
00136                 return;
00137
00138             if (ret >= 0)
00139                 // request has been sent to hardware
00140                 front.release();
00141
00142             // element was processed, remove it from queue
00143             _queue.pop_front();
00144         }
00145     }
00146
00147 private:
00148     std::deque<cxx::unique_ptr<Pending_request>> _queue;
00149 };
00150
00151 } // namespace

```

16.242 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     L4IO_MEM_USE_RESERVED_AREA = 0x40 « 8,
00029

```

```

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 int l4io_resource_handle_t;
00061
00069 typedef l4vbus_resource_t l4io_resource_t;
00070
00074 typedef l4vbus_device_t l4io_device_t;

```

16.243 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

```

16.244 l4/sys/types.h File Reference

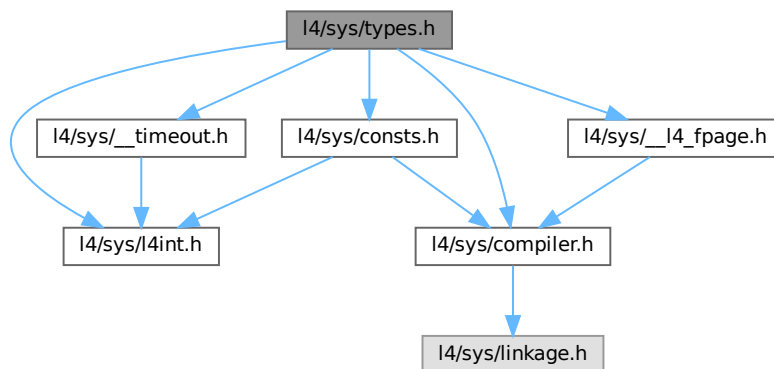
Common [L4](#) ABI Data Types.

```

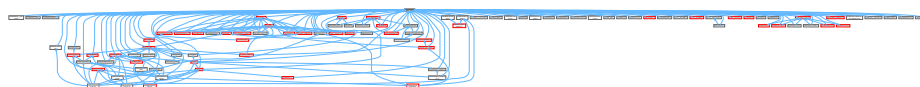
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
#include <l4/sys/consts.h>
#include <l4/sys/__l4_fpage.h>
#include <l4/sys/__timeout.h>

```

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 }
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.
- 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.

16.244.1 Detailed Description

Common [L4](#) ABI Data Types.

Definition in file [types.h](#).

16.244.2 Function Documentation

16.244.2.1 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 [479](#) of file [types.h](#).

References [L4_CAP_OFFSET](#).

16.245 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 };
00078
00079 enum L4_varg_type
00080 {
00081     L4_VARG_TYPE_NIL = 0x00,
00082     L4_VARG_TYPE_UMWORD = 0x01,
00083     L4_VARG_TYPE_MWORD = 0x81,
00084     L4_VARG_TYPE_STRING = 0x02,
00085     L4_VARG_TYPE_FPAGE = 0x03,
00086
00087     L4_VARG_TYPE_SIGN = 0x80,
00088 };
00089
00090
00095 enum L4_msgtag_flags
00096 {
00097     // flags for received IPC
00102     L4_MSGTAG_ERROR = 0x8000,
00103
00104     // flags for sending IPC
00114     L4_MSGTAG_TRANSFER_FPU = 0x1000,
00123     L4_MSGTAG_SCHEDULE = 0x2000,
00136     L4_MSGTAG_PROPAGATE = 0x4000,
00137
00142     L4_MSGTAG_FLAGS = 0xf000,
00143 };
00144
00145
00162 typedef struct l4_msgtag_t
00163 {
00164     l4_mword_t raw;
00165 #ifdef __cplusplus
00167     long label() const L4_NOTHROW { return raw >> 16; }
00169     void label(long v) L4_NOTHROW { raw = (raw & 0xffff) | ((l4_umword_t)v << 16); }
00171     unsigned words() const L4_NOTHROW { return raw & 0x3f; }
00173     unsigned items() const L4_NOTHROW { return (raw >> 6) & 0x3f; }
00180     unsigned flags() const L4_NOTHROW { return raw & 0xf000; }
00182     bool is_page_fault() const L4_NOTHROW { return label() == L4_PROTO_PAGE_FAULT; }
00184     bool is_preemption() const L4_NOTHROW { return label() == L4_PROTO_PREEMPTION; }
00186     bool is_sys_exception() const L4_NOTHROW { return label() == L4_PROTO_SYS_EXCEPTION; }
00188     bool is_exception() const L4_NOTHROW { return label() == L4_PROTO_EXCEPTION; }
00190     bool is_sigma0() const L4_NOTHROW { return label() == L4_PROTO_SIGMA0; }
00192     bool is_io_page_fault() const L4_NOTHROW { return label() == L4_PROTO_IO_PAGE_FAULT; }
00194     unsigned has_error() const L4_NOTHROW { return raw & L4_MSGTAG_ERROR; }
00195 #endif
00196 } l4_msgtag_t;
00197
00198
00199
00211 L4_INLINE l4_msgtag_t l4_msgtag(long label, unsigned words, unsigned items,
00212                                unsigned flags) L4_NOTHROW;
00213
00222 L4_INLINE long l4_msgtag_label(l4_msgtag_t t) L4_NOTHROW;
00223
00232 L4_INLINE unsigned l4_msgtag_words(l4_msgtag_t t) L4_NOTHROW;
00233
00242 L4_INLINE unsigned l4_msgtag_items(l4_msgtag_t t) L4_NOTHROW;
00243
00254 L4_INLINE unsigned l4_msgtag_flags(l4_msgtag_t t) L4_NOTHROW;
00255

```

```

00268 L4_INLINE unsigned l4_msgtag_has_error(l4_msgtag_t t) L4_NOTHROW;
00269
00278 L4_INLINE unsigned l4_msgtag_is_page_fault(l4_msgtag_t t) L4_NOTHROW;
00279
00287 L4_INLINE unsigned l4_msgtag_is_preemption(l4_msgtag_t t) L4_NOTHROW;
00288
00297 L4_INLINE unsigned l4_msgtag_is_sys_exception(l4_msgtag_t t) L4_NOTHROW;
00298
00307 L4_INLINE unsigned l4_msgtag_is_exception(l4_msgtag_t t) L4_NOTHROW;
00308
00317 L4_INLINE unsigned l4_msgtag_is_sigma0(l4_msgtag_t t) L4_NOTHROW;
00318
00327 L4_INLINE unsigned l4_msgtag_is_io_page_fault(l4_msgtag_t t) L4_NOTHROW;
00328
00358 typedef unsigned long l4_cap_idx_t;
00359
00369 L4_INLINE unsigned l4_is_invalid_cap(l4_cap_idx_t c) L4_NOTHROW;
00370
00380 L4_INLINE unsigned l4_is_valid_cap(l4_cap_idx_t c) L4_NOTHROW;
00381
00395 L4_INLINE unsigned l4_capability_equal(l4_cap_idx_t c1, l4_cap_idx_t c2) L4_NOTHROW;
00396
00405 L4_INLINE l4_cap_idx_t l4_capability_next(l4_cap_idx_t c) L4_NOTHROW;
00406
00407 /* ***** */
00408 /* Implementation */
00409
00410 L4_INLINE unsigned
00411 l4_is_invalid_cap(l4_cap_idx_t c) L4_NOTHROW
00412 { return c & L4_INVALID_CAP_BIT; }
00413
00414 L4_INLINE unsigned
00415 l4_is_valid_cap(l4_cap_idx_t c) L4_NOTHROW
00416 { return !(c & L4_INVALID_CAP_BIT); }
00417
00418 L4_INLINE unsigned
00419 l4_capability_equal(l4_cap_idx_t c1, l4_cap_idx_t c2) L4_NOTHROW
00420 { return (c1 >> L4_CAP_SHIFT) == (c2 >> L4_CAP_SHIFT); }
00421
00422
00426 L4_INLINE
00427 l4_msgtag_t l4_msgtag(long label, unsigned words, unsigned items,
00428                      unsigned flags) L4_NOTHROW
00429 {
00430     return (l4_msgtag_t){ (l4_mword_t)((l4_umword_t)label << 16)
00431                          | (l4_mword_t)(words & 0x3f)
00432                          | (l4_mword_t)((items & 0x3f) << 6)
00433                          | (l4_mword_t)(flags & 0xf000)};
00434 }
00435
00436
00437
00438 L4_INLINE
00439 long l4_msgtag_label(l4_msgtag_t t) L4_NOTHROW
00440 { return t.raw >> 16; }
00441
00442 L4_INLINE
00443 unsigned l4_msgtag_words(l4_msgtag_t t) L4_NOTHROW
00444 { return t.raw & 0x3f; }
00445
00446 L4_INLINE
00447 unsigned l4_msgtag_items(l4_msgtag_t t) L4_NOTHROW
00448 { return (t.raw >> 6) & 0x3f; }
00449
00450 L4_INLINE
00451 unsigned l4_msgtag_flags(l4_msgtag_t t) L4_NOTHROW
00452 { return t.raw & 0xf000; }
00453
00454
00455 L4_INLINE
00456 unsigned l4_msgtag_has_error(l4_msgtag_t t) L4_NOTHROW
00457 { return t.raw & L4_MSGTAG_ERROR; }
00458
00459
00460
00461 L4_INLINE unsigned l4_msgtag_is_page_fault(l4_msgtag_t t) L4_NOTHROW
00462 { return l4_msgtag_label(t) == L4_PROTO_PAGE_FAULT; }
00463
00464 L4_INLINE unsigned l4_msgtag_is_preemption(l4_msgtag_t t) L4_NOTHROW
00465 { return l4_msgtag_label(t) == L4_PROTO_PREEMPTION; }
00466
00467 L4_INLINE unsigned l4_msgtag_is_sys_exception(l4_msgtag_t t) L4_NOTHROW
00468 { return l4_msgtag_label(t) == L4_PROTO_SYS_EXCEPTION; }
00469
00470 L4_INLINE unsigned l4_msgtag_is_exception(l4_msgtag_t t) L4_NOTHROW
00471 { return l4_msgtag_label(t) == L4_PROTO_EXCEPTION; }
00472

```

```

00473 L4_INLINE unsigned l4_msgtag_is_sigma0(l4_msgtag_t t) L4_NOTHROW
00474 { return l4_msgtag_label(t) == L4_PROTO_SIGMA0; }
00475
00476 L4_INLINE unsigned l4_msgtag_is_io_page_fault(l4_msgtag_t t) L4_NOTHROW
00477 { return l4_msgtag_label(t) == L4_PROTO_IO_PAGE_FAULT; }
00478
00479 L4_INLINE l4_cap_idx_t l4_capability_next(l4_cap_idx_t c) L4_NOTHROW
00480 { return c + L4_CAP_OFFSET; }
00481
00482 #include <l4/sys/__l4_fpage.h>
00483 #include <l4/sys/__timeout.h>

```

16.246 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_queue.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    public Pending_request::Owner
00031 {
00032 protected:
00033     class Generic_pending_request : public Pending_request
00034     {
00035     protected:
00036         int check_error(int result)
00037         {
00038             if (result < 0 && result != -L4_EBUSY)
00039                 client->handle_request_error(result, this);
00040             return result;
00041         }
00042     };
00043
00044 public:
00045     explicit Generic_pending_request(Virtio_client *c, cxx::unique_ptr<Request> &&req)
00046     : request(cxx::move(req)), client(c)
00047     {}
00048
00049     void fail_request() override
00050     {
00051         client->finalize_request(cxx::move(request), 0, L4VIRTIO_BLOCK_S_IOERR);
00052     }
00053
00054     bool is_owner(Pending_request::Owner *owner) override
00055     { return static_cast<Pending_request::Owner*>(client) == owner; }
00056
00057     cxx::unique_ptr<Request> request;
00058     Virtio_client *client;
00059 };
00060
00061 struct Pending_inout_request : public Generic_pending_request
00062 {
00063     Inout_block blocks;
00064
00065     using Generic_pending_request::Generic_pending_request;
00066
00067     L4Re::Dma_space::Direction dir() const
00068     {
00069         return this->request->header().type == L4VIRTIO_BLOCK_T_OUT

```

```

00070         ? L4Re::Dma_space::Direction::To_device
00071         : L4Re::Dma_space::Direction::From_device;
00072     }
00073
00074     int handle_request() override
00075     { return this->check_error(this->client->inout_request(this)); }
00076
00077 };
00078
00079 struct Pending_flush_request : public Generic_pending_request
00080 {
00081     using Generic_pending_request::Generic_pending_request;
00082
00083     int handle_request() override
00084     { return this->check_error(this->client->flush_request(this)); }
00085 };
00086
00087 struct Pending_cmd_request : public Generic_pending_request
00088 {
00089     Inout_block blocks;
00090
00091     using Generic_pending_request::Generic_pending_request;
00092
00093     int handle_request() override
00094     {
00095         return this->check_error(this->client->discard_cmd_request(this, 0));
00096     }
00097 };
00098
00099 public:
00100     using Device_type = DEV;
00101
00110     Virtio_client(cxx::Ref_ptr<Device_type> const &dev, unsigned numds, bool readonly)
00111     : L4virtio::Svr::Block_dev_base<Mem_region_info>(L4VIRTIO_VENDOR_KK, 0x100,
00112                                                     dev->capacity() > 9,
00113                                                     dev->is_read_only()
00114                                                         || readonly),
00115       _numds(numds),
00116       _device(dev),
00117       _pending(dev->request_queue()),
00118       _in_flight(0)
00119     {
00120         reset_client();
00121         init_discard_info(0);
00122     }
00123
00127     void reset_device() override
00128     {
00129         if (_pending)
00130             _pending->drain_queue_for(this, false);
00131         _device->reset();
00132         _negotiated_features.raw = 0;
00133     }
00134
00138     void reset_client()
00139     {
00140         init_mem_info(_numds);
00141         set_seg_max(_device->max_segments());
00142         set_size_max(_device->max_size());
00143         set_flush();
00144         set_config_wce(0); // starting in write-through mode
00145         _shutdown_state = Shutdown_type::Running;
00146         _negotiated_features.raw = 0;
00147     }
00148
00149     bool queue_stopped() override
00150     { return false; }
00151
00152     bool process_request(cxx::unique_ptr<Request> &&req) override
00153     {
00154         auto trace = Dbg::trace("virtio");
00155
00156         if (_shutdown_state != Shutdown_type::Running)
00157         {
00158             trace.printf("Failing requests as the client is shutting down\n");
00159             this->finalize_request(cxx::move(req), 0, L4VIRTIO_BLOCK_S_IOERR);
00160             return false;
00161         }
00162
00163         trace.printf("request received: type 0x%x, sector 0x%llx\n",
00164                     req->header().type, req->header().sector);
00165         switch (req->header().type)
00166         {
00167             case L4VIRTIO_BLOCK_T_OUT:
00168             case L4VIRTIO_BLOCK_T_IN:
00169             {
00170                 auto pending = cxx::make_unique<Pending_inout_request>(this, cxx::move(req));

```

```

00171         int ret = build_inout_blocks(pending.get());
00172         if (ret >= 0)
00173         {
00174             if (_pending && !_pending->empty())
00175                 ret = -L4_EBUSY; // make sure to keep request order
00176             else
00177                 ret = inout_request(pending.get());
00178         } else
00179             release_dma(pending.get());
00180         return handle_request_result(ret, cxx::move(pending));
00181     }
00182     case L4VIRTIO_BLOCK_T_FLUSH:
00183     {
00184         auto pending = cxx::make_unique<Pending_flush_request>(this, cxx::move(req));
00185         int ret = check_flush_request(pending.get());
00186         if (ret == L4_EOK)
00187         {
00188             if (_pending && !_pending->empty())
00189                 ret = -L4_EBUSY; // make sure to keep request order
00190             else
00191                 ret = flush_request(pending.get());
00192         }
00193         return handle_request_result(ret, cxx::move(pending));
00194     }
00195     case L4VIRTIO_BLOCK_T_WRITE_ZEROES:
00196     case L4VIRTIO_BLOCK_T_DISCARD:
00197     {
00198         auto pending = cxx::make_unique<Pending_cmd_request>(this, cxx::move(req));
00199         return handle_discard(cxx::move(pending), 0);
00200     }
00201     default:
00202         finalize_request(cxx::move(req), 0, L4VIRTIO_BLOCK_S_UNSUPP);
00203     }
00204
00205     return true;
00206 }
00207
00208 void task_finished(Generic_pending_request *preq, int error, l4_size_t sz)
00209 {
00210     _in_flight--;
00211
00212     // move on to the next request
00213
00214     // Only finalize if the client is still alive
00215     if (_shutdown_state != Client_gone)
00216         finalize_request(cxx::move(preq->request), sz, error);
00217
00218     if (_pending)
00219         _pending->process_pending();
00220
00221     // pending request can be dropped
00222     cxx::unique_ptr<Pending_request> ureq(preq);
00223 }
00224
00228 void shutdown_event(Shutdown_type type)
00229 {
00230     // If the client is already in the Client_gone state, it means that it was
00231     // already shutdown and this is another go at its removal. This situation
00232     // can occur because at the time of its previous removal attempt there were
00233     // still I/O requests in progress.
00234     if (_shutdown_state == Client_gone)
00235         return;
00236
00237     // Transitions from System_shutdown are also not allowed, the initiator
00238     // should take care of graceful handling of this.
00239     l4_assert(_shutdown_state != System_shutdown);
00240     // If we are transitioning from System_suspend, it must be only to Running,
00241     // the initiator should handle this gracefully.
00242     l4_assert(_shutdown_state != System_suspend
00243             || type == Shutdown_type::Running);
00244
00245     // Update shutdown state of the client
00246     _shutdown_state = type;
00247
00248     if (type == Shutdown_type::Client_shutdown)
00249     {
00250         reset();
00251         reset_client();
00252         // Client_shutdown must transit to the Running state
00253         l4_assert(_shutdown_state == Shutdown_type::Running);
00254     }
00255
00256     if (type != Shutdown_type::Running)
00257     {
00258         if (_pending)
00259             _pending->drain_queue_for(this, type != Client_gone);
00260         _device->reset();

```

```

00261     }
00262 }
00263
00277 L4::Cap<void> register_obj(L4::Registry_iface *registry,
00278                          char const *service = 0)
00279 {
00280     L4Re::chkcap(registry->register_irq_obj(this->irq_iface()));
00281     L4::Cap<void> ret;
00282     if (service)
00283         ret = registry->register_obj(this, service);
00284     else
00285         ret = registry->register_obj(this);
00286     L4Re::chkcap(ret);
00287
00288     return ret;
00289 }
00290
00291 L4::Cap<void> register_obj(L4::Registry_iface *registry,
00292                          L4::Cap<L4::Rcv_endpoint> ep)
00293 {
00294     L4Re::chkcap(registry->register_irq_obj(this->irq_iface()));
00295
00296     return L4Re::chkcap(registry->register_obj(this, ep));
00297 }
00298
00304 void unregister_obj(L4::Registry_iface *registry)
00305 {
00306     // We need to delete the IRQ object created in register_irq_obj() ourselves
00307     L4::Cap<L4::Task> (L4Re::This_task)
00308         ->unmap(this->irq_iface()->obj_cap().fpage(),
00309               L4_FP_ALL_SPACES | L4_FP_DELETE_OBJ);
00310     registry->unregister_obj(this->irq_iface());
00311     registry->unregister_obj(this);
00312 }
00313
00314 bool busy() const
00315 {
00316     return _in_flight != 0;
00317 }
00318
00319 protected:
00320 L4::Ipc_svr::Server_iface *server_iface() const override
00321 {
00322     return this->L4::Epiface::server_iface();
00323 }
00324
00325 private:
00326 void release_dma(Pending_inout_request *req)
00327 {
00328     // unmap DMA regions
00329     Inout_block *cur = &req->blocks;
00330     while (cur)
00331     {
00332         if (cur->num_sectors)
00333             _device->dma_unmap(cur->dma_addr, cur->num_sectors, req->dir());
00334         cur = cur->next.get();
00335     }
00336 }
00337
00338 int build_inout_blocks(Pending_inout_request *preq)
00339 {
00340     auto *req = preq->request.get();
00341     l4_size_t sps = _device->sector_size() >> 9;
00342     l4_uint64_t current_sector = req->header().sector / sps;
00343     l4_uint64_t sectors = _device->capacity() / _device->sector_size();
00344     auto dir = preq->dir();
00345
00346     l4_uint32_t flags = 0;
00347     if (req->header().type == L4VIRTIO_BLOCK_T_OUT)
00348     {
00349         // If RO was offered, every write must fail
00350         if (device_features().ro())
00351             return -L4_EIO;
00352
00353         // Figure out whether the write has a write-through or write-back semantics
00354         if (_negotiated_features.config_wce())
00355         {
00356             if (get_writeback() == 1)
00357                 flags = Block_device::Inout_f_wb;
00358         }
00359         else if (_negotiated_features.flush())
00360             flags = Block_device::Inout_f_wb;
00361     }
00362
00363     // Check alignment of the first sector
00364     if (current_sector * sps != req->header().sector)
00365         return -L4_EIO;

```

```

00366
00367     Inout_block *last_blk = nullptr;
00368
00369     size_t seg = 0;
00370
00371     while (req->has_more())
00372     {
00373         Request::Data_block b;
00374
00375         if (++seg > _device->max_segments())
00376             return -L4_EIO;
00377
00378         try
00379         {
00380             b = req->next_block();
00381         }
00382         catch (L4virtio::Svr::Bad_descriptor const &e)
00383         {
00384             return -L4_EIO;
00385         }
00386
00387         l4_size_t off = b.mem->ds_offset() + (l4_addr_t) b.addr
00388             - (l4_addr_t) b.mem->local_base();
00389
00390         l4_size_t sz = b.len / _device->sector_size();
00391
00392         if (sz * _device->sector_size() != b.len)
00393         {
00394             Dbg::warn().printf("Bad block size 0x%x\n", b.len);
00395             return -L4_EIO;
00396         };
00397
00398         // Check bounds
00399         if (sz > sectors)
00400             return -L4_EIO;
00401         if (current_sector > sectors - sz)
00402             return -L4_EIO;
00403
00404         Inout_block *blk;
00405         if (last_blk)
00406         {
00407             last_blk->next = cxx::make_unique<Inout_block>();
00408             blk = last_blk->next.get();
00409         }
00410         else
00411             blk = &preq->blocks;
00412
00413         L4Re::Dma_space::Dma_addr phys;
00414         long ret = _device->dma_map(b.mem, off, sz, dir, &phys);
00415         if (ret < 0)
00416             return ret;
00417
00418         blk->dma_addr = phys;
00419         blk->virt_addr = (void *) ((l4_addr_t)b.mem->local_base() + off);
00420         blk->num_sectors = sz;
00421         current_sector += sz;
00422         blk->flags = flags;
00423
00424         last_blk = blk;
00425     }
00426
00427     return L4_EOK;
00428 }
00429
00430 void maintain_cache_before_req(Pending_inout_request const *preq)
00431 {
00432     if (preq->dir() == L4Re::Dma_space::None)
00433         return;
00434     for (Inout_block const *cur = &preq->blocks; cur; cur = cur->next.get())
00435     {
00436         l4_addr_t vstart = (l4_addr_t)cur->virt_addr;
00437         if (vstart)
00438         {
00439             l4_size_t vsize = cur->num_sectors * _device->sector_size();
00440             if (preq->dir() == L4Re::Dma_space::From_device)
00441                 l4_cache_inv_data(vstart, vstart + vsize);
00442             else if (preq->dir() == L4Re::Dma_space::To_device)
00443                 l4_cache_clean_data(vstart, vstart + vsize);
00444             else // L4Re::Dma_space::Bidirectional
00445                 l4_cache_flush_data(vstart, vstart + vsize);
00446         }
00447     }
00448 }
00449
00450 void maintain_cache_after_req(Pending_inout_request const *preq)
00451 {
00452     if (preq->dir() == L4Re::Dma_space::None)

```

```

00453     return;
00454     for (Inout_block const *cur = &preq->blocks; cur; cur = cur->next.get())
00455     {
00456         l4_addr_t vstart = (l4_addr_t)cur->virt_addr;
00457         if (vstart)
00458         {
00459             l4_size_t vsize = cur->num_sectors * _device->sector_size();
00460             if (preq->dir() != L4Re::Dma_space::To_device)
00461                 l4_cache_inv_data(vstart, vstart + vsize);
00462         }
00463     }
00464 }
00465
00466 int inout_request(Pending_inout_request *preq)
00467 {
00468     auto *req = preq->request.get();
00469     l4_uint64_t sector = req->header().sector / (_device->sector_size() >> 9);
00470
00471     maintain_cache_before_req(preq);
00472     int res = _device->inout_data(
00473         sector, preq->blocks,
00474         [this, preq](int error, l4_size_t sz) {
00475             release_dma(preq);
00476             maintain_cache_after_req(preq);
00477             task_finished(preq, error, sz);
00478         },
00479         preq->dir());
00480
00481     // request successfully submitted to device
00482     if (res >= 0)
00483         _in_flight++;
00484
00485     return res;
00486 }
00487
00488 int check_flush_request(Pending_flush_request *preq)
00489 {
00490     if (!_negotiated_features.flush())
00491         return -L4_ENOSYS;
00492
00493     auto *req = preq->request.get();
00494
00495     // sector must be zero for FLUSH
00496     if (req->header().sector)
00497         return -L4_ENOSYS;
00498
00499     return L4_EOK;
00500 }
00501
00502 int flush_request(Pending_flush_request *preq)
00503 {
00504     int res = _device->flush([this, preq](int error, l4_size_t sz) {
00505         task_finished(preq, error, sz);
00506     });
00507
00508     // request successfully submitted to device
00509     if (res >= 0)
00510         _in_flight++;
00511
00512     return res;
00513 }
00514
00515 bool check_features(void) override
00516 {
00517     _negotiated_features = negotiated_features();
00518     return true;
00519 }
00520
00521 template <typename T = Device_type>
00522 void init_discard_info(long) {}
00523
00524 template <typename T = Device_type>
00525 auto init_discard_info(int)
00526     -> decltype(((T*)0)->discard_info(), void())
00527 {
00528     _di = _device->discard_info();
00529
00530     // Convert sector sizes to virtio 512-byte sectors.
00531     size_t sps = _device->sector_size() >> 9;
00532     if (_di.max_discard_sectors)
00533         set_discard(_di.max_discard_sectors * sps, _di.max_discard_seg,
00534             _di.discard_sector_alignment * sps);
00535     if (_di.max_write_zeroes_sectors)
00536         set_write_zeroes(_di.max_write_zeroes_sectors * sps,
00537             _di.max_write_zeroes_seg, _di.write_zeroes_may_unmap);
00538 }
00539

```



```

00540 bool handle_discard(cxx::unique_ptr<Pending_cmd_request> &&pending, int)
00541 {
00542     int ret = build_discard_cmd_blocks(pending.get());
00543     if (ret >= 0)
00544     {
00545         if (this->_pending && !this->_pending->empty())
00546             ret = -L4_EBUSY; // make sure to keep request order
00547         else
00548             ret = discard_cmd_request(pending.get(), 0);
00549     }
00550
00551     return this->handle_request_result(ret, cxx::move(pending));
00552 }
00553
00554 int build_discard_cmd_blocks(Pending_cmd_request *preq)
00555 {
00556     auto *req = preq->request.get();
00557     bool discard = (req->header().type == L4VIRTIO_BLOCK_T_DISCARD);
00558
00559     if (this->device_features().ro())
00560         return -L4_EIO;
00561
00562     // sector is used only for inout requests, it must be zero for WzD
00563     if (req->header().sector)
00564         return -L4_ENOSYS;
00565
00566     if (discard)
00567     {
00568         if (!negotiated_features.discard())
00569             return -L4_ENOSYS;
00570     }
00571     else
00572     {
00573         if (!negotiated_features.write_zeroes())
00574             return -L4_ENOSYS;
00575     }
00576
00577     auto *d = _device.get();
00578
00579     size_t seg = 0;
00580     size_t max_seg = discard ? _di.max_discard_seg : _di.max_write_zeroes_seg;
00581
00582     l4_size_t sps = d->sector_size() >> 9;
00583     l4_uint64_t sectors = d->capacity() / d->sector_size();
00584
00585     Inout_block *last_blk = nullptr;
00586
00587     while (req->has_more())
00588     {
00589         Request::Data_block b;
00590
00591         try
00592         {
00593             b = req->next_block();
00594         }
00595         catch (L4virtio::Svr::Bad_descriptor const &e)
00596         {
00597             return -L4_EIO;
00598         }
00599
00600         auto *payload = reinterpret_cast<l4virtio_block_discard_t *>(b.addr);
00601
00602         size_t items = b.len / sizeof(payload[0]);
00603         if (items * sizeof(payload[0]) != b.len)
00604             return -L4_EIO;
00605
00606         if (seg + items > max_seg)
00607             return -L4_EIO;
00608         seg += items;
00609
00610         for (auto i = 0u; i < items; i++)
00611         {
00612             auto p = cxx::access_once<l4virtio_block_discard_t>(&payload[i]);
00613
00614             // Check sector size alignment. Discard sector alignment is not
00615             // strictly enforced as it is merely a hint to the driver.
00616             if (p.sector % sps != 0)
00617                 return -L4_EIO;
00618             if (p.num_sectors % sps != 0)
00619                 return -L4_EIO;
00620
00621             // Convert to the device sector size
00622             p.sector /= sps;
00623             p.num_sectors /= sps;
00624
00625             // Check bounds
00626             if (p.num_sectors > sectors)

```

```

00627         return -L4_EIO;
00628     if (p.sector > sectors - p.num_sectors)
00629         return -L4_EIO;
00630
00631     if (p.flags & L4VIRTIO_BLOCK_DISCARD_F_RESERVED)
00632         return -L4_ENOSYS;
00633
00634     if (discard)
00635     {
00636         if (p.flags & L4VIRTIO_BLOCK_DISCARD_F_UNMAP)
00637             return -L4_ENOSYS;
00638         if (p.num_sectors > _di.max_discard_sectors)
00639             return -L4_EIO;
00640     }
00641     else
00642     {
00643         if (p.flags & L4VIRTIO_BLOCK_DISCARD_F_UNMAP
00644             && !_di.write_zeroes_may_unmap)
00645             return -L4_ENOSYS;
00646         if (p.num_sectors > _di.max_write_zeroes_sectors)
00647             return -L4_EIO;
00648     }
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     if (p.flags & L4VIRTIO_BLOCK_DISCARD_F_UNMAP)
00662         blk->flags = Inout_f_unmap;
00663
00664     last_blk = blk;
00665 }
00666
00667 return L4_EOK;
00668 }
00669
00670 template <typename T = Device_type>
00671 int discard_cmd_request(Pending_cmd_request *, long)
00672 { return -L4_EIO; }
00673
00674 template <typename T = Device_type>
00675 auto discard_cmd_request(Pending_cmd_request *preq, int)
00676     -> decltype(((T*)0)->discard_info(), int())
00677 {
00678     auto *req = preq->request.get();
00679     bool discard = (req->header().type == L4VIRTIO_BLOCK_T_DISCARD);
00680
00681     int res = _device->discard(
00682         0, preq->blocks,
00683         [this, preq](int error, l4_size_t sz) { task_finished(preq, error, sz); },
00684         discard);
00685
00686     // request successfully submitted to device
00687     if (res >= 0)
00688         _in_flight++;
00689
00690     return res;
00691 }
00692
00693 template <typename REQ>
00694 bool handle_request_result(int error, cxx::unique_ptr<REQ> &&pending)
00695 {
00696     if (error == -L4_EBUSY && _pending)
00697     {
00698         Dbg::trace("virtio").printf("Port busy, queueing request.\n");
00699         _pending->add_to_queue(cxx::unique_ptr<Pending_request>(pending.release()));
00700     }
00701     else if (error < 0)
00702         handle_request_error(error, pending.get());
00703     else
00704     {
00705         // request has been successfully sent to hardware
00706         // which now has ownership of Request pointer, so release here
00707         pending.release();
00708     }
00709
00710     return true;
00711 }
00712
00713

```

```

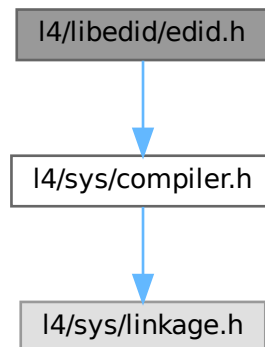
00714 // only use on errors that are not busy
00715 void handle_request_error(int error, Generic_pending_request *pending)
00716 {
00717     auto trace = Dbg::trace("virtio");
00718
00719     if (error == -L4_ENOSYS)
00720     {
00721         trace.printf("Unsupported operation.\n");
00722         finalize_request(cxx::move(pending->request), 0,
00723                         L4VIRTIO_BLOCK_S_UNSUPP);
00724     }
00725     else
00726     {
00727         trace.printf("Got IO error: %d\n", error);
00728         finalize_request(cxx::move(pending->request), 0, L4VIRTIO_BLOCK_S_IOERR);
00729     }
00730 }
00731
00732 protected:
00733     unsigned _numds;
00734     Shutdown_type _shutdown_state;
00735     cxx::Ref_ptr<Device_type> _device;
00736     Request_queue *_pending;
00737     Device_discard_feature::Discard_info _di;
00738
00739     L4virtio::Svr::Block_features _negotiated_features;
00740
00741     unsigned _in_flight;
00742 };
00743
00744 } //name space

```

16.247 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.

16.248 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

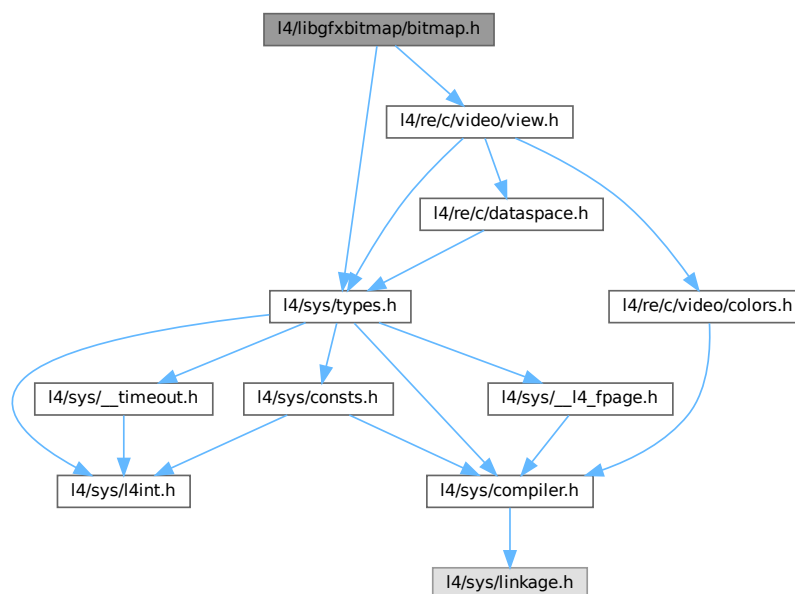
```

16.249 I4/libgfxbitmap/bitmap.h File Reference

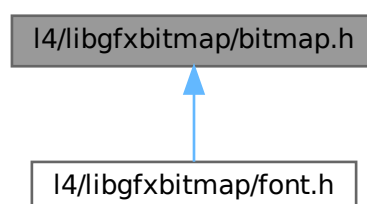
Bitmap renderer header file.

```
#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.

16.249.1 Detailed Description

Bitmap renderer header file.

Definition in file [bitmap.h](#).

16.249.2 Macro Definition Documentation

16.249.2.1 pSLIM_BMAP_START_LSB

```
#define pSLIM_BMAP_START_LSB 0x01
```

the other way round

Definition at line 41 of file [bitmap.h](#).

16.249.3 Typedef Documentation

16.249.3.1 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](#).

16.249.3.2 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](#).

16.249.4 Function Documentation

16.249.4.1 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](#).

16.249.4.2 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.

16.249.4.3 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.

16.249.4.4 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.

16.249.4.5 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.

16.250 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

```

16.251 l4/libgfxbitmap/font.h File Reference

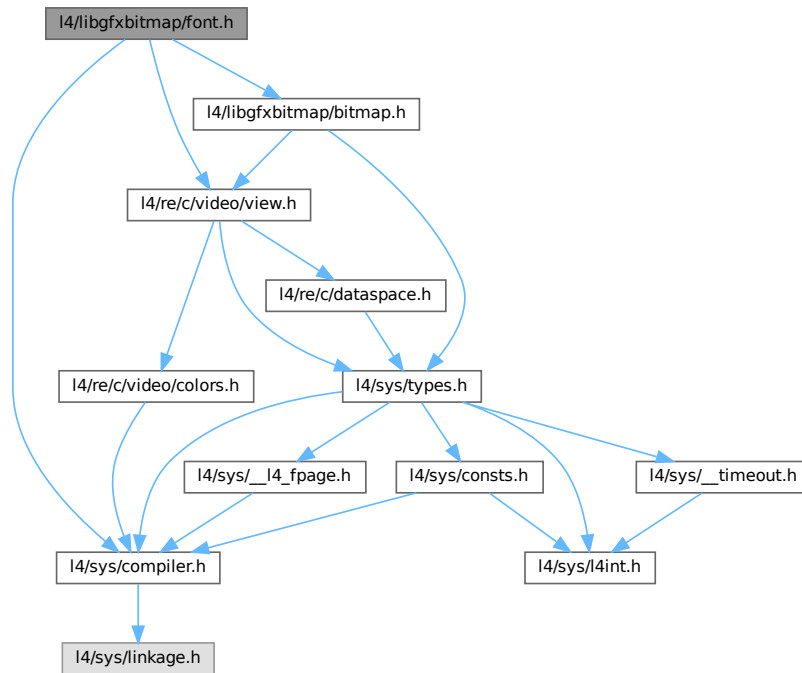
Bitmap font renderer header file.

```

#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.

16.251.1 Detailed Description

Bitmap font renderer header file.

Definition in file [font.h](#).

16.251.2 Enumeration Type Documentation

16.251.2.1 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](#).

16.251.3 Function Documentation

16.251.3.1 [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.

16.251.3.2 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.

16.251.3.3 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.

16.251.3.4 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

16.251.3.5 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.

16.251.3.6 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> _{<i>x</i>}	Horizontal scale factor.
<i>scale</i> _{<i>y</i>}	Vertical scale factor.

16.251.3.7 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.

16.252 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 *
```


Date

2009

Author

Adam Lackorzynski adam@os.inf.tu-dresden.deDefinition in file [support](#).

16.254 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

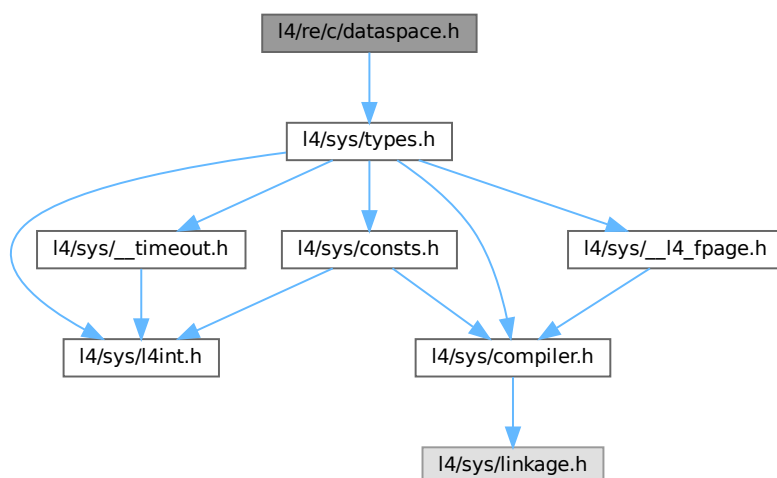
```

16.255 l4/re/c/dataspace.h File Reference

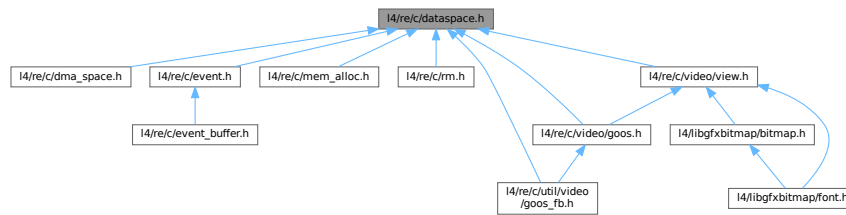
Data space C interface.

#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.

16.255.1 Detailed Description

Data space C interface.

Definition in file [dataspace.h](#).

16.256 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
00160 EXTERN_C_END

```

16.257 l4/re/c/dma_space.h File Reference

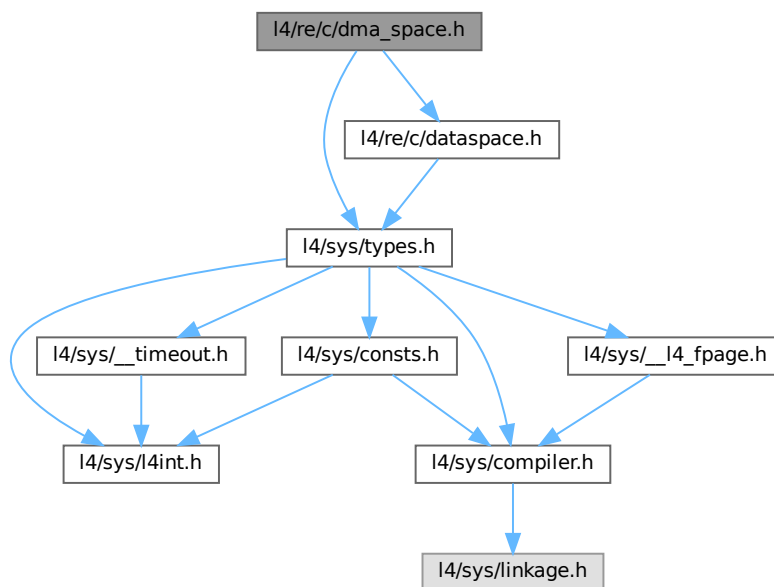
DMA space C interface.

```

#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.

16.257.1 Detailed Description

DMA space C interface.

Definition in file [dma_space.h](#).

16.257.2 Enumeration Type Documentation

16.257.2.1 l4re_dma_space_direction

enum [l4re_dma_space_direction](#)

Direction of the DMA transfers.

Enumerator

L4RE_DMA_SPACE_BIDIRECTIONAL	device reads and writes to the memory
L4RE_DMA_SPACE_TO_DEVICE	device reads the memory
L4RE_DMA_SPACE_FROM_DEVICE	device writes to the memory
L4RE_DMA_SPACE_NONE	device is coherently connected

Definition at line 37 of file [dma_space.h](#).

16.257.2.2 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](#).

16.258 dma_space.h

[Go to the documentation of this file.](#)

```

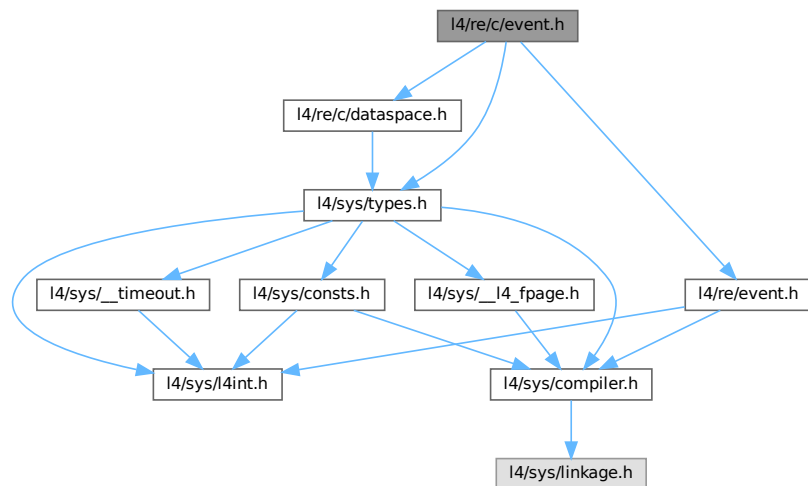
00001
00005 /*
00006  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00007  *
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU General Public License 2.
00010  * Please see the COPYING-GPL-2 file for details.
00011  *
00012  * As a special exception, 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

```

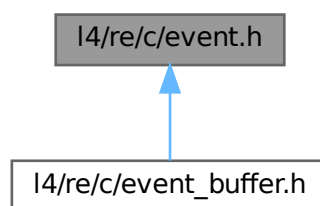
16.259 l4/re/c/event.h File Reference

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.

16.259.1 Detailed Description

Event C interface.

Definition in file [event.h](#).

16.260 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

```

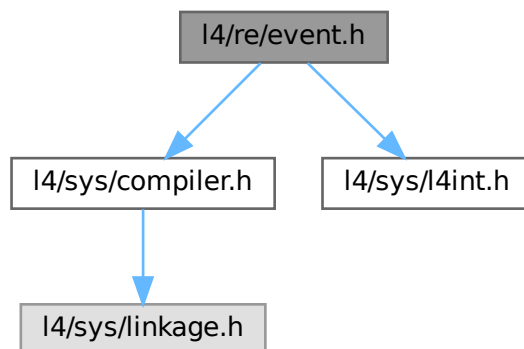
16.261 l4/re/event.h File Reference

Events.

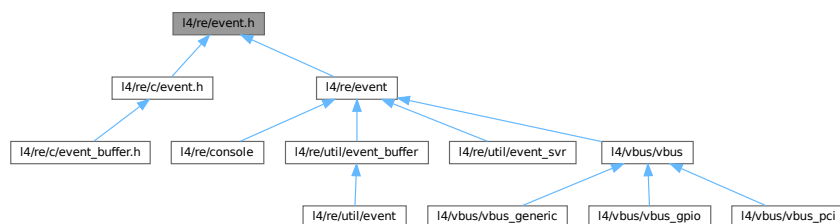
```
#include <l4/sys/compiler.h>
```

```
#include <l4/sys/l4int.h>
```

Include dependency graph for event.h:



This graph shows which files directly or indirectly include this file:



16.261.1 Detailed Description

Events.

Definition in file [event.h](#).

16.262 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
00061 #define __UNUM_B(x) ((x+1) + sizeof(unsigned long)*8 - 1) / (sizeof(unsigned long)*8)
00062
00063 typedef struct L4_EXPORT_TYPE l4re_event_stream_info_t
00064 {
00065     l4_umword_t stream_id;
00066     char name[32];
00067     char phys[32];
00068     l4re_event_stream_id_t id;
00069
00070     unsigned long propbits[__UNUM_B(L4RE_EVENT_PROP_MAX)];
00071
00072     unsigned long evbits[__UNUM_B(L4RE_EVENT_EV_MAX)];

```

```

00073 unsigned long keybits[__UNUM_B(L4RE_EVENT_KEY_MAX)];
00074 unsigned long relbits[__UNUM_B(L4RE_EVENT_REL_MAX)];
00075 unsigned long absbits[__UNUM_B(L4RE_EVENT_ABS_MAX)];
00076 unsigned long swbits[__UNUM_B(L4RE_EVENT_SW_MAX)];
00077
00078 } l4re_event_stream_info_t;
00079
00080 typedef struct L4_EXPORT_TYPE l4re_event_stream_state_t
00081 {
00082     unsigned long keybits[__UNUM_B(L4RE_EVENT_KEY_MAX)];
00083     unsigned long swbits[__UNUM_B(L4RE_EVENT_SW_MAX)];
00084 } l4re_event_stream_state_t;
00085
00086 #undef __UNUM_B
00087

```

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

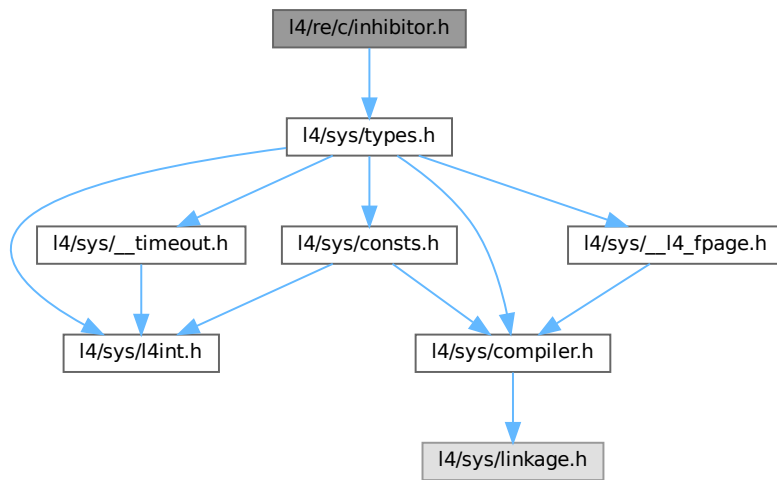
```

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 L4Re::Inhibitor)
<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 L4Re::Inhibitor)
<i>name</i>	A pointer to a buffer for the name of the lock.
<i>len</i>	The length of the available buffer (usually L4Re::Inhibitor::Name_max 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 L4Re::Inhibitor).
<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

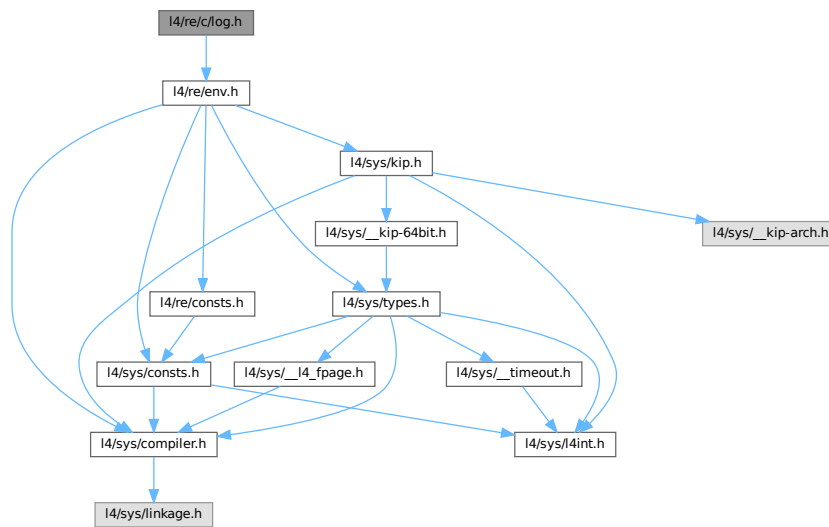
```

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

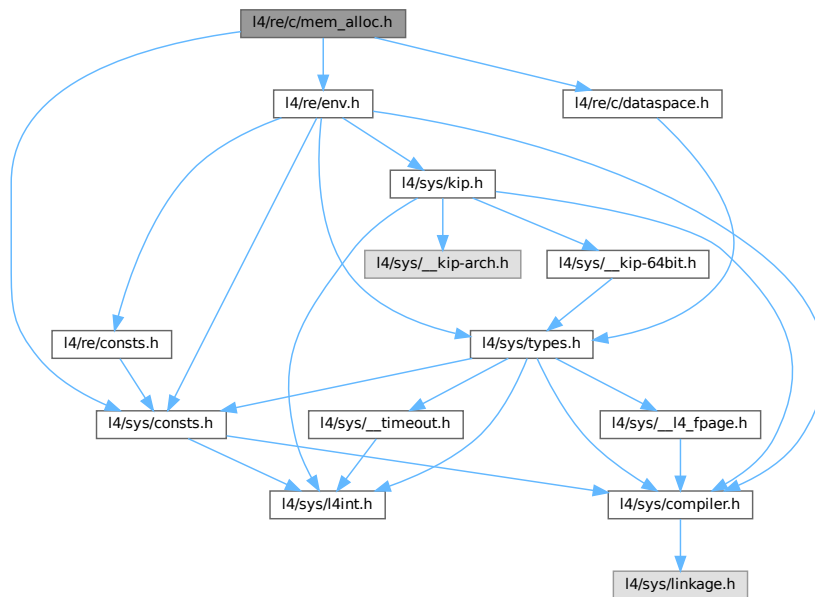
Memory allocator C interface.

```

#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

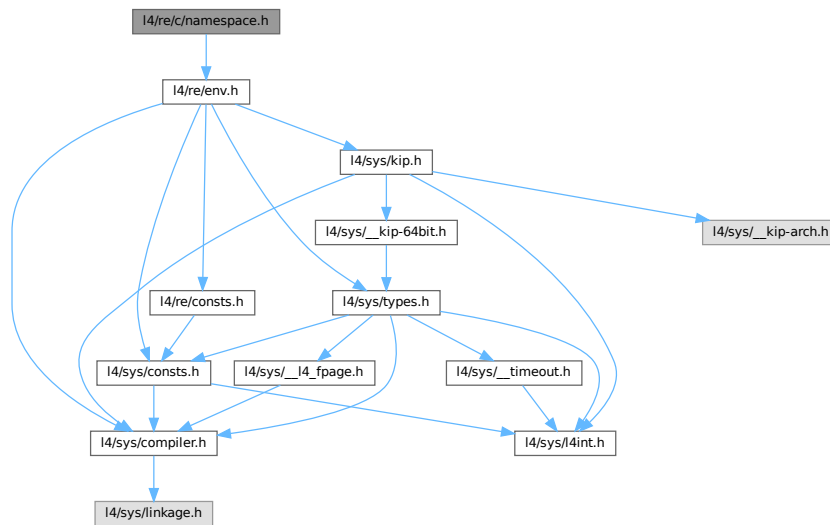
```

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/rm.h File Reference

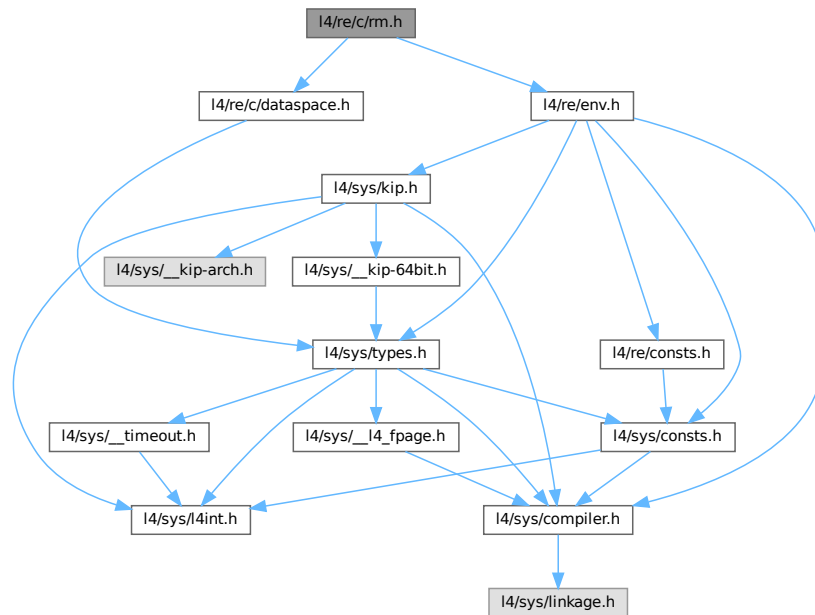
Region map interface, C interface.

```

#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` = `0x20000` ,
`L4RE_RM_F_IN_AREA` = `0x40000` , `L4RE_RM_F_EAGER_MAP` = `0x80000` , `L4RE_RM_F_ATTACH_FLAGS`
= `0xf0000` }

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.

16.272.1 Detailed Description

Region map interface, C interface.

Definition in file [rm.h](#).

16.273 rm.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 #include <l4/re/env.h>
00023 #include <l4/re/c/dataspace.h>
00024
00025 EXTERN_C_BEGIN
00026
00027 enum l4re_rm_flags_values {
00028     L4RE_RM_F_R    = L4RE_DS_F_R,
00029     L4RE_RM_F_W    = L4RE_DS_F_W,
00030     L4RE_RM_F_X    = L4RE_DS_F_X,
00031     L4RE_RM_F_RX   = L4RE_DS_F_RX,
00032

```

```

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 = 0x20000,
00069 L4RE_RM_F_IN_AREA     = 0x40000,
00070 L4RE_RM_F_EAGER_MAP   = 0x80000,
00071 L4RE_RM_F_ATTACH_FLAGS = 0xf0000,
00072 };
00073
00074 typedef l4_uint32_t l4re_rm_flags_t;
00075 typedef l4_uint64_t l4re_rm_offset_t;
00076
00085 L4_CV L4_INLINE int
00086 l4re_rm_reserve_area(l4_addr_t *start, unsigned long size,
00087                     l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW;
00088
00097 L4_CV L4_INLINE int
00098 l4re_rm_free_area(l4_addr_t addr) L4_NOTHROW;
00099
00108 L4_CV L4_INLINE int
00109 l4re_rm_attach(void **start, unsigned long size, l4re_rm_flags_t flags,
00110               l4re_ds_t mem,
00111               l4re_rm_offset_t offs,
00112               unsigned char align) L4_NOTHROW;
00113
00114
00132 L4_CV L4_INLINE int
00133 l4re_rm_detach(void *addr) L4_NOTHROW;
00134
00154 L4_CV L4_INLINE int
00155 l4re_rm_detach_ds(void *addr, l4re_ds_t *ds) L4_NOTHROW;
00156
00168 L4_CV L4_INLINE int
00169 l4re_rm_detach_unmap(l4_addr_t addr, l4_cap_idx_t task) L4_NOTHROW;
00170
00185 L4_CV L4_INLINE int
00186 l4re_rm_detach_ds_unmap(void *addr, l4re_ds_t *ds,
00187                         l4_cap_idx_t task) L4_NOTHROW;
00188
00189
00196 L4_CV L4_INLINE int
00197 l4re_rm_find(l4_addr_t *addr, unsigned long *size,
00198             l4re_rm_offset_t *offset,
00199             l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW;
00200
00207 L4_CV L4_INLINE void
00208 l4re_rm_show_lists(void) L4_NOTHROW;
00209
00210
00211 /*
00212  * Variants of functions that also take a capability of the region map
00213  * service.
00214  */
00215
00216
00221 L4_CV int
00222 l4re_rm_reserve_area_srv(l4_cap_idx_t rm, l4_addr_t *start, unsigned long size,
00223                         l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW;
00224
00229 L4_CV int
00230 l4re_rm_free_area_srv(l4_cap_idx_t rm, l4_addr_t addr) L4_NOTHROW;
00231
00236 L4_CV int
00237 l4re_rm_attach_srv(l4_cap_idx_t rm, void **start, unsigned long size,
00238                  l4re_rm_flags_t flags, l4re_ds_t mem,
00239                  l4re_rm_offset_t offs,
00240                  unsigned char align) L4_NOTHROW;
00241
00242
00247 L4_CV int
00248 l4re_rm_detach_srv(l4_cap_idx_t rm, l4_addr_t addr,
00249                  l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW;
00250
00251

```

```

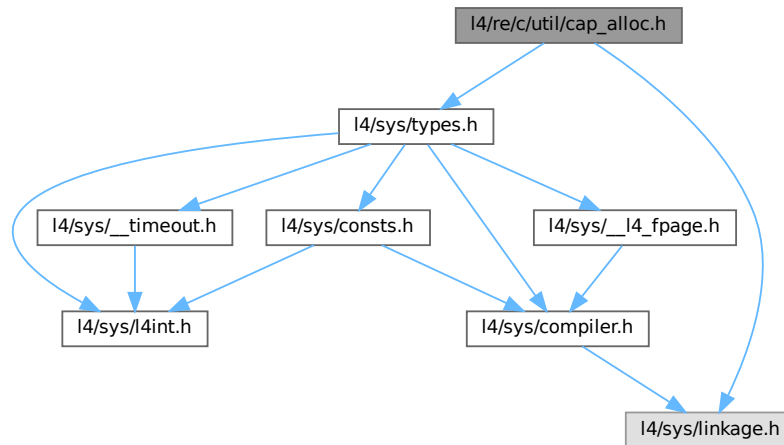
00256 L4_CV int
00257 l4re_rm_find_srv(l4_cap_idx_t rm, l4_addr_t *addr,
00258                 unsigned long *size, l4re_rm_offset_t *offset,
00259                 l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW;
00260
00265 L4_CV void
00266 l4re_rm_show_lists_srv(l4_cap_idx_t rm) L4_NOTHROW;
00267
00268
00269 /***** Implementations *****/
00270
00271 L4_CV L4_INLINE int
00272 l4re_rm_reserve_area(l4_addr_t *start, unsigned long size,
00273                    l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW
00274 {
00275     return l4re_rm_reserve_area_srv(l4re_global_env->rm, start, size,
00276                                     flags, align);
00277 }
00278
00279 L4_CV L4_INLINE int
00280 l4re_rm_free_area(l4_addr_t addr) L4_NOTHROW
00281 {
00282     return l4re_rm_free_area_srv(l4re_global_env->rm, addr);
00283 }
00284
00285 L4_CV L4_INLINE int
00286 l4re_rm_attach(void **start, unsigned long size, l4re_rm_flags_t flags,
00287              l4re_ds_t mem, l4re_rm_offset_t offs,
00288              unsigned char align) L4_NOTHROW
00289 {
00290     return l4re_rm_attach_srv(l4re_global_env->rm, start, size,
00291                              flags, mem, offs, align);
00292 }
00293
00294
00295 L4_CV L4_INLINE int
00296 l4re_rm_detach(void *addr) L4_NOTHROW
00297 {
00298     return l4re_rm_detach_srv(l4re_global_env->rm,
00299                             (l4_addr_t)addr, 0, L4_BASE_TASK_CAP);
00300 }
00301
00302 L4_CV L4_INLINE int
00303 l4re_rm_detach_unmap(l4_addr_t addr, l4_cap_idx_t task) L4_NOTHROW
00304 {
00305     return l4re_rm_detach_srv(l4re_global_env->rm, addr, 0, task);
00306 }
00307
00308 L4_CV L4_INLINE int
00309 l4re_rm_detach_ds(void *addr, l4re_ds_t *ds) L4_NOTHROW
00310 {
00311     return l4re_rm_detach_srv(l4re_global_env->rm, (l4_addr_t)addr,
00312                             ds, L4_BASE_TASK_CAP);
00313 }
00314
00315 L4_CV L4_INLINE int
00316 l4re_rm_detach_ds_unmap(void *addr, l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW
00317 {
00318     return l4re_rm_detach_srv(l4re_global_env->rm, (l4_addr_t)addr,
00319                             ds, task);
00320 }
00321
00322 L4_CV L4_INLINE int
00323 l4re_rm_find(l4_addr_t *addr, unsigned long *size,
00324             l4re_rm_offset_t *offset,
00325             l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW
00326 {
00327     return l4re_rm_find_srv(l4re_global_env->rm, addr, size, offset, flags, m);
00328 }
00329
00330 L4_CV L4_INLINE void
00331 l4re_rm_show_lists(void) L4_NOTHROW
00332 {
00333     l4re_rm_show_lists_srv(l4re_global_env->rm);
00334 }
00335
00336 EXTERN_C_END

```

16.274 l4re/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.

16.274.1 Detailed Description

Capability allocator C interface.

Definition in file [cap_alloc.h](#).

16.275 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

```

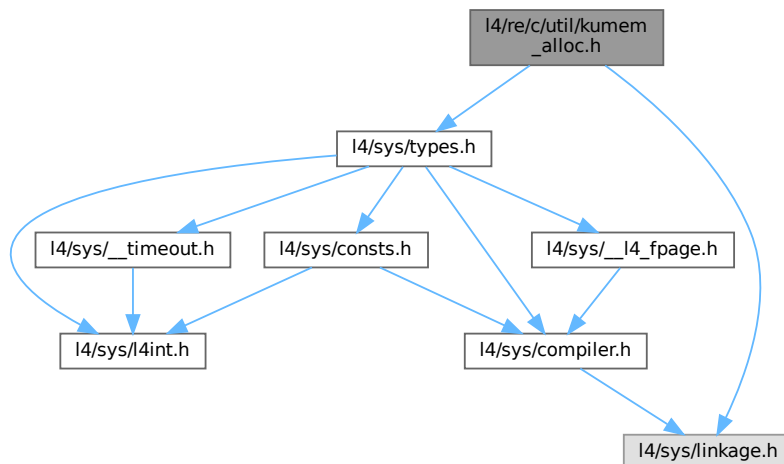
16.276 l4/re/c/util/kumem_alloc.h File Reference

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.

16.276.1 Detailed Description

Kumem allocator utility C interface.

Definition in file [kumem_alloc.h](#).

16.276.2 Function Documentation

16.276.2.1 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.
	<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.

Examples

[examples/sys/aliens/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

16.277 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

```

16.278 l4/re/c/util/video/goos_fb.h File Reference

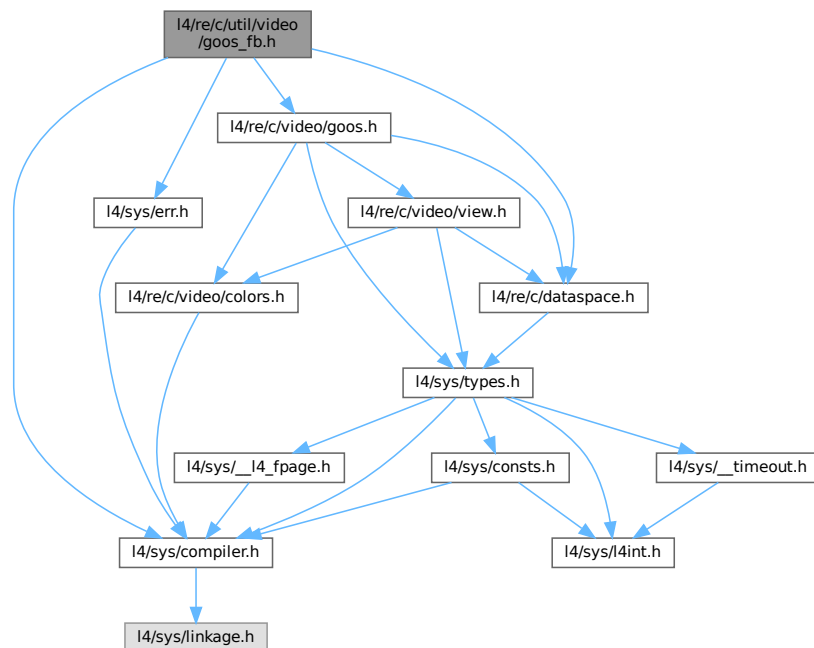
Framebuffer utility functionality.

```

#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:



16.278.1 Detailed Description

Framebuffer utility functionality.

Definition in file [goos_fb.h](#).

16.279 goos_fb.h

[Go to the documentation of this file.](#)

```

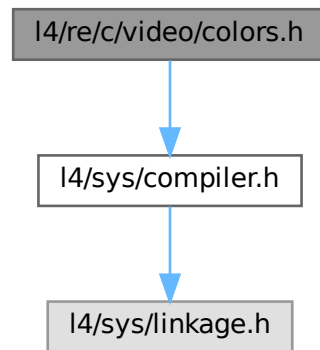
00001
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 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
00024 #include <l4/sys/compiler.h>
00025 #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

```

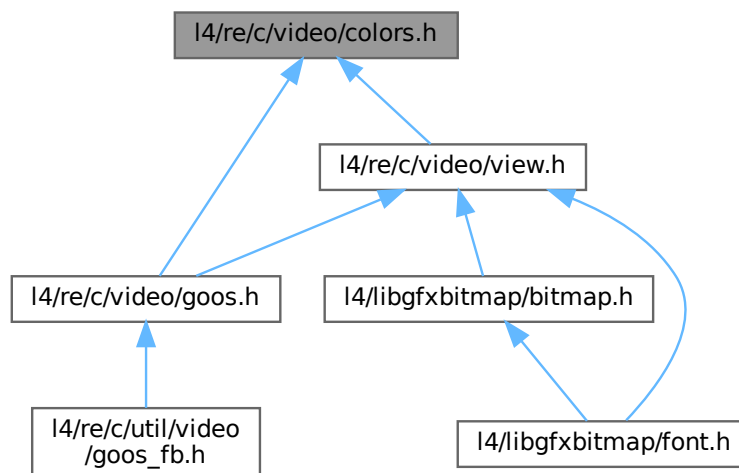
16.280 l4/re/c/video/colors.h File Reference

```
#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.

16.280.1 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](#).

16.281 colors.h

[Go to the documentation of this file.](#)

```

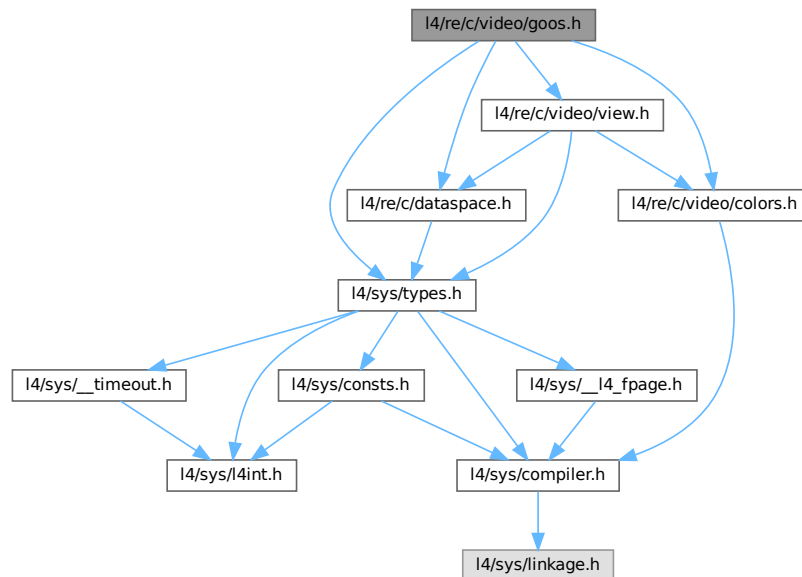
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the 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

```

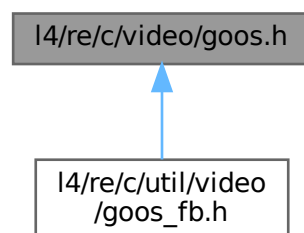
16.282 I4/re/c/video/goos.h File Reference

```
#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.

16.282.1 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](#).

16.283 goos.h

[Go to the documentation of this file.](#)

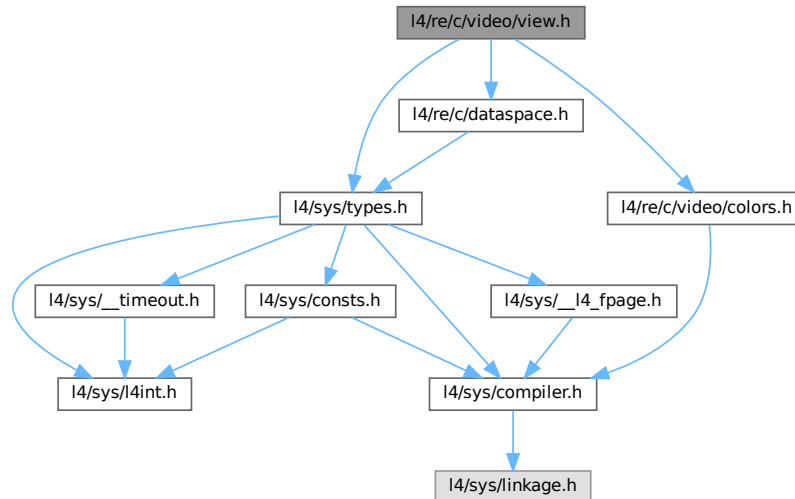
```

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

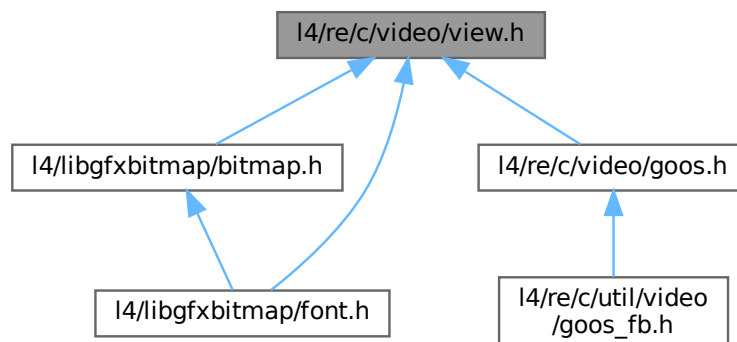
```

16.284 I4/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 `I4re_video_view_info_t`
View information structure.
- struct `I4re_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.

16.284.1 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 [view.h](#).

16.285 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

```

16.286 console

```

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/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

```

16.287 l4/re/dataspace File Reference

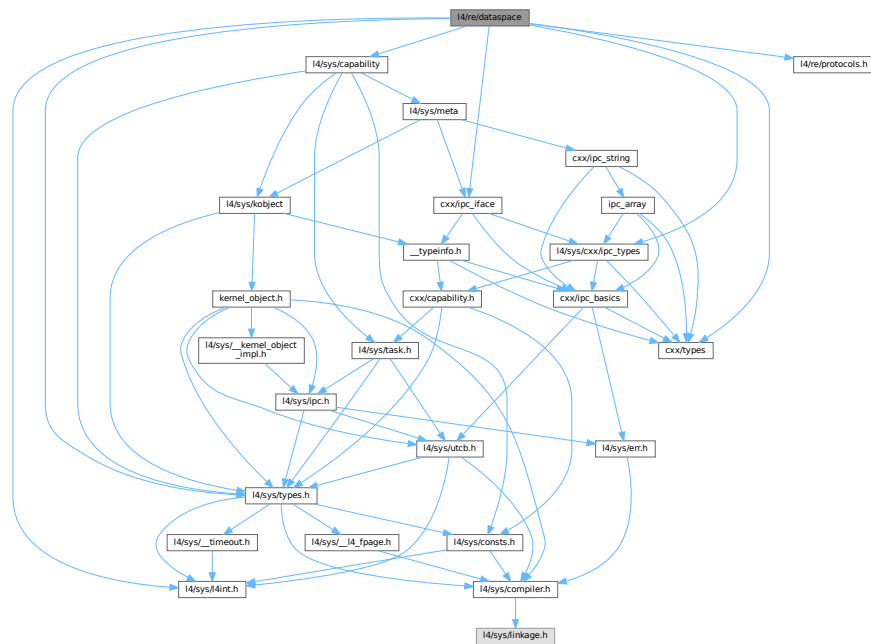
Dataspace interface.

```

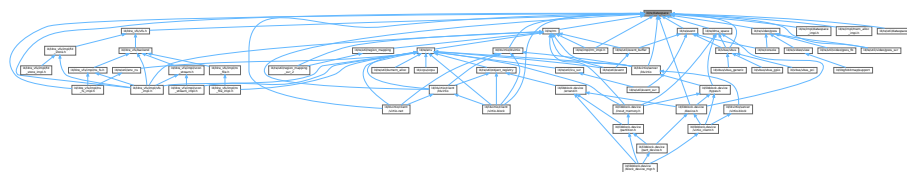
#include <l4/sys/types.h>
#include <l4/sys/l4int.h>
#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/types>

```

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.

16.287.1 Detailed Description

Dataspace interface.

Definition in file [dataspace](#).

16.288 dataspace

[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  *               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 #pragma once
00029
00030 #include <l4/sys/types.h>
00031 #include <l4/sys/l4int.h>
00032 #include <l4/sys/capability>
00033 #include <l4/re/protocols.h>
00034 #include <l4/sys/cxx/ipc_types>
00035 #include <l4/sys/cxx/ipc_iface>
00036 #include <l4/sys/cxx/types>
00037
00038 namespace L4Re
00039 {
00040
00041     // MISSING:
00042     // * size support in map, mapped size in reply
00043
00060     class L4_EXPORT Dataspace :
00061     public L4::Kobject_t<Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE,
00062                          L4::Type_info::Demand_t<1> >
00063     {
00064     public:
00065
00067         struct F
00068         {
00069             enum
00070             {
00071                 Caching_shift = 4,
00072             };
00073
00080             enum Flags
00081             {
00083                 R   = L4_FPAGE_RO,
00085                 Ro  = L4_FPAGE_RO,
00087                 RW  = L4_FPAGE_RW,
00089                 W   = L4_FPAGE_W,
00091                 X   = L4_FPAGE_X,
00093                 RX  = L4_FPAGE_RX,
00095                 RWX = L4_FPAGE_RWX,
00097                 Rights_mask = 0x0f,
00098
00100                 Normal      = 0x00,
00102                 Cacheable   = Normal,
00104                 Bufferable   = 0x10,
00106                 Uncacheable = 0x20,
00108                 Caching_mask = 0x30,
00109             };
00110
00111             L4_TYPES_FLAGS_OPS_DEF(Flags);
00112         };
00113
00114         struct Flags : L4::Types::Flags_ops_t<Flags>
00115         {
00116             unsigned long raw;
00117             Flags() = default;
00118             explicit constexpr Flags(unsigned long f) : raw(f) {}
00119             constexpr Flags(F::Flags f) : raw(f) {}
00120             constexpr bool r() const { return raw & L4_FPAGE_RO; }
00121             constexpr bool w() const { return raw & L4_FPAGE_W; }
00122             constexpr bool x() const { return raw & L4_FPAGE_X; }

```



```

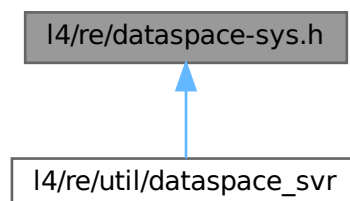
00123
00124     constexpr unsigned long fpage_rights() const
00125     { return raw & 0xf; }
00126 };
00127
00128 typedef l4_uint64_t Size;
00129 typedef l4_uint64_t Offset;
00130 typedef l4_uint64_t Map_addr;
00131
00135 struct Stats
00136 {
00137     Size size;
00138     Flags flags;
00139 };
00140
00141
00164 long map(Offset offset, Flags flags, Map_addr local_addr,
00165         Map_addr min_addr, Map_addr max_addr) const noexcept;
00166
00192 long map_region(Offset offset, Flags flags,
00193                Map_addr min_addr, Map_addr max_addr) const noexcept;
00194
00212 L4_RPC(long, clear, (Offset offset, Size size));
00213
00233 L4_RPC(long, allocate, (Offset offset, Size size));
00234
00253 L4_RPC(long, copy_in, (Offset dst_offs, L4::Ipc::Cap<Dataspace> src,
00254                      Offset src_offs, Size size));
00255
00261 Size size() const noexcept;
00262
00271 Flags flags() const noexcept;
00272
00281 L4_RPC(long, info, (Stats *stats));
00282
00283 L4_RPC_NF(long, map, (Offset offset, Map_addr spot,
00284                    Flags flags, L4::Ipc::Rcv_fpage r,
00285                    L4::Ipc::Snd_fpage &fp));
00286
00287 private:
00288
00289     long __map(Offset offset, unsigned char *order, Flags flags,
00290               Map_addr local_addr) const noexcept;
00291
00292 public:
00293     typedef L4::Typeid::Rpc<map_t, clear_t, info_t, copy_in_t,
00294                      allocate_t> Rpc;
00295
00296 };
00297
00298 }
00299

```

16.289 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.

16.289.1 Detailed Description

Dataspace protocol defintion.

Definition in file [dataspace-sys.h](#).

16.290 dataspace-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 Dataspace_
00028     {
00034         enum Opcodes { Map, Clear, Stats, Copy, Take, Release, Allocate };
00035     };
00036 };
00037

```

16.291 l4/re/dma_space File Reference

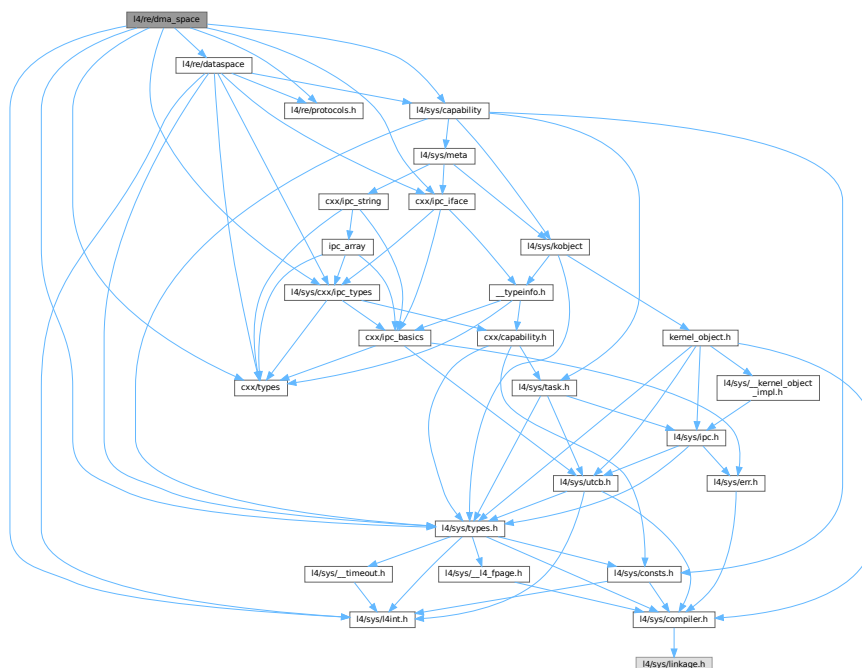
```

#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>

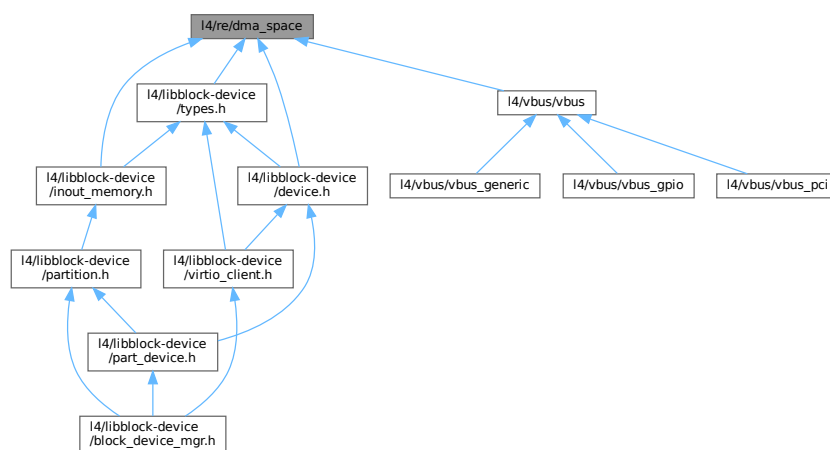
```

```
#include <l4/sys/cxx/ipc_iface>
```

Include dependency graph for dma_space:



This graph shows which files directly or indirectly include this file:



Data Structures

- class `L4Re::Dma_space`
Managed DMA Address Space.

Namespaces

- namespace **L4Re**
L4Re C++ Interfaces.

16.292 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(

```

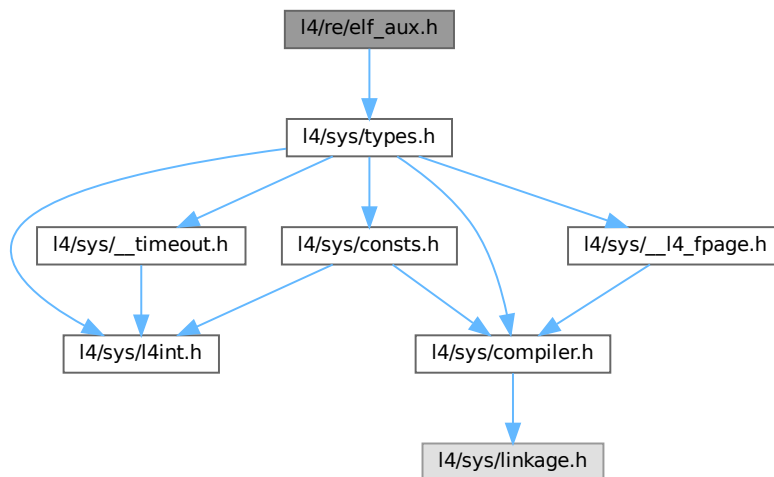
```

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 }
```

16.293 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](#) }

16.293.1 Detailed Description

Auxiliary information for binaries.

Definition in file [elf_aux.h](#).

16.294 elf_aux.h

[Go to the documentation of this file.](#)

```

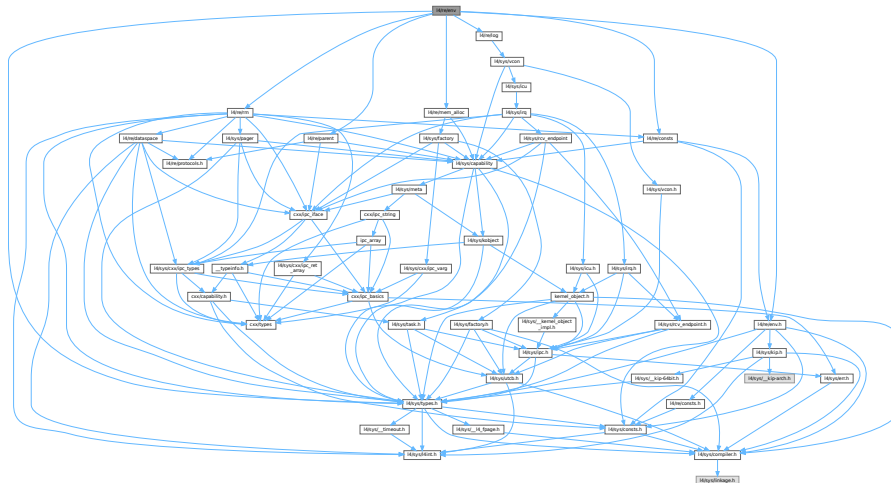
00001
00005 /*
00006  * (c) 2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 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,

```

16.295 I4/re/env File Reference

```
#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 };

```

16.297 I4/re/env.h File Reference

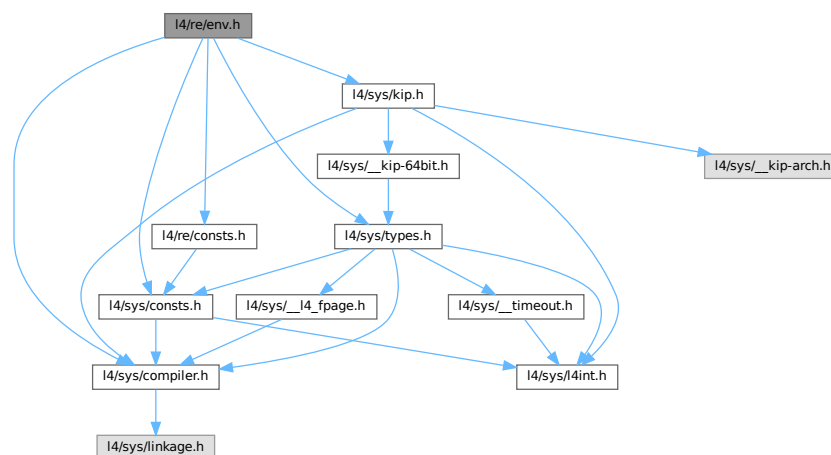
Environment interface.

```

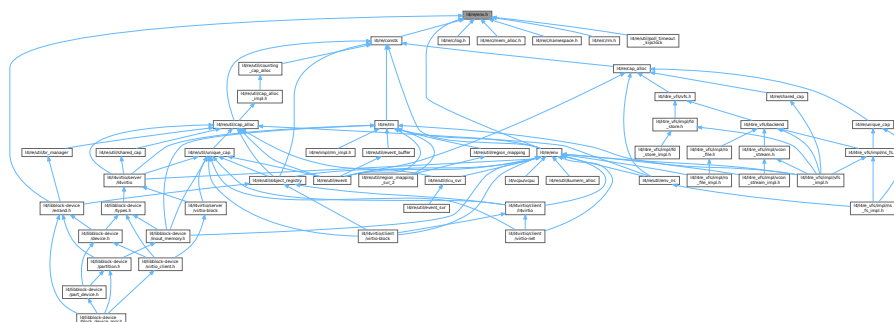
#include <l4/sys/consts.h>
#include <l4/sys/types.h>
#include <l4/sys/kip.h>
#include <l4/sys/compiler.h>
#include <l4/re/consts.h>

```

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.

16.297.1 Detailed Description

Environment interface.

Definition in file [env.h](#).

16.297.2 Typedef Documentation

16.297.2.1 [l4re_env_t](#)

```
typedef struct l4re\_env\_t l4re\_env\_t
```

Initial environment data structure.

See also

[Initial environment](#)

16.298 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(~0) {}
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

```

16.299 l4/re/error_helper File Reference

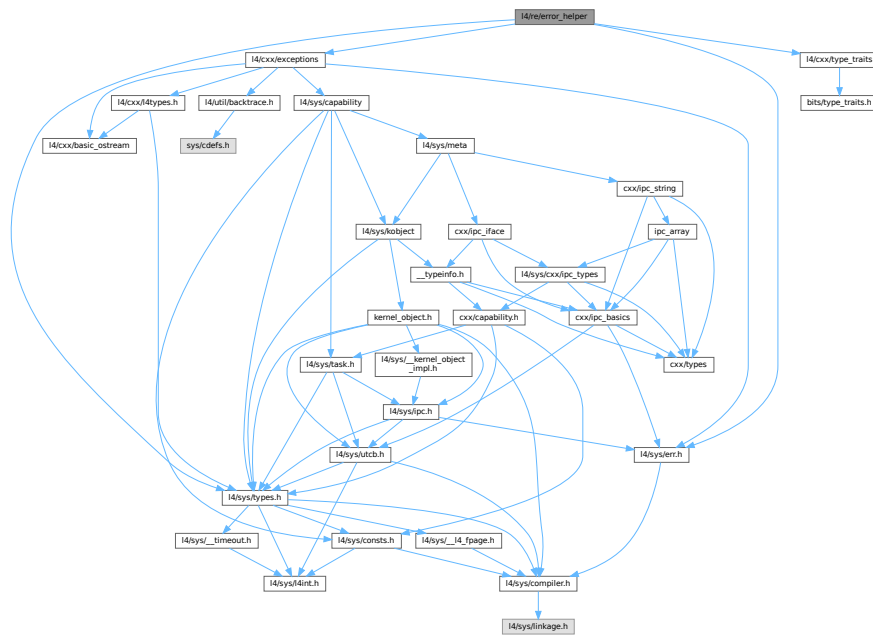
Error helper.

```

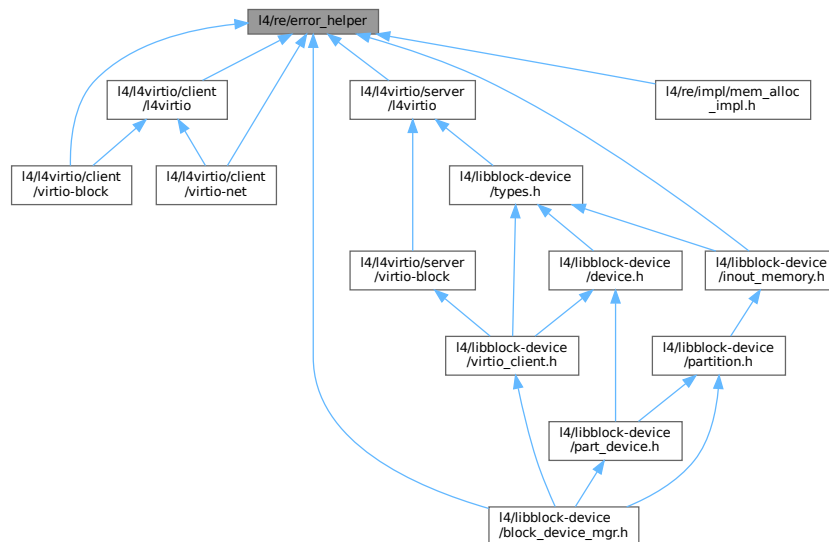
#include <l4/sys/types.h>
#include <l4/cxx/exceptions>
#include <l4/cxx/type_traits>
#include <l4/sys/err.h>

```

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.

16.299.1 Detailed Description

Error helper.

Definition in file [error_helper](#).

16.300 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 }
```

```

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
00073     return err;
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 }

```

16.301 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 };

```

16.302 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_KPPPLUS = 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_SCROLLLL         = 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 };

```

16.303 l4/re/impl/dataspace_impl.h File Reference

Dataspace client stub implementation.

```

#include <l4/re/dataspace>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/cxx/consts>

```



```

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
00032 namespace L4Re {
00033
00034     long
00035     Dataspace::__map(Dataspace::Offset offset, unsigned char *size,
00036                     Dataspace::Flags flags,
00037                     Dataspace::Map_addr local_addr) const noexcept
00038     {
00039         Map_addr spot = local_addr & ~(~0ULL << l4_umword_t(*size));
00040         Map_addr base = local_addr & (~0ULL << l4_umword_t(*size));
00041         L4::Ipc::Rcv_fpage r;
00042         r = L4::Ipc::Rcv_fpage::mem(base, *size, 0);
00043
00044         L4::Ipc::Snd_fpage fp;
00045         long err = map_t::call(c(), offset, spot, flags, r, fp, l4_utcb());
00046         if (L4_UNLIKELY(err < 0))
00047             return err;
00048
00049         *size = fp.rcv_order();
00050         return err;
00051     }
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) const noexcept
00058 {
00059     min_addr = L4::trunc_page(min_addr);
00060     max_addr = L4::round_page(max_addr);
00061     unsigned char order = L4_LOG2_PAGESIZE;
00062
00063     long err = 0;
00064
00065     while (min_addr < max_addr)
00066     {
00067         unsigned char order_mapped;
00068         order_mapped = order
00069             = L4::max_order(order, min_addr, min_addr, max_addr, min_addr);
00070
00071         err = __map(offset, &order_mapped, flags, min_addr);
00072         if (L4_UNLIKELY(err < 0))
00073             return err;
00074
00075         if (order > order_mapped)
00076             order = order_mapped;
00077
00078         min_addr += Map_addr(1) << order;
00079         offset += Map_addr(1) << order;
00080
00081         if (min_addr >= max_addr)
00082             return 0;
00083
00084         while (min_addr != L4::trunc_order(min_addr, order)
00085              || max_addr < L4::round_order(min_addr + 1, order))
00086             --order;
00087     }
00088
00089     return 0;
00090 }
00091
00092 long
00093 Dataspace::map(Dataspace::Offset offset, Dataspace::Flags flags,
00094                Dataspace::Map_addr local_addr,
00095                Dataspace::Map_addr min_addr,
00096                Dataspace::Map_addr max_addr) const noexcept
00097 {
00098     min_addr = L4::trunc_page(min_addr);
00099     max_addr = L4::round_page(max_addr);
00100     local_addr = L4::trunc_page(local_addr);

```


16.305.1 Detailed Description

Memory allocator client stub implementation.

Definition in file [mem_alloc_impl.h](#).

16.306 mem_alloc_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/mem_alloc>
00024 #include <l4/re/mem_alloc-sys.h>
00025 #include <l4/re/dataspace>
00026 #include <l4/re/error_helper>
00027
00028 #include <l4/sys/factory>
00029
00030
00031 namespace L4Re
00032 {
00033
00034     long
00035     Mem_alloc::alloc(long size,
00036                     L4::Cap<Dataspace> mem, unsigned long flags,
00037                     unsigned long align) const noexcept
00038     {
00039         L4::Cap<L4::Factory> f(cap());
00040         return l4_error(f->create(mem, L4Re::Dataspace::Protocol)
00041                        « l4_mword_t(size)
00042                        « l4_umword_t(flags)
00043                        « l4_umword_t(align));
00044     }
00045
00046 };

```

16.307 l4/re/impl/namespace_impl.h File Reference

Namespace client stub implementation.

```

#include <l4/re/namespace>
#include <l4/util/util.h>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/assert.h>

```



```

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
00083     if (L4_UNLIKELY(timeout < 0))
00084         return -L4_EINVAL;
00085
00086     long ret;
00087     long rem = timeout;
00088     long to = 0;
00089
00090     if (rem)
00091         to = 10;
00092
00093     do
00094     {
00095         ret = _query(name, len, target, local_id, iterate);
00096
00097         if (ret >= 0)
00098             return ret;
00099
00100         if (L4_UNLIKELY(ret != -L4_EAGAIN))
00101             return ret;
00102
00103         if (rem == to)
00104             return ret;
00105
00106         l4_sleep(to);
00107     }

```


16.309.1 Detailed Description

Region map client stub implementation.

Definition in file [rm_impl.h](#).

16.310 rm_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/rm>
00024 #include <l4/re/dataspace>
00025
00026 #include <l4/sys/cxx/ipc_client>
00027
00028 #include <l4/sys/task>
00029 #include <l4/sys/err.h>
00030
00031 L4_RPC_DEF(L4Re::Rm::reserve_area);
00032 L4_RPC_DEF(L4Re::Rm::free_area);
00033 L4_RPC_DEF(L4Re::Rm::attach);
00034 L4_RPC_DEF(L4Re::Rm::detach);
00035 L4_RPC_DEF(L4Re::Rm::get_regions);
00036 L4_RPC_DEF(L4Re::Rm::get_areas);
00037 L4_RPC_DEF(L4Re::Rm::find);
00038
00039 namespace L4Re
00040 {
00041
00042     long
00043     Rm::attach(l4_addr_t *start, unsigned long size, Rm::Flags flags,
00044               L4::Ipc::Cap<Dataspace> mem, Rm::Offset offs,
00045               unsigned char align) const noexcept
00046     {
00047         if ((flags & F::Rights_mask) == Flags(0) || (flags & F::Reserved))
00048             mem = L4::Ipc::Cap<L4Re::Dataspace>();
00049
00050         long e = attach_t::call(c(), start, size, flags, mem, offs, align, mem.cap().cap());
00051         if (e < 0)
00052             return e;
00053
00054         if (flags & F::Eager_map)
00055             e = mem.cap()->map_region(offs, map_flags(flags), *start, *start + size);
00056
00057         return e;
00058     }
00059
00060     int
00061     Rm::detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
00062               L4::Cap<L4::Task> task, unsigned flags) const noexcept
00063     {
00064         l4_addr_t rstart = 0, rsize = 0;
00065         l4_cap_idx_t mem_cap = L4_INVALID_CAP;
00066         long e = detach_t::call(c(), start, size, flags, rstart, rsize, mem_cap);
00067         if (L4_UNLIKELY(e < 0))
00068             return e;
00069
00070         if (mem)
00071             *mem = L4::Cap<L4Re::Dataspace>(mem_cap);
00072     }

```

```

00073     if (!task.is_valid())
00074         return e;
00075
00076     rsize = l4_round_page(rsize);
00077     unsigned order = L4_LOG2_PAGESIZE;
00078     unsigned long sz = (1UL << order);
00079     for (unsigned long p = rstart; rsize; p += sz, rsize -= sz)
00080     {
00081         while (sz > rsize)
00082         {
00083             --order;
00084             sz >>= 1;
00085         }
00086
00087         for (;;)
00088         {
00089             unsigned long m = sz << 1;
00090             if (m > rsize)
00091                 break;
00092
00093             if (p & (m - 1))
00094                 break;
00095
00096             ++order;
00097             sz <<= 1;
00098         }
00099
00100         task->unmap(l4_fpage(p, order, L4_FPAGE_RWX),
00101                   L4_FP_ALL_SPACES);
00102     }
00103
00104     return e;
00105 }
00106 }

```

16.311 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::Ipc::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::Ipc::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::Ipc::In_out<l4_mword_t *> current_id,
00098                                         L4::Ipc::String<char> &name));
00099
00100 typedef L4::Typeid::Rpcs<acquire_t, release_t, next_lock_info_t> Rpcs;
00101 };

```



```
00102
00103 }
```

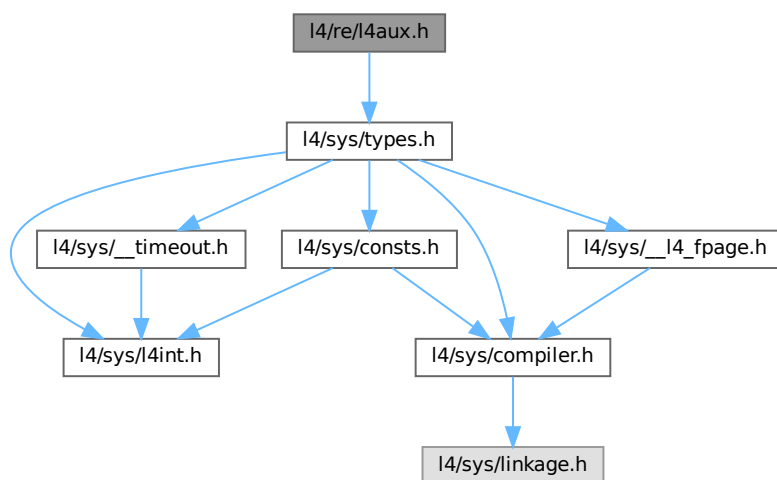
16.312 inhibitor-sys.h

```
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 }
```

16.313 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.

16.313.1 Detailed Description

Auxiliary definitions.

Definition in file [l4aux.h](#).

16.314 l4aux.h

[Go to the documentation of this file.](#)

```

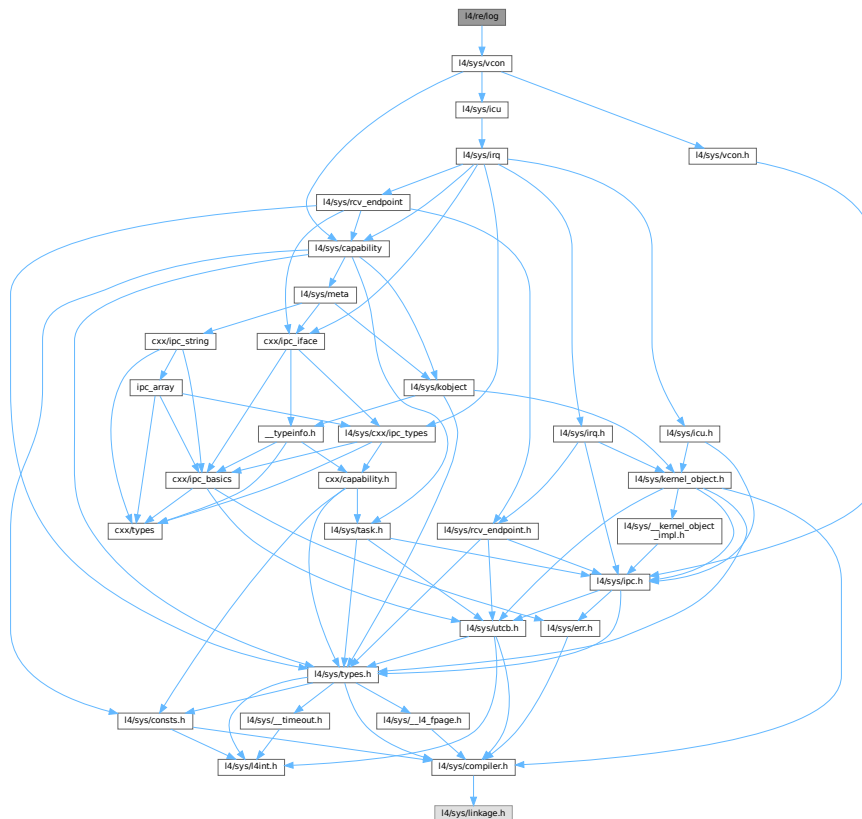
00001 #pragma once
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  * Björn Döbel <doebel@os.inf.tu-dresden.de>
00010  * economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part 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 } l4re_aux_t;
00058

```

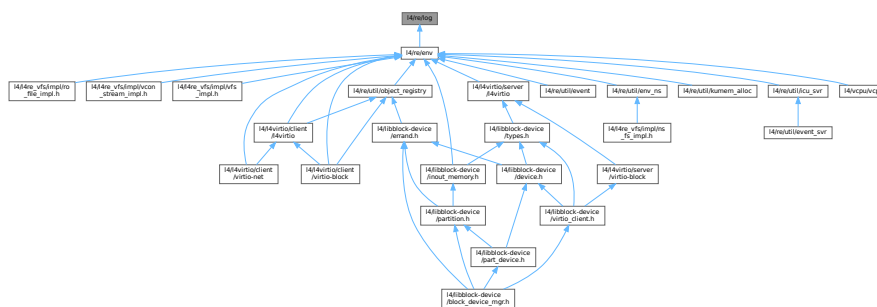
16.315 l4/re/log File Reference

Log interface.

```
#include <14/sys/vcon>
Include dependency graph for log:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- class `L4Re::Log`
Log interface class.

Namespaces

- namespace **L4Re**
L4Re C++ Interfaces.

16.315.1 Detailed Description

Log interface.

Definition in file [log](#).

16.316 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 }
```

16.317 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.

16.317.1 Detailed Description

Log protocol definition.

Definition in file [log-sys.h](#).

16.318 log-sys.h

[Go to the documentation of this file.](#)

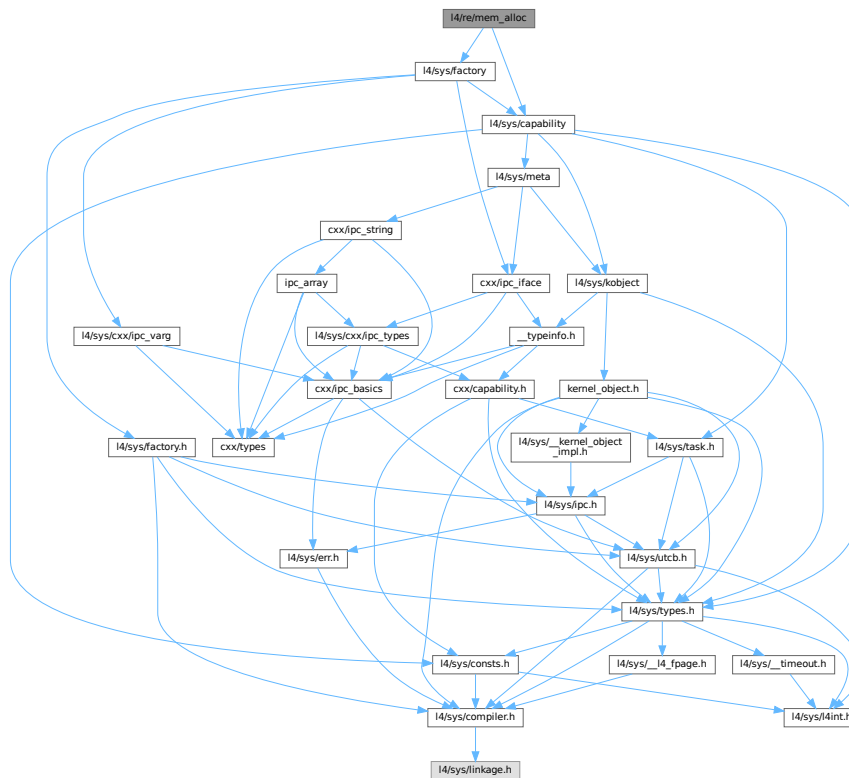
```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
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 };
```

16.319 l4/re/mem_alloc File Reference

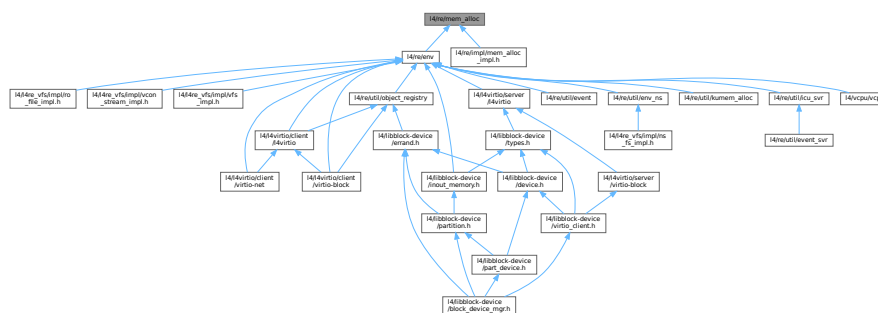
Memory allocator interface.

```
#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.

16.319.1 Detailed Description

Memory allocator interface.

Definition in file [mem_alloc](#).

16.320 mem_alloc

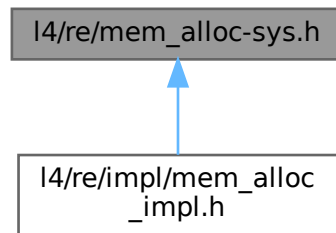
[Go to the documentation of this file.](#)

```
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     };
00077
00104     long alloc(long size, L4::Cap<Dataspace> mem,
00105                 unsigned long flags = 0, unsigned long align = 0) const noexcept;
00106 };
00107
00108 };
```

16.321 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.

16.321.1 Detailed Description

Memory allocator protocol definitions.

Definition in file [mem_alloc-sys.h](#).

16.322 mem_alloc-sys.h

[Go to the documentation of this file.](#)

```

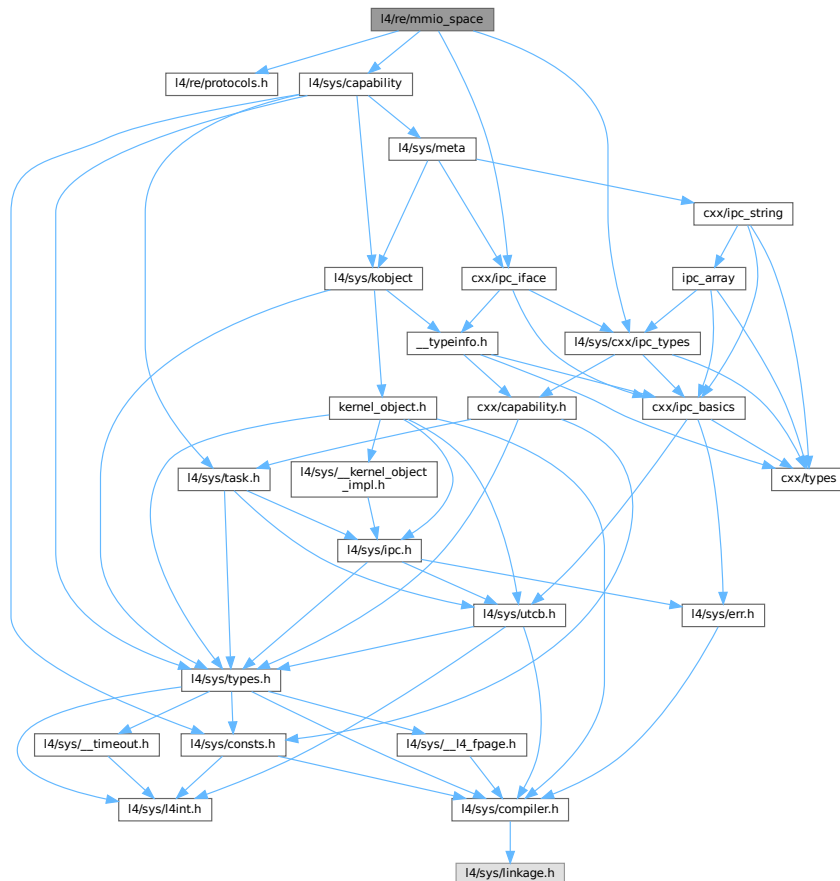
00001
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * 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 };

```


16.323 l4/re/mmio_space File Reference

Interface definition to emit MMIO-like accesses via IPC.

```
#include <l4/re/protocols.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>
Include dependency graph for mmio_space:
```



Data Structures

- struct [L4Re::Mmio_space](#)
Interface for memory-like address space accessible via IPC.

Namespaces

- namespace [L4Re](#)
L4Re C++ Interfaces.

16.323.1 Detailed Description

Interface definition to emit MMIO-like accesses via IPC.

Definition in file [mmio_space](#).

16.324 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> Rpc;
00094 };
00095
00096 }
```

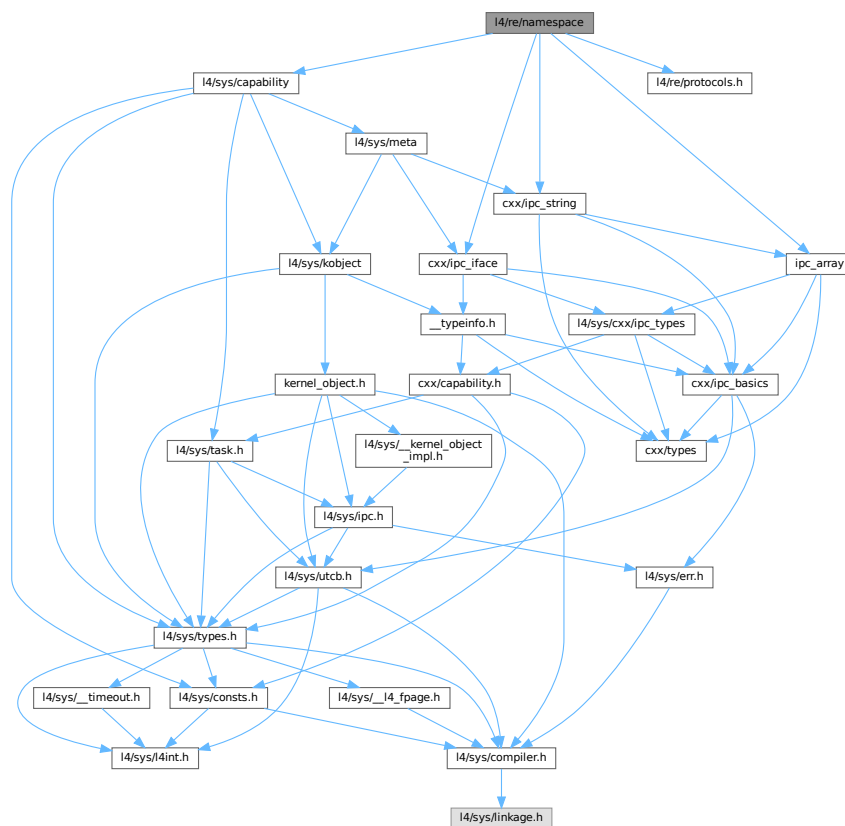
16.325 l4/re/namespace File Reference

Namespace interface.

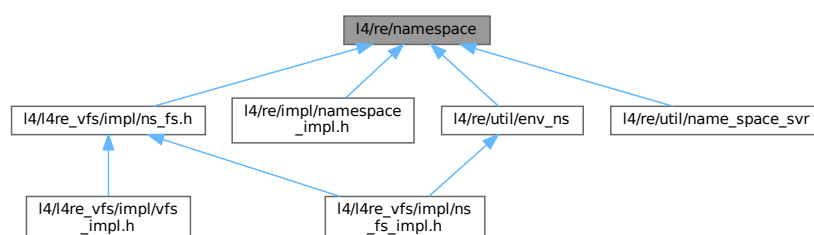
```
#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_array>
```

```
#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.

16.325.1 Detailed Description

Namespace interface.

Definition in file [namespace](#).

16.326 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 };

```

16.327 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.

16.327.1 Detailed Description

Namespace protocol definitions.

Definition in file [namespace-sys.h](#).

16.328 namespace-sys.h

[Go to the documentation of this file.](#)

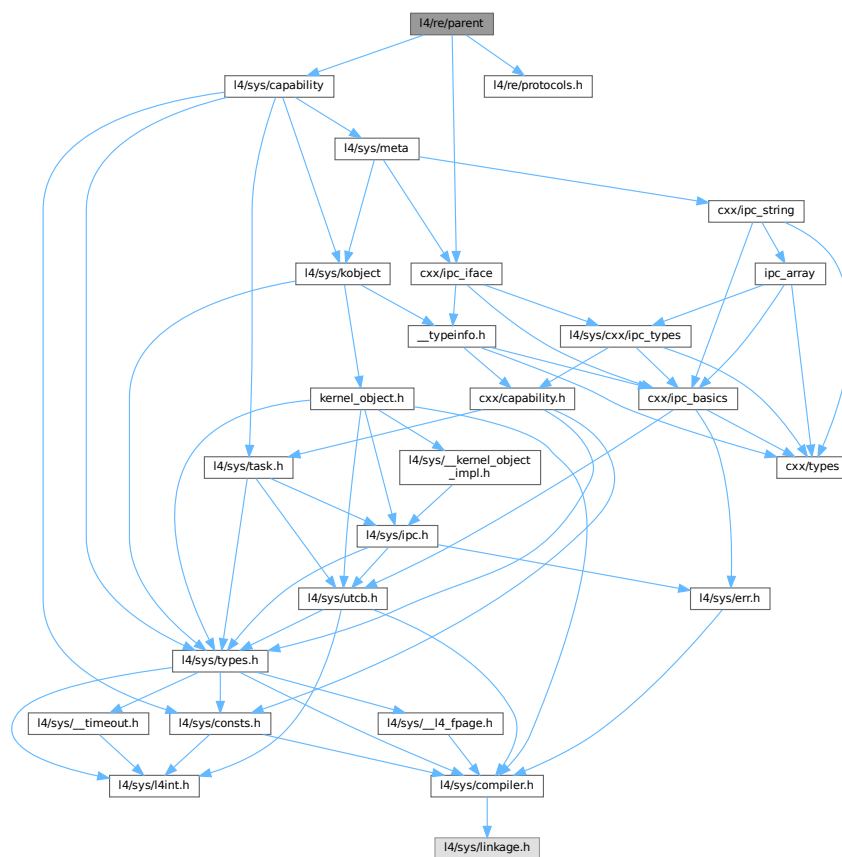
```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
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 };
```

16.329 l4/re/parent File Reference

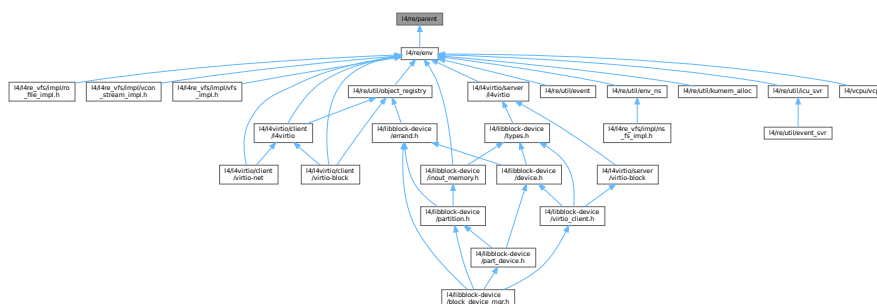
Parent interface.

```
#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.

16.329.1 Detailed Description

Parent interface.

Definition in file [parent](#).

16.330 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
```

16.331 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.

16.331.1 Detailed Description

Parent protocol definition.

Definition in file [parent-sys.h](#).

16.332 parent-sys.h

[Go to the documentation of this file.](#)

```

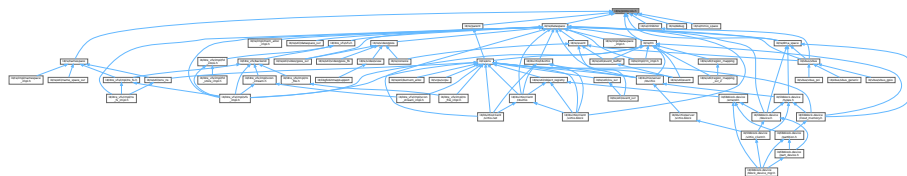
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * 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 };

```

16.333 l4/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 }

16.333.1 Detailed Description

[L4Re](#) Protocol Constants (C version)

Definition in file [protocols.h](#).

16.333.2 Enumeration Type Documentation

16.333.2.1 L4re_protocols

enum [L4re_protocols](#)

Enumerator

L4RE_PROTO_DATASPACE	ID for L4Re::Dataspace RPCs
L4RE_PROTO_NAMESPACE	ID for L4Re::Namespace RPCs
L4RE_PROTO_PARENT	ID for L4Re::Parent RPCs
L4RE_PROTO_GOOS	ID for L4Re::Video::Goos RPCs
L4RE_PROTO_RSVD_1	Reserved ID
L4RE_PROTO_RM	ID for L4Re::Rm RPCs
L4RE_PROTO_EVENT	ID for L4Re::Event RPCs
L4RE_PROTO_INHIBITOR	ID for L4Re::Inhibitor RPCs
L4RE_PROTO_DMA_SPACE	ID for L4Re::Dma_space RPCs
L4RE_PROTO_MMIO_SPACE	ID for L4Re::Mmio_space
L4RE_PROTO_DEBUG	ID for debugging RPCs

Definition at line 30 of file [protocols.h](#).

16.334 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

```

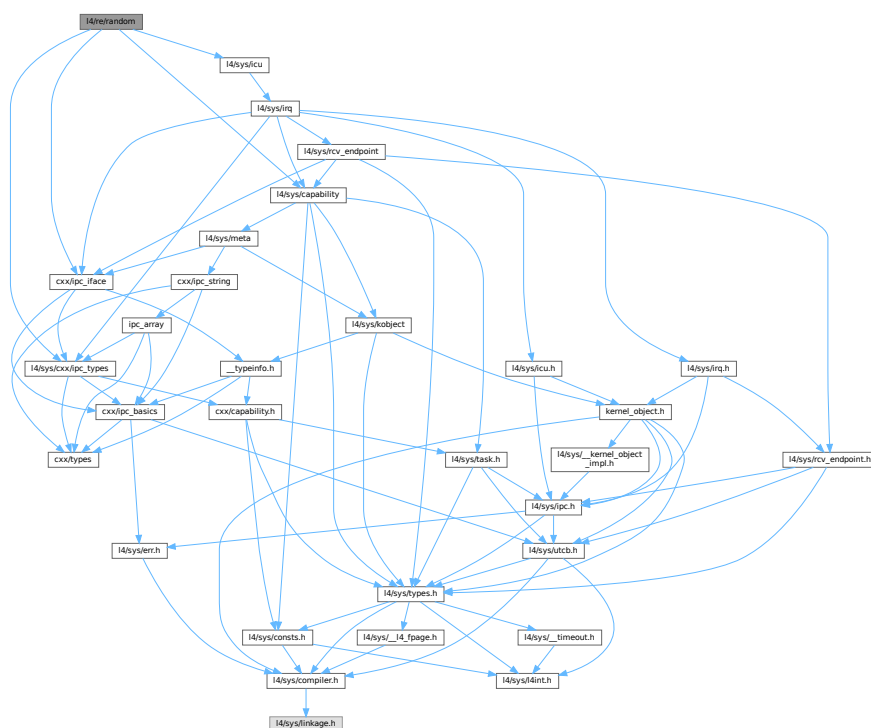
```
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00030 enum L4re_protocols
00031 {
00032     L4RE_PROTO_DATASPACE = 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
```

16.335 I4/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>
```

Include dependency graph for random:



Data Structures

- struct L4Re::Random

Low-bandwidth interface for random number generators.

Namespaces

- namespace [L4Re](#)
L4Re C++ Interfaces.

16.335.1 Detailed Description

Random number generator interface definition.

Definition in file [random](#).

16.336 random

[Go to the documentation of this file.](#)

```
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  *
00007  */
00012 #pragma once
00013
00014 #include <l4/sys/capability>
00015 #include <l4/sys/cxx/ipc_types>
00016 #include <l4/sys/cxx/ipc_iface>
00017 #include <l4/sys/icu>
00018
00019 namespace L4Re
00020 {
00021
00033 struct L4_EXPORT Random
00034 : public L4::Kobject_t<Random, L4::Icu>
00035 {
00060     L4_INLINE_RPC(long, get_random, (l4_size_t size,
00061                                     L4::Ipc::Array<char, unsigned long> *buffer));
00062
00063     typedef L4::Typeid::Rpc<get_random_t> Rpc;
00064 };
00065
00066 } // namespace
```

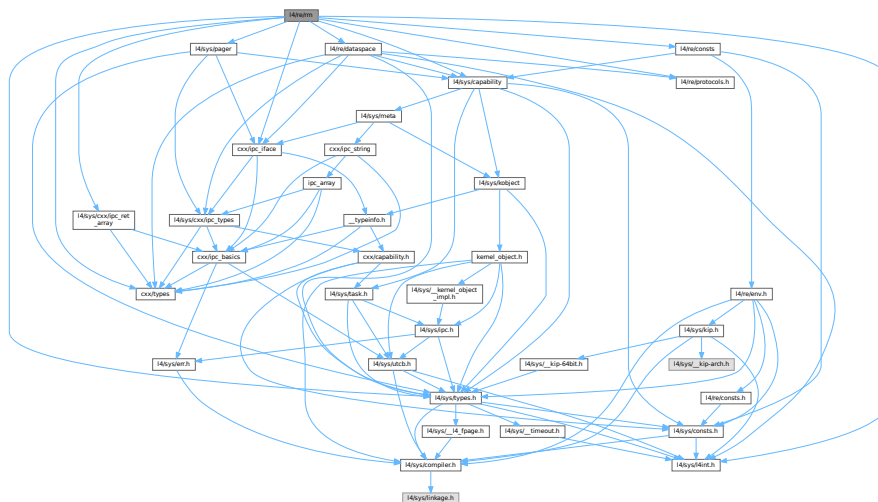
16.337 l4/re/rm File Reference

Region mapper interface.

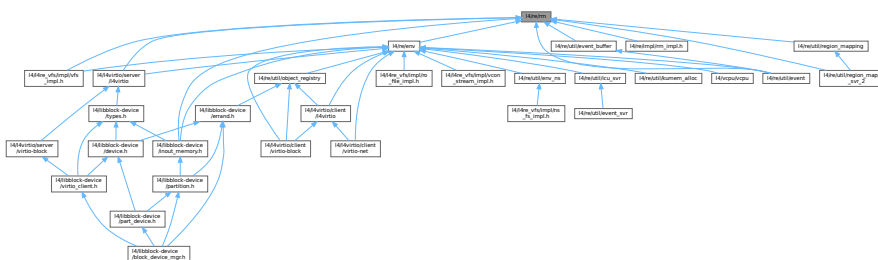
```
#include <l4/sys/types.h>
#include <l4/sys/l4int.h>
#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/pager>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_ret_array>
#include <l4/sys/cxx/types>
#include <l4/re/consts>
```

```
#include <l4/re/dataspace>
```

Include dependency graph for rm:



This graph shows which files directly or indirectly include this file:



Data Structures

- class `L4Re::Rm`
Region map.
- struct `L4Re::Rm::F`
Rm flags definitions.
- class `L4Re::Rm::Unique_region< T >`
Unique region.
- struct `L4Re::Rm::Range`
A range of virtual addresses.

Namespaces

- namespace **L4Re**
L4Re C++ Interfaces.

16.337.1 Detailed Description

Region mapper interface.

Definition in file [rm](#).

16.338 rm

[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  *      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 #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
00085 class L4_EXPORT Rm :
00086     public L4::Kobject_t<Rm, L4::Pager, L4RE_PROTO_RM,
00087         L4::Type_info::Demand_t<1> >
00088 {
00089 public:
00090     typedef L4Re::Dataspace::Offset Offset;
00091
00093     enum Detach_result
00094     {
00095         Detached_ds = 0,
00096         Kept_ds = 1,
00097         Split_ds = 2,
00098         Detach_result_mask = 3,
00099
00100         Detach_again = 4,
00101     };
00102
00105     enum Region_flag_shifts
00106     {
00108         Caching_shift = Dataspace::F::Caching_shift,
00109     };
00110
00112     struct F
00113     {
00115         enum Attach_flags : l4_uint32_t
00116         {
00118             Search_addr = 0x20000,
00120             In_area = 0x40000,
00122             Eager_map = 0x80000,
00124             Attach_mask = 0xf0000,
00125         };
00126     };
00127 }
```

```

00126
00127     L4_TYPES_FLAGS_OPS_DEF(Attach_flags);
00128
00129     enum Region_flags : l4_uint16_t
00130     {
00132         Rights_mask = 0x0f,
00134         R           = Dataspace::F::R,
00136         W           = Dataspace::F::W,
00138         X           = Dataspace::F::X,
00140         RW          = Dataspace::F::RW,
00142         RX          = Dataspace::F::RX,
00144         RWX         = Dataspace::F::RWX,
00145
00147         Detach_free = 0x200,
00149         Pager       = 0x400,
00151         Reserved    = 0x800,
00152
00153
00155         Caching_mask = Dataspace::F::Caching_mask,
00157         Cache_normal = Dataspace::F::Normal,
00159         Cache_buffered = Dataspace::F::Bufferable,
00161         Cache_uncached = Dataspace::F::Uncacheable,
00162
00164         Ds_map_mask   = 0xff,
00165
00167         Region_flags_mask = 0xffff,
00168     };
00169
00170     L4_TYPES_FLAGS_OPS_DEF(Region_flags);
00171
00172     friend constexpr Dataspace::Flags map_flags(Region_flags rf)
00173     {
00174         return Dataspace::Flags((l4_uint16_t)rf & Ds_map_mask);
00175     }
00176
00177     struct Flags : L4::Types::Flags_ops_t<Flags>
00178     {
00179         l4_uint32_t raw;
00180         Flags() = default;
00181         explicit constexpr Flags(l4_uint32_t f) : raw(f) {}
00182         constexpr Flags(Attach_flags rf) : raw((l4_uint32_t)rf) {}
00183         constexpr Flags(Region_flags rf) : raw((l4_uint32_t)rf) {}
00184
00185         friend constexpr Dataspace::Flags map_flags(Flags f)
00186         {
00187             return Dataspace::Flags(f.raw & Ds_map_mask);
00188         }
00189
00190         constexpr Region_flags region_flags() const
00191         {
00192             return Region_flags(raw & Region_flags_mask);
00193         }
00194
00195         constexpr Attach_flags attach_flags() const
00196         {
00197             return Attach_flags(raw & Attach_mask);
00198         }
00199
00200         constexpr bool r() const { return raw & L4_FPAGE_RO; }
00201         constexpr bool w() const { return raw & L4_FPAGE_W; }
00202         constexpr bool x() const { return raw & L4_FPAGE_X; }
00203         constexpr unsigned cap_rights() const
00204         { return w() ? L4_CAP_FPAGE_RW : L4_CAP_FPAGE_RO; }
00205     };
00206
00207     friend constexpr Flags operator | (Region_flags l, Attach_flags r)
00208     { return Flags(l) | Flags(r); }
00209
00210     friend constexpr Flags operator | (Attach_flags l, Region_flags r)
00211     { return Flags(l) | Flags(r); }
00212 };
00213
00214 using Attach_flags = F::Attach_flags;
00215 using Region_flags = F::Region_flags;
00216 using Flags = F::Flags;
00217
00219     enum Detach_flags
00220     {
00230         Detach_exact = 1,
00240         Detach_overlap = 2,
00241
00249         Detach_keep = 4,
00250     };
00251
00278     long reserve_area(l4_addr_t *start, unsigned long size,
00279                      Flags flags = Flags(0),
00280                      unsigned char align = L4_PAGESHIFT) const noexcept

```

```

00281 { return reserve_area_t::call(c(), start, size, flags, align); }
00282
00283 L4_RPC_NF(long, reserve_area, (L4::Ipc::In_out<l4_addr_t *> start,
00284                               unsigned long size,
00285                               Flags flags,
00286                               unsigned char align));
00287
00303 template< typename T >
00304 long reserve_area(T **start, unsigned long size,
00305                  Flags flags = Flags(0),
00306                  unsigned char align = L4_PAGESHIFT) const noexcept
00307 { return reserve_area_t::call(c(), (l4_addr_t*)start, size, flags, align); }
00308
00321 L4_RPC(long, free_area, (l4_addr_t addr));
00322
00323 L4_RPC_NF(long, attach, (L4::Ipc::In_out<l4_addr_t *> start,
00324                          unsigned long size, Flags flags,
00325                          L4::Ipc::Opt<L4::Ipc::Cap<Dataspace> > mem,
00326                          Offset offs, unsigned char align,
00327                          L4::Ipc::Opt<l4_cap_idx_t> client_cap));
00328
00329 L4_RPC_NF(long, detach, (l4_addr_t addr, unsigned long size, unsigned flags,
00330                          l4_addr_t &start, l4_addr_t &rsz,
00331                          l4_cap_idx_t &mem_cap));
00332
00383 long attach(l4_addr_t *start, unsigned long size, Flags flags,
00384             L4::Ipc::Cap<Dataspace> mem, Offset offs = 0,
00385             unsigned char align = L4_PAGESHIFT) const noexcept;
00386
00390 template< typename T >
00391 long attach(T **start, unsigned long size, Flags flags,
00392            L4::Ipc::Cap<Dataspace> mem, Offset offs = 0,
00393            unsigned char align = L4_PAGESHIFT) const noexcept
00394 {
00395     union X { l4_addr_t a; T* t; };
00396     X *x = reinterpret_cast<X*>(start);
00397     return attach(&x->a, size, flags, mem, offs, align);
00398 }
00399
00400 #if __cplusplus >= 201103L
00411 template< typename T >
00412 class Unique_region
00413 {
00414 private:
00415     T _addr;
00416     L4::Cap<Rm> _rm;
00417
00418 public:
00419     Unique_region(Unique_region const &) = delete;
00420     Unique_region &operator = (Unique_region const &) = delete;
00421
00425     Unique_region() noexcept
00426     : _addr(0), _rm(L4::Cap<Rm>::Invalid) {}
00427
00433     explicit Unique_region(T addr) noexcept
00434     : _addr(addr), _rm(L4::Cap<Rm>::Invalid) {}
00435
00442     Unique_region(T addr, L4::Cap<Rm> const &rm) noexcept
00443     : _addr(addr), _rm(rm) {}
00444
00450     Unique_region(Unique_region &&o) noexcept : _addr(o.get()), _rm(o._rm)
00451     { o.release(); }
00452
00458     Unique_region &operator = (Unique_region &&o) noexcept
00459     {
00460         if (&o != this)
00461         {
00462             if (_rm.is_valid())
00463                 _rm->detach(l4_addr_t(_addr), 0);
00464             _rm = o._rm;
00465             _addr = o.release();
00466         }
00467         return *this;
00468     }
00469
00475     ~Unique_region() noexcept
00476     {
00477         if (_rm.is_valid())
00478             _rm->detach(l4_addr_t(_addr), 0);
00479     }
00480
00486     T get() const noexcept
00487     { return _addr; }
00488
00494     T release() noexcept
00495     {
00496         _rm = L4::Cap<Rm>::Invalid;

```



```

00497     return _addr;
00498 }
00499
00500 void reset(T addr, L4::Cap<Rm> const &rm) noexcept
00501 {
00502     if (_rm.is_valid())
00503         _rm->detach(l4_addr_t(_addr), 0);
00504
00505     _rm = rm;
00506     _addr = addr;
00507 }
00508
00509 void reset() noexcept
00510 { reset(0, L4::Cap<Rm>::Invalid); }
00511
00512 bool is_valid() const noexcept
00513 { return _rm.is_valid(); }
00514
00515 T operator * () const noexcept { return _addr; }
00516
00517 T operator -> () const noexcept { return _addr; }
00518 };
00519
00520 template< typename T >
00521 long attach(Unique_region<T> *start, unsigned long size, Flags flags,
00522             L4::Ipc::Cap<Dataspace> mem, Offset offs = 0,
00523             unsigned char align = L4_PAGESHIFT) const noexcept
00524 {
00525     l4_addr_t addr = (l4_addr_t)start->get();
00526
00527     long res = attach(&addr, size, flags, mem, offs, align);
00528     if (res < 0)
00529         return res;
00530
00531     start->reset((T)addr, L4::Cap<Rm>(cap()));
00532     return res;
00533 }
00534 #endif
00535
00536 int detach(l4_addr_t addr, L4::Cap<Dataspace> *mem,
00537            L4::Cap<L4::Task> const &task = This_task) const noexcept;
00538
00539 int detach(void *addr, L4::Cap<Dataspace> *mem,
00540            L4::Cap<L4::Task> const &task = This_task) const noexcept;
00541
00542 int detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
00543            L4::Cap<L4::Task> const &task) const noexcept;
00544
00545 long find(l4_addr_t *addr, unsigned long *size, Offset *offset,
00546           L4Re::Rm::Flags *flags, L4::Cap<Dataspace> *m) noexcept
00547 { return find_t::call(c(), addr, size, flags, offset, m); }
00548
00549 L4_RPC_NF(long, find, (L4::Ipc::In_out<l4_addr_t *> addr,
00550                      L4::Ipc::In_out<unsigned long *> size,
00551                      L4Re::Rm::Flags *flags, Offset *offset,
00552                      L4::Ipc::As_value<L4::Cap<Dataspace> > *m));
00553
00554 struct Range
00555 {
00556     l4_addr_t start;
00557     l4_addr_t end;
00558 };
00559
00560 using Region = Range;
00561
00562 using Area = Range;
00563
00564 L4_RPC(long, get_regions, (l4_addr_t start, L4::Ipc::Ret_array<Range> regions));
00565
00566 L4_RPC(long, get_areas, (l4_addr_t start, L4::Ipc::Ret_array<Range> areas));
00567
00568 int detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
00569            L4::Cap<L4::Task> task, unsigned flags) const noexcept;
00570
00571 typedef L4::Typeid::Rpcs<attach_t, detach_t, find_t,
00572                          reserve_area_t, free_area_t,
00573                          get_regions_t, get_areas_t> Rpcs;
00574 };
00575
00576 inline int
00577 Rm::detach(l4_addr_t addr, L4::Cap<Dataspace> *mem,
00578            L4::Cap<L4::Task> const &task) const noexcept
00579 { return detach(addr, 1, mem, task, Detach_overlap); }
00580
00581 inline int
00582 Rm::detach(void *addr, L4::Cap<Dataspace> *mem,

```

```

00728         L4::Cap<L4::Task> const &task) const noexcept
00729 {   return detach((l4_addr_t)addr, 1, mem, task, Detach_overlap); }
00730
00731 inline int
00732 Rm::detach(l4_addr_t addr, unsigned long size, L4::Cap<Dataspace> *mem,
00733           L4::Cap<L4::Task> const &task) const noexcept
00734 {   return detach(addr, size, mem, task, Detach_exact); }
00735
00736 };

```

16.339 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.

16.339.1 Detailed Description

Region mapper protocol definitions.

Definition in file [rm-sys.h](#).

16.340 rm-sys.h

[Go to the documentation of this file.](#)

```

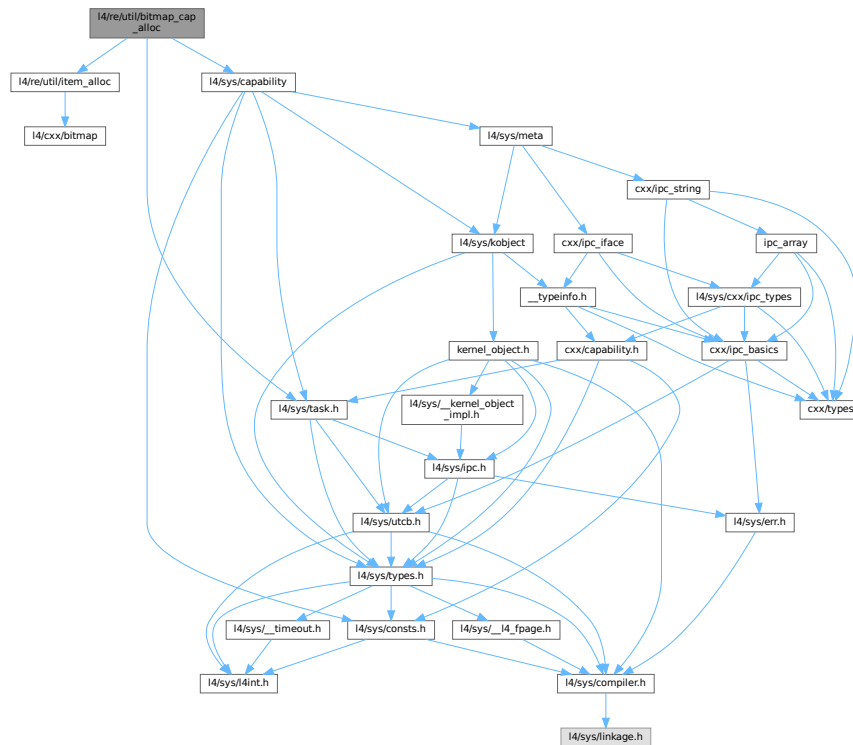
00001
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * 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 Rm_
00025     {
00026         enum Opcodes
00027         {
00028             Attach, Detach, Find, Attach_area, Detach_area, Get_regions, Get_areas
00029         };
00030     };
00031 };

```

16.341 I4/re/util/bitmap_cap_alloc File Reference

Bitmap capability allocator.

```
#include <l4/re/util/item_alloc>
#include <l4/sys/capability>
#include <l4/sys/task.h>
Include dependency graph for bitmap_cap_alloc:
```



Data Structures

- class [L4Re::Util::Cap_alloc_base](#)
Capability allocator.

Namespaces

- namespace [L4Re](#)
L4Re C++ Interfaces.
- namespace [L4Re::Util](#)
Documentation of the L4 Runtime Environment utility functionality in C++.

16.341.1 Detailed Description

Bitmap capability allocator.

Definition in file [bitmap_cap_alloc](#).

16.342 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/capability>
00026 #include <l4/sys/task.h>
00027
00028 namespace L4Re { namespace Util {
00029
00030 class Cap_alloc_base
00031 {
00032 private:
00033     long _bias;
00034     Item_alloc_base _items;
00035 public:
00036     enum State { Free = 0, Allocated, Unknown };
00037     Cap_alloc_base(long max, void *mem, long bias = 0)
00038         noexcept : _bias(bias), _items(max, mem) {}
00039
00040     L4::Cap<void> alloc() noexcept
00041     {
00042         long cap = _items.alloc();
00043         if (cap < 0)
00044             return L4::Cap<void>::Invalid;
00045
00046         return L4::Cap<void>((cap + _bias) << L4_CAP_SHIFT);
00047     }
00048
00049     long hint() const { return _items.hint(); }
00050
00051 template< typename T >
00052 L4::Cap<T> alloc() noexcept
00053 { return L4::Cap<T>(alloc().cap()); }
00054
00055 State is_allocated(L4::Cap<void> c) const noexcept
00056 {
00057     long idx = (c.cap() >> L4_CAP_SHIFT);
00058
00059     if (idx < _bias)
00060         return Unknown;
00061
00062     idx -= _bias;
00063     return _items.is_allocated(idx) ? Allocated : Free;
00064 }
00065
00066 template< typename T >
00067 void free(L4::Cap<T> const &cap, l4_cap_idx_t task = L4_INVALID_CAP,
00068          l4_umword_t unmap_flags = L4_FP_ALL_SPACES) noexcept
00069 {
00070     long idx = (cap.cap() >> L4_CAP_SHIFT);
00071     if (idx < _bias)
00072         return;
00073
00074     idx -= _bias;
00075
00076     _items.free(idx);
00077
00078     if (l4_is_valid_cap(task))
00079         l4_task_unmap(task, cap.fpage(), unmap_flags | 2);
00080 }
00081
00082

```

```

00097 // since we have no counters assume counter always > 0
00098 void take(L4::Cap<void>) noexcept {}
00099 bool release(L4::Cap<void>, l4_cap_idx_t task = L4_INVALID_CAP,
00100             unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept
00101 { (void)task; (void)unmap_flags; return false; }
00102
00103 long last() noexcept
00104 {
00105     return _items.size() + _bias - 1;
00106 }
00107 };
00108
00109 template< long Size >
00110 class Cap_alloc : public Cap_alloc_base
00111 {
00112 private:
00113     typename Bitmap_base::Word<Size>::Type _bits[Bitmap_base::Word<Size>::Size];
00114
00115 public:
00116     explicit Cap_alloc(long bias = 0) noexcept
00117         : Cap_alloc_base(Size, _bits, bias) {}
00118
00119 };
00120
00121 }
00122 }

```

16.343 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
00136
00137  int add_timeout(L4::Ipc_svr::Timeout *, l4_kernel_clock_t) override
00138  { return -L4_ENOSYS; }
00139
00140
00141  int remove_timeout(L4::Ipc_svr::Timeout *) override
00142  { return -L4_ENOSYS; }
00143
00144
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
00163 private:
00164     unsigned short _caps;
00165     unsigned long _cap_flags;
00166
00167     l4_umword_t _brs[L4_UTCB_GENERIC_BUFFERS_SIZE];
00168 };
00169
00176 struct Br_manager_hooks
00177 : L4::Ipc_svr::Ignore_errors,
00178   L4::Ipc_svr::Default_timeout,
00179   L4::Ipc_svr::Compound_reply,
00180   Br_manager
00181 {};
00182
00190 struct Br_manager_timeout_hooks :
00191     public L4::Ipc_svr::Timeout_queue_hooks<Br_manager_timeout_hooks, Br_manager>,
00192     public L4::Ipc_svr::Ignore_errors
00193 {
00194 public:
00195     static l4_kernel_clock_t now()
00196     { return l4_kip_clock(l4re_kip()); }
00197 };
00198
00199 }}
00200

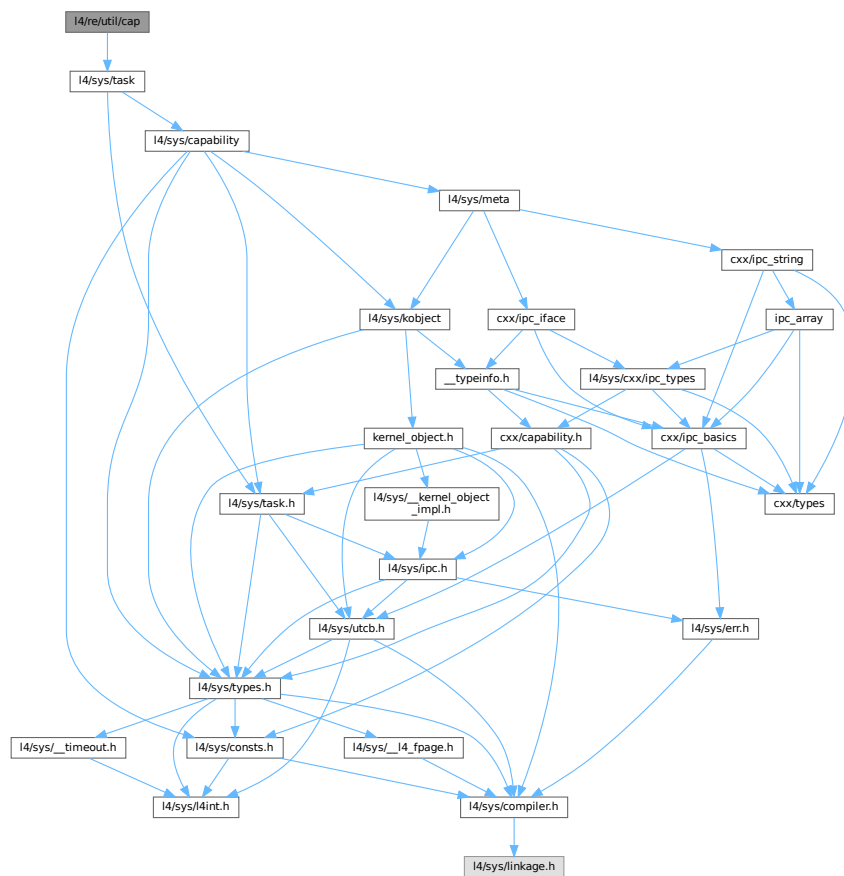
```

16.344 l4/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++.

16.344.1 Detailed Description

Capability utility functions.

Definition in file [cap](#).

16.345 cap

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2009 Adam Lackorzynski <adam@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
00029 namespace L4Re { namespace Util {
00030
00038 L4_CV static inline l4_msgtag_t cap_release(L4::Cap<void> cap)
00039 {
00040     return l4_task_unmap(L4_BASE_TASK_CAP,
00041                         l4_obj_fpage(cap.cap(), 0, L4_FPAGE_RWX),
00042                         L4_FP_ALL_SPACES);
00043 }
00044
00045 }}
```

16.346 l4/re/cap_alloc File Reference

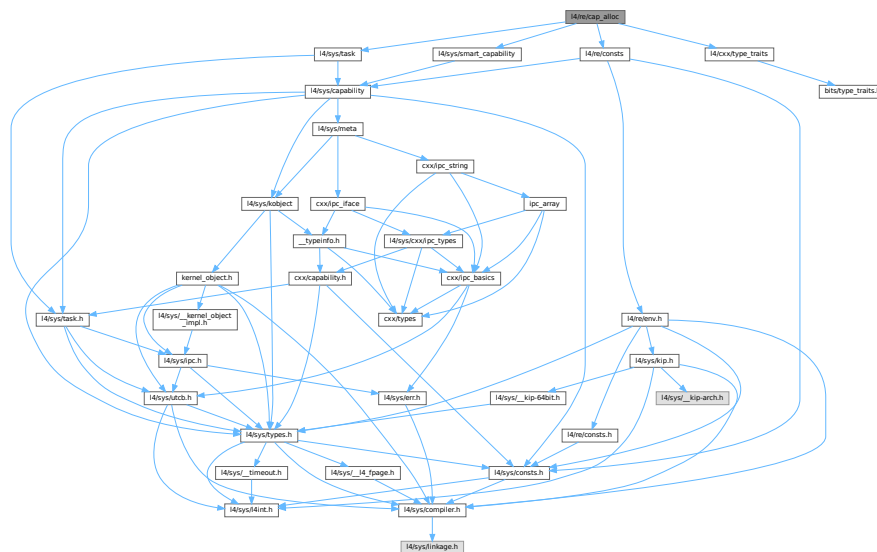
Abstract capability-allocator interface.

```

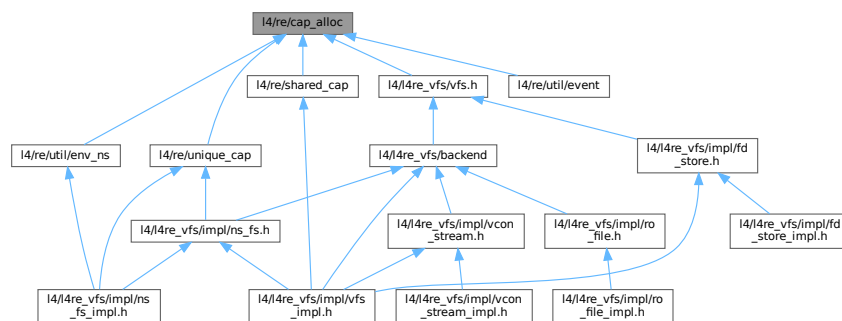
#include <l4/sys/task>
#include <l4/sys/smart_capability>
#include <l4/re/consts>
```



```
#include <l4/cxx/type_traits>
Include dependency graph for cap_alloc:
```



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.

16.346.1 Detailed Description

Abstract capability-allocator interface.

Definition in file [cap_alloc](#).

16.347 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     static L4::Cap_base copy(L4::Cap_base const &src)
00166     {
00167         L4::Cap_base r = src;
00168         invalidate(const_cast<L4::Cap_base &>(src));
00169         return r;
00170     }
00171 };
00172
00173 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00174 class Smart_count_cap
00175 {
00176 private:
00177     Cap_alloc *_ca;
00178
00179 public:
00180     Smart_count_cap() : _ca(nullptr) {}
00181     Smart_count_cap(Cap_alloc *ca) : _ca(ca) {}
00182     void free(L4::Cap_base &c) noexcept
00183     {
00184         if (c.is_valid())
00185         {
00186             if (_ca && _ca->release(L4::Cap<void>(c.cap()), This_task, Unmap_flags))
00187                 c.invalidate();
00188         }
00189     }
00190 }

```


16.348.1 Detailed Description

Capability allocator.

Definition in file [cap_alloc](#).

16.349 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/cap_alloc_impl.h>
00025 #include <l4/sys/smart_capability>
00026 #include <l4/sys/task>
00027 #include <l4/re/consts>
00028
00029 namespace L4Re { namespace Util {
00030
00031     extern _Cap_alloc &cap_alloc;
00032
00033     template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00034     class Smart_cap_auto
00035     {
00036     public:
00037         static void free(L4::Cap_base &c)
00038         {
00039             if (c.is_valid())
00040             {
00041                 cap_alloc.free(L4::Cap<void>(c.cap()), This_task, Unmap_flags);
00042                 c.invalidate();
00043             }
00044         }
00045
00046         static void invalidate(L4::Cap_base &c)
00047         {
00048             if (c.is_valid())
00049                 c.invalidate();
00050         }
00051
00052         static L4::Cap_base copy(L4::Cap_base const &src)
00053         {
00054             L4::Cap_base r = src;
00055             invalidate(const_cast<L4::Cap_base &>(src));
00056             return r;
00057         }
00058     };
00059
00060     template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00061     class Smart_count_cap
00062     {
00063     public:
00064         static void free(L4::Cap_base &c) noexcept
00065         {
00066             if (c.is_valid())
00067             {
00068                 if (cap_alloc.release(L4::Cap<void>(c.cap()), This_task, Unmap_flags))

```

16.350 l4/re/util/cap_alloc_impl.h File Reference

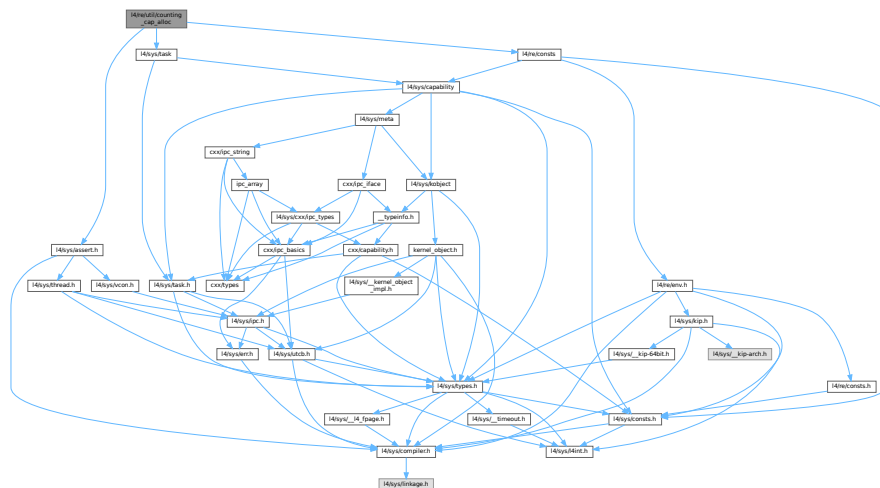
```
#include <l4/re/util/counting_cap_alloc>
```

Include dependency graph for cap_alloc_impl.h:



16.352 l4/re/util/counting_cap_alloc File Reference

```
#include <l4/sys/task>
#include <l4/sys/assert.h>
#include <l4/re/consts>
Include dependency graph for counting_cap_alloc:
```



16.353 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     void inc() { ++_cnt; }
00049     Type dec() { return --_cnt; }
00050     bool try_alloc()
00051     {
00052         if (_cnt == 0)
00053         {
00054             _cnt = 1;
00055             return true;
00056         }
00057         return false;
00058     }
00059 };
00060
00072 template< typename COUNTER = unsigned char >
00073 struct Counter_atomic
00074 {
00075     typedef COUNTER Type;
00076     Type _cnt;
00077
00078     static Type nil() { return 0; }
00079     static Type unused() { return 1; }
00080
00081     bool is_free() const { return __atomic_load_n(&_cnt, __ATOMIC_RELAXED) == 0; }
00082
00083     bool try_alloc()
00084     {
00085         Type expected = nil();
00086         // Use "acquire" memory ordering. Any operations tied to the capability slot
00087         // must only be observable after the slot has been occupied.
00088         return __atomic_compare_exchange_n(&_cnt, &expected, 2, false,
00089                                           __ATOMIC_ACQUIRE, __ATOMIC_RELAXED);
00090     }
00091
00092     void inc() { __atomic_add_fetch(&_cnt, 1, __ATOMIC_RELAXED); }
00093     Type dec() { return __atomic_sub_fetch(&_cnt, 1, __ATOMIC_RELAXED); }
00094
00095     void free()
00096     {
00097         // Use "release" memory ordering to make sure that any operations tied to
00098         // the capability slot are observable by other threads before the slot can
00099         // be reused.
00100         __atomic_store_n(&_cnt, 0, __ATOMIC_RELEASE);
00101     }
00102 };
```

```

00103
00128 template <typename COUNTERTYPE>
00129 class Counting_cap_alloc
00130 {
00131 private:
00132     void operator = (Counting_cap_alloc const &) { }
00133     typedef COUNTERTYPE Counter;
00134
00135     COUNTERTYPE *_items;
00136     long _free_hint;
00137     long _bias;
00138     long _capacity;
00139
00140 public:
00141
00142     template <unsigned COUNT>
00143     struct Counter_storage
00144     {
00145         COUNTERTYPE _buf[COUNT];
00146         typedef COUNTERTYPE Buf_type[COUNT];
00147         enum { Size = COUNT };
00148     };
00149
00150 protected:
00151
00152     Counting_cap_alloc() noexcept
00153     : _items(0), _free_hint(0), _bias(0), _capacity(0)
00154     {}
00155
00156     void setup(void *m, long capacity, long bias) noexcept
00157     {
00158         _items = (Counter*)m;
00159         _capacity = capacity;
00160         _bias = bias;
00161     }
00162
00163 public:
00164     L4::Cap<void> alloc() noexcept
00165     {
00166         long free_hint = __atomic_load_n(&_free_hint, __ATOMIC_RELAXED);
00167
00168         for (long i = free_hint; i < _capacity; ++i)
00169             if (_items[i].try_alloc())
00170             {
00171                 _free_hint = i + 1;
00172                 return L4::Cap<void>((i + _bias) << L4_CAP_SHIFT);
00173             }
00174
00175         // _free_hint is not necessarily correct in case of multi-threading! Make
00176         // sure we don't miss any potentially free slots.
00177         for (long i = 0; i < free_hint && i < _capacity; ++i)
00178             if (_items[i].try_alloc())
00179             {
00180                 _free_hint = i + 1;
00181                 return L4::Cap<void>((i + _bias) << L4_CAP_SHIFT);
00182             }
00183
00184         return L4::Cap<void>::Invalid;
00185     }
00186
00187     template <typename T>
00188     L4::Cap<T> alloc() noexcept
00189     {
00190         return L4::cap_cast<T>(alloc());
00191     }
00192
00193     void take(L4::Cap<void> cap) noexcept
00194     {
00195         long c;
00196         if (!range_check_and_get_idx(cap, &c))
00197             return;
00198
00199         _items[c].inc();
00200     }
00201
00202     bool free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00203              unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept
00204     {
00205         long c;
00206         if (!range_check_and_get_idx(cap, &c))
00207             return false;
00208
00209         l4_assert(!_items[c].is_free());
00210     }

```

```

00261     if (l4_is_valid_cap(task))
00262         l4_task_unmap(task, cap.fpage(), unmap_flags);
00263
00264     if (c < _free_hint)
00265         _free_hint = c;
00266
00267     _items[c].free();
00268
00269     return true;
00270 }
00271
00290 bool release(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00291             unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept
00292 {
00293     long c;
00294     if (!range_check_and_get_idx(cap, &c))
00295         return false;
00296
00297     l4_assert(!_items[c].is_free());
00298
00299     if (_items[c].dec() == Counter::unused())
00300     {
00301         if (task != L4_INVALID_CAP)
00302             l4_task_unmap(task, cap.fpage(), unmap_flags);
00303
00304         if (c < _free_hint)
00305             _free_hint = c;
00306
00307         // Let others allocate this slot only after the l4_task_unmap() has
00308         // finished.
00309         _items[c].free();
00310
00311         return true;
00312     }
00313     return false;
00314 }
00315
00319 long last() noexcept
00320 {
00321     return _capacity + _bias - 1;
00322 }
00323
00324 private:
00325 bool range_check_and_get_idx(L4::Cap<void> cap, long *c)
00326 {
00327     *c = cap.cap() >> L4_CAP_SHIFT;
00328     if (*c < _bias)
00329         return false;
00330
00331     *c -= _bias;
00332
00333     return *c < _capacity;
00334 }
00335 };
00336
00337 }}
00338

```

16.354 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/sys/types.h>
00026 #include <l4/cxx/list>
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 private:
00043     typedef L4::Ipc::Gen_fpage<L4::Ipc::Snd_item> Snd_fpage;
00044 public:
00045     L4_RPC_LEGACY_DISPATCH(L4Re::Dataspace);
00046
00047     typedef Snd_fpage::Map_type Map_type;
00048     typedef Snd_fpage::Cacheopt Cache_type;
00049
00050     Dataspace_svr() noexcept
00051     : _ds_start(0), _ds_size(0), _map_flags(Snd_fpage::Map),
00052       _cache_flags(Snd_fpage::Cached)
00053     {}
00054
00055     virtual ~Dataspace_svr() noexcept {}
00056
00070     int map(Dataspace::Offset offset,
00071            Dataspace::Map_addr local_addr,
00072            Dataspace::Flags flags,
00073            Dataspace::Map_addr min_addr,
00074            Dataspace::Map_addr max_addr,
00075            L4::Ipc::Snd_fpage &memory);
00076
00089     virtual int map_hook(Dataspace::Offset offs,
00090                        Dataspace::Flags flags,
00091                        Dataspace::Map_addr min,
00092                        Dataspace::Map_addr max)
00093     {
00094         (void)offs; (void)flags; (void)min; (void)max;
00095         return 0;
00096     }
00097
00103     virtual void take() noexcept
00104     {}
00105
00113     virtual unsigned long release() noexcept
00114     { return 0; }
00115
00128     virtual long copy(l4_addr_t dst_offs, l4_umword_t src_id,
00129                     l4_addr_t src_offs, unsigned long size) noexcept
00130     {
00131         (void)dst_offs; (void)src_id; (void)src_offs; (void)size;
00132         return -L4_ENODEV;
00133     }
00134
00144     virtual long clear(unsigned long offs, unsigned long size) const noexcept;
00145
00157     virtual long allocate(l4_addr_t offset, l4_size_t size, unsigned access) noexcept
00158     { (void)offset; (void)size; (void)access; return -L4_ENODEV; }
00159
00165     virtual unsigned long page_shift() const noexcept
00166     { return L4_LOG2_PAGESIZE; }
00167
00173     virtual bool is_static() const noexcept
00174     { return true; }
00175
00176
00177     long op_map(L4Re::Dataspace::Rights rights,
00178                L4Re::Dataspace::Offset offset,
00179                L4Re::Dataspace::Map_addr spot,
00180                L4Re::Dataspace::Flags flags,
00181                L4::Ipc::Snd_fpage &fp)
00182     {
00183         auto rf = map_flags(rights);
00184
00185         if (!rf.w() && flags.w())
00186             return -L4_EPERM;
00187
00188         return map(offset, spot, flags & rf, 0, ~0, fp);
00189     }
00190
00191     long op_allocate(L4Re::Dataspace::Rights rights,
00192                    L4Re::Dataspace::Offset offset,
00193                    L4Re::Dataspace::Size size)
00194     { return allocate(offset, size, rights & 3); }
00195

```

```

00196 long op_copy_in(L4Re::Dataspace::Rights rights,
00197                L4Re::Dataspace::Offset dst_offs,
00198                L4::Ipc::Snd_fpage const &src_cap,
00199                L4Re::Dataspace::Offset src_offs,
00200                L4Re::Dataspace::Size sz)
00201 {
00202     if (!src_cap.id_received())
00203         return -L4_EINVAL;
00204
00205     if (!(rights & L4_CAP_FPAGE_W))
00206         return -L4_EACCESS;
00207
00208     if (sz == 0)
00209         return L4_EOK;
00210
00211     return copy(dst_offs, src_cap.data(), src_offs, sz);
00212 }
00213
00214 long op_info(L4Re::Dataspace::Rights rights, L4Re::Dataspace::Stats &s)
00215 {
00216     s.size = size();
00217     // only return writable if really writable
00218     s.flags = Dataspace::Flags(0);
00219     if (map_flags(rights).w())
00220         s.flags |= Dataspace::F::W;
00221     return L4_EOK;
00222 }
00223
00224 long op_clear(L4Re::Dataspace::Rights rights,
00225              L4Re::Dataspace::Offset offset,
00226              L4Re::Dataspace::Size size)
00227 {
00228     if (!map_flags(rights).w())
00229         return -L4_EACCESS;
00230
00231     return clear(offset, size);
00232 }
00233
00234
00235 protected:
00236 unsigned long size() const noexcept
00237 { return _ds_size; }
00238 unsigned long map_flags() const noexcept
00239 { return _map_flags; }
00240 unsigned long page_size() const noexcept
00241 { return 1UL < page_shift(); }
00242 unsigned long round_size() const noexcept
00243 { return l4_round_size(size(), page_shift()); }
00244 bool check_limit(l4_addr_t offset) const noexcept
00245 { return offset < round_size(); }
00246
00247 L4Re::Dataspace::Flags
00248 map_flags(L4Re::Dataspace::Rights rights = L4_CAP_FPAGE_W) const noexcept
00249 {
00250     auto f = (_rw_flags & L4Re::Dataspace::Flags(0x0f)) | L4Re::Dataspace::F::Caching_mask;
00251     if (!(rights & L4_CAP_FPAGE_W))
00252         f &= ~L4Re::Dataspace::F::W;
00253
00254     return f;
00255 }
00256
00257 protected:
00258 void size(unsigned long size) noexcept { _ds_size = size; }
00259
00260 l4_addr_t _ds_start;
00261 l4_size_t _ds_size;
00262 Map_type _map_flags;
00263 Cache_type _cache_flags;
00264 L4Re::Dataspace::Flags _rw_flags;
00265 };
00266
00267 }

```

16.355 l4/re/debug File Reference

Debug interface.

```

#include <l4/sys/capability>
#include <l4/re/protocols.h>

```


16.356 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 }
```

16.357 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 { cprintf("%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 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 # ifdef __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 # ifdef __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
00118     Dbg() : _m(1), _component(0), _subsys(0) { };
00119
00120     explicit
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
00140     Dbg() {}
00141
00142     explicit
00143     Dbg(unsigned long, char const *, char const *) {}
00144
00145 #endif
00146
00147 };
00148
00149 }}
00150

```

16.358 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             char const *n = name;
00031             for (; len && *n != '/'; ++n, --len)
00032                 ;
00033
00034             Cap_entry const *e = _env->get(name, n - name);
00035             if (!e)
00036                 return L4::Cap<void>(-L4_ENOENT);
00037
00038             if (len > 0 && *n == '/')
00039             {
00040                 L4::Cap<L4Re::Namespace> ns(e->cap);
00041                 L4::Cap<void> cap = _ca->alloc<void>();
00042
00043                 if (!cap.is_valid())
00044                     return L4::Cap<void>(-L4_ENOMEM);
00045
00046                 long r = ns->query(n + 1, len - 1, cap, timeout, local_id, iterate);
00047                 if (r >= 0)
00048                     return cap;
00049
00050                 _ca->free(cap);
00051
00052                 return L4::Cap<void>(r);
00053             }
00054
00055             return L4::Cap<void>(e->cap);

```

```

00056     }
00057
00058     L4::Cap<void>
00059     query(char const *name, int timeout = Namespace::To_default,
00060           l4_umword_t *local_id = 0, bool iterate = true) const noexcept
00061     { return query(name, __builtin_strlen(name), timeout, local_id, iterate); }
00062
00063     template<typename T >
00064     L4::Cap<T>
00065     query(char const *name, int timeout = Namespace::To_default,
00066           l4_umword_t *local_id = 0, bool iterate = true) const noexcept
00067     {
00068         return L4::cap_cast<T>(query(name, __builtin_strlen(name),
00069                                     timeout, local_id, iterate));
00070     }
00071 };
00072
00073 }}

```

16.359 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
00032     typedef l4re_event_stream_id_t Event_stream_id;
00033     typedef l4re_event_absinfo_t Event_absinfo;
00034
00035     class L4_EXPORT Event_stream_bitmap_h
00036     {
00037     protected:
00038         static unsigned __get_idx(unsigned idx)
00039         { return idx / (sizeof(unsigned long)*8); }
00040
00041         static unsigned long __get_mask(unsigned idx)
00042         { return 1ul << (idx % (sizeof(unsigned long)*8)); }
00043
00044         static bool __get_bit(unsigned long const *bm, unsigned max, unsigned idx)
00045         {
00046             if (idx <= max)
00047                 return bm[__get_idx(idx)] & __get_mask(idx);
00048             return false;
00049         }
00050
00051         static void __set_bit(unsigned long *bm, unsigned max, unsigned idx, bool v)
00052         {
00053             if (idx > max)
00054                 return;
00055
00056             if (v)
00057                 bm[__get_idx(idx)] |= __get_mask(idx);
00058             else
00059                 bm[__get_idx(idx)] &= ~__get_mask(idx);
00060         }
00061     };
00062
00063     };
00064
00065     };
00066
00067     };
00068
00069     };
00070
00071     };
00072
00073     };
00074
00075     };
00076
00077     };
00078
00079     };
00080
00081     };
00082
00083     };
00084
00085     };
00086
00087     };
00088
00089     };
00090
00091     };
00092
00093     };
00094
00095     };
00096
00097     };
00098
00099     };
00100
00101     };
00102
00103     };
00104
00105     };
00106
00107     };
00108
00109     };
00110
00111     };
00112
00113     };
00114
00115     };
00116
00117     };
00118
00119     };
00120
00121     };
00122
00123     };
00124
00125     };
00126
00127     };
00128
00129     };
00130
00131     };
00132
00133     };
00134
00135     };
00136
00137     };
00138
00139     };
00140
00141     };
00142
00143     };
00144
00145     };
00146
00147     };
00148
00149     };
00150
00151     };
00152
00153     };
00154
00155     };
00156
00157     };
00158
00159     };
00160
00161     };
00162
00163     };
00164
00165     };
00166
00167     };
00168
00169     };
00170
00171     };
00172
00173     };
00174
00175     };
00176
00177     };
00178
00179     };
00180
00181     };
00182
00183     };
00184
00185     };
00186
00187     };
00188
00189     };
00190
00191     };
00192
00193     };
00194
00195     };
00196
00197     };
00198
00199     };
00200
00201     };
00202
00203     };
00204
00205     };
00206
00207     };
00208
00209     };
00210
00211     };
00212
00213     };
00214
00215     };
00216
00217     };
00218
00219     };
00220
00221     };
00222
00223     };
00224
00225     };
00226
00227     };
00228
00229     };
00230
00231     };
00232
00233     };
00234
00235     };
00236
00237     };
00238
00239     };
00240
00241     };
00242
00243     };
00244
00245     };
00246
00247     };
00248
00249     };
00250
00251     };
00252
00253     };
00254
00255     };
00256
00257     };
00258
00259     };
00260
00261     };
00262
00263     };
00264
00265     };
00266
00267     };
00268
00269     };
00270
00271     };
00272
00273     };
00274
00275     };
00276
00277     };
00278
00279     };
00280
00281     };
00282
00283     };
00284
00285     };
00286
00287     };
00288
00289     };
00290
00291     };
00292
00293     };
00294
00295     };
00296
00297     };
00298
00299     };
00300
00301     };
00302
00303     };
00304
00305     };
00306
00307     };
00308
00309     };
00310
00311     };
00312
00313     };
00314
00315     };
00316
00317     };
00318
00319     };
00320
00321     };
00322
00323     };
00324
00325     };
00326
00327     };
00328
00329     };
00330
00331     };
00332
00333     };
00334
00335     };
00336
00337     };
00338
00339     };
00340
00341     };
00342
00343     };
00344
00345     };
00346
00347     };
00348
00349     };
00350
00351     };
00352
00353     };
00354
00355     };
00356
00357     };
00358
00359     };
00360
00361     };
00362
00363     };
00364
00365     };
00366
00367     };
00368
00369     };
00370
00371     };
00372
00373     };
00374
00375     };
00376
00377     };
00378
00379     };
00380
00381     };
00382
00383     };
00384
00385     };
00386
00387     };
00388
00389     };
00390
00391     };
00392
00393     };
00394
00395     };
00396
00397     };
00398
00399     };
00400
00401     };
00402
00403     };
00404
00405     };
00406
00407     };
00408
00409     };
00410
00411     };
00412
00413     };
00414
00415     };
00416
00417     };
00418
00419     };
00420
00421     };
00422
00423     };
00424
00425     };
00426
00427     };
00428
00429     };
00430
00431     };
00432
00433     };
00434
00435     };
00436
00437     };
00438
00439     };
00440
00441     };
00442
00443     };
00444
00445     };
00446
00447     };
00448
00449     };
00450
00451     };
00452
00453     };
00454
00455     };
00456
00457     };
00458
00459     };
00460
00461     };
00462
00463     };
00464
00465     };
00466
00467     };
00468
00469     };
00470
00471     };
00472
00473     };
00474
00475     };
00476
00477     };
00478
00479     };
00480
00481     };
00482
00483     };
00484
00485     };
00486
00487     };
00488
00489     };
00490
00491     };
00492
00493     };
00494
00495     };
00496
00497     };
00498
00499     };
00500
00501     };
00502
00503     };
00504
00505     };
00506
00507     };
00508
00509     };
00510
00511     };
00512
00513     };
00514
00515     };
00516
00517     };
00518
00519     };
00520
00521     };
00522
00523     };
00524
00525     };
00526
00527     };
00528
00529     };
00530
00531     };
00532
00533     };
00534
00535     };
00536
00537     };
00538
00539     };
00540
00541     };
00542
00543     };
00544
00545     };
00546
00547     };
00548
00549     };
00550
00551     };
00552
00553     };
00554
00555     };
00556
00557     };
00558
00559     };
00560
00561     };
00562
00563     };
00564
00565     };
00566
00567     };
00568
00569     };
00570
00571     };
00572
00573     };
00574
00575     };
00576
00577     };
00578
00579     };
00580
00581     };
00582
00583     };
00584
00585     };
00586
00587     };
00588
00589     };
00590
00591     };
00592
00593     };
00594
00595     };
00596
00597     };
00598
00599     };
00600
00601     };
00602
00603     };
00604
00605     };
00606
00607     };
00608
00609     };
00610
00611     };
00612
00613     };
00614
00615     };
00616
00617     };
00618
00619     };
00620
00621     };
00622
00623     };
00624
00625     };
00626
00627     };
00628
00629     };
00630
00631     };
00632
00633     };
00634
00635     };
00636
00637     };
00638
00639     };
00640
00641     };
00642
00643     };
00644
00645     };
00646
00647     };
00648
00649     };
00650
00651     };
00652
00653     };
00654
00655     };
00656
00657     };
00658
00659     };
00660
00661     };
00662
00663     };
00664
00665     };
00666
00667     };
00668
00669     };
00670
00671     };
00672
00673     };
00674
00675     };
00676
00677     };
00678
00679     };
00680
00681     };
00682
00683     };
00684
00685     };
00686
00687     };
00688
00689     };
00690
00691     };
00692
00693     };
00694
00695     };
00696
00697     };
00698
00699     };
00700
00701     };
00702
00703     };
00704
00705     };
00706
00707     };
00708
00709     };
00710
00711     };
00712
00713     };
00714
00715     };
00716
00717     };
00718
00719     };
00720
00721     };
00722
00723     };
00724
00725     };
00726
00727     };
00728
00729     };
00730
00731     };
00732
00733     };
00734
00735     };
00736
00737     };
00738
00739     };
00740
00741     };
00742
00743     };
00744
00745     };
00746
00747     };
00748
00749     };
00750
00751     };
00752
00753     };
00754
00755     };
00756
00757     };
00758
00759     };
00760
00761     };
00762
00763     };
00764
00765     };
00766
00767     };
00768
00769     };
00770
00771     };
00772
00773     };
00774
00775     };
00776
00777     };
00778
00779     };
00780
00781     };
00782
00783     };
00784
00785     };
00786
00787     };
00788
00789     };
00790
00791     };
00792
00793     };
00794
00795     };
00796
00797     };
00798
00799     };
00800
00801     };
00802
00803     };
00804
00805     };
00806
00807     };
00808
00809     };
00810
00811     };
00812
00813     };
00814
00815     };
00816
00817     };
00818
00819     };
00820
00821     };
00822
00823     };
00824
00825     };
00826
00827     };
00828
00829     };
00830
00831     };
00832
00833     };
00834
00835     };
00836
00837     };
00838
00839     };
00840
00841     };
00842
00843     };
00844
00845     };
00846
00847     };
00848
00849     };
00850
00851     };
00852
00853     };
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
00881     };
00882
00883     };
00884
00885     };
00886
00887     };
00888
00889     };
00890
00891     };
00892
00893     };
00894
00895     };
00896
00897     };
00898
00899     };
00900
00901     };
00902
00903     };
00904
00905     };
00906
00907     };
00908
00909     };
00910
00911     };
00912
00913     };
00914
00915     };
00916
00917     };
00918
00919     };
00920
00921     };
00922
00923     };
00924
00925     };
00926
00927     };
00928
00929     };
00930
00931     };
00932
00933     };
00934
00935     };
00936
00937     };
00938
00939     };
00940
00941     };
00942
00943     };
00944
00945     };
00946
00947     };
00948
00949     };
00950
00951     };
00952
00953     };
00954
00955     };
00956
00957     };
00958
00959     };
00960
00961     };
00962
00963     };
00964
00965     };
00966
00967     };
00968
00969     };
00970
00971     };
00972
00973     };
00974
00975     };
00976
00977     };
00978
00979     };
00980
00981     };
00982
00983     };
00984
00985     };
00986
00987     };
00988
00989     };
00990
00991     };
00992
00993     };
00994
00995     };
00996
00997     };
00998
00999     };

```

```

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
00148 class L4_EXPORT Event :
00149 public L4::Kobject_t<Event, L4::Icu, L4RE_PROTO_EVENT>
00150 {
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((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

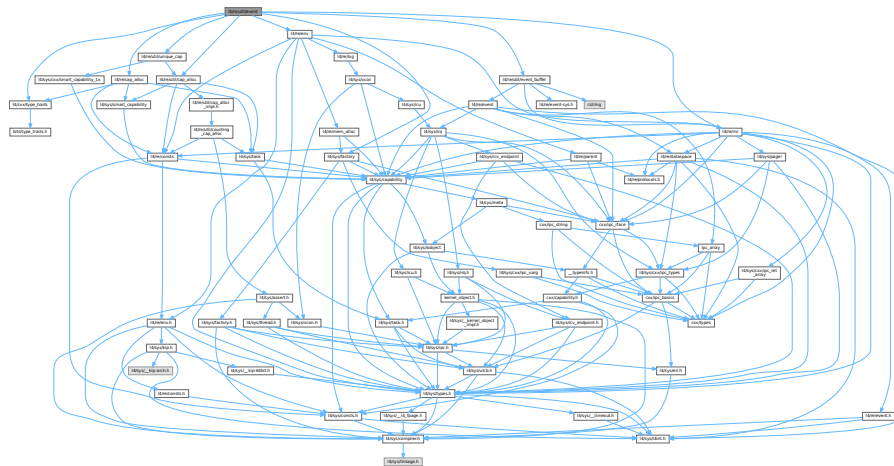
```

```
00346 }
00347
00348
```

16.360 I4/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++.

16.361 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(L4Re::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 }

```

16.362 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
00035 template< typename PAYLOAD >
00036 class Event_buffer_t : public L4Re::Event_buffer_t<PAYLOAD>

```

```

00037 {
00038 private:
00039     void *_buf;
00040 public:
00041     void *buf() const noexcept { return _buf; }
00042
00043     long attach(L4::Cap<L4Re::Dataspace> ds, L4::Cap<L4Re::Rm> rm) noexcept
00044     {
00045         l4_addr_t sz = ds->size();
00046         _buf = 0;
00047
00048         long r = rm->attach(&_buf, sz,
00049                             L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00050                             L4::Ipc::make_cap_rw(ds));
00051
00052         if (r < 0)
00053             return r;
00054
00055         *(L4Re::Event_buffer_t<PAYLOAD>*)this = L4Re::Event_buffer_t<PAYLOAD>(_buf, sz);
00056         return 0;
00057     }
00058
00059     long detach(L4::Cap<L4Re::Rm> rm) noexcept
00060     {
00061         L4::Cap<L4Re::Dataspace> ds;
00062         if (_buf)
00063             return rm->detach(_buf, &ds);
00064         return 0;
00065     }
00066 };
00067
00068 template< typename PAYLOAD >
00069 class Event_buffer_consumer_t : public Event_buffer_t<PAYLOAD>
00070 {
00071 public:
00072     template< typename CB, typename D >
00073     void foreach_available_event(CB const &cb, D data = D())
00074     {
00075         typename Event_buffer_t<PAYLOAD>::Event *e;
00076         while ((e = Event_buffer_t<PAYLOAD>::next()))
00077         {
00078             cb(e, data);
00079             e->free();
00080         }
00081     }
00082
00083     template< typename CB, typename D >
00084     void process(L4::Cap<L4::Irq> irq,
00085                  L4::Cap<L4::Thread> thread,
00086                  CB const &cb, D data = D())
00087     {
00088         if (l4_error(irq->bind_thread(thread, 0)))
00089             return;
00090
00091         while (1)
00092         {
00093             long r;
00094             r = l4_ipc_error(l4_irq_receive(irq.cap(), L4_IPC_NEVER),
00095                             l4_utcb());
00096
00097             if (r)
00098                 continue;
00099
00100             foreach_available_event(cb, data);
00101         }
00102     }
00103 };
00104
00105 typedef Event_buffer_t<Default_event_payload> Event_buffer;
00106 typedef Event_buffer_consumer_t<Default_event_payload> Event_buffer_consumer;
00107
00108 }

```

16.363 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
00038 template< typename SVR >
00039 class Event_svr : public Icu_cap_array_svr<SVR>
00040 {
00041 private:
00042     typedef Icu_cap_array_svr<SVR> Icu_svr;
00043
00044 protected:
00045     L4::Cap<L4Re::Dataspace> _ds;
00046     typename Icu_svr::Irq _irq;
00047
00048 public:
00049     Event_svr() : Icu_svr(1, &_irq) {}
00050
00051     L4_RPC_LEGACY_DISPATCH(L4Re::Event);
00052     L4_RPC_LEGACY_USING(Icu_svr);
00053
00055     long op_get_buffer(L4Re::Event::Rights, L4::Ipc::Cap<L4Re::Dataspace> &ds)
00056     {
00057         static_cast<SVR*>(this)->reset_event_buffer();
00058         ds = L4::Ipc::Cap<L4Re::Dataspace>(_ds, L4_CAP_FPAGE_RW);
00059         return 0;
00060     }
00061
00062     long op_get_num_streams(L4Re::Event::Rights)
00063     { return static_cast<SVR*>(this)->get_num_streams(); }
00064
00065     long op_get_stream_info(L4Re::Event::Rights, int idx, Event_stream_info &info)
00066     { return static_cast<SVR*>(this)->get_stream_info(idx, &info); }
00067
00068     long op_get_stream_info_for_id(L4Re::Event::Rights, l4_umword_t id,
00069                                     Event_stream_info &info)
00070     { return static_cast<SVR*>(this)->get_stream_info_for_id(id, &info); }
00071
00072     long op_get_axis_info(L4Re::Event::Rights, l4_umword_t id,
00073                           L4::Ipc::Array_in_buf<unsigned, unsigned long> const &axes,
00074                           L4::Ipc::Array_ref<Event_absinfo, unsigned long> &info)
00075     {
00076         unsigned naxes = cxx::min<unsigned>(L4RE_ABS_MAX, axes.length);
00077
00078         info.length = 0;
00079
00080         Event_absinfo _info[naxes];
00081         int r = static_cast<SVR*>(this)->get_axis_info(id, naxes, axes.data, _info);
00082         if (r < 0)
00083             return r;
00084
00085         for (unsigned i = 0; i < naxes; ++i)
00086             info.data[i] = _info[i];
00087
00088         info.length = naxes;
00089         return r;
00090     }
00091
00092     long op_get_stream_state_for_id(L4Re::Event::Rights, l4_umword_t stream_id,
00093                                     Event_stream_state &state)
00094     { return static_cast<SVR*>(this)->get_stream_state_for_id(stream_id, &state); }
00095
00096     int get_num_streams() const { return 0; }
00097     int get_stream_info(int, L4Re::Event_stream_info *)
00098     { return -L4_EINVAL; }
00099     int get_stream_info_for_id(l4_umword_t, L4Re::Event_stream_info *)
00100     { return -L4_EINVAL; }

```

```

00101 int get_axis_info(l4_umword_t, unsigned /*naxes*/, unsigned const * /*axes*/,
00102                  L4Re::Event_absinfo *)
00103 { return -L4_EINVAL; }
00104 int get_stream_state_for_id(l4_umword_t, L4Re::Event_stream_state *)
00105 { return -L4_EINVAL; }
00106 };
00107
00108 }

```

16.364 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
00064     return irq->bind(this_icu(), irq_fp);
00065 }
00066
00067 template<typename ICU> inline
00068 int
00069 Icu_svr<ICU>::op_unbind(L4::Icu::Rights, l4_umword_t irqnum,
00070                        L4::Ipc::Snd_fpage irq_fp)
00071 {
00072     typename ICU::Irq *irq = this_icu()->icu_get_irq(irqnum);

```

```

00073     if (!irq)
00074         return -L4_EINVAL;
00075
00076     return irq->unbind(this_icu(), irq_fp);
00077 }
00078
00079 template<typename ICU> inline
00080 int
00081 Icu_svr<ICU>::op_info(L4::Icu::Rights, L4::Icu::_Info &info)
00082 {
00083     l4_icu_info_t i;
00084     this_icu()->icu_get_info(&i);
00085     info.features = i.features;
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         { if (_cap) _cap->trigger(); }
00145
00146         int bind(ICU *, L4::Ipc::Snd_fpage const &irq_fp);
00147         int unbind(ICU *, L4::Ipc::Snd_fpage const &irq_fp);
00148         void mask(bool mask) const
00149         { (void)mask; }
00150
00151         int msi_info(l4_uint64_t, l4_icu_msi_info_t *) const
00152         { return -L4_EINVAL; }
00153
00154         L4::Cap<L4::Irq> cap() const { return _cap; }
00155
00156     private:
00157         L4::Cap<L4::Irq> _cap;
00158     };
00159

```

```

00160 private:
00161     Irq *_irqs;
00162     unsigned _nr_irqs;
00163 public:
00164     Icu_cap_array_svr(unsigned nr_irqs, Irq *_irqs)
00165     : _irqs(irqs), _nr_irqs(nr_irqs)
00166     {}
00167     Irq *icu_get_irq(l4_umword_t irqnum)
00168     {
00169         if (irqnum >= _nr_irqs)
00170             return 0;
00171         return _irqs + irqnum;
00172     }
00173     void icu_get_info(l4_icu_info_t *inf)
00174     {
00175         inf->features = 0;
00176         inf->nr_irqs = _nr_irqs;
00177         inf->nr_msis = 0;
00178     }
00179 };
00180 template< typename ICU >
00181 int
00182 Icu_cap_array_svr<ICU>::Irq::bind(ICU *cfb, L4::Ipc::Snd_fpage const &irq_fp)
00183 {
00184     if (!irq_fp.cap_received())
00185         return -L4_EINVAL;
00186     L4::Cap<L4::Irq> irq = cfb->server_iface()->template_rcv_cap<L4::Irq>(0);
00187     if (!irq)
00188         return -L4_EINVAL;
00189     int r = cfb->server_iface()->realloc_rcv_cap(0);
00190     if (r < 0)
00191         return r;
00192     ICU::free_irq_cap(_cap);
00193     _cap = irq;
00194     return 0;
00195 }
00196 template< typename ICU >
00197 int
00198 Icu_cap_array_svr<ICU>::Irq::unbind(ICU *, L4::Ipc::Snd_fpage const &/*irq_fp*/)
00199 {
00200     ICU::free_irq_cap(_cap);
00201     _cap = L4::Cap<L4::Irq>::Invalid;
00202     return 0;
00203 }
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216 }

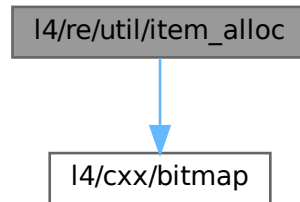
```

16.365 I4/re/util/item_alloc File Reference

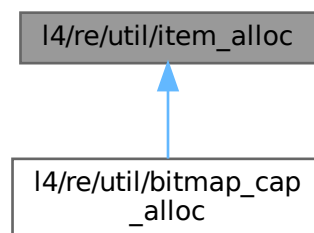
Item allocator.

```
#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++.

16.365.1 Detailed Description

Item allocator.

Definition in file [item_alloc](#).

16.366 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 public:
00046     bool is_allocated(long item) const noexcept
00047     { return _bits[item]; }
00048
00049     long hint() const { return _free_hint; }
00050
00051     bool alloc(long item) noexcept
00052     {
00053         if (!_bits[item])
00054         {
00055             _bits.set_bit(item);
00056             return true;
00057         }
00058         return false;
00059     }
00060
00061     void free(long item) noexcept
00062     {
00063         if (item < _free_hint)
00064             _free_hint = item;
00065
00066         _bits.clear_bit(item);
00067     }
00068
00069     Item_alloc_base(long size, void *mem) noexcept
00070         : _capacity(size), _free_hint(0), _bits(mem)
00071     {}
00072
00073     long alloc() noexcept
00074     {
00075         if (_free_hint >= _capacity)
00076             return -1;
00077
00078         long free = _bits.scan_zero(_capacity, _free_hint);
00079         if (free >= 0)
00080         {
00081             _bits.set_bit(free);
00082             _free_hint += 1;
00083         }
00084         return free;
00085     }
00086
00087     long size() const noexcept
00088     {
00089         return _capacity;

```



```

00090     }
00091 };
00092
00093 template< long Bits >
00094 class Item_alloc : public Item_alloc_base
00095 {
00096 private:
00097     typename Bitmap_base::Word<Bits>::_Type _bits[Bitmap_base::Word<Bits>::_Size];
00098
00099 public:
00100     Item_alloc() noexcept : Item_alloc_base(Bits, _bits) {}
00101 };
00102
00103 }

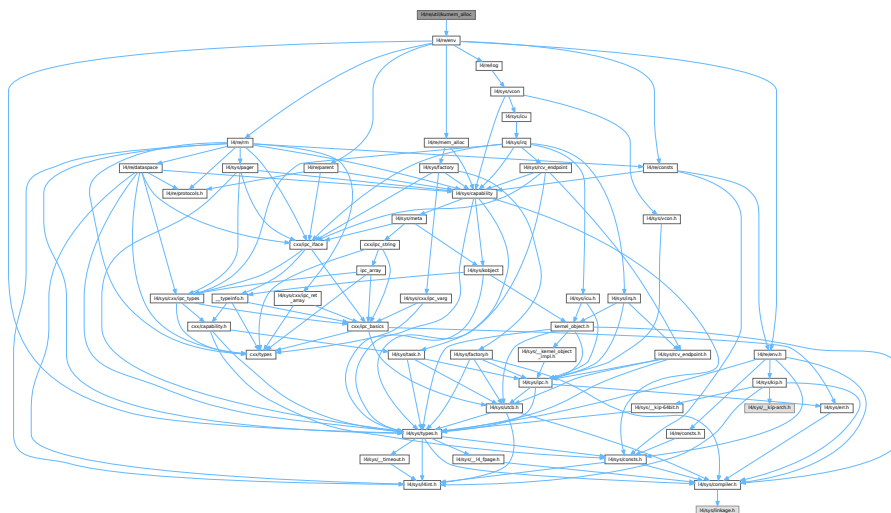
```

16.367 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.

16.367.1 Detailed Description

Kumem allocator helper.

Definition in file [kumem_alloc](#).

16.368 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 }}
```

16.369 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
00022 #include <l4/cxx/avl_tree>
00023 #include <l4/cxx/std_ops>
00024 #include <l4/sys/cxx/ipc_epiface>
00025 #include <l4/cxx/string>
00026
00027 #include <l4/sys/capability>
00028 #include <l4/re/namespace>
00029
```

```

00030 #include <cstdint>
00031
00032 namespace L4Re { namespace Util {
00033
00034 class Dbg;
00035 class Err;
00036
00037 namespace Names {
00038
00042 class Name : public cxx::String
00043 {
00044 public:
00045
00046     Name(const char *name = "") : String(name, __builtin_strlen(name)) {}
00047     Name(const char *name, unsigned long len) : String(name, len) {}
00048     Name(cxx::String const &n) : String(n) {}
00049     char const *name() const { return start(); }
00050     bool operator < (Name const &r) const;
00051 };
00052
00053
00057 class Obj
00058 {
00059 protected:
00060     unsigned _f;
00061     union
00062     {
00063         l4_cap_idx_t _cap;
00064         L4::Epiface *_obj;
00065     };
00066
00067 public:
00068     enum Flags
00069     {
00070         F_rw          = L4Re::Namespace::Rw,
00071         F_strong       = L4Re::Namespace::Strong,
00072
00073         F_trusted      = L4Re::Namespace::Trusted,
00074
00075         F_rights_mask = F_rw | F_strong | F_trusted,
00076
00077         F_cap          = 0x100,
00078         F_local        = 0x200,
00079         F_replacable   = 0x400,
00080         F_base_mask    = 0xf00,
00081     };
00082
00083     unsigned flags() const { return _f; }
00086     void restrict_flags(unsigned max_rights)
00087     { _f &= (~F_rights_mask | (max_rights & F_rights_mask)); }
00088
00089     bool is_rw() const { return (_f & F_rw) == F_rw; }
00090     bool is_strong() const { return _f & F_strong; }
00091
00092     bool is_valid() const { return _f & F_cap; }
00093     bool is_complete() const { return is_valid(); }
00094     bool is_local() const { return _f & F_local; }
00095     bool is_replacable() const { return _f & F_replacable; }
00096     bool is_trusted() const { return _f & F_trusted; }
00097
00098     L4::Epiface *obj() const { if (is_local()) return _obj; return 0; }
00099     L4::Cap<void> cap() const
00100     {
00101         if (!is_local())
00102             return L4::Cap<void>(_cap);
00103         if (!_obj)
00104             return L4::Cap<void>::Invalid;
00105         return _obj->obj_cap();
00106     }
00107
00108     void set(Obj const &o, unsigned flags)
00109     {
00110         *this = o;
00112         restrict_flags(flags);
00113     }
00114
00115     explicit Obj(unsigned flags = 0)
00116     : _f(flags), _cap(L4_INVALID_CAP)
00117     {}
00118
00119     Obj(unsigned f, L4::Cap<void> const &cap)
00120     : _f((f & ~F_base_mask) | F_cap), _cap(cap.cap())
00121     {}
00122

```

```

00123     Obj(unsigned f, L4::Epiface *o)
00124     : _f((f & ~F_base_mask) | F_cap | F_local), _obj(o)
00125     {}
00126
00127     void reset(unsigned flags)
00128     {
00129         _f = (_f & F_replacable) | (flags & ~(F_cap | F_local));
00130         _cap = L4_INVALIDID_CAP;
00131     }
00132
00133
00134 };
00135
00136
00140 class Entry : public cxx::Avl_tree_node
00141 {
00142 private:
00143     friend class Name_space;
00144     Name _n;
00145     Obj _o;
00146
00147     bool _dynamic;
00148
00149 public:
00150     Entry(Name const &n, Obj const &o, bool dynamic = false)
00151     : _n(n), _o(o), _dynamic(dynamic) {}
00152
00153     Name const &name() const { return _n; }
00154     Obj const *obj() const { return &_amp;o; }
00155     Obj *obj() { return &_amp;o; }
00156     void obj(Obj const &o) { _o = o; }
00157
00158     bool is_placeholder() const
00159     { return !obj()->is_complete(); }
00160
00161     bool is_dynamic() const { return _dynamic; }
00162
00163     void set(Obj const &o);
00164
00165 private:
00166     void * operator new (size_t s);
00167     void operator delete(void *b);
00168
00169 };
00170
00171 struct Names_get_key
00172 {
00173     typedef Name Key_type;
00174     static Key_type const &key_of(Entry const *e)
00175     { return e->name(); }
00176 };
00177
00178
00186 class Name_space
00187 {
00188     friend class Entry;
00189
00190 private:
00191     typedef cxx::Avl_tree<Entry, Names_get_key> Tree;
00192     Tree _tree;
00193
00194 protected:
00195     L4Re::Util::Dbg const &_dbg;
00196     L4Re::Util::Err const &_err;
00197
00198 public:
00199
00200     typedef Tree::Const_iterator Const_iterator;
00201
00202     Const_iterator begin() const { return _tree.begin(); }
00203     Const_iterator end() const { return _tree.end(); }
00204
00205     Name_space(L4Re::Util::Dbg const &dbg, L4Re::Util::Err const &err)
00206     : _dbg(dbg), _err(err)
00207     {}
00208
00212     virtual ~Name_space() {}
00213
00214     Entry *find(Name const &name) const { return _tree.find_node(name); }
00215     Entry *remove(Name const &name) { return _tree.remove(name); }
00216     Entry *find_iter(Name const &name) const;
00217     bool insert(Entry *e) { return _tree.insert(e).second; }
00218
00219     void dump(bool rec = false, int indent = 0) const;
00220
00221 protected:
00222     // server support -----

```

```

00235     virtual Entry *alloc_dynamic_entry(Name const &n, unsigned flags) = 0;
00236
00242     virtual void free_dynamic_entry(Entry *e) = 0;
00243
00266     virtual int get_epiface(l4_umword_t data, bool is_local, L4::Epiface **lo) = 0;
00267
00280     virtual int copy_receive_cap(L4::Cap<void> *cap) = 0;
00281
00290     virtual void free_capability(L4::Cap<void> cap) = 0;
00291
00300     virtual void free_epiface(L4::Epiface *epiface) = 0;
00301
00302     int insert_entry(Name const &name, unsigned flags, Entry **e);
00303
00304 public:
00305     // server interface -----
00306     int op_query(L4Re::Namespace::Rights,
00307                 L4::Ipc::Array_in_buf<char, unsigned long> const &name,
00308                 L4::Ipc::Snd_fpage &snd_cap, L4::Ipc::Opt<L4::Opcode> &,
00309                 L4::Ipc::Opt<L4::Ipc::Array_ref<char, unsigned long> > &out_name);
00310
00311     int op_register_obj(L4Re::Namespace::Rights, unsigned flags,
00312                        L4::Ipc::Array_in_buf<char, unsigned long> const &name,
00313                        L4::Ipc::Snd_fpage &cap);
00314
00315     int op_unlink(L4Re::Namespace::Rights r,
00316                  L4::Ipc::Array_in_buf<char, unsigned long> const &name);
00317 };
00318
00319 }}}
00320

```

16.370 object_registry

```

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/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
00079     Object_registry(L4::Ipc_svr::Server_iface *sif,
00080                    L4::Cap<L4::Thread> server,
00081                    L4::Cap<L4::Factory> factory)
00082     : _server(server), _factory(factory), _sif(sif)
00083     {}
00084
00085 private:
00086     typedef L4::Ipc_svr::Server_iface Server_iface;
00087     typedef Server_iface::Demand Demand;
00088
00089     L4::Cap<L4::Rcv_endpoint>
00090     _register_ep(L4::Epiface *o, L4::Cap<L4::Rcv_endpoint> ep,
00091                 Demand const &demand)
00092     {
00093         int err = _sif->alloc_buffer_demand(demand);
00094         if (err < 0)
00095             return L4::Cap<L4::Rcv_endpoint>(err | L4_INVALID_CAP_BIT);
00096
00097         err = o->set_server(_sif, ep);
00098         if (err < 0)
00099             return L4::Cap<L4::Rcv_endpoint>(err | L4_INVALID_CAP_BIT);
00100
00101         l4_umword_t id = l4_umword_t(o);
00102         err = l4_error(ep->bind_thread(_server, id));
00103         if (err < 0)
00104             return L4::Cap<L4::Rcv_endpoint>(err | L4_INVALID_CAP_BIT);
00105
00106         return ep;
00107     }
00108
00109     L4::Cap<void> _register_ep(L4::Epiface *o, char const *service,
00110                              Demand const &demand)
00111     {
00112         L4::Cap<L4::Rcv_endpoint> cap = L4Re::Env::env()->get_cap<L4::Rcv_endpoint>(service);
00113         if (!cap.is_valid())
00114             return cap;
00115
00116         return _register_ep(o, cap, demand);
00117     }
00118
00119     L4::Cap<void> _register_gate(L4::Epiface *o, Demand const &demand)
00120     {
00121         int err = _sif->alloc_buffer_demand(demand);
00122         if (err < 0)
00123             return L4::Cap<void>(err | L4_INVALID_CAP_BIT);
00124
00125         auto cap = L4Re::Util::make_unique_cap<L4::Kobject>();
00126
00127         if (!cap.is_valid())
00128             return cap.get();
00129
00130         l4_umword_t id = l4_umword_t(o);
00131         err = l4_error(_factory->create_gate(cap.get(), _server, id));
00132         if (err < 0)
00133             return L4::Cap<void>(err | L4_INVALID_CAP_BIT);
00134
00135         err = o->set_server(_sif, cap.get(), true);
00136         if (err < 0)
00137             return L4::Cap<void>(err | L4_INVALID_CAP_BIT);
00138
00139         return cap.release();
00140     }
00141
00142     L4::Cap<L4::Irq> _register_irq(L4::Epiface *o,
00143                                   Demand const &demand)
00144     {
00145         int err = _sif->alloc_buffer_demand(demand);
00146         if (err < 0)
00147             return L4::Cap<L4::Irq>(err | L4_INVALID_CAP_BIT);
00148
00149         auto cap = L4Re::Util::make_unique_cap<L4::Irq>();
00150
00151         if (!cap.is_valid())
00152             return cap.get();
00153
00154         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>((L4::Epiface*)&_null_handler));
00292     _server->modify_senders(todo);
00293
00294     // we use bit 4 to indicated an internally allocated cap
00295     if (c.managed())
00296         cap_alloc.free(c);
00297
00298     o->set_server(0, L4::Cap<void>::Invalid);
00299 }
00300 };
00301
00305 template< typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks >
00306 class Registry_server : public L4::Server<LOOP_HOOKS>
00307 {
00308 private:
00309     typedef L4::Server<LOOP_HOOKS> Base;
00310     Object_registry _registry;
00311
00312 public:
00318 Registry_server() : _registry(this)
00319 {}
00320

```

```

00330 Registry_server(l4_utcb_t *, L4::Cap<L4::Thread> server,
00331                  L4::Cap<L4::Factory> factory)
00332 : _registry(this, server, factory)
00333 {}
00334
00341 Registry_server(L4::Cap<L4::Thread> server,
00342                  L4::Cap<L4::Factory> factory)
00343 : _registry(this, server, factory)
00344 {}
00345
00347 Object_registry const *registry() const { return &_amp;registry; }
00349 Object_registry *registry() { return &_amp;registry; }
00350
00357 void L4_NORETURN loop(l4_utcb_t *utcb = l4_utcb())
00358 { Base::template loop<L4::Runtime_error, Object_registry &>(_registry, utcb); }
00359 };
00360
00361 }

```

16.371 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
00038 class Poll_timeout_kipclock
00039 {
00040 public:
00045 Poll_timeout_kipclock(unsigned poll_time_us)
00046 {
00047     set(poll_time_us);
00048 }
00049
00054 void set(unsigned poll_time_us)
00055 {
00056     _timeout = l4_kip_clock(l4re_kip()) + poll_time_us;
00057     _last_check = true;
00058 }
00059
00068 bool test(bool expression = true)
00069 {
00070     if (!expression)
00071         return false;
00072
00073     return _last_check = l4_kip_clock(l4re_kip()) < _timeout;
00074 }
00075
00080 bool timed_out() const { return !_last_check; }
00081
00082 private:
00083     l4_cpu_time_t _timeout;
00084     bool _last_check;
00085 };
00086 }

```

16.372 l4/re/util/region_mapping File Reference

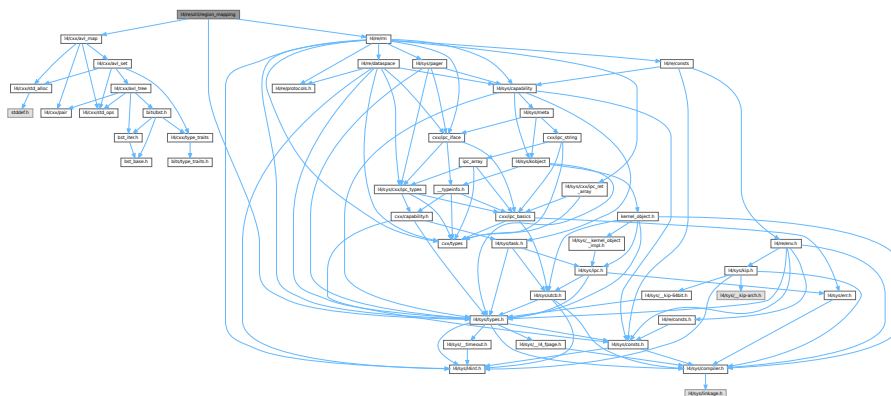
Region handling.

```

#include <l4/cxx/avl_map>
#include <l4/sys/types.h>

```


Include dependency graph for region_mapping:



```

graph BT
    A[14/re/util/region_mapping_svr_2] --> B[14/re/util/region_mapping]
  
```

- namespace **L4Re**
L4Re C++ Interfaces.
- namespace **L4Re::Util**

16.372.1 Detailed Description

Definition in file [region_mapping](#).

16.373 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
00023 #pragma once
00024
00025 #include <l4/cxx/avl_map>
00026 #include <l4/sys/types.h>
00027 #include <l4/re/rm>
00028
00029 namespace L4Re { namespace Util {
00030 class Region
00031 {
00032 private:
00033     l4_addr_t _start, _end;
00034 public:
00035     Region() noexcept : _start(~0UL), _end(~0UL) {}
00036     Region(l4_addr_t addr) noexcept : _start(addr), _end(addr) {}
00037     Region(l4_addr_t start, l4_addr_t end) noexcept
00038         : _start(start), _end(end) {}
00039     l4_addr_t start() const noexcept { return _start; }
00040     l4_addr_t end() const noexcept { return _end; }
00041     unsigned long size() const noexcept { return end() - start() + 1; }
00042     bool invalid() const noexcept { return _start == ~0UL && _end == ~0UL; }
00043     bool operator < (Region const &o) const noexcept
00044     { return end() < o.start(); }
00045     bool contains(Region const &o) const noexcept
00046     { return o.start() >= start() && o.end() <= end(); }
00047     bool operator == (Region const &o) const noexcept
00048     { return o.start() == start() && o.end() == end(); }
00049     ~Region() noexcept {}
00050 };
00051
00052 template< typename DS, typename OPS >
00053 class Region_handler
00054 {
00055 private:
00056     L4Re::Rm::Offset _offs;
00057     DS _mem;
00058     l4_cap_idx_t _client_cap = L4_INVALID_CAP;
00059     L4Re::Rm::Region_flags _flags;
00060 public:
00061     typedef DS Dataspace;
00062     typedef OPS Ops;
00063     typedef typename OPS::Map_result Map_result;
00064
00065     Region_handler() noexcept : _offs(0), _mem(), _flags() {}
00066     Region_handler(Dataspace const &mem, l4_cap_idx_t client_cap,
00067         L4Re::Rm::Offset offset = 0,
00068         L4Re::Rm::Region_flags flags = L4Re::Rm::Region_flags(0)) noexcept
00069         : _offs(offset), _mem(mem), _client_cap(client_cap), _flags(flags)
00070     {}
00071
00072     Dataspace const &memory() const noexcept
00073     {
00074         return _mem;
00075     }
00076
00077     l4_cap_idx_t client_cap_idx() const noexcept
00078     {
00079         return _client_cap;
00080     }

```

```

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 };
00126
00127
00128 template< typename Hdlr, template<typename T> class Alloc >
00129 class Region_map
00130 {
00131 protected:
00132     typedef cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc > Tree;
00133     Tree _rm;
00134     Tree _am;
00135
00136 private:
00137     l4_addr_t _start;
00138     l4_addr_t _end;
00139
00140 protected:
00141     void set_limits(l4_addr_t start, l4_addr_t end) noexcept
00142     {
00143         _start = start;
00144         _end = end;
00145     }
00146
00147 public:
00148     typedef typename Tree::Item_type Item;
00149     typedef typename Tree::Node Node;
00150     typedef typename Tree::Key_type Key_type;
00151     typedef Hdlr Region_handler;
00152
00153     typedef typename Tree::Iterator Iterator;
00154     typedef typename Tree::Const_iterator Const_iterator;
00155     typedef typename Tree::Rev_iterator Rev_iterator;
00156     typedef typename Tree::Const_rev_iterator Const_rev_iterator;
00157
00158     Iterator begin() noexcept { return _rm.begin(); }
00159     Const_iterator begin() const noexcept { return _rm.begin(); }
00160     Iterator end() noexcept { return _rm.end(); }
00161     Const_iterator end() const noexcept { return _rm.end(); }
00162
00163     Iterator area_begin() noexcept { return _am.begin(); }
00164     Const_iterator area_begin() const noexcept { return _am.begin(); }
00165     Iterator area_end() noexcept { return _am.end(); }
00166     Const_iterator area_end() const noexcept { return _am.end(); }
00167     Node area_find(Key_type const &c) const noexcept { return _am.find_node(c); }
00168
00169     l4_addr_t min_addr() const noexcept { return _start; }
00170     l4_addr_t max_addr() const noexcept { return _end; }
00171
00172
00173     Region_map(l4_addr_t start, l4_addr_t end) noexcept : _start(start), _end(end) {}

```

```

00174
00175 Node find(Key_type const &key) const noexcept
00176 {
00177     Node n = _rm.find_node(key);
00178     if (!n)
00179         return Node();
00180
00181     // 'find' should find any region overlapping with the searched one, the
00182     // caller should check for further requirements
00183     if (0)
00184         if (!n->first.contains(key))
00185             return Node();
00186
00187     return n;
00188 }
00189
00190 Node lower_bound(Key_type const &key) const noexcept
00191 {
00192     Node n = _rm.lower_bound_node(key);
00193     return n;
00194 }
00195
00196 Node lower_bound_area(Key_type const &key) const noexcept
00197 {
00198     Node n = _am.lower_bound_node(key);
00199     return n;
00200 }
00201
00202 L4_addr_t attach_area(L4_addr_t addr, unsigned long size,
00203                      L4Re::Rm::Flags flags = L4Re::Rm::Flags(0),
00204                      unsigned char align = L4_PAGESHIFT) noexcept
00205 {
00206     if (size < 2)
00207         return L4_INVALID_ADDR;
00208
00209     Region c;
00210
00211     if (!(flags & L4Re::Rm::F::Search_addr))
00212     {
00213         c = Region(addr, addr + size - 1);
00214         Node r = _am.find_node(c);
00215         if (r)
00216             return L4_INVALID_ADDR;
00217     }
00218
00219     while (flags & L4Re::Rm::F::Search_addr)
00220     {
00221         if (addr < min_addr() || (addr + size - 1) > max_addr())
00222             addr = min_addr();
00223         addr = find_free(addr, max_addr(), size, align, flags);
00224         if (addr == L4_INVALID_ADDR)
00225             return L4_INVALID_ADDR;
00226
00227         c = Region(addr, addr + size - 1);
00228         Node r = _am.find_node(c);
00229         if (!r)
00230             break;
00231
00232         if (r->first.end() >= max_addr())
00233             return L4_INVALID_ADDR;
00234
00235         addr = r->first.end() + 1;
00236     }
00237
00238     if (_am.insert(c, Hdlr(typename Hdlr::Dataspace(), 0, 0, flags.region_flags()).second == 0)
00239         return addr;
00240
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
00249     return true;
00250 }
00251
00252 void *attach(void *addr, unsigned long size, Hdlr const &hdlr,
00253             L4Re::Rm::Flags flags = L4Re::Rm::Flags(0),
00254             unsigned char align = L4_PAGESHIFT) noexcept
00255 {
00256     if (size < 2)
00257         return L4_INVALID_PTR;
00258
00259     L4_addr_t end = max_addr();

```

```

00261     l4_addr_t beg = (l4_addr_t)addr;
00262
00263     if (flags & L4Re::Rm::F::In_area)
00264     {
00265         Node r = _am.find_node(Region(beg, beg + size - 1));
00266         if (!r || (r->second.flags() & L4Re::Rm::F::Reserved))
00267             return L4_INVALID_PTR;
00268
00269         end = r->first.end();
00270     }
00271
00272     if (flags & L4Re::Rm::F::Search_addr)
00273     {
00274         beg = find_free(beg, end, size, align, flags);
00275         if (beg == L4_INVALID_ADDR)
00276             return L4_INVALID_PTR;
00277     }
00278
00279     if (!(flags & (L4Re::Rm::F::Search_addr | L4Re::Rm::F::In_area))
00280         && _am.find_node(Region(beg, beg + size - 1)))
00281         return L4_INVALID_PTR;
00282
00283     if (beg < min_addr() || beg + size - 1 > end)
00284         return L4_INVALID_PTR;
00285
00286     if (_rm.insert(Region(beg, beg + size - 1), hdlr).second == 0)
00287         return (void*)beg;
00288
00289     return L4_INVALID_PTR;
00290 }
00291
00292 int detach(void *addr, unsigned long sz, unsigned flags,
00293            Region *reg, Hdlr *hdlr) noexcept
00294 {
00295     Region dr((l4_addr_t)addr, (l4_addr_t)addr + sz - 1);
00296     Region res(~0UL, 0);
00297
00298     Node r = find(dr);
00299     if (!r)
00300         return -L4_ENOENT;
00301
00302     Region g = r->first;
00303     Hdlr const &h = r->second;
00304
00305     if (flags & L4Re::Rm::Detach_overlap || dr.contains(g))
00306     {
00307         if (_rm.remove(g))
00308             return -L4_ENOENT;
00309
00310         if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::F::Detach_free))
00311             h.free(0, g.size());
00312
00313         if (hdlr) *hdlr = h;
00314         if (reg) *reg = g;
00315
00316         if (find(dr))
00317             return Rm::Detached_ds | Rm::Detach_again;
00318         else
00319             return Rm::Detached_ds;
00320     }
00321     else if (dr.start() <= g.start())
00322     {
00323         // move the start of a region
00324
00325         if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::F::Detach_free))
00326             h.free(0, dr.end() + 1 - g.start());
00327
00328         unsigned long sz = dr.end() + 1 - g.start();
00329         Item *cn = const_cast<Item*>((Item const *)r);
00330         cn->first = Region(dr.end() + 1, g.end());
00331         cn->second = cn->second + sz;
00332         if (hdlr) *hdlr = Hdlr();
00333         if (reg) *reg = Region(g.start(), dr.end());
00334         if (find(dr))
00335             return Rm::Kept_ds | Rm::Detach_again;
00336         else
00337             return Rm::Kept_ds;
00338     }
00339     else if (dr.end() >= g.end())
00340     {
00341         // move the end of a region
00342
00343         if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::F::Detach_free))
00344             h.free(dr.start() - g.start(), g.end() + 1 - dr.start());
00345
00346         Item *cn = const_cast<Item*>((Item const *)r);
00347         cn->first = Region(g.start(), dr.start() - 1);

```

```

00348     if (hdlr) *hdlr = Hdlr();
00349     if (reg) *reg = Region(dr.start(), g.end());
00350
00351     if (find(dr))
00352         return Rm::Kept_ds | Rm::Detach_again;
00353     else
00354         return Rm::Kept_ds;
00355     }
00356     else if (g.contains(dr))
00357     {
00358         // split a single region that contains the new region
00359
00360         if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::F::Detach_free))
00361             h.free(dr.start() - g.start(), dr.size());
00362
00363         // first move the end off the existing region before the new one
00364         const_cast<Item*>((Item const *)r)->first = Region(g.start(), dr.start()-1);
00365
00366         int err;
00367
00368         // insert a second region for the remaining tail of
00369         // the old existing region
00370         err = _rm.insert(Region(dr.end() + 1, g.end()), h + (dr.end() + 1 - g.start())).second;
00371
00372         if (err)
00373             return err;
00374
00375         if (hdlr) *hdlr = h;
00376         if (reg) *reg = dr;
00377         return Rm::Split_ds;
00378     }
00379     return -L4_ENOENT;
00380 }
00381
00382 l4_addr_t find_free(l4_addr_t start, l4_addr_t end, l4_addr_t size,
00383                    unsigned char align, L4Re::Rm::Flags flags) const noexcept;
00384
00385 };
00386
00387
00388 template< typename Hdlr, template<typename T> class Alloc >
00389 l4_addr_t
00390 Region_map<Hdlr, Alloc>::find_free(l4_addr_t start, l4_addr_t end,
00391                                   unsigned long size, unsigned char align, L4Re::Rm::Flags flags) const noexcept
00392 {
00393     l4_addr_t addr = start;
00394
00395     if (addr == ~0UL || addr < min_addr() || addr >= end)
00396         addr = min_addr();
00397
00398     addr = l4_round_size(addr, align);
00399     Node r;
00400
00401     for(;;)
00402     {
00403         if (addr > 0 && addr - 1 > end - size)
00404             return L4_INVALID_ADDR;
00405
00406         Region c(addr, addr + size - 1);
00407         r = _rm.find_node(c);
00408
00409         if (!r)
00410         {
00411             if (!(flags & L4Re::Rm::F::In_area) && (r = _am.find_node(c)))
00412             {
00413                 if (r->first.end() > end - size)
00414                     return L4_INVALID_ADDR;
00415
00416                 addr = l4_round_size(r->first.end() + 1, align);
00417                 continue;
00418             }
00419             break;
00420         }
00421         else if (r->first.end() > end - size)
00422             return L4_INVALID_ADDR;
00423
00424         addr = l4_round_size(r->first.end() + 1, align);
00425     }
00426
00427     if (!r)
00428         return addr;
00429
00430     return L4_INVALID_ADDR;
00431 }
00432
00433 }

```

16.374 region_mapping_svr_2

```

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 <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
00033 public:
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         if (!(flags & Rm::F::Reserved))
00041         {
00042             if (long r = rm()->validate_ds(static_cast<DERIVED*>(this)->server_iface(), ds_cap,
00043             flags.region_flags(), &ds))
00044                 return r;
00045         }
00046         size = l4_round_page(size);
00047         l4_addr_t start = l4_trunc_page(_start);
00048         if (size < L4_PAGESIZE)
00049             return -L4_EINVAL;
00050         Rm::Region_flags r_flags = flags.region_flags();
00051         Rm::Attach_flags a_flags = flags.attach_flags();
00052         start = l4_addr_t(rm()->attach((void*)start, size,
00053                                     typename DERIVED::Region_handler(ds, client_cap_idx, offs,
00054                                     r_flags),
00055                                     a_flags, align));
00056         if (start == L4_INVALID_ADDR)
00057             return -L4_EADDRNOTAVAIL;
00058         _start = start;
00059         return L4_EOK;
00060     }
00061
00062     long op_free_area(L4Re::Rm::Rights, l4_addr_t start)
00063     {
00064         if (!rm()->detach_area(start))
00065             return -L4_ENOENT;
00066         return L4_EOK;
00067     }
00068
00069     long op_find(L4Re::Rm::Rights, l4_addr_t &addr, unsigned long &size,
00070                 L4Re::Rm::Flags &flags, L4Re::Rm::Offset &offset,
00071                 L4::Cap<L4Re::Dataspace> &m)
00072     {
00073         if (!DERIVED::Have_find)
00074             return -L4_EPERM;
00075         Rm::Flags flag_area { 0 };

```

```

00098
00099     typename DERIVED::Node r = rm()->find(Region(addr, addr + size - 1));
00100     if (!r)
00101     {
00102         r = rm()->area_find(Region(addr, addr + size - 1));
00103         if (!r)
00104             return -L4_ENOENT;
00105         flag_area = Rm::F::In_area;
00106     }
00107
00108     addr = r->first.start();
00109     size = r->first.end() + 1 - addr;
00110
00111     flags = r->second.flags() | flag_area;
00112     offset = r->second.offset();
00113     m = L4::Cap<L4Re::Dataspace>(DERIVED::find_res(r->second.memory()));
00114     return L4_EOK;
00115 }
00116
00120 long op_detach(L4Re::Rm::Rights, l4_addr_t addr,
00121               unsigned long size, unsigned flags,
00122               l4_addr_t &start, l4_addr_t &rsize,
00123               l4_cap_idx_t &mem_cap)
00124 {
00125     Region r;
00126     typename DERIVED::Region_handler h;
00127     int err = rm()->detach((void*)addr, size, flags, &r, &h);
00128     if (err < 0)
00129     {
00130         start = rsize = mem_cap = 0;
00131         return err;
00132     }
00133
00134     if (r.invalid())
00135     {
00136         start = rsize = mem_cap = 0;
00137         return -L4_ENOENT;
00138     }
00139
00140     start = r.start();
00141     rsize = r.size();
00142     mem_cap = h.client_cap_idx();
00143     return err;
00144 }
00145
00149 long op_reserve_area(L4Re::Rm::Rights, l4_addr_t &start, unsigned long size,
00150                     L4Re::Rm::Flags flags, unsigned char align)
00151 {
00152     start = rm()->attach_area(start, size, flags, align);
00153     if (start == L4_INVALID_ADDR)
00154         return -L4_EADDRNOTAVAIL;
00155     return L4_EOK;
00156 }
00157
00161 long op_get_regions(L4Re::Rm::Rights, l4_addr_t addr,
00162                    L4::Ipc::Ret_array<L4Re::Rm::Region> regions)
00163 {
00164     typename DERIVED::Node r;
00165     unsigned num = 0;
00166     while ((r = rm()->lower_bound(Region(addr))))
00167     {
00168         Rm::Region &x = regions.value[num];
00169         x.start = r->first.start();
00170         x.end = r->first.end();
00171
00172         if (++num >= regions.max)
00173             break;
00174
00175         if (x.end >= rm()->max_addr())
00176             break;
00177         addr = x.end + 1;
00178     }
00179     return num;
00180 }
00181
00185 long op_get_areas(L4Re::Rm::Rights, l4_addr_t addr,
00186                  L4::Ipc::Ret_array<L4Re::Rm::Area> areas)
00187 {
00188     typename DERIVED::Node r;
00189     unsigned num = 0;
00190     while ((r = rm()->lower_bound_area(Region(addr))))
00191     {
00192         Rm::Area &x = areas.value[num];
00193         x.start = r->first.start();
00194         x.end = r->first.end();
00195
00196         if (++num >= areas.max)

```



```

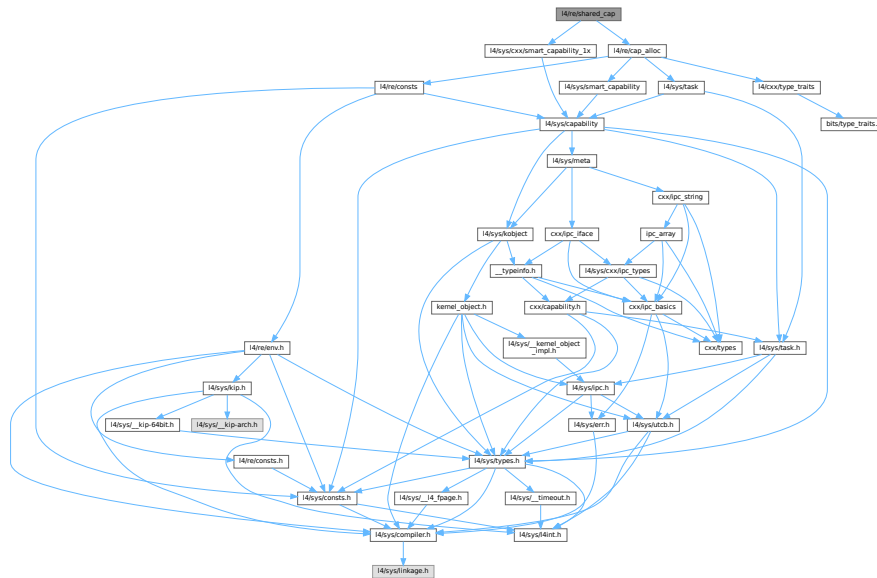
00197         break;
00198
00199         if (x.end >= rm()->max_addr())
00200             break;
00201
00202         addr = x.end + 1;
00203     }
00204     return num;
00205 }
00206
00207 private:
00208     static void pager_set_result(L4::Ipc::Opt<L4::Ipc::Snd_fpage> *fp,
00209                                 L4::Ipc::Snd_fpage const &f)
00210     { *fp = f; }
00211
00212     static void pager_set_result(L4::Ipc::Opt<L4::Ipc::Snd_fpage> *, ...)
00213     {}
00214 public:
00215
00219     long op_io_page_fault(L4::Io_pager::Rights, l4_fpage_t, l4_umword_t,
00220                           L4::Ipc::Opt<L4::Ipc::Snd_fpage> &)
00221     {
00222         // generate exception
00223         return -L4_ENOMEM;
00224     }
00225
00226     long op_page_fault(L4::Pager::Rights, l4_umword_t addr, l4_umword_t pc,
00227                        L4::Ipc::Opt<L4::Ipc::Snd_fpage> &fp)
00228     {
00229         Dbg(Dbg::Server).printf("page fault: %lx pc=%lx\n", addr, pc);
00230
00231         bool need_w = addr & 2;
00232         bool need_x = addr & 4;
00233
00234         typename DERIVED::Node n = rm()->find(addr);
00235
00236         if (!n || !n->second.memory())
00237         {
00238             Dbg(Dbg::Warn, "rm").printf("unhandled %s page fault at 0x%lx pc=0x%lx\n",
00239                                         need_w ? "write" :
00240                                         need_x ? "instruction" : "read", addr, pc);
00241             // generate exception
00242             return -L4_ENOMEM;
00243         }
00244
00245         if (!(n->second.flags() & L4Re::Rm::F::W) && need_w)
00246         {
00247             Dbg(Dbg::Warn, "rm").printf("write page fault in readonly region at 0x%lx pc=0x%lx\n",
00248                                         addr, pc);
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             // generate exception
00258             return -L4_EACCESS;
00259         }
00260
00261         typename DERIVED::Region_handler::Ops::Map_result map_res;
00262         if (int err = n->second.map(addr, n->first, need_w, &map_res))
00263         {
00264             Dbg(Dbg::Warn, "rm").printf("mapping for page fault failed with error %d at 0x%lx pc=0x%lx\n",
00265                                         err, addr, pc);
00266             // generate exception
00267             return -L4_ENOMEM;
00268         }
00269
00270         pager_set_result(&fp, map_res);
00271         return L4_EOK;
00272     }
00273 };
00274
00275 {}

```

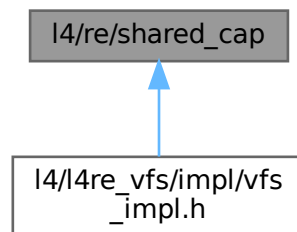
16.375 l4/re/shared_cap File Reference

Shared_cap / Shared_del_cap.

```
#include <l4/re/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.

Typedefs

- `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.

16.375.1 Detailed Description

Shared_cap / Shared_del_cap.

Definition in file [shared_cap](#).

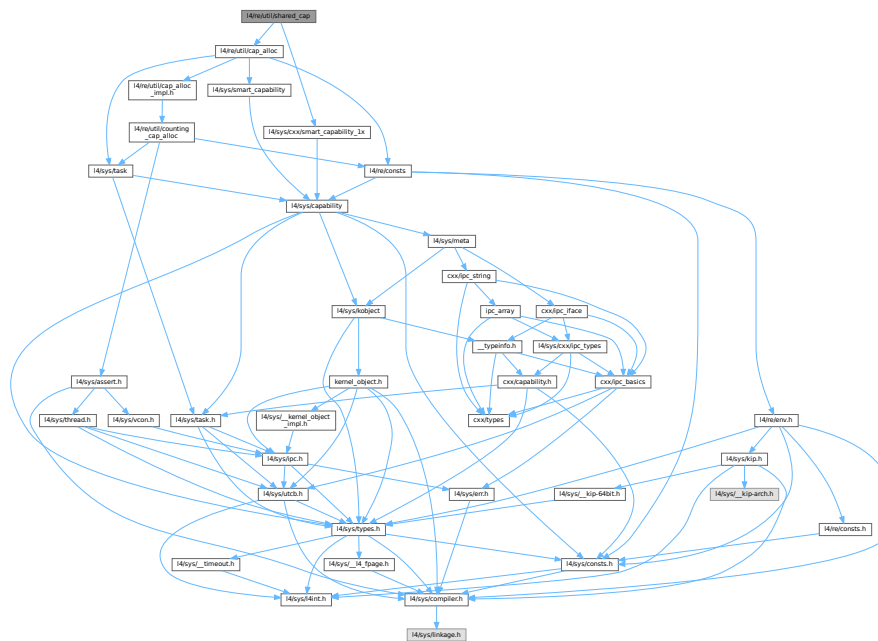
16.376 shared_cap

[Go to the documentation of this file.](#)

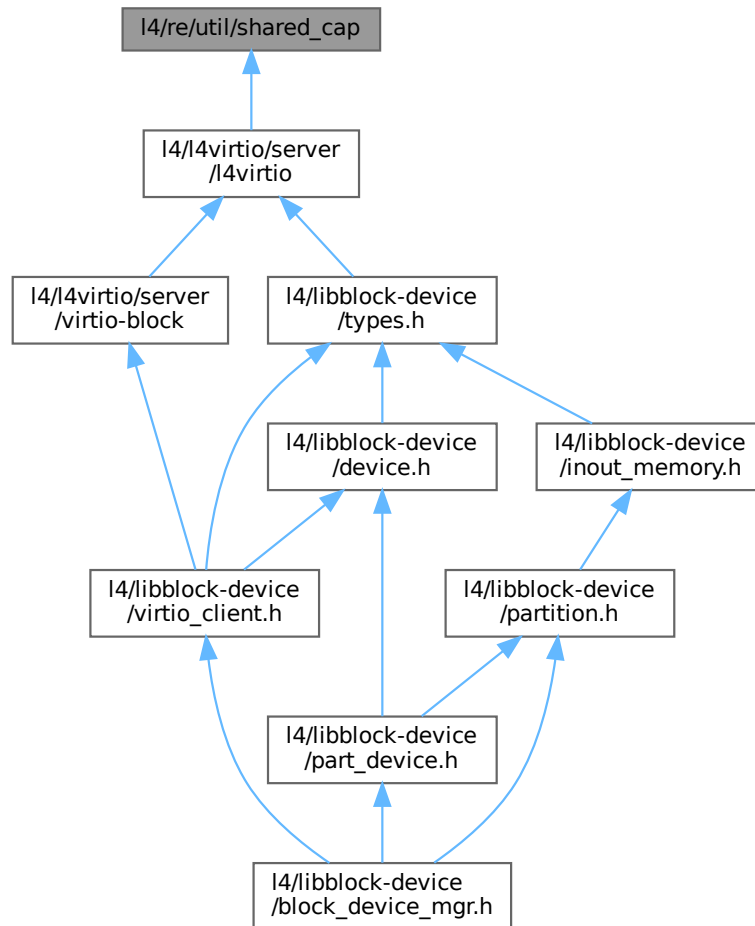
```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2018 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
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_lx>
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>
```

16.377 I4/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.

16.377.1 Detailed Description

Shared_cap / Shared_del_cap.

Definition in file [shared_cap](#).

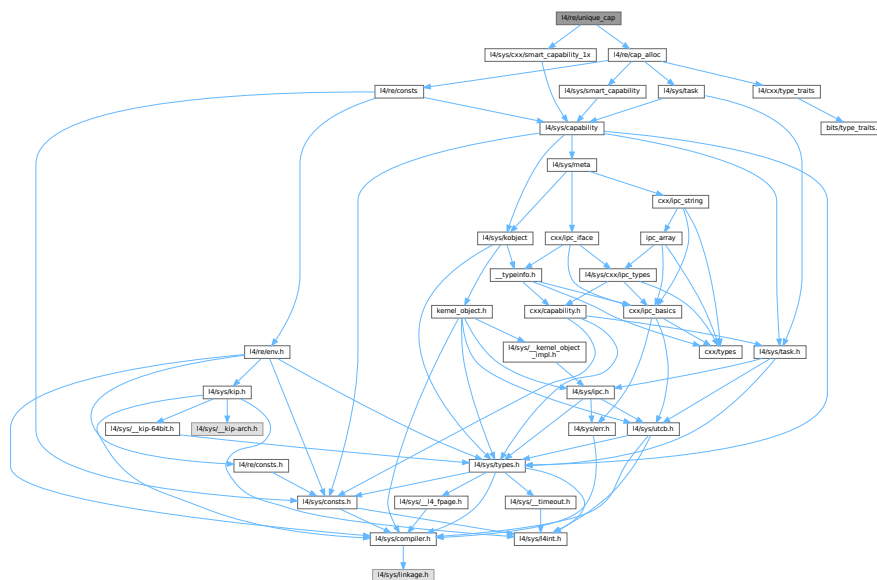
16.378 shared_cap

[Go to the documentation of this file.](#)

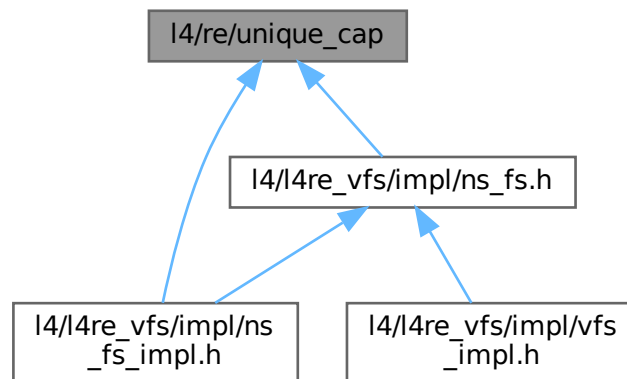
```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
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_1x>
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
```

16.379 I4/re/unique_cap File Reference

```
#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](#).

16.379.1 Detailed Description

Unique_cap / Unique_del_cap.

Definition in file [unique_cap](#).

16.380 unique_cap

[Go to the documentation of this file.](#)

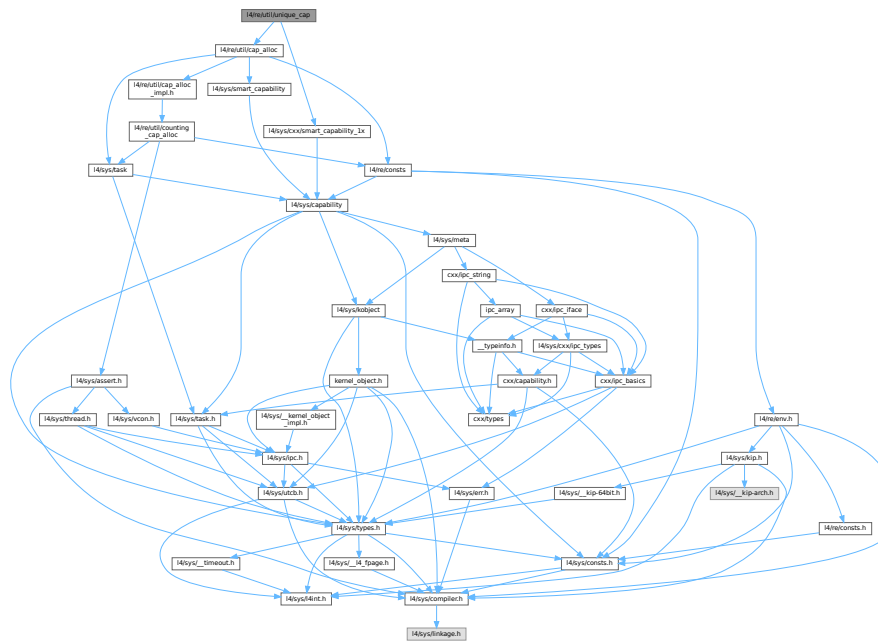
```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
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/cap_alloc>
00026 #include <l4/sys/cxx/smart_capability_1x>
00027
00028 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 }
```

16.381 l4/re/util/unique_cap File Reference

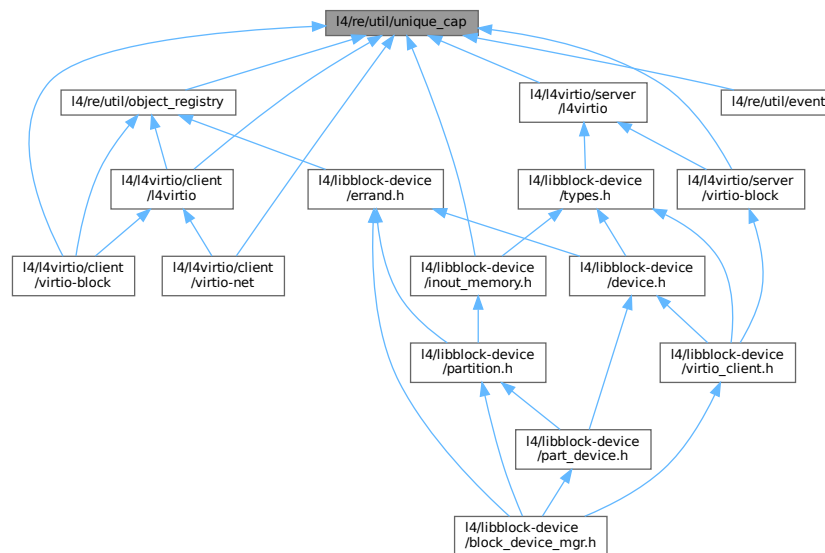
Unique_cap / Unique_del_cap.

```
#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.

16.381.1 Detailed Description

Unique_cap / Unique_del_cap.

Definition in file [unique_cap](#).

16.382 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

```

16.383 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
00027 namespace L4Re { namespace Util {
00028
00045 template< typename SVR >
00046 class Vcon_svr
00047 {
00048 public:
00049     L4_RPC_LEGACY_DISPATCH(L4::Vcon);
00050
00051     l4_msgtag_t op_dispatch(l4_utcb_t *utcb, l4_msgtag_t tag, L4::Vcon::Rights)
00052     {
00053         l4_msgregs_t *m = l4_utcb_mr_u(utcb);
00054         L4::Opcode op = m->mr[0];
00055
00056         switch (op)
00057         {
00058             case L4_VCON_WRITE_OP:
00059                 if (tag.words() < 3)
00060                     return l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00061
00062                 this_vcon()->vcon_write((char const *) &m->mr[2], m->mr[1]);
00063                 return l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00064
00065             case L4_VCON_SET_ATTR_OP:
00066                 if (tag.words() < 4)
00067                     return l4_msgtag(-L4_EINVAL, 0, 0, 0);
00068
00069                 return l4_msgtag(this_vcon()->vcon_set_attr((l4_vcon_attr_t const *) &m->mr[1]),
00070                                0, 0, 0);
00071

```

```

00071
00072     case L4_VCON_GET_ATTR_OP:
00073         return l4_msgtag(this_vcon()->vcon_get_attr((l4_vcon_attr_t *)&m->mr[1]),
00074             4, 0, 0);
00075
00076     default:
00077         break;
00078     }
00079
00080     unsigned size = op » 16;
00081
00082     if (size > (L4_UTCB_GENERIC_DATA_SIZE - 1) * sizeof(l4_utcb_mr()->mr[0]))
00083         size = (L4_UTCB_GENERIC_DATA_SIZE - 1) * sizeof(l4_utcb_mr()->mr[0]);
00084
00085     char buf[size];
00086     // Hmm, could we avoid the double copy here?
00087     l4_umword_t v = this_vcon()->vcon_read(buf, size);
00088     unsigned bytes = v & L4_VCON_READ_SIZE_MASK;
00089
00090     if (bytes < size)
00091         v |= L4_VCON_READ_STAT_DONE;
00092
00093     m->mr[0] = v;
00094     __builtin_memcpy(&m->mr[1], buf, bytes);
00095
00096     return l4_msgtag(0,
00097         (bytes + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t) + 1,
00098         0, 0);
00099 }
00100
00101 unsigned vcon_read(char *buf, unsigned size) noexcept;
00102 void vcon_write(const char *buf, unsigned size) noexcept;
00103 int vcon_set_attr(l4_vcon_attr_t const *) noexcept
00104 { return -L4_EOK; }
00105 int vcon_get_attr(l4_vcon_attr_t *attr) noexcept
00106 {
00107     attr->l_flags = attr->o_flags = attr->i_flags = 0;
00108     return -L4_EOK;
00109 }
00110
00111 private:
00112     SVR const *this_vcon() const { return static_cast<SVR const *>(this); }
00113     SVR *this_vcon() { return static_cast<SVR *>(this); }
00114 };
00115
00116 }}

```

16.384 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/video/goos>
00023
00024 namespace L4Re { namespace Util { namespace Video {
00025
00026     class Goos_fb
00027     {
00028     private:
00029         L4::Cap<L4Re::Video::Goos> _goos;
00030         L4Re::Video::View _view;
00031         L4::Cap<L4Re::Dataspace> _buffer;
00032
00033     public:
00034         enum Flags

```

```

00035     F_dyn_buffer = 0x01,
00036     F_dyn_view   = 0x02,
00037     F_dyn_goos   = 0x04,
00038 };
00039 unsigned _flags;
00040
00041 unsigned _buffer_index;
00042
00043 private:
00044     void init();
00045
00046     Goos_fb(Goos_fb const &);
00047     void operator = (Goos_fb const &);
00048
00049 public:
00050     Goos_fb() : _goos(L4_INVALID_CAP), _buffer(L4_INVALID_CAP), _flags(0) {}
00051
00052     explicit Goos_fb(L4::Cap<L4Re::Video::Goos> goos);
00053     explicit Goos_fb(char const *name);
00054
00055     void setup(L4::Cap<L4Re::Video::Goos> goos);
00056     void setup(char const *name);
00057
00058     ~Goos_fb();
00059
00060     int view_info(L4Re::Video::View::Info *info)
00061     { return _view.info(info); }
00062
00063     L4Re::Video::View const *view() const { return &_amp;view; }
00064     L4Re::Video::View *view() { return &_amp;view; }
00065
00066     L4::Cap<L4Re::Dataspace> buffer() const { return _buffer; }
00067     void *attach_buffer();
00068
00069     int refresh(int x, int y, int w, int h)
00070     { return _view.refresh(x, y, w, h); }
00071
00072     L4::Cap<L4Re::Video::Goos> goos() const { return _goos; }
00073 };
00074 }}}

```

16.385 goos_svr

```

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/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 }}}

```

16.386 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 » (unsigned long)_shift)
00077             & ~(~0UL « (unsigned long)_bits)) « (unsigned long)(16 - _bits);
00078     }
00079
00085     long unsigned set(int v) const
00086     { return (v » (unsigned long)(16 - _bits)) « (unsigned long)_shift; }
00087
00093     template< typename OUT >
00094     void dump(OUT &s) const
00095     {
00096         s.printf("%d(%d)", (int)size(), (int)shift());
00097     }
00098 } __attribute__((packed));
00099
00107 class L4_EXPORT Pixel_info
00108 {
00109 private:
00110     Color_component _r, _g, _b, _a;
00111     unsigned char _bpp;
00112
00113 public:
00118     Color_component const &r() const { return _r; }
00119
00124     Color_component const &g() const { return _g; }
00125
00130     Color_component const &b() const { return _b; }
00131
00136     Color_component const &a() const { return _a; }
00137
00144     Color_component const padding() const
00145     {
00146         unsigned char top_bit = cxx::max<unsigned char>(_r.size() + _r.shift(),
00147             _g.size() + _g.shift());
00148         top_bit = cxx::max<unsigned char>(top_bit, _b.size() + _b.shift());
00149         top_bit = cxx::max<unsigned char>(top_bit, _a.size() + _a.shift());
00150
00151         unsigned char bits = _bpp * 8;

```



```

00152     if (top_bit < bits)
00153         return Color_component(bits - top_bit, top_bit);
00154     return Color_component(0, 0);
00155 }
00156 unsigned char bytes_per_pixel() const { return _bpp; }
00157
00158 unsigned char bits_per_pixel() const
00159 { return _r.size() + _g.size() + _b.size() + _a.size(); }
00160
00161 bool has_alpha() const { return _a.size() > 0; }
00162
00163 void r(Color_component const &c) { _r = c; }
00164
00165 void g(Color_component const &c) { _g = c; }
00166
00167 void b(Color_component const &c) { _b = c; }
00168
00169 void a(Color_component const &c) { _a = c; }
00170
00171 void bytes_per_pixel(unsigned char bpp) { _bpp = bpp; }
00172
00173 Pixel_info() = default;
00174
00175 Pixel_info(unsigned char bpp, char r, char rs, char g, char gs,
00176            char b, char bs, char a = 0, char as = 0)
00177 : _r(r, rs), _g(g, gs), _b(b, bs), _a(a, as), _bpp(bpp)
00178 {}
00179
00180 template<typename VBI>
00181 explicit Pixel_info(VBI const *vbi)
00182 : _r(vbi->red_mask_size, vbi->red_field_position),
00183   _g(vbi->green_mask_size, vbi->green_field_position),
00184   _b(vbi->blue_mask_size, vbi->blue_field_position),
00185   _bpp((vbi->bits_per_pixel + 7) / 8)
00186 {}
00187
00188 bool operator == (Pixel_info const &o) const
00189 {
00190     return _r == o._r && _g == o._g && _b == o._b && _a == o._a && _bpp == o._bpp;
00191 }
00192
00193 template< typename OUT >
00194 void dump(OUT &s) const
00195 {
00196     s.printf("RGBA(%d):%d(%d):%d(%d):%d(%d):%d(%d)",
00197             (int)bytes_per_pixel(),
00198             (int)r().size(), (int)r().shift(),
00199             (int)g().size(), (int)g().shift(),
00200             (int)b().size(), (int)b().shift(),
00201             (int)a().size(), (int)a().shift());
00202 }
00203 };
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274

```

16.387 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;
00097         unsigned view_index;
00098
00099         unsigned long xpos;
00100         unsigned long ypos;
00101         unsigned long width;
00102         unsigned long height;
00103         unsigned long buffer_offset;
00104         unsigned long bytes_per_line;
00105         Pixel_info pixel_info;
00106         unsigned buffer_index;
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<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

```

```

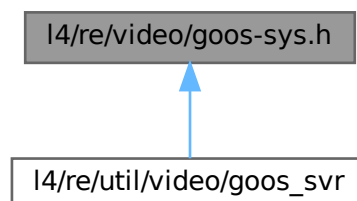
00352 // those are used by the View
00353 L4_INLINE_RPC(long, view_info, (unsigned index, View::Info *info));
00354 L4_INLINE_RPC(long, set_view_info, (unsigned index, View::Info const &info));
00355 L4_INLINE_RPC(long, view_stack, (unsigned index, unsigned pivot, bool behind));
00356 L4_INLINE_RPC(long, 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 }

```

16.388 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.

16.388.1 Detailed Description

Goos protocol definition.

Definition in file [goos-sys.h](#).

16.389 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 }}
```

16.390 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

```

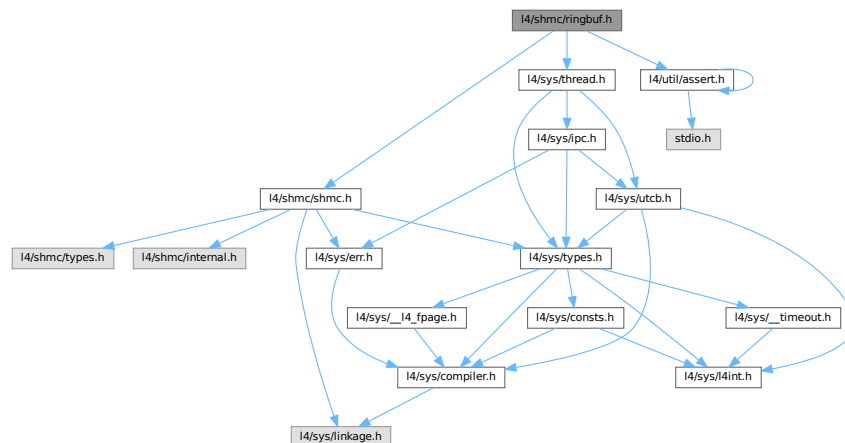
16.391 l4/shmc/ringbuf.h File Reference

```

#include <l4/shmc/shmc.h>
#include <l4/util/assert.h>
#include <l4/sys/thread.h>

```

Include dependency graph for ringbuf.h:



Data Structures

- struct [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.

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.

16.391.1 Macro Definition Documentation

16.391.1.1 L4SHMC_RINGBUF_DATA

```
#define L4SHMC_RINGBUF_DATA(  
    ringbuf ) (L4SHMC_RINGBUF_HEAD(ringbuf)->data)
```

Get ring buffer data pointer.

Parameters

<i>ringbuf</i>	l4shmc_ringbuf_t struct
----------------	---

Definition at line 113 of file [ringbuf.h](#).

16.391.1.2 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>	l4shmc_ringbuf_t struct
----------------	---

Definition at line 122 of file [ringbuf.h](#).

16.391.1.3 L4SHMC_RINGBUF_HEAD

```
#define L4SHMC_RINGBUF_HEAD(  
    ringbuf ) ((l4shmc_ringbuf_head_t*)((ringbuf)->_addr))
```

Get ring buffer head pointer.

Parameters

<i>ringbuf</i>	l4shmc_ringbuf_t struct
----------------	---

Definition at line 104 of file [ringbuf.h](#).

16.391.2 Function Documentation

16.391.2.1 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

16.391.2.2 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

16.391.2.3 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
------------	-------------------------------

16.391.2.4 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

16.391.2.5 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

16.391.2.6 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

16.391.2.7 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
------------	-------------------------------

16.391.2.8 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

16.391.2.9 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

16.391.2.10 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
-------------	-------------------------------

16.391.2.11 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
------------	-------------------------------

16.391.2.12 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

16.391.2.13 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

16.392 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

```

16.393 l4/shmc/shmc.h File Reference

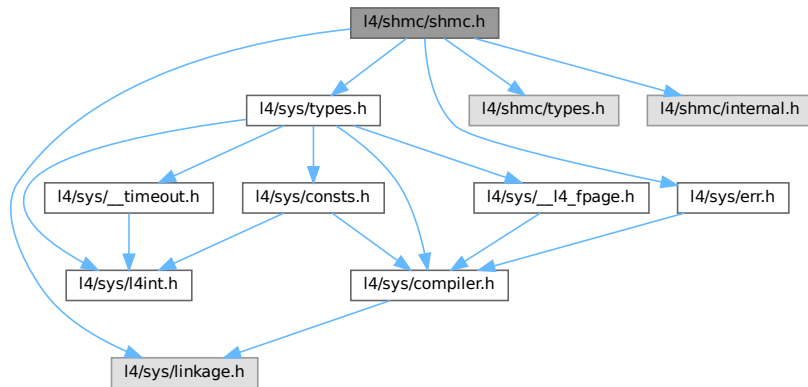
Shared memory library header file.

```

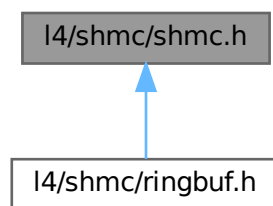
#include <l4/sys/linkage.h>
#include <l4/sys/types.h>

```

```
#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.*

16.393.1 Detailed Description

Shared memory library header file.

Definition in file [shmc.h](#).

16.394 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 *shmc_name);
00088
00110 L4_CV long
00111 l4shmc_attach(char const *shmc_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

```

16.395 l4/sigma0/sigma0.h File Reference

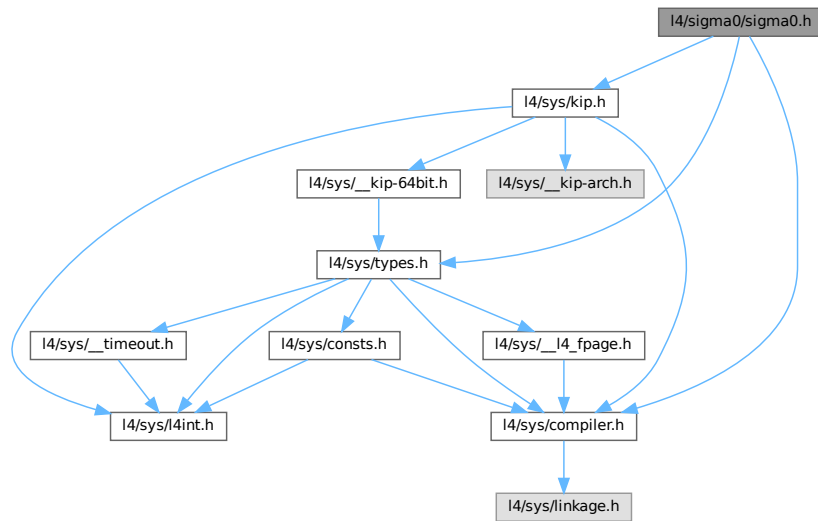
Sigma0 interface.

```

#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.

16.395.1 Detailed Description

Sigma0 interface.

Definition in file [sigma0.h](#).

16.396 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(dl) \
00054     ((dl & 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 */

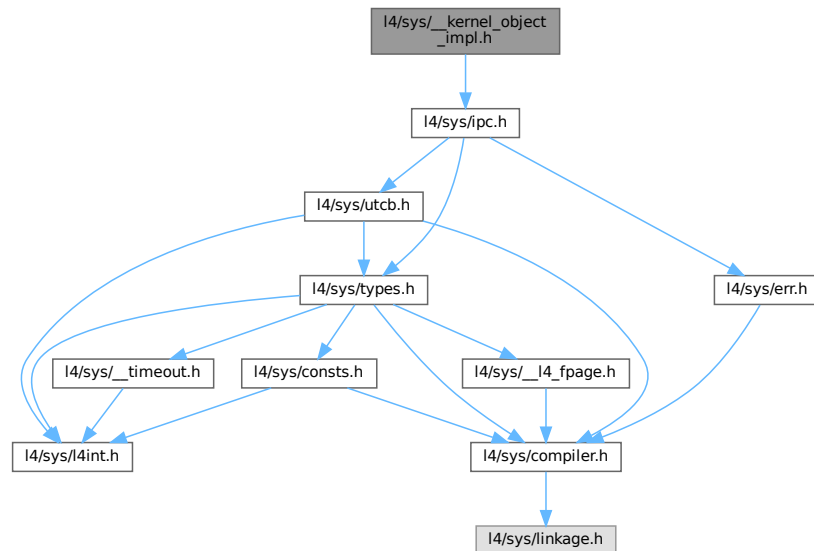
```

16.397 l4/sys/__kernel_object_impl.h File Reference

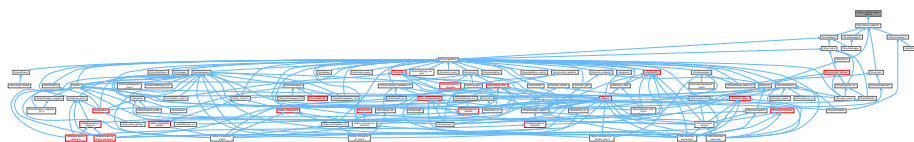
Low-level kernel debugger functions.

```
#include <l4/sys/ipc.h>
```

Include dependency graph for __kernel_object_impl.h:



This graph shows which files directly or indirectly include this file:



16.397.1 Detailed Description

Low-level kernel debugger functions.

Definition in file [__kernel_object_impl.h](#).

16.398 __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

```

16.399 __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       fill1[3];
00047
00048     /* the following stuff is undocumented; we assume that the kernel
00049      * info page is located at offset 0x1000 into the L4 kernel boot
00050      * image so that these declarations are consistent with section 2.9
00051      * of the L4 Reference Manual */
00052
00053     /* offset 0x10 */
00054     /* Kernel debugger */
00055     l4_umword_t      scheduler_granularity;
00056     l4_umword_t      _res00[3];
00057
00058     /* offset 0x20 */
00059     /* Sigma0 */
00060     l4_umword_t      sigma0_esp;
00061     l4_umword_t      sigma0_eip;
00062     l4_umword_t      _res01[2];
00063
00064     /* offset 0x30 */
00065     /* Sigma1 */
00066     l4_umword_t      sigma1_esp;
00067     l4_umword_t      sigma1_eip;
00068     l4_umword_t      _res02[2];
00069
00070     /* offset 0x40 */

```

```

00073  /* Root task */
00074  l4_umword_t      root_esp;
00075  l4_umword_t      root_eip;
00076  l4_umword_t      _res03[2];
00077
00078  /* offset 0x50 */
00079  /* L4 configuration */
00080  l4_umword_t      _res50[1];
00081  l4_umword_t      mem_info;
00082  l4_umword_t      _res58[2];
00083
00084  /* offset 0x60 */
00085  l4_umword_t      _res04[16];
00086
00087  /* offset 0xA0 */
00088  volatile l4_cpu_time_t _clock_val;
00089  l4_umword_t      _res05[2];
00090
00091  /* offset 0xB0 */
00092  l4_umword_t      frequency_cpu;
00093  l4_umword_t      frequency_bus;
00094
00095  /* offset 0xB8 */
00096  l4_umword_t      _res06[10];
00097
00098  /* offset 0xE0 */
00099  l4_umword_t      user_ptr;
00100  l4_umword_t      vhw_offset;
00101  l4_umword_t      _res07[2];
00102
00103  /* offset 0xF0 */
00104  struct l4_kip_platform_info platform_info;
00105 } l4_kernel_info_t;

```

16.400 __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;
00040     l4_uint64_t      version;
00041     l4_uint8_t      offset_version_strings;
00042     l4_uint8_t      fill2[7];
00043     l4_uint8_t      kip_sys_calls;
00044     l4_uint8_t      fill3[7];
00045
00046     /* the following stuff is undocumented; we assume that the kernel
00047     info page is located at offset 0x1000 into the L4 kernel boot
00048     image so that these declarations are consistent with section 2.9
00049     of the L4 Reference Manual */
00050
00051     /* offset 0x20 */
00052     /* Kernel debugger */
00053     l4_umword_t      scheduler_granularity;
00054     l4_umword_t      _res00[3];
00055
00056     /* offset 0x40 */

```

```

00059  /* Sigma0 */
00060  l4_umword_t      sigma0_esp;
00061  l4_umword_t      sigma0_eip;
00062  l4_umword_t      _res01[2];
00063
00064  /* offset 0x60 */
00065  /* Sigma1 */
00066  l4_umword_t      sigma1_esp;
00067  l4_umword_t      sigma1_eip;
00068  l4_umword_t      _res02[2];
00069
00070  /* offset 0x80 */
00071  /* Root task */
00072  l4_umword_t      root_esp;
00073  l4_umword_t      root_eip;
00074  l4_umword_t      _res03[2];
00075
00076  /* offset 0xA0 */
00077  /* L4 configuration */
00078  l4_umword_t      _res_a0[1];
00079  l4_umword_t      mem_info;
00080  l4_umword_t      _res_b0[2];
00081
00082  /* offset 0xC0 */
00083  l4_umword_t      _res04[16];
00084
00085  /* offset 0x140 */
00086  volatile l4_cpu_time_t _clock_val;
00087  l4_umword_t      _res05[1];
00088  l4_umword_t      frequency_cpu;
00089  l4_umword_t      frequency_bus;
00090
00091  /* offset 0x160 */
00092  l4_umword_t      _res06[12];
00093
00094  /* offset 0x1C0 */
00095  l4_umword_t      user_ptr;
00096  l4_umword_t      vhw_offset;
00097  l4_umword_t      _res07[2];
00098
00099  /* offset 0x1E0 */
00100  struct l4_kip_platform_info platform_info;
00101 } l4_kernel_info_t;

```

16.401 l4/sys/__ktrace-impl.h File Reference

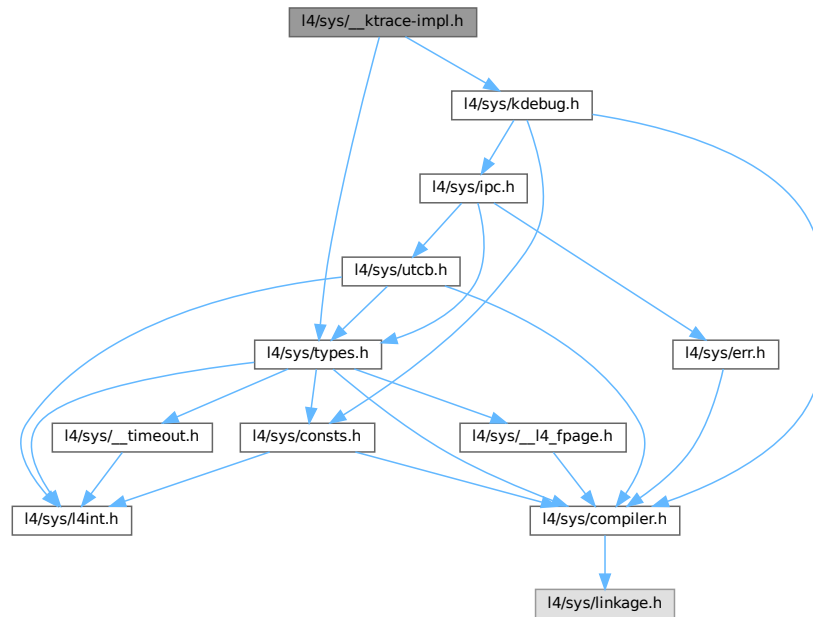
[L4](#) kernel event tracing.

```

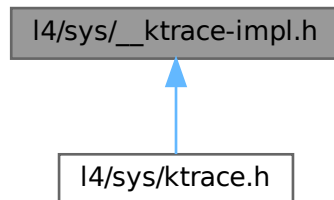
#include <l4/sys/types.h>
#include <l4/sys/kdebug.h>

```

Include dependency graph for `__ktrace-impl.h`:



This graph shows which files directly or indirectly include this file:



Functions

- `I4_umword_t fiasco_tbuf_log` (const char *text)
Create new trace-buffer entry with describing <text>.
- `I4_umword_t fiasco_tbuf_log_3val` (const char *text, `I4_umword_t` v1, `I4_umword_t` v2, `I4_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.
- `I4_umword_t fiasco_tbuf_log_binary` (const unsigned char *data)
Create new trace-buffer entry with binary data.

16.401.1 Detailed Description

[L4](#) kernel event tracing.

Definition in file [__ktrace-impl.h](#).

16.402 __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

```

16.403 __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) & 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 }

```

16.404 __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
00022 l4_task_vgicc_map_u(l4_cap_idx_t task, l4_fpage_t vgicc_fpage,
00023                    l4_utcb_t *u) L4_NOTHROW;
00024
00025 L4_INLINE l4_msgtag_t
00026 l4_task_vgicc_map(l4_cap_idx_t task, l4_fpage_t vgicc_fpage) L4_NOTHROW;
00027
00028 /* IMPLEMENTATION ----- */
00029 #include <l4/sys/ipc.h>
00030
00031 L4_INLINE l4_msgtag_t
00032 l4_task_vgicc_map_u(l4_cap_idx_t task, l4_fpage_t vgicc_fpage,
00033                    l4_utcb_t *u) L4_NOTHROW
00034 {
00035     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00036     v->mr[0] = L4_TASK_MAP_VGICC_ARM_OP;
00037     v->mr[1] = vgicc_fpage.raw;
00038     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0), L4_IPC_NEVER);
00039 }
00040
00041 L4_INLINE l4_msgtag_t
00042 l4_task_vgicc_map(l4_cap_idx_t task, l4_fpage_t vgicc_fpage) L4_NOTHROW
00043 {
00044     return l4_task_vgicc_map_u(task, vgicc_fpage, l4_utcb());
00045 }

```

16.405 __timeout.h

```

00001
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 #ifndef L4_SYS_TIMEOUT_H__
00022 #define L4_SYS_TIMEOUT_H__

```

```

00027
00028 #include <l4/sys/l4int.h>
00029
00047 typedef struct l4_timeout_s {
00048     l4_uint16_t t;
00049 } __attribute__((packed)) l4_timeout_s;
00050
00051
00059 typedef union l4_timeout_t {
00060     l4_uint32_t raw;
00061     struct
00062     {
00063 #ifdef __BIG_ENDIAN__
00064         l4_timeout_s snd;
00065         l4_timeout_s rcv;
00066 #else
00067         l4_timeout_s rcv;
00068         l4_timeout_s snd;
00069 #endif
00070     } p;
00071 } l4_timeout_t;
00072
00073
00079 #define L4_IPC_TIMEOUT_0 ((l4_timeout_s){0x0400})
00080 #define L4_IPC_TIMEOUT_NEVER ((l4_timeout_s){0})
00081 #define L4_IPC_NEVER_INITIALIZER {0}
00082 #define L4_IPC_NEVER ((l4_timeout_t){0})
00083 #define L4_IPC_RECV_TIMEOUT_0 ((l4_timeout_t){0x00000400})
00084 #define L4_IPC_SEND_TIMEOUT_0 ((l4_timeout_t){0x04000000})
00085 #define L4_IPC_BOTH_TIMEOUT_0 ((l4_timeout_t){0x04000400})
00091 #define L4_TIMEOUT_US_NEVER (~0U)
00092
00096 #define L4_TIMEOUT_US_MAX    (~0U - 1)
00097
00109 L4_INLINE
00110 l4_timeout_s l4_timeout_rel(unsigned man, unsigned exp) L4_NOTHROW;
00111
00112
00122 L4_INLINE
00123 l4_timeout_t l4_ipc_timeout(unsigned snd_man, unsigned snd_exp,
00124                             unsigned rcv_man, unsigned rcv_exp) L4_NOTHROW;
00125
00135 L4_INLINE
00136 l4_timeout_t l4_timeout(l4_timeout_s snd, l4_timeout_s rcv) L4_NOTHROW;
00137
00145 L4_INLINE
00146 void l4_snd_timeout(l4_timeout_s snd, l4_timeout_t *to) L4_NOTHROW;
00147
00155 L4_INLINE
00156 void l4_rcv_timeout(l4_timeout_s rcv, l4_timeout_t *to) L4_NOTHROW;
00157
00166 L4_INLINE
00167 l4_kernel_clock_t l4_timeout_rel_get(l4_timeout_s to) L4_NOTHROW;
00168
00169
00178 L4_INLINE
00179 unsigned l4_timeout_is_absolute(l4_timeout_s to) L4_NOTHROW;
00180
00190 L4_INLINE
00191 l4_kernel_clock_t l4_timeout_get(l4_kernel_clock_t cur, l4_timeout_s to) L4_NOTHROW;
00192
00200 L4_INLINE
00201 l4_timeout_s l4_timeout_from_us(l4_uint32_t us) L4_NOTHROW;
00202
00203 /*
00204  * Implementation
00205  */
00206
00207 L4_INLINE
00208 l4_timeout_t l4_ipc_timeout(unsigned snd_man, unsigned snd_exp,
00209                             unsigned rcv_man, unsigned rcv_exp) L4_NOTHROW
00210 {
00211     l4_timeout_t t;
00212     t.p.snd.t = (snd_man & 0x3ff) | ((snd_exp << 10) & 0x7c00);
00213     t.p.rcv.t = (rcv_man & 0x3ff) | ((rcv_exp << 10) & 0x7c00);
00214     return t;
00215 }
00216
00217
00218 L4_INLINE
00219 l4_timeout_t l4_timeout(l4_timeout_s snd, l4_timeout_s rcv) L4_NOTHROW
00220 {
00221     l4_timeout_t t;
00222     t.p.snd = snd;
00223     t.p.rcv = rcv;
00224     return t;
00225 }

```

```

00226
00227
00228 L4_INLINE
00229 void l4_snd_timeout(l4_timeout_s snd, l4_timeout_t *to) L4_NOTHROW
00230 {
00231     to->p.snd = snd;
00232 }
00233
00234
00235 L4_INLINE
00236 void l4_rcv_timeout(l4_timeout_s rcv, l4_timeout_t *to) L4_NOTHROW
00237 {
00238     to->p.rcv = rcv;
00239 }
00240
00241
00242 L4_INLINE
00243 l4_timeout_s l4_timeout_rel(unsigned man, unsigned exp) L4_NOTHROW
00244 {
00245     return (l4_timeout_s){(l4_uint16_t)((man & 0x3ff) | ((exp << 10) & 0x7c00))};
00246 }
00247
00248
00249 L4_INLINE
00250 l4_kernel_clock_t l4_timeout_rel_get(l4_timeout_s to) L4_NOTHROW
00251 {
00252     if (to.t == 0)
00253         return ~0ULL;
00254     return (l4_kernel_clock_t)(to.t & 0x3ff) << ((to.t > 10) & 0x1f);
00255 }
00256
00257
00258 L4_INLINE
00259 unsigned l4_timeout_is_absolute(l4_timeout_s to) L4_NOTHROW
00260 {
00261     return to.t & 0x8000;
00262 }
00263
00264
00265 L4_INLINE
00266 l4_kernel_clock_t l4_timeout_get(l4_kernel_clock_t cur, l4_timeout_s to) L4_NOTHROW
00267 {
00268     if (l4_timeout_is_absolute(to))
00269         return 0; /* We cannot retrieve the value ... */
00270     else
00271         return cur + l4_timeout_rel_get(to);
00272 }
00273
00274 L4_INLINE
00275 l4_timeout_s l4_timeout_from_us(l4_uint32_t us) L4_NOTHROW
00276 {
00277     static_assert(sizeof(us) <= 4,
00278         "Verify the correctness of log2(us) and the number of bits for e!");
00279     l4_timeout_s t;
00280     if (us == 0)
00281         t = L4_IPC_TIMEOUT_0;
00282     else if (us == L4_TIMEOUT_US_NEVER)
00283         t = L4_IPC_TIMEOUT_NEVER;
00284     else
00285     {
00286         /* Here it is certain that at least one bit in 'us' is set. */
00287
00288         unsigned m;
00289         int e = (31 - __builtin_clz(us)) - 9;
00290         if (e < 0) e = 0;
00291
00292         /* Here it is certain that '0 <= e <= 22' and '1 <= 2^e <= 2^22'. */
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          *
00303          * Dividing by (1<e) ensures that for all us < 2^32: m < 2^10.
00304          *
00305          * What about sizeof(us) == 8? 'e = log2(us) - 9':
00306          * o 2^63 <= us <= 2^64-1: e = 54; 512 <= us/2^54 <= 1023.
00307          *
00308          * That means that this function would even work for 64-bit values of
00309          * 'us' as long as __builtin_clz(us) works correctly for that range.
00310          * But the number of bits available for the exponent is limited:
00311          * o bits 0..9 (10 bits) are used for 'm'
00312          * o bits 10..14 (5 bits) are used for 'e'

```

```

00313      * o bit 15 is used to distinguish between absolute timeouts and
00314      * relative timeouts (see l4_timeout_is_absolute())
00315      *
00316      * That means 'e <= 31' and thus it's not possible to encode timeouts
00317      * represented by 64-bit values.
00318      */
00319
00320      t.t = (e << 10) | m;
00321  }
00322  return t;
00323 }
00324
00325 #endif

```

16.406 l4/sys/__typeinfo.h File Reference

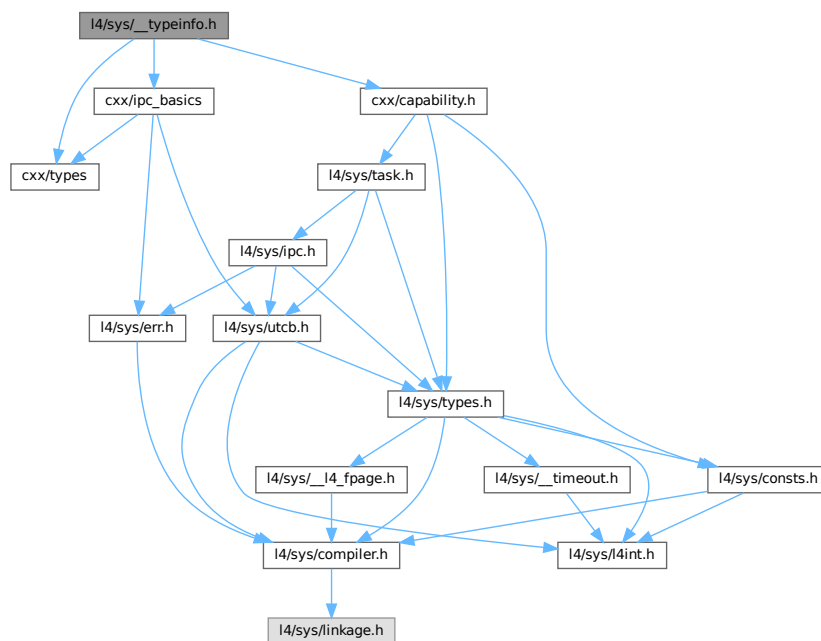
Type information handling.

```

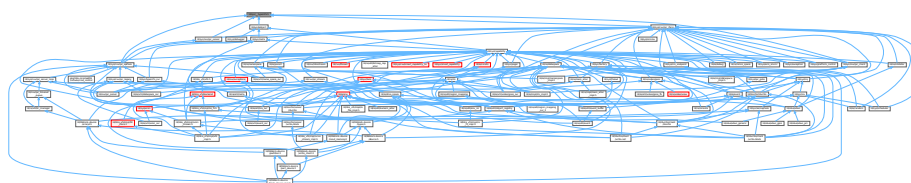
#include "cxx/types"
#include "cxx/ipc_basics"
#include "cxx/capability.h"

```

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.*

16.406.1 Detailed Description

Type information handling.

Definition in file [__typeinfo.h](#).

16.407 __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 nd 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,
00060         PROTO_EMPTY = -19,
00061     };
00062
00077     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
00260
00261
00262

```

```

00263     template< typename L1, typename L2 >
00264     struct Conflict : _Conflict<typename L1::type, typename L2::type> {};
00265
00266     // to be removed -----
00267     // p_dispatch code -- for legacy dispatch -----
00268     /***** */
00269     /*
00270     * \internal
00271     * helper: Dispatch helper for calling server-side p_dispatch() functions.
00272     */
00273     template<typename LIST>
00274     struct _P_dispatch;
00275
00276     // No matching dispatcher found
00277     template<>
00278     struct _P_dispatch<Iface_list_end>
00279     {
00280         template< typename THIS, typename A1, typename A2 >
00281         static int f(THIS *, long, A1, A2 &) noexcept
00282         { return -L4_EBADPROTO; }
00283     };
00284
00285     // call matching p_dispatch() function
00286     template< typename I, typename LIST >
00287     struct _P_dispatch<Iface_list<I, LIST> >
00288     {
00289         // special handling for the meta protocol, to avoid 'using' murx
00290         template< typename THIS, typename A1, typename A2 >
00291         static int _f(THIS self, A1, A2 &a2, True::type)
00292         {
00293             return self->dispatch_meta_request(a2);
00294         }
00295
00296         // normal p_dispatch() dispatching
00297         template< typename THIS, typename A1, typename A2 >
00298         static int _f(THIS self, A1 a1, A2 &a2, False::type)
00299         {
00300             return self->p_dispatch(reinterpret_cast<typename I::iface_type *>(0),
00301                                     a1, a2);
00302         }
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, Bool<I::Proto == (long)L4_PROTO_META>());
00311
00312         return _P_dispatch<typename LIST::type>::f(self, proto, a1, a2);
00313     }
00314 };
00315
00316     template<typename LIST>
00317     struct P_dispatch : _P_dispatch<typename LIST::type> {};
00318 // end: p_dispatch -----
00319 // end: to be removed -----
00320
00321     template<typename RPC> struct Default_op;
00322
00323     namespace Detail {
00324
00325     struct Rpc_end
00326     {
00327         typedef void opcode_type;
00328         typedef Rpc_end rpc;
00329         typedef Rpc_end type;
00330     };
00331
00332     template<typename O1, typename O2, typename RPCS>
00333     struct _Rpc : _Rpc<typename RPCS::next::rpc, O2, typename RPCS::next::type> {};
00334
00335     template<typename O1, typename O2>
00336     struct _Rpc<O1, O2, Rpc_end> {};
00337
00338     template<typename OP, typename RPCS>
00339     struct _Rpc<OP, OP, RPCS> : RPCS
00340     {
00341         typedef _Rpc type;
00342     };
00343
00344     template<typename OP, typename RPCS>
00345     struct Rpc : _Rpc<typename RPCS::rpc, OP, RPCS> {};
00346
00347     template<typename T, unsigned CODE>
00348     struct _Get_opcode
00349     {

```

```

00354     template<bool, typename> struct Invalid_opcode {};
00355     template<typename X> struct Invalid_opcode<true, X>;
00356
00357 private:
00358     template<typename U, U> struct _chk;
00359     template<typename U> static long _opc(_chk<int, U::Opcode> *);
00360     template<typename U> static char _opc(...);
00361
00362     template<unsigned SZ, typename U>
00363     struct _Opc { enum { value = CODE }; };
00364
00365     template<typename U>
00366     struct _Opc<sizeof(long), U> { enum { value = U::Opcode }; };
00367
00368 public:
00369     enum { value = _Opc<sizeof(_opc<T>)(0), T>::value };
00370     Invalid_opcode<(value < CODE), T> invalid_opcode;
00371 };
00372
00373 template<typename OPCODE, unsigned O, typename ...X>
00374 struct _Rpcs : Rpcs_end {};
00375
00376 template<typename OPCODE, unsigned O, typename R, typename ...X>
00377 struct _Rpcs<OPCODE, O, R, X...>
00378 {
00379     typedef _Rpcs type;
00380     typedef OPCODE opcode_type;
00381     typedef R rpc;
00382     typedef typename _Rpcs<OPCODE, _Get_opcode<R, O>::value + 1, X...>::type next;
00383     enum { Opcode = _Get_opcode<R, O>::value };
00384     template<typename Y> struct Rpc : Typeid::Detail::Rpc<Y, _Rpcs> {};
00385 };
00386
00387 template<typename OPCODE, unsigned O, typename R>
00388 struct _Rpcs<OPCODE, O, Default_op<R> >
00389 {
00390     typedef _Rpcs type;
00391     typedef void opcode_type;
00392     typedef R rpc;
00393     typedef Rpcs_end next;
00394     enum { Opcode = -99 };
00395     template<typename Y> struct Rpc : Typeid::Detail::Rpc<Y, _Rpcs> {};
00396 };
00397
00398 } // namespace Detail
00399
00400 template<typename CLASS>
00401 struct Raw_ipc
00402 {
00403     typedef Raw_ipc type;
00404     typedef Detail::Rpcs_end next;
00405     typedef void opcode_type;
00406 };
00407
00408 template<typename ...RPCS>
00409 struct Rpcs : Detail::_Rpcs<L4::Opcode, 0, RPCS...> {};
00410
00411 template<typename OPCODE_TYPE>
00412 struct Rpcs_code
00413 {
00414     template<typename ...RPCS>
00415     struct F : Detail::_Rpcs<OPCODE_TYPE, 0, RPCS...> {};
00416 };
00417
00418 template<typename OPERATION>
00419 struct Rpc_nocode : Detail::_Rpcs<void, 0, OPERATION> {};
00420
00421 template<typename ...ARG>
00422 struct Rpcs_sys : Detail::_Rpcs<l4_umword_t, 0, ARG...> {};
00423
00424 template<typename CLASS>
00425 struct Rights
00426 {
00427     unsigned rights;
00428     Rights(unsigned rights) noexcept : rights(rights) {}
00429     unsigned operator & (unsigned rhs) const noexcept { return rights & rhs; }
00430 };
00431
00432 } // namespace Typeid
00433
00434 struct L4_EXPORT Type_info
00435 {
00436     class L4_EXPORT Demand
00437     {
00438     private:
00439         static unsigned char max(unsigned char a, unsigned char b) noexcept
00440         { return a > b ? a : b; }
00441     };
00442 };

```

```

00522
00523 public:
00524     unsigned char caps;
00525     unsigned char flags;
00526     unsigned char mem;
00527     unsigned char ports;
00528
00536     explicit
00537     Demand(unsigned char caps = 0, unsigned char flags = 0,
00538            unsigned char mem = 0, unsigned char ports = 0) noexcept
00539     : caps(caps), flags(flags), mem(mem), ports(ports) {}
00540
00542     bool no_demand() const noexcept
00543     { return caps == 0 && mem == 0 && ports == 0 && flags == 0; }
00544
00546     Demand operator | (Demand const &rhs) const noexcept
00547     {
00548         return Demand(max(caps, rhs.caps), flags | rhs.flags,
00549                        max(mem, rhs.mem), max(ports, rhs.ports));
00550     }
00551 };
00552
00561     template<unsigned char CAPS = 0, unsigned char FLAGS = 0,
00562             unsigned char MEM = 0, unsigned char PORTS = 0>
00563     struct Demand_t : Demand
00564     {
00565         enum
00566         {
00567             Caps = CAPS,
00568             Flags = FLAGS,
00569             Mem = MEM,
00570             Ports = PORTS
00571         };
00572         Demand_t() noexcept : Demand(CAPS, FLAGS, MEM, PORTS) {}
00573     };
00574
00582     template<typename D1, typename D2>
00583     struct Demand_union_t : Demand_t<__I::Max<D1::Caps, D2::Caps>::Res,
00584                                     D1::Flags | D2::Flags,
00585                                     __I::Max<D1::Mem, D2::Mem>::Res,
00586                                     __I::Max<D1::Ports, D2::Ports>::Res>
00587     {};
00588
00589     L4_std_type_info_ptr _type;
00590     Type_info const *const *_bases;
00591     unsigned _num_bases;
00592     long _proto;
00593
00594     L4_std_type_info_ptr type() const noexcept { return _type; }
00595     Type_info const *base(unsigned idx) const noexcept { return _bases[idx]; }
00596     unsigned num_bases() const noexcept { return _num_bases; }
00597     long proto() const noexcept { return _proto; }
00598     char const *name() const noexcept { return L4_kobject_type_name(type()); }
00599     bool has_proto(long proto) const noexcept
00600     {
00601         if (_proto && _proto == proto)
00602             return true;
00603
00604         if (!proto)
00605             return false;
00606
00607         for (unsigned i = 0; i < _num_bases; ++i)
00608             if (base(i)->has_proto(proto))
00609                 return true;
00610
00611         return false;
00612     }
00613 };
00614
00620     template<typename T> struct Kobject_typeid
00621     {
00632         typedef typename T::__Kobject_typeid::Demand Demand;
00633         typedef typename T::__Iface::iface_type Iface;
00634         typedef typename T::__Iface_list Iface_list;
00635
00640         static Type_info const *id() noexcept { return &T::__Kobject_typeid::_m; }
00641
00649         static Type_info::Demand demand() noexcept
00650         { return T::__Kobject_typeid::Demand(); }
00651
00652         // to be removed -----
00653         // p_dispatch -----
00669         template<typename THIS, typename A1, typename A2>
00670         static int proto_dispatch(THIS *self, long proto, A1 a1, A2 &a2)
00671         { return Typeid::P_dispatch<typename T::__Iface_list>::f(self, proto, a1, a2); }
00672         // p_dispatch -----
00673         // end: to be removed -----

```

```

00674 };
00675
00677 template<> struct Kobject_typeid<void>
00678 {
00679     typedef Type_info::Demand_t<> Demand;
00680 };
00681
00690 template<typename T>
00691 inline
00692 Type_info const *kobject_typeid() noexcept
00693 { return Kobject_typeid<T>::id(); }
00694
00699 #define L4___GEN_TI(t...)
00700 Type_info const t::__Kobject_typeid::_m =
00701 {
00702     L4_KOBJECT_META_RTII(Derived),
00703     &t::__Kobject_typeid::_b[0],
00704     sizeof(t::__Kobject_typeid::_b) / sizeof(t::__Kobject_typeid::_b[0]),
00705     PROTO
00706 }
00707
00712 #define L4___GEN_TI_MEMBERS(BASE_DEMAND...)
00713 private:
00714     template< typename T > friend struct Kobject_typeid;
00715 protected:
00716     struct __Kobject_typeid {
00717         typedef Type_info::Demand_union_t<S_DEMAND, BASE_DEMAND> Demand;
00718         static Type_info const *const _b[];
00719         static Type_info const _m;
00720     };
00721 public:
00722     static long const Protocol = PROTO;
00723     typedef L4::Typeid::Rights<Class> Rights;
00724
00753 template<
00754     typename Derived,
00755     typename Base,
00756     long PROTO = PROTO_ANY,
00757     typename S_DEMAND = Type_info::Demand_t<>
00758 >
00759 class Kobject_t : public Base
00760 {
00761 protected:
00762     typedef Derived Class;
00763     typedef Typeid::Iface<PROTO, Derived> __Iface;
00764     typedef Typeid::Merge_list<
00765         Typeid::Iface_list<__Iface>, typename Base::__Iface_list
00766     > __Iface_list;
00767
00772 static void __check_protocols__() noexcept
00773 {
00774     typedef Typeid::Iface_conflict<__Iface, typename Base::__Iface_list> Base_conflict;
00775     static_assert(!Base_conflict::value, "ambiguous protocol ID: protocol also used by Base");
00776 }
00777
00779 L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }
00780
00781 // Generate the remaining type information
00782 L4___GEN_TI_MEMBERS(typename Base::__Kobject_typeid::Demand)
00783 };
00784
00785
00787 template< typename Derived, typename Base, long PROTO, typename S_DEMAND>
00788 Type_info const *const
00789 Kobject_t<Derived, Base, PROTO, S_DEMAND>::
00790     __Kobject_typeid::_b[] = { &Base::__Kobject_typeid::_m };
00791
00792
00797 template< typename Derived, typename Base, long PROTO, typename S_DEMAND>
00798 L4___GEN_TI(Kobject_t<Derived, Base, PROTO, S_DEMAND>);
00799
00800
00830 template<
00831     typename Derived,
00832     typename Base1,
00833     typename Base2,
00834     long PROTO = PROTO_ANY,
00835     typename S_DEMAND = Type_info::Demand_t<>
00836 >
00837 class Kobject_2t : public Base1, public Base2
00838 {
00839 protected:
00840     typedef Derived Class;
00841     typedef Typeid::Iface<PROTO, Derived> __Iface;
00842     typedef Typeid::Merge_list<
00843         Typeid::Iface_list<__Iface>,
00844         Typeid::Merge_list<
00845             typename Base1::__Iface_list,

```

```

00849     typename Base2::__Iface_list
00850     >
00851 > __Iface_list;
00852
00854 static void __check_protocols__() noexcept
00855 {
00856     typedef typename Base1::__Iface_list Base1_proto_list;
00857     typedef typename Base2::__Iface_list Base2_proto_list;
00858
00859     typedef Typeid::Iface_conflict<__Iface, Base1_proto_list> Base1_conflict;
00860     typedef Typeid::Iface_conflict<__Iface, Base2_proto_list> Base2_conflict;
00861     static_assert(!Base1_conflict::value, "ambiguous protocol ID, also in Base1");
00862     static_assert(!Base2_conflict::value, "ambiguous protocol ID, also in Base2");
00863
00864     typedef Typeid::Conflict<Base1_proto_list, Base2_proto_list> Bases_conflict;
00865     static_assert(!Bases_conflict::value, "ambiguous protocol IDs in base classes");
00866 }
00867
00868 // disambiguate cap()
00869 l4_cap_idx_t cap() const noexcept
00870 { return Base1::cap(); }
00871
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 Base1::__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<Base1 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(((Base1*)0)->dec_refcnt(diff, utcb)))
00888     { return Base1::dec_refcnt(diff, utcb); }
00889 };
00890
00891
00893 template< typename Derived, typename Base1, typename Base2,
00894     long PROTO, typename S_DEMAND >
00895 Type_info const *const
00896 Kobject_2t<Derived, Base1, Base2, PROTO, S_DEMAND>::__Kobject_typeid::b[] =
00897 {
00898     &Base1::__Kobject_typeid::_m,
00899     &Base2::__Kobject_typeid::_m
00900 };
00901
00902
00907 template< typename Derived, typename Base1, typename Base2,
00908     long PROTO, typename S_DEMAND >
00909 L4__GEN_TI(Kobject_2t<Derived, Base1, Base2, PROTO, S_DEMAND>);
00910
00911
00912
00932 template<
00933     typename Derived,
00934     typename Base1,
00935     typename Base2,
00936     typename Base3,
00937     long PROTO = PROTO_ANY,
00938     typename S_DEMAND = Type_info::Demand_t<>
00939 >
00940 struct Kobject_3t : Base1, Base2, Base3
00941 {
00942     protected:
00943         typedef Derived Class;
00944         typedef Typeid::Iface<PROTO, Derived> __Iface;
00945         typedef Typeid::Merge_list<
00946             Typeid::Iface_list<__Iface>,
00947             Typeid::Merge_list<
00948                 typename Base1::__Iface_list,
00949                 Typeid::Merge_list<
00950                     typename Base2::__Iface_list,
00951                     typename Base3::__Iface_list
00952                 >
00953             >
00954         > __Iface_list;
00955
00956     >
00957 > __Iface_list;
00958
00960 static void __check_protocols__() noexcept
00961 {
00962     typedef typename Base1::__Iface_list Base1_proto_list;
00963     typedef typename Base2::__Iface_list Base2_proto_list;
00964     typedef typename Base3::__Iface_list Base3_proto_list;
00965
00966     typedef Typeid::Iface_conflict<__Iface, Base1_proto_list> Base1_conflict;

```

```

00967     typedef Typeid::Iface_conflict<__Iface, Base2_proto_list> Base2_conflict;
00968     typedef Typeid::Iface_conflict<__Iface, Base3_proto_list> Base3_conflict;
00969
00970     static_assert(!Base1_conflict::value, "ambiguous protocol ID, also in Base1");
00971     static_assert(!Base2_conflict::value, "ambiguous protocol ID, also in Base2");
00972     static_assert(!Base3_conflict::value, "ambiguous protocol ID, also in Base3");
00973
00974     typedef Typeid::Conflict<Base1_proto_list, Base2_proto_list> Conflict_bases12;
00975     typedef Typeid::Conflict<Base1_proto_list, Base3_proto_list> Conflict_bases13;
00976     typedef Typeid::Conflict<Base2_proto_list, Base3_proto_list> Conflict_bases23;
00977
00978     static_assert(!Conflict_bases12::value, "ambiguous protocol IDs in base classes: Base1 and
Base2");
00979     static_assert(!Conflict_bases13::value, "ambiguous protocol IDs in base classes: Base1 and
Base3");
00980     static_assert(!Conflict_bases23::value, "ambiguous protocol IDs in base classes: Base2 and
Base3");
00981 }
00982
00983 // disambiguate cap()
00984 l4_cap_idx_t cap() const noexcept
00985 { return Base1::cap(); }
00986
00987 L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }
00988
00989 L4__GEN_TI_MEMBERS(Type_info::Demand_union_t<Type_info::Demand_union_t<
00990     typename Base1::__Kobject_typeid::Demand,
00991     typename Base2::__Kobject_typeid::Demand>,
00992     typename Base3::__Kobject_typeid::Demand>
00993 )
00994 )
00995
00996 public:
00997     // Provide non-ambiguous conversion to Kobject
00998     operator Kobject const & () const noexcept
00999     { return *static_cast<Base1 const *>(this); }
01000
01001     // Provide non-ambiguous access of dec_refcnt()
01002     l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
01003     noexcept(noexcept(((Base1*)0)->dec_refcnt(diff, utcb)))
01004     { return Base1::dec_refcnt(diff, utcb); }
01005 };
01006
01007
01008 template< typename Derived, typename Base1, typename Base2, typename Base3,
01009     long PROTO, typename S_DEMAND >
01010 Type_info const *const
01011 Kobject_3t<Derived, Base1, Base2, Base3, PROTO, S_DEMAND>::__Kobject_typeid::b[] =
01012 {
01013     &Base1::__Kobject_typeid::_m,
01014     &Base2::__Kobject_typeid::_m,
01015     &Base3::__Kobject_typeid::_m
01016 };
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 #if __cplusplus >= 201103L
01025
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>

```

```

01064 struct Merge_list<L1, L2> : Typeid::Merge_list<L1, L2> {};
01065
01066 template<typename L1, typename L2, typename ...LISTS>
01067 struct Merge_list<L1, L2, LISTS...> :
01068     Merge_list<typename Typeid::Merge_list<L1, L2>::type, LISTS...> {};
01069
01070 template< typename I, typename ...LIST >
01071 struct Iface_conflict;
01072
01073 template< typename I >
01074 struct Iface_conflict<I> : Typeid::False {};
01075
01076 template< typename I, typename L, typename ...LIST >
01077 struct Iface_conflict<I, L, LIST...> :
01078     Typeid::Bool<Typeid::Iface_conflict<typename I::type, typename L::type>::value
01079         || Iface_conflict<I, LIST...>::value>
01080 {};
01081
01082 template< typename ...LIST >
01083 struct Conflict;
01084
01085 template< typename L >
01086 struct Conflict<L> : Typeid::False {};
01087
01088 template< typename L1, typename L2, typename ...LIST >
01089 struct Conflict<L1, L2, LIST...> :
01090     Typeid::Bool<Typeid::Conflict<typename L1::type, typename L2::type>::value
01091         || Conflict<L1, LIST...>::value
01092         || Conflict<L2, LIST...>::value>
01093 {};
01094
01095 template< typename T >
01096 struct Is_demand
01097 {
01098     static long test(Type_info::Demand const *);
01099     static char test(...);
01100     enum { value = sizeof(test((T*)0)) == sizeof(long) };
01101 };
01102
01103 template< typename T, typename ... >
01104 struct First : T { typedef T type; };
01105 } // Typeid
01106
01112 template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>
01113 struct __Kobject_base : BASES...
01114 {
01115 protected:
01116     typedef Derived Class;
01117     typedef Typeid::Iface<PROTO, Derived> __Iface;
01118     typedef Typeid_xx::Merge_list<
01119         Typeid::Iface_list<__Iface>,
01120         typename BASES::__Iface_list...
01121     > __Iface_list;
01122
01123 static void __check_protocols__() noexcept
01124 {
01125     typedef Typeid_xx::Iface_conflict<__Iface, typename BASES::__Iface_list...> Conflict;
01126     static_assert(!Conflict::value, "ambiguous protocol ID, protocol also used in base class");
01127
01128     typedef Typeid_xx::Conflict<typename BASES::__Iface_list...> Base_conflict;
01129     static_assert(!Base_conflict::value, "ambiguous protocol IDs in base classes");
01130 }
01131
01132 // disambiguate cap()
01133 l4_cap_idx_t cap() const noexcept
01134 { return Typeid_xx::First<BASES...>::type::cap(); }
01135
01136 L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }
01137
01138 L4__GEN_TI_MEMBERS(Kobject_demand<BASES...>)
01139
01140 private:
01141     // This function returns the first base class (used below)
01142     template<typename B1, typename ...> struct Basel { typedef B1 type; };
01143 public:
01144     // Provide non-ambiguous conversion to Kobject
01145     operator Kobject const & () const noexcept
01146     { return *static_cast<typename Basel<BASES...>::type const *>(this); }
01147
01148     // Provide non-ambiguous access of dec_refcnt()
01149     l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
01150     noexcept(noexcept(((typename Basel<BASES...>::type *)0)->dec_refcnt(diff, utcb)))
01151     { return Basel<BASES...>::type::dec_refcnt(diff, utcb); }
01152 };
01153
01154
01156 template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>

```



```

01157 Type_info const *const
01158 __Kobject_base<Derived, PROTO, S_DEMAND, BASES...>::__Kobject_typeid::b[] =
01159 {
01160     (&BASES::__Kobject_typeid::m)...
01161 };
01162
01163
01164 template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>
01165 L4_____GEN_TI(__Kobject_base<Derived, PROTO, S_DEMAND, BASES...>);
01166
01167
01168 // Test if the there is a Demand argument to Kobject_x
01169 template< typename Derived, long PROTO, bool HAS_DEMAND, typename DEMAND, typename ...ARGS >
01170 struct __Kobject_x_proto;
01171
01172 // YES: pass it to __Kobject_base
01173 template< typename Derived, long PROTO, typename DEMAND, typename ...BASES>
01174 struct __Kobject_x_proto<Derived, PROTO, true, DEMAND, BASES...> :
01175     __Kobject_base<Derived, PROTO, DEMAND, BASES...> {};
01176
01177 // NO: pass it empty Type_info::Demand_t
01178 template< typename Derived, long PROTO, typename B1, typename ...BASES>
01179 struct __Kobject_x_proto<Derived, PROTO, false, B1, BASES...> :
01180     __Kobject_base<Derived, PROTO, Type_info::Demand_t<>, B1, BASES...> {};
01181
01182
01183 template< long P = PROTO_EMPTY >
01184 struct Proto_t {};
01185
01186
01187 template< typename Derived, typename ...ARGS >
01188 struct Kobject_x;
01189
01190
01191 template< typename Derived, typename A, typename ...ARGS >
01192 struct Kobject_x<Derived, A, ARGS...> :
01193     __Kobject_x_proto<Derived, PROTO_ANY, Typeid_xx::Is_demand<A>::value, A, ARGS...>
01194 {};
01195
01196
01197 template< typename Derived, long PROTO, typename A, typename ...ARGS >
01198 struct Kobject_x<Derived, Proto_t<PROTO>, A, ARGS...> :
01199     __Kobject_x_proto<Derived, PROTO, Typeid_xx::Is_demand<A>::value, A, ARGS...>
01200 {};
01201
01202 }
01203 #endif
01204
01205 #undef L4_____GEN_TI
01206 #undef L4_____GEN_TI_MEMBERS
01207

```

16.408 __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 L4_INLINE void *l4_vcpu_e_ptr(void const *vcpu, unsigned id) L4_NOTHROW
00031 { return (void *)((l4_addr_t)vcpu + 0x400 + (id & 0xfff)); }
00032
00033 enum L4_vcpu_e_consts
00034 {

```

```

00035     L4_VCPU_E_NUM_LR = 4,
00036 };
00037
00038 L4_INLINE l4_arm_vcpu_e_info_t const *
00039 l4_vcpu_e_info(void const *vcpu) L4_NOTHROW;
00040
00041 L4_INLINE l4_arm_vcpu_e_info_t const *
00042 l4_vcpu_e_info(void const *vcpu) L4_NOTHROW
00043 {
00044     return (l4_arm_vcpu_e_info_t const *)((l4_addr_t)vcpu + 0x200);
00045 }
00046
00047 L4_INLINE l4_umword_t *
00048 l4_vcpu_e_info_user(void *vcpu) L4_NOTHROW;
00049
00050 L4_INLINE l4_umword_t *
00051 l4_vcpu_e_info_user(void *vcpu) L4_NOTHROW
00052 {
00053     return ((l4_arm_vcpu_e_info_t *)((l4_addr_t)vcpu + 0x200))->user;
00054 }
00055
00056
00064 L4_INLINE l4_uint32_t
00065 l4_vcpu_e_read_32(void const *vcpu, unsigned id) L4_NOTHROW;
00066
00067 L4_INLINE l4_uint32_t
00068 l4_vcpu_e_read_32(void const *vcpu, unsigned id) L4_NOTHROW
00069 { return *(l4_uint32_t const *)l4_vcpu_e_ptr(vcpu, id); }
00070
00078 L4_INLINE void
00079 l4_vcpu_e_write_32(void *vcpu, unsigned id, l4_uint32_t val) L4_NOTHROW;
00080
00081 L4_INLINE void
00082 l4_vcpu_e_write_32(void *vcpu, unsigned id, l4_uint32_t val) L4_NOTHROW
00083 { *((l4_uint32_t *)l4_vcpu_e_ptr(vcpu, id)) = val; }
00084
00092 L4_INLINE l4_uint64_t
00093 l4_vcpu_e_read_64(void const *vcpu, unsigned id) L4_NOTHROW;
00094
00095 L4_INLINE l4_uint64_t
00096 l4_vcpu_e_read_64(void const *vcpu, unsigned id) L4_NOTHROW
00097 { return *(l4_uint64_t const *)l4_vcpu_e_ptr(vcpu, id); }
00098
00106 L4_INLINE void
00107 l4_vcpu_e_write_64(void *vcpu, unsigned id, l4_uint64_t val) L4_NOTHROW;
00108
00109 L4_INLINE void
00110 l4_vcpu_e_write_64(void *vcpu, unsigned id, l4_uint64_t val) L4_NOTHROW
00111 { *((l4_uint64_t *)l4_vcpu_e_ptr(vcpu, id)) = val; }
00112
00120 L4_INLINE l4_umword_t
00121 l4_vcpu_e_read(void const *vcpu, unsigned id) L4_NOTHROW;
00122
00123 L4_INLINE l4_umword_t
00124 l4_vcpu_e_read(void const *vcpu, unsigned id) L4_NOTHROW
00125 { return *(l4_umword_t const *)l4_vcpu_e_ptr(vcpu, id); }
00126
00134 L4_INLINE void
00135 l4_vcpu_e_write(void *vcpu, unsigned id, l4_umword_t val) L4_NOTHROW;
00136
00137 L4_INLINE void
00138 l4_vcpu_e_write(void *vcpu, unsigned id, l4_umword_t val) L4_NOTHROW
00139 { *((l4_umword_t *)l4_vcpu_e_ptr(vcpu, id)) = val; }

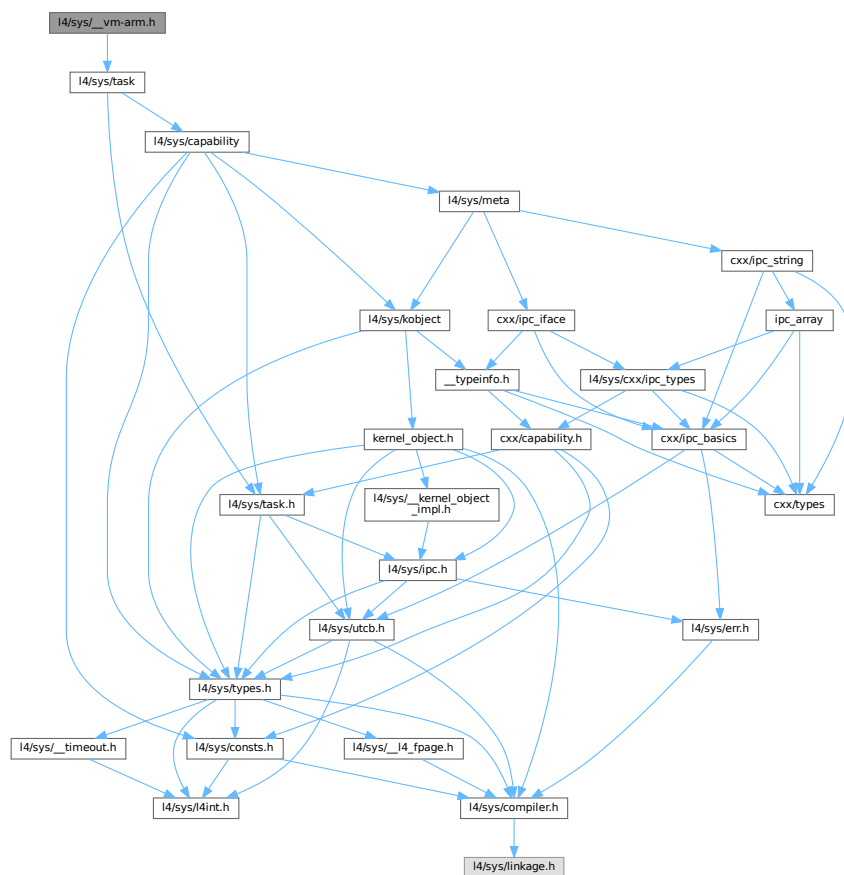
```

16.409 l4/sys/__vm-arm.h File Reference

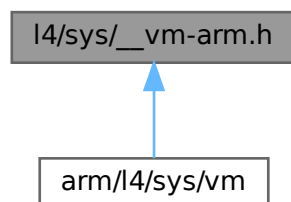
Virtualization interface.

```
#include <l4/sys/task>
```

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.

16.409.1 Detailed Description

Virtualization interface.

Definition in file [__vm-arm.h](#).

16.410 __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 }
```

16.411 __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
00165 typedef struct l4_vm_svm_vmc_b_t
00166 {
00167     l4_vm_svm_vmc_b_control_area_t    control_area;
00168     l4_vm_svm_vmc_b_state_save_area_t state_save_area;
00169 } l4_vm_svm_vmc_b_t;

```

16.412 __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_NUM_CAPS_REGS
00055 };
00056
00057
00062 enum L4_vm_vmx_dfl1_regs
00063 {
00064     L4_VM_VMX_PINBASED_CTL5_DFL1_REG = 0x1,
00065     L4_VM_VMX_PROCBASED_CTL5_DFL1_REG = 0x2,
00066     L4_VM_VMX_EXIT_CTL5_DFL1_REG = 0x3,
00067     L4_VM_VMX_ENTRY_CTL5_DFL1_REG = 0x4,
00068     L4_VM_VMX_NUM_DFL1_REGS
00069 };
00070
00079 L4_INLINE
00080 l4_uint64_t
00081 l4_vm_vmx_get_caps(void const *vcpu_state, unsigned cap_msr) L4_NOTHROW;
00082
00091 L4_INLINE
00092 l4_uint32_t
00093 l4_vm_vmx_get_caps_default1(void const *vcpu_state, unsigned cap_msr) L4_NOTHROW;
00094
00095
00105 enum
00106 {
00114     L4_VM_VMX_VMCS_CR2 = 0x683e,
00116     L4_VM_VMX_VMCS_XCR0 = 0x2840,
00118     L4_VM_VMX_VMCS_MSR_SYSCALL_MASK = 0x2842,
00120     L4_VM_VMX_VMCS_MSR_LSTAR = 0x2844,
00122     L4_VM_VMX_VMCS_MSR_CSTAR = 0x2846,
00124     L4_VM_VMX_VMCS_MSR_TSC_AUX = 0x2848,
00126     L4_VM_VMX_VMCS_MSR_STAR = 0x284a,
00128     L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE = 0x284c,
00129 };
00130
00138 L4_INLINE
00139 unsigned
00140 l4_vm_vmx_field_len(unsigned field) L4_NOTHROW;
00141
00149 L4_INLINE
00150 unsigned
00151 l4_vm_vmx_field_order(unsigned field) L4_NOTHROW;
00152
00163 L4_INLINE
00164 void *
00165 l4_vm_vmx_field_ptr(void *vmcs, unsigned field) L4_NOTHROW;
00166
00176 L4_INLINE
00177 void
00178 l4_vm_vmx_clear(void *vmcs, void *user_vmcs) L4_NOTHROW;
00179
00189 L4_INLINE
00190 void
00191 l4_vm_vmx_ptr_load(void *vmcs, void *user_vmcs) L4_NOTHROW;
00192
00193
00208 L4_INLINE
00209 l4_uint32_t
00210 l4_vm_vmx_get_cr2_index(void const *vmcs) L4_NOTHROW;
00211
00221 L4_INLINE
00222 l4_umword_t
00223 l4_vm_vmx_read_nat(void *vmcs, unsigned field) L4_NOTHROW;
00224
00234 L4_INLINE
00235 l4_uint16_t
00236 l4_vm_vmx_read_16(void *vmcs, unsigned field) L4_NOTHROW;
00237
00247 L4_INLINE
00248 l4_uint32_t
00249 l4_vm_vmx_read_32(void *vmcs, unsigned field) L4_NOTHROW;
00250
00260 L4_INLINE
00261 l4_uint64_t
00262 l4_vm_vmx_read_64(void *vmcs, unsigned field) L4_NOTHROW;
00263
00273 L4_INLINE
00274 l4_uint64_t
00275 l4_vm_vmx_read(void *vmcs, unsigned field) L4_NOTHROW;
00276
00285 L4_INLINE
00286 void
00287 l4_vm_vmx_write_nat(void *vmcs, unsigned field, l4_umword_t val) L4_NOTHROW;
00288
00297 L4_INLINE
00298 void
00299 l4_vm_vmx_write_16(void *vmcs, unsigned field, l4_uint16_t val) L4_NOTHROW;
00300
```

```

00309 L4_INLINE
00310 void
00311 l4_vm_vmx_write_32(void *vmcs, unsigned field, l4_uint32_t val) L4_NOTHROW;
00312
00321 L4_INLINE
00322 void
00323 l4_vm_vmx_write_64(void *vmcs, unsigned field, l4_uint64_t val) L4_NOTHROW;
00324
00333 L4_INLINE
00334 void
00335 l4_vm_vmx_write(void *vmcs, unsigned field, l4_uint64_t val) L4_NOTHROW;
00336
00337
00338 /* Implementations */
00339
00340 L4_INLINE
00341 unsigned
00342 l4_vm_vmx_field_order(unsigned field) L4_NOTHROW
00343 {
00344     switch (field >> 13)
00345     {
00346         case 0: return 1;
00347         case 1: return 3;
00348         case 2: return 2;
00349         case 3: if (sizeof(l4_umword_t) == 8) return 3; else return 2;
00350         default: return 0;
00351     }
00352 }
00353
00354
00355 L4_INLINE
00356 unsigned
00357 l4_vm_vmx_field_len(unsigned field) L4_NOTHROW
00358 {
00359     return 1 << l4_vm_vmx_field_order(field);
00360 }
00361
00362
00363 /* Internal VCPU state layout:
00364 *
00365 * VCPU State:
00366 * 0 - xxx: normal IA32 VCPU state (l4_vcpu_state_t)
00367 * 200h: VMX capabilities (see l4_vm_vmx_get_caps)
00368 * 400h: Fiasco.OC VMCS
00369 *
00370 * Fiasco.OC VMCS:
00371 * 0h - 7h: Reserved
00372 * 8h - Fh: ignored by kernel, stores current VMCS for l4_vm_vmx_clear...
00373 * 10h - 13h: L4_VM_VMX_VMCS_CR2 value used by the kernel
00374 * 14h - 1Fh: Reserved
00375 * 20h - 3Fh: VMCS field offset table
00376 * 40h - BFFh: Data (VMCS field data)
00377 *
00378 * VMCS field offset table:
00379 * 0h - 2h: 3 offsets for 16bit fields:
00380 *         0: Control fields, 1: read-only fields, 2: guest state
00381 *         all offsets in 64byte granules relative to the start of the VMCS
00382 *         3h: Reserved
00383 * 4h - 7h: Index shift values for 16bit, 64bit, 32bit, and natural width fields
00384 * 8h - Ah: 3 offsets for 64bit fields
00385 * Bh - Fh: Reserved
00386 * 10h - 12h: 3 offsets for 32bit fields
00387 * 13h - 17h: Reserved
00388 * 18h - 1Ah: 3 offsets for natural width fields
00389 * 1Bh: Reserved
00390 * 1Ch: Offset of first VMCS field
00391 * 1Dh: Full size of VMCS fields
00392 * 1Eh - 1Fh: Reserved
00393 *
00394 */
00395
00396 L4_INLINE
00397 unsigned
00398 l4_vm_vmx_field_offset(void const *vmcs, unsigned field) L4_NOTHROW
00399 {
00400     // the offset table is at 0x20 offset
00401     enum { Si = 4 };
00402     l4_uint8_t const *offsets = (l4_uint8_t const *)vmcs;
00403     offsets += 0x20;
00404     return (unsigned)offsets[field >> 10] * 64 + ((field & 0x3ff) << offsets[Si + (field >> 13)]);
00405 }
00406
00407 L4_INLINE
00408 void *
00409 l4_vm_vmx_field_ptr(void *vmcs, unsigned field) L4_NOTHROW
00410 {
00411     return (void *)((char *)vmcs + l4_vm_vmx_field_offset(vmcs, field));

```



```

00412 }
00413
00418 L4_INLINE
00419 void
00420 l4_vm_vmx_copy_state(void const *vmcs, void *_dst, void const *_src) L4_NOTHROW
00421 {
00422     l4_uint8_t const *offsets = (l4_uint8_t const *)vmcs + 0x20;
00423
00424     unsigned offs = offsets[28] * 64;
00425     unsigned size = offsets[29] * 64;
00426     char *const dst = (char*)_dst + offs;
00427     char const *const src = (char const *)_src + offs;
00428     __builtin_memcpy(dst, src, size);
00429 }
00430
00431 L4_INLINE
00432 void
00433 l4_vm_vmx_clear(void *vmcs, void *user_vmcs) L4_NOTHROW
00434 {
00435     void **current_vmcs = (void **)((char *)vmcs + 8);
00436     if (*current_vmcs != user_vmcs)
00437         return;
00438
00439     l4_vm_vmx_copy_state(vmcs, user_vmcs, vmcs);
00440     *current_vmcs = 0;
00441 }
00442
00443 L4_INLINE
00444 void
00445 l4_vm_vmx_ptr_load(void *vmcs, void *user_vmcs) L4_NOTHROW
00446 {
00447     void **current_vmcs = (void **)((char *)vmcs + 8);
00448     if (*current_vmcs == user_vmcs)
00449         return;
00450
00451     if (*current_vmcs && *current_vmcs != user_vmcs)
00452         l4_vm_vmx_clear(vmcs, *current_vmcs);
00453
00454     *current_vmcs = user_vmcs;
00455     l4_vm_vmx_copy_state(vmcs, vmcs, user_vmcs);
00456 }
00457
00458
00459 L4_INLINE
00460 l4_umword_t
00461 l4_vm_vmx_read_nat(void *vmcs, unsigned field) L4_NOTHROW
00462 { return *(l4_umword_t*)(l4_vm_vmx_field_ptr(vmcs, field)); }
00463
00464 L4_INLINE
00465 l4_uint16_t
00466 l4_vm_vmx_read_16(void *vmcs, unsigned field) L4_NOTHROW
00467 { return *(l4_uint16_t*)(l4_vm_vmx_field_ptr(vmcs, field)); }
00468
00469 L4_INLINE
00470 l4_uint32_t
00471 l4_vm_vmx_read_32(void *vmcs, unsigned field) L4_NOTHROW
00472 { return *(l4_uint32_t*)(l4_vm_vmx_field_ptr(vmcs, field)); }
00473
00474 L4_INLINE
00475 l4_uint64_t
00476 l4_vm_vmx_read_64(void *vmcs, unsigned field) L4_NOTHROW
00477 { return *(l4_uint64_t*)(l4_vm_vmx_field_ptr(vmcs, field)); }
00478
00479 L4_INLINE
00480 l4_uint64_t
00481 l4_vm_vmx_read(void *vmcs, unsigned field) L4_NOTHROW
00482 {
00483     switch(field >> 13)
00484     {
00485         case 0: return l4_vm_vmx_read_16(vmcs, field);
00486         case 1: return l4_vm_vmx_read_64(vmcs, field);
00487         case 2: return l4_vm_vmx_read_32(vmcs, field);
00488         case 3: return l4_vm_vmx_read_nat(vmcs, field);
00489     }
00490     __builtin_trap();
00491 }
00492
00493 L4_INLINE
00494 void
00495 l4_vm_vmx_write_nat(void *vmcs, unsigned field, l4_umword_t val) L4_NOTHROW
00496 { *(l4_umword_t*)(l4_vm_vmx_field_ptr(vmcs, field)) = val; }
00497
00498 L4_INLINE
00499 void
00500 l4_vm_vmx_write_16(void *vmcs, unsigned field, l4_uint16_t val) L4_NOTHROW
00501 { *(l4_uint16_t*)(l4_vm_vmx_field_ptr(vmcs, field)) = val; }
00502

```

```

00503 L4_INLINE
00504 void
00505 l4_vm_vmx_write_32(void *vmcs, unsigned field, l4_uint32_t val) L4_NOTHROW
00506 { *(l4_uint32_t*)(l4_vm_vmx_field_ptr(vmcs, field)) = val; }
00507
00508 L4_INLINE
00509 void
00510 l4_vm_vmx_write_64(void *vmcs, unsigned field, l4_uint64_t val) L4_NOTHROW
00511 { *(l4_uint64_t*)(l4_vm_vmx_field_ptr(vmcs, field)) = val; }
00512
00513
00514 L4_INLINE
00515 void
00516 l4_vm_vmx_write(void *vmcs, unsigned field, l4_uint64_t val) L4_NOTHROW
00517 {
00518     switch(field >> 13)
00519     {
00520         case 0: l4_vm_vmx_write_16(vmcs, field, val); break;
00521         case 1: l4_vm_vmx_write_64(vmcs, field, val); break;
00522         case 2: l4_vm_vmx_write_32(vmcs, field, val); break;
00523         case 3: l4_vm_vmx_write_nat(vmcs, field, val); break;
00524     }
00525 }
00526
00527 L4_INLINE
00528 l4_uint64_t
00529 l4_vm_vmx_get_caps(void const *vcpu_state, unsigned cap_msr) L4_NOTHROW
00530 {
00531     l4_uint64_t const *caps = (l4_uint64_t const *)((char const *) (vcpu_state) +
00532     L4_VCPU_OFFSET_EXT_INFOS);
00533     return caps[cap_msr & 0xf];
00534 }
00535
00536 L4_INLINE
00537 l4_uint32_t
00538 l4_vm_vmx_get_caps_default1(void const *vcpu_state, unsigned cap_msr) L4_NOTHROW
00539 {
00540     l4_uint32_t const *caps = (l4_uint32_t const *)((char const *) (vcpu_state) +
00541     L4_VCPU_OFFSET_EXT_INFOS);
00542     return caps[L4_VM_VMX_NUM_CAPS_REGS * 2 + ((cap_msr & 0xf) - L4_VM_VMX_PINBASED_CTL5_DFL1_REG)];
00543 }
00544
00545 L4_INLINE
00546 l4_uint32_t
00547 l4_vm_vmx_get_cr2_index(void const *vmcs) L4_NOTHROW
00548 {
00549     l4_uint32_t const *infos = (l4_uint32_t const *)vmcs;
00550     return infos[3];
00551 }

```

16.413 l4/sys/arm_smccc File Reference

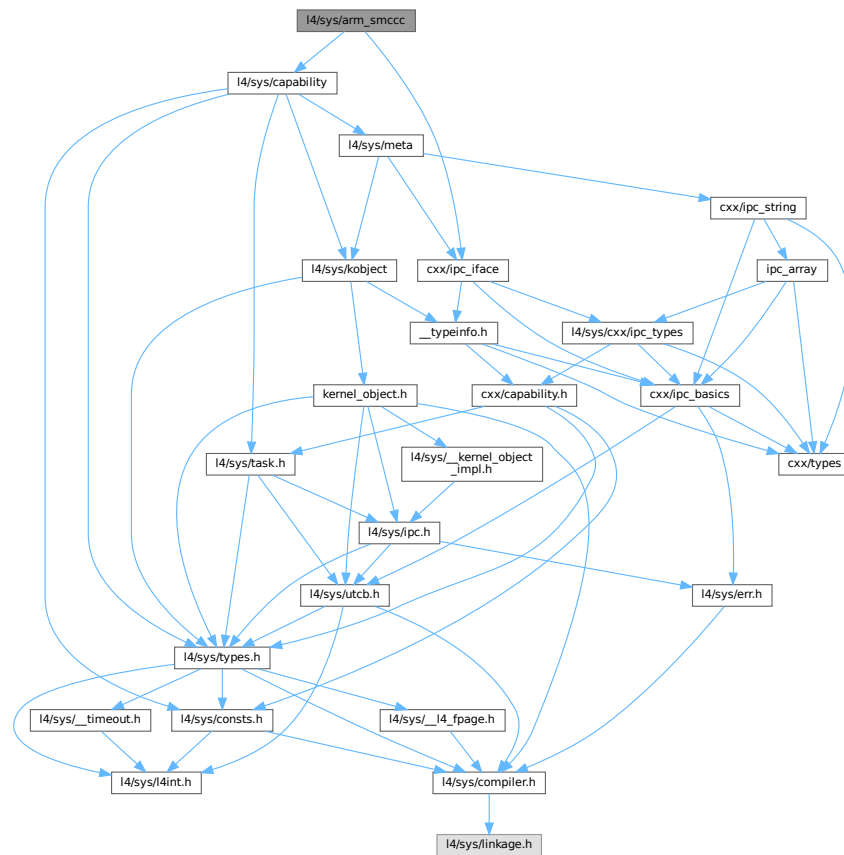
ARM secure monitor call functions.

```

#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.

16.413.1 Detailed Description

ARM secure monitor call functions.

Definition in file [arm_smccc](#).

16.414 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 }

```

16.415 l4/sys/arm_smccc.h File Reference

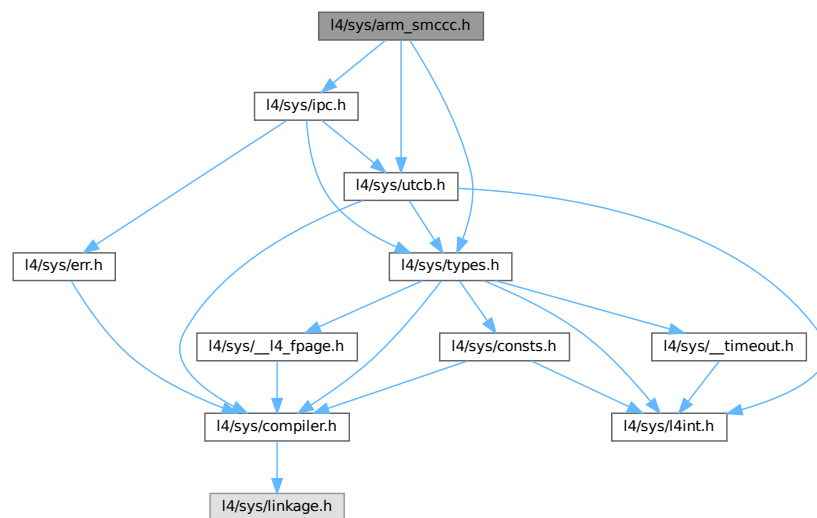
ARM secure monitor call functions.

```

#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.

16.415.1 Detailed Description

ARM secure monitor call functions.

Definition in file [arm_smccc.h](#).

16.415.2 Function Documentation

16.415.2.1 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"> • 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. • Bit 30 defines the calling convention: • Bit 30 == 1: 64-bit calling convention. • Bit 30 == 0: 32-bit calling convention. • Bits 24..29 determine the service call ID. Only service IDs $\geq 0x30000000$ (<i>Trusted Application Calls</i> and <i>Trusted OS Calls</i>) are allowed.
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.

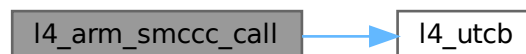
Return values

-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:



16.416 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      l4_msgtag_t ret;
00063      l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00064      v->mr[0] = func;
00065      v->mr[1] = in0;
00066      v->mr[2] = in1;
00067      v->mr[3] = in2;
00068      v->mr[4] = in3;
00069      v->mr[5] = in4;
00070      v->mr[6] = in5;
00071      v->mr[7] = client_id;
00072
00073      ret = l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_SMCCC, 8, 0, 0),
00074                      L4_IPC_NEVER);
00075
00076      if (l4_error(ret) >= 0)
00077      {
00078          *out0 = v->mr[0];
00079          *out1 = v->mr[1];
00080          *out2 = v->mr[2];
00081          *out3 = v->mr[3];
00082      }
00083
00084      return ret;
00085  }
00086  }

```

16.417 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.

16.417.1 Detailed Description

Cache functions.

Definition in file [cache.h](#).

16.418 cache.h

[Go to the documentation of this file.](#)

```

00001
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 #ifndef __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__
00020 #define __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__
00021
00022 #include_next <l4/sys/cache.h>
00023
00024 L4_INLINE int
00025 l4_cache_clean_data(unsigned long start,
00026                    unsigned long end) L4_NOTHROW
00027 {
00028     (void)start; (void)end;
00029     return 0;
00030 }
00031
00032 L4_INLINE int
00033 l4_cache_flush_data(unsigned long start,
00034                    unsigned long end) L4_NOTHROW
00035 {
00036     (void)start; (void)end;
00037     return 0;
00038 }
00039
00040 L4_INLINE int
00041 l4_cache_inv_data(unsigned long start,
00042                  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__ */

```

16.419 arm/l4/sys/cache.h File Reference

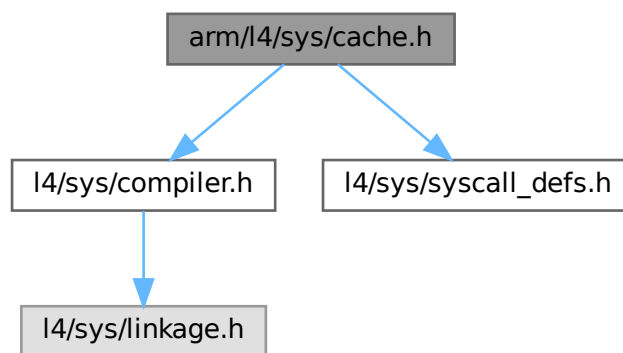
Cache functions.

```

#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.

16.419.1 Detailed Description

Cache functions.

Date

2007-11

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de

Definition in file [cache.h](#).

16.420 cache.h

[Go to the documentation of this file.](#)

```

00001
00009 /*
00010  * (c) 2007-2009 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 #ifndef __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__
00027 #define __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__
00028
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/syscall_defs.h>
00031
00032 #include_next <l4/sys/cache.h>
00033
00037 L4_INLINE void
00038 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    lr, pc                \n\t"
00054   "mvn    pc, %[sc]             \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", "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__ */

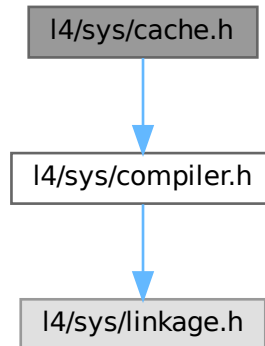
```

16.421 l4/sys/cache.h File Reference

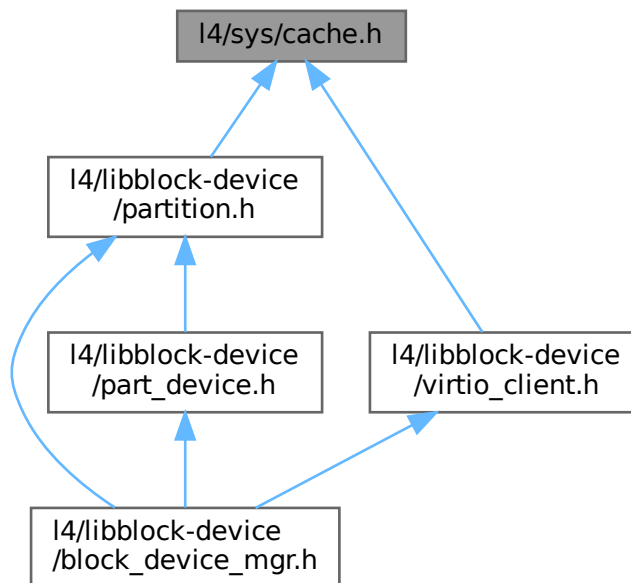
Cache-consistency functions.

```
#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.

16.421.1 Detailed Description

Cache-consistency functions.

Date

2007-11

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de

Definition in file [cache.h](#).

16.422 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
00142 L4_INLINE int
00143 l4_cache_dma_coherent_full(void) L4_NOTHROW;
00144
00145 EXTERN_C_END
00146
00147 #endif /* ! __L4SYS__INCLUDE__CACHE_H__ */

```

16.423 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.

16.423.1 Detailed Description

Cache functions.

Definition in file [cache.h](#).

16.424 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__ */

```

16.425 l4/sys/capability File Reference

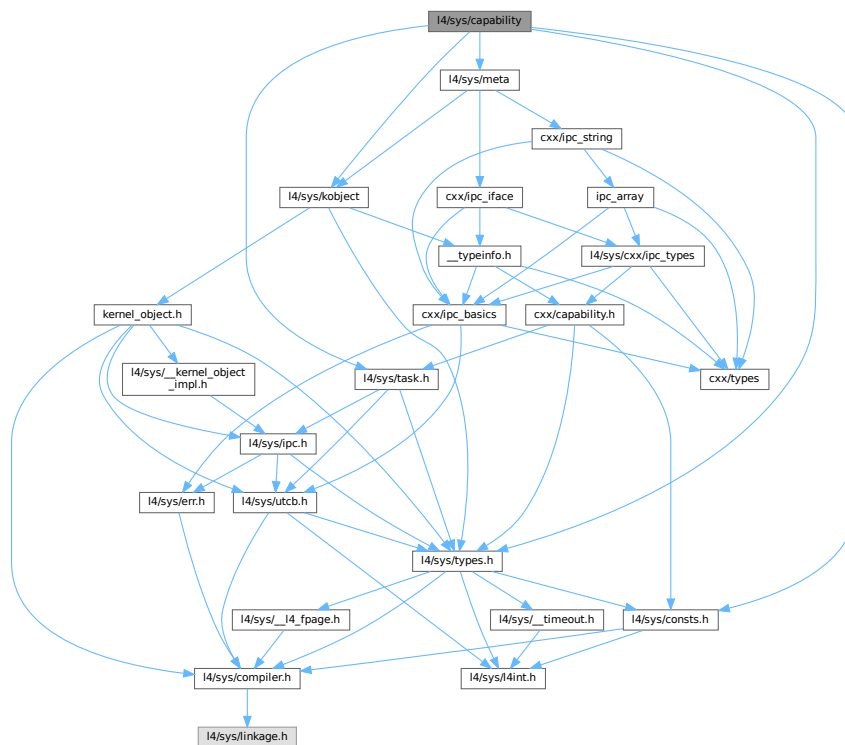
[L4::Cap](#) related definitions.

```

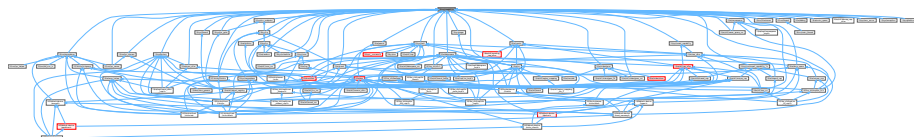
#include <l4/sys/consts.h>
#include <l4/sys/types.h>
#include <l4/sys/kobject>
#include <l4/sys/task.h>
#include <l4/sys/meta>

```

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.

16.425.1 Detailed Description

[L4::Cap](#) related definitions.

Author

Alexander Warg alexander.warg@os.inf.tu-dresden.de

Definition in file [capability](#).

16.425.2 Macro Definition Documentation

16.425.2.1 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 61 of file [capability](#).

16.426 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 Kobject;
00044
00060  #if __cplusplus >= 201103L
00061  #   define L4_DISABLE_COPY(_class) \
00062      public: \
00063          _class(_class const &) = delete; \
00064          _class operator = (_class const &) = delete; \
00065      private:
00066  #else
00067  #   define L4_DISABLE_COPY(_class) \
00068      private: \
00069          _class(_class const &); \
00070          _class operator = (_class const &);
00071  #endif
00072
00073
00074  #define L4_KOBJECT_DISABLE_COPY(_class) \
00075      protected: \
00076          _class(); \
00077          L4_DISABLE_COPY(_class)
00078
00079
00080  #define L4_KOBJECT(_class) L4_KOBJECT_DISABLE_COPY(_class)
00081
00082  inline l4_msgtag_t
00083  Cap_base::validate(Cap<Task> task, l4_utcb_t *u) const noexcept
00084  {
00085      return is_valid() ? l4_task_cap_valid_u(task.cap(), _c, u)
00086          : l4_msgtag(0, 0, 0, 0);
00087  }
00088
00089  inline l4_msgtag_t
00090  Cap_base::validate(l4_utcb_t *u) const noexcept
00091  {
00092      return is_valid() ? l4_task_cap_valid_u(L4_BASE_TASK_CAP, _c, u)
00093          : l4_msgtag(0, 0, 0, 0);
00094  }
00095
00096  }; // namespace L4
00097
00098  #include <l4/sys/meta>
00099
00100  namespace L4 {
00101
00123  template< typename T, typename F >
00124  inline
00125  Cap<T> cap_dynamic_cast(Cap<F> const &c) noexcept
00126  {
00127      if (!c.is_valid())
00128          return Cap<T>::Invalid;
00129
00130      Cap<Meta> mc = cap_reinterpret_cast<Meta>(c);
00131      Type_info const *m = kobject_typeid<T>();
00132      if (m->proto() && l4_error(mc->supports(m->proto())) > 0)
00133          return Cap<T>(c.cap());
00134
00135      // FIXME: use generic checker
00136      #if 0
00137      if (l4_error(mc->supports(T::kobject_proto())) > 0)
00138          return Cap<T>(c.cap());
00139      #endif
00140
00141      return Cap<T>::Invalid;
00142  }
00143
00144  }

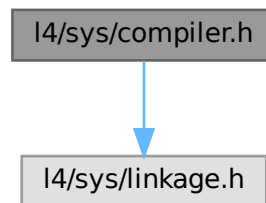
```

16.427 l4/sys/compiler.h File Reference

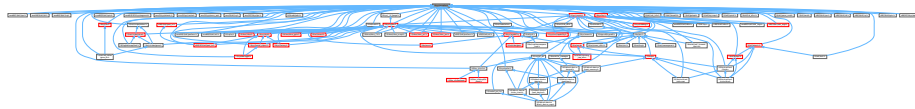
L4 compiler related defines.

```
#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_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.

16.427.1 Detailed Description

[L4](#) compiler related defines.

Definition in file [compiler.h](#).

16.428 compiler.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  * Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00006  * Jork Löser <jork@os.inf.tu-dresden.de>,
00007  * Ronald Aigner <ra3@os.inf.tu-dresden.de>
00008  *
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
```

```

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  #       ifdef __GNUC_STDC_INLINE__
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  #   ifndef __BEGIN_DECLS
00180  #       define __BEGIN_DECLS
00181  #   endif
00182  #   ifndef __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  #define L4_NORETURN __attribute__((noreturn))
00209
00210  #define L4_PURE __attribute__((pure))
00211
00216  #define L4_NOINSTRUMENT __attribute__((no_instrument_function))
00217  #ifndef L4_HIDDEN
00218  #   define L4_HIDDEN __attribute__((visibility("hidden")))
00219  #endif
00220  #if !defined L4_EXPORT || defined DOXYGEN
00221  #   define L4_EXPORT __attribute__((visibility("default")))
00222  #endif
00223  #ifndef L4_EXPORT_TYPE
00224  #   ifdef __cplusplus
00225  #       define L4_EXPORT_TYPE __attribute__((visibility("default")))
00226  #   else
00227  #       define L4_EXPORT_TYPE
00228  #   endif
00229  #endif

```

```

00230 #define L4_STRONG_ALIAS(name, aliasname) L4__STRONG_ALIAS(name, aliasname)
00231 #define L4__STRONG_ALIAS(name, aliasname) \
00232     extern __typeof (name) aliasname __attribute__ ((alias (#name)));
00233
00241 #if defined(ARCH_x86) || defined(ARCH_amd64) || \
00242     defined(ARCH_arm) || defined(ARCH_arm64) || \
00243     defined(ARCH_mips) || defined(ARCH_ppc32) || defined(ARCH_sparc)
00244 #define L4_STACK_ALIGN __BIGGEST_ALIGNMENT__
00245 #else
00246 #error Define L4_STACK_ALIGN for this target!
00247 #endif
00248
00266 #if defined(ARCH_x86) || defined(ARCH_amd64)
00267 L4_INLINE unsigned long l4_align_stack_for_direct_fncall(unsigned long stack)
00268 {
00269     if ((stack & (L4_STACK_ALIGN - 1)) == (L4_STACK_ALIGN - sizeof(unsigned long)))
00270         return stack;
00271     return (stack & ~(L4_STACK_ALIGN)) - sizeof(unsigned long);
00272 }
00273 #else
00274 L4_INLINE unsigned long l4_align_stack_for_direct_fncall(unsigned long stack)
00275 {
00276     return stack & ~(L4_STACK_ALIGN);
00277 }
00278 #endif
00279
00280 #endif /* !__ASSEMBLY__ */
00281
00282 #include <l4/sys/linkage.h>
00283
00284 #define L4_LIKELY(x) __builtin_expect((x),1)
00285 #define L4_UNLIKELY(x) __builtin_expect((x),0)
00286
00287 /* Make sure that the function is not removed by optimization. Without the
00288  * "used" attribute, unreferenced static functions are removed. */
00289 #define L4_STICKY(x) __attribute__((used)) x
00290 #define L4_DEPRECATED(s) __attribute__((deprecated(s)))
00291
00292 #ifndef static_assert
00293 #if !defined(__cplusplus)
00294 #define static_assert(x, y) _Static_assert(x, y)
00295 #elif __cplusplus < 201103L
00296 #define static_assert(x, y) \
00297     extern int l4_static_assert[!(x)] __attribute__((unused))
00298 #endif
00299 #endif
00300
00301 #define L4_stringify_helper(x) #x
00302 #define L4_stringify(x) L4_stringify_helper(x)
00303
00304 #ifndef __ASSEMBLER__
00308 L4_INLINE void l4_barrier(void);
00309
00313 L4_INLINE void l4_mb(void);
00314
00318 L4_INLINE void l4_wmb(void);
00319
00323 L4_INLINE L4_NORETURN void l4_infinite_loop(void);
00324
00325
00326 /* Implementations */
00327 L4_INLINE void l4_barrier(void)
00328 {
00329     __asm__ __volatile__ ("": : : "memory");
00330 }
00331
00332 L4_INLINE void l4_mb(void)
00333 {
00334     __asm__ __volatile__ ("": : : "memory");
00335 }
00336
00337 L4_INLINE void l4_wmb(void)
00338 {
00339     __asm__ __volatile__ ("": : : "memory");
00340 }
00341
00342 L4_INLINE L4_NORETURN void l4_infinite_loop(void)
00343 {
00344     while (1)
00345         l4_barrier();
00346 }
00347 #endif
00348
00351 #endif /* !__L4_COMPILER_H__ */

```

16.429 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.

16.429.1 Detailed Description

Common [L4](#) constants, amd64 version.

Definition in file [consts.h](#).

16.430 consts.h

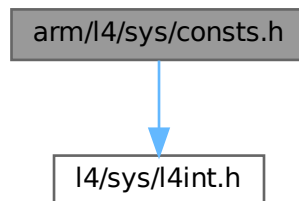
[Go to the documentation of this file.](#)

```
00001 /*****
00002 */
00003 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00005 * 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 #ifndef __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__
00025 #define __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__
00026
00027 #define L4_PAGESHIFT 12
00028
00029 #define L4_SUPERPAGESHIFT 21
00030
00031 #include_next <l4/sys/consts.h>
00032
00033 #endif /* ! __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__ */
```

16.431 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.

16.431.1 Detailed Description

Common L4 constants, arm version.

Definition in file [consts.h](#).

16.432 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 */

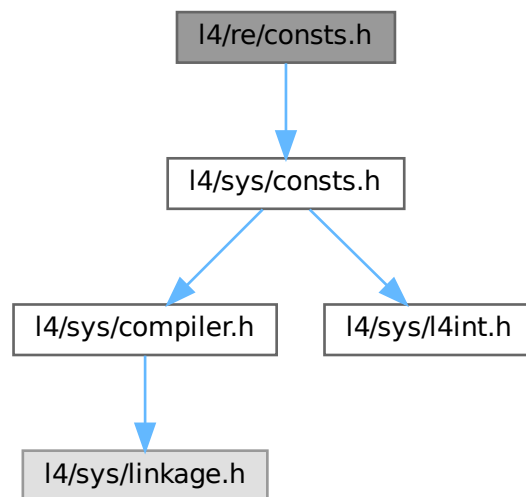
```

16.433 l4/re/consts.h File Reference

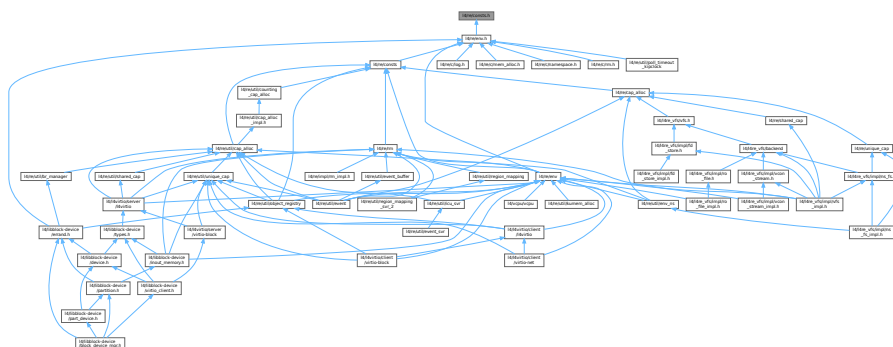
Constants.

```
#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.

16.433.1 Detailed Description

Constants.

Definition in file [consts.h](#).

16.433.2 Enumeration Type Documentation

16.433.2.1 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](#).

16.434 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
```

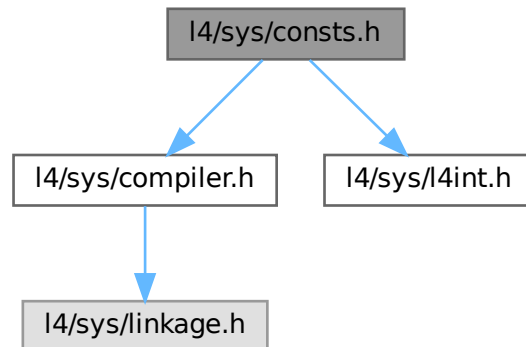
16.435 l4/sys/consts.h File Reference

Common constants.

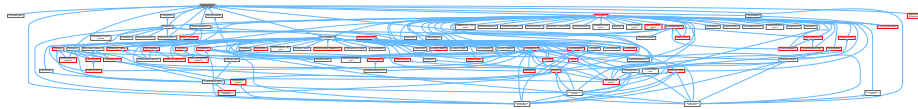
```
#include <l4/sys/compiler.h>
```

```
#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_SUPERPAGE_SIZE**
Size of a large page.
- **#define L4_SUPERPAGEMASK**
Mask for the number of a large page.
- **#define L4_LOG2_SUPERPAGE_SIZE**
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.

16.435.1 Detailed Description

Common constants.

Definition in file [consts.h](#).

16.436 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
00262     L4_RCV_ITEM_SINGLE_CAP = L4_ITEM_MAP | 2,
00263
00264
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)
00506 #else
00507 # define NULL 0
00508 #endif
00509 #endif
00510 #endif
00511 #endif
00512 #endif
00513
00514 #endif /* ! __L4_SYS__INCLUDE__CONSTS_H__ */

```

16.437 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.

16.437.1 Detailed Description

Common [L4](#) constants, x86 version.

Definition in file [consts.h](#).

16.438 consts.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 * Björn Döbel <doebel@os.inf.tu-dresden.de>,
00006 * Lars Reuther <reuther@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 #ifndef __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__
00025 #define __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__
00026
00027 #define L4_PAGESHIFT 12
00028
00029 #define L4_SUPERPAGESHIFT 22
00030
00031 #include_next <l4/sys/consts.h>
00032
00033 #endif /* ! __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__ */

```

16.439 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 private:
00028     struct Invalid_conversion;
00029
00030 public:
00032     enum No_init_type
00033     {
00037         No_init
00038     };
00039
00043     enum Cap_type
00044     {
00045         Invalid = L4_INVALID_CAP
00046     };
00047
00052     l4_cap_idx_t cap() const noexcept { return _c; }
00053
00060     bool is_valid() const noexcept { return !(_c & L4_INVALID_CAP_BIT); }
00061
00062     operator Invalid_conversion * () const noexcept
00063     { return (Invalid_conversion*)(!(_c & L4_INVALID_CAP_BIT)); }
00064
00072     l4_fpage_t fpage(unsigned rights = L4_CAP_FPAGE_RWS) const noexcept
00073     { return l4_obj_fpage(_c, 0, rights); }
00074
00084     l4_umword_t snd_base(unsigned grant = L4_MAP_ITEM_MAP,
00085                          l4_cap_idx_t base = L4_INVALID_CAP) const noexcept
00086     {
00087         if (base == L4_INVALID_CAP)
00088             base = _c;
00089         return l4_map_obj_control(base, grant);
00090     }
00091
00092
00096     bool operator == (Cap_base const &o) const noexcept
00097     { return _c == o._c; }
00098
00102     bool operator != (Cap_base const &o) const noexcept
00103     { return _c != o._c; }
00104
00118     inline l4_msgtag_t validate(l4_utcb_t *u = l4_utcb()) const noexcept;
00119
00134     inline l4_msgtag_t validate(Cap<Task> task,
00135                                l4_utcb_t *u = l4_utcb()) const noexcept;
00136
00140     void invalidate() noexcept { _c = L4_INVALID_CAP; }
00141 protected:
00147     explicit Cap_base(l4_cap_idx_t c) noexcept : _c(c) {}
00151     explicit Cap_base(Cap_type cap) noexcept : _c(cap) {}
00152
00158     explicit Cap_base(l4_default_caps_t cap) noexcept : _c(cap) {}
00159
00163     explicit Cap_base() noexcept {}
00164
00174     void move(Cap_base const &src) const
00175     {
00176         if (!is_valid() || !src.is_valid())
00177             return;
00178
00179         l4_task_map(L4_BASE_TASK_CAP, L4_BASE_TASK_CAP, src.fpage(L4_CAP_FPAGE_RWS),
00180                    snd_base(L4_MAP_ITEM_GRANT) | L4_FPAGE_C_OBJ_RIGHTS);
00181     }
00182
00190     void copy(Cap_base const &src) const
00191     {
00192         if (!is_valid() || !src.is_valid())
00193             return;
00194
00195         l4_task_map(L4_BASE_TASK_CAP, L4_BASE_TASK_CAP, src.fpage(L4_CAP_FPAGE_RWS),
00196                    snd_base() | L4_FPAGE_C_OBJ_RIGHTS);

```



```

00197     }
00198
00201     l4_cap_idx_t _c;
00202 };
00203
00204
00220 template< typename T >
00221 class L4_EXPORT Cap : public Cap_base
00222 {
00223 private:
00224     friend class L4::Kobject;
00225
00237     explicit Cap(T const *p) noexcept
00238     : Cap_base(reinterpret_cast<l4_cap_idx_t>(p)) {}
00239
00240 public:
00241
00246     template< typename O >
00247     Cap(Cap<O> const &o) noexcept : Cap_base(o.cap())
00248     { T* __t = ((O*)100); (void)__t; }
00249
00254     Cap(Cap_type cap) noexcept : Cap_base(cap) {}
00255
00260     Cap(l4_default_caps_t cap) noexcept : Cap_base(cap) {}
00261
00266     explicit Cap(l4_cap_idx_t idx = L4_INVALID_CAP) noexcept : Cap_base(idx) {}
00267
00271     explicit Cap(No_init_type) noexcept {}
00272
00279     Cap move(Cap const &src) const
00280     {
00281         Cap_base::move(src);
00282         return *this;
00283     }
00284
00289     Cap copy(Cap const &src) const
00290     {
00291         Cap_base::copy(src);
00292         return *this;
00293     }
00294
00298     T *operator -> () const noexcept { return reinterpret_cast<T*>(_c); }
00299 };
00300
00301
00312 template<
00313 class L4_EXPORT Cap<void> : public Cap_base
00314 {
00315 public:
00316
00317     explicit Cap(void const *p) noexcept
00318     : Cap_base(reinterpret_cast<l4_cap_idx_t>(p)) {}
00319
00323     Cap(Cap_type cap) noexcept : Cap_base(cap) {}
00324
00329     Cap(l4_default_caps_t cap) noexcept : Cap_base(cap) {}
00330
00335     explicit Cap(l4_cap_idx_t idx = L4_INVALID_CAP) noexcept : Cap_base(idx) {}
00336     explicit Cap(No_init_type) noexcept {}
00337
00344     Cap move(Cap const &src) const
00345     {
00346         Cap_base::move(src);
00347         return *this;
00348     }
00349
00354     Cap copy(Cap const &src) const
00355     {
00356         Cap_base::copy(src);
00357         return *this;
00358     }
00359
00360     template< typename T >
00361     Cap(Cap<T> const &o) noexcept : Cap_base(o.cap()) {}
00362 };
00363
00380 template< typename T, typename F >
00381 inline
00382 Cap<T> cap_cast(Cap<F> const &c) noexcept
00383 {
00384     (void)static_cast<T const *>(reinterpret_cast<F const *>(100));
00385     return Cap<T>(c.cap());
00386 }
00387
00388 // gracefully deal with L4::Kobject ambiguity
00389 template< typename T >
00390 inline

```


Namespaces

- namespace [L4Re](#)
[L4Re C++ Interfaces](#).

16.440.1 Detailed Description

Constants.

Definition in file [consts](#).

16.441 consts

[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
00022 #include <l4/sys/capability>
00023 #include <l4/sys/consts.h>
00024 #include <l4/re/env.h>
00025
00026 namespace L4Re {
00027     static L4::Cap<L4::Task>::Cap_type const This_task
00028         = (L4::Cap<L4::Task>::Cap_type) (L4RE_THIS_TASK_CAP);
00029 }
```

16.442 consts

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002
00003 #pragma once
00004
00005 #include <l4/sys/consts.h>
00006
00007 namespace L4 {
00008
00009     template<typename T>
00010     constexpr T trunc_order(T val, unsigned char order)
00011     {
00012         return val & ((~T(0)) << order);
00013     }
00014
00015     template<typename T>
00016     constexpr T round_order(T val, unsigned char order)
00017     {
00018         return (val + (T(1) << order) - T(1)) & ((~T(0)) << order);
00019     }
00020
00021     template<typename T>
00022     constexpr T trunc_page(T val)
00023     {
```

```

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 }

```

16.443 ipc_array

```

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 "types"
00021 #include "ipc_basics"
00022 #include "ipc_types"
00023
00024 namespace L4 { namespace Ipc L4_EXPORT {
00025
00026     typedef unsigned short Array_len_default;
00027
00028     template< typename ELEM_TYPE, typename LEN_TYPE = Array_len_default >
00029     struct Array_ref
00030     {
00031         typedef ELEM_TYPE *ptr_type;
00032         typedef LEN_TYPE len_type;
00033
00034         len_type length;
00035         ptr_type data;
00036         Array_ref() = default;
00037         Array_ref(len_type length, ptr_type data)
00038             : length(length), data(data)
00039         {}
00040
00041         template<typename X> struct Non_const
00042         { typedef Array_ref<X, LEN_TYPE> type; };
00043
00044         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 {
00094     Array() {}
00095     Array(LEN_TYPE length, ELEM_TYPE *data)
00096     : Array_ref<ELEM_TYPE, LEN_TYPE>(length, data)
00097     {}
00098
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];
00144     LEN_TYPE length;
00145
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 {
00196     typedef Array_ref<A, LEN> &arg_type;
00197     typedef Array_ref<typename L4::Types::Remove_const<A>::type, LEN> svr_type;
00200     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             for (LEN i = 0; i < a.length; ++i)
00230                 data[i] = a.data[i];
00231
00232         return offset + a.length * sizeof(A);
00233     }
00234 };
00235 } // namespace Detail
00236
00237 template<typename A, typename LEN>
00238 struct Clnt_val_ops<Array<A, LEN>, Dir_in, Cls_data> :
00239     Detail::Clnt_val_ops_d_in<A, LEN, Array<A, LEN>, false> {};
00240
00241 template<typename A, typename LEN>
00242 struct Clnt_val_ops<Array_ref<A, LEN>, Dir_in, Cls_data> :
00243     Detail::Clnt_val_ops_d_in<A, LEN, Array_ref<A, LEN>, true> {};
00244
00245 template<typename A, typename LEN, typename CLASS>
00246 struct Svr_val_ops< Array_ref<A, LEN>, Dir_in, CLASS >
00247 : Svr_noops< Array_ref<A, LEN> >
00248 {
00249     typedef Array_ref<A, LEN> svr_type;
00250
00251     using Svr_noops<svr_type>::to_svr;
00252     static int to_svr(char *msg, unsigned offset, unsigned limit,
00253                     svr_type &a, Dir_in, Cls_data)
00254     {
00255         offset = align_to<LEN>(offset);
00256         if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00257             return -L4_MSGTOOSHORT;
00258         a.length = *reinterpret_cast<LEN*>(msg + offset);
00259         offset = align_to<A>(offset + sizeof(LEN));
00260         if (L4_UNLIKELY(!check_size<A>(offset, limit, a.length)))
00261             return -L4_MSGTOOSHORT;
00262         a.data = reinterpret_cast<A*>(msg + offset);
00263         return offset + a.length * sizeof(A);
00264     }
00265 };
00266
00267 template<typename A, typename LEN>
00268 struct Svr_xmit< Array<A, LEN> > : Svr_xmit< Array_ref<A, LEN> > {};
00269
00270 template<typename A, typename LEN>
00271 struct Clnt_val_ops<Array<A, LEN>, Dir_out, Cls_data> : Clnt_noops<Array<A, LEN> >
00272 {
00273     typedef Array<A, LEN> type;
00274
00275     using Clnt_noops<type>::from_msg;
00276     static int from_msg(char *msg, unsigned offset, unsigned limit, long,
00277                     type &a, Dir_out, Cls_data)
00278     {
00279         offset = align_to<LEN>(offset);
00280         if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00281             return -L4_MSGTOOSHORT;
00282

```

```

00283     LEN l = *reinterpret_cast<LEN *>(msg + offset);
00284
00285     offset = align_to<A>(offset + sizeof(LEN));
00286     if (L4_UNLIKELY(!check_size<A>(offset, limit, 1)))
00287         return -L4_MSGTOOSHORT;
00288
00289     A *data = reinterpret_cast<A*>(msg + offset);
00290
00291     if (l > a.length)
00292         l = a.length;
00293     else
00294         a.length = l;
00295
00296     for (unsigned i = 0; i < l; ++i)
00297         a.data[i] = data[i];
00298
00299     return offset + l * sizeof(A);
00300 };
00301 };
00302
00303 template<typename A, typename LEN>
00304 struct Clnt_val_ops<Array_ref<A, LEN>, Dir_out, Cls_data> :
00305     Clnt_noops<Array_ref<A, LEN>>
00306 {
00307     typedef Array_ref<A, LEN> type;
00308
00309     using Clnt_noops<type>::from_msg;
00310     static int from_msg(char *msg, unsigned offset, unsigned limit, long,
00311                         type &a, Dir_out, Cls_data)
00312     {
00313         offset = align_to<LEN>(offset);
00314         if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00315             return -L4_MSGTOOSHORT;
00316
00317         LEN l = *reinterpret_cast<LEN *>(msg + offset);
00318
00319         offset = align_to<A>(offset + sizeof(LEN));
00320         if (L4_UNLIKELY(!check_size<A>(offset, limit, 1)))
00321             return -L4_MSGTOOSHORT;
00322
00323         a.data = reinterpret_cast<A*>(msg + offset);
00324         a.length = l;
00325         return offset + l * sizeof(A);
00326     };
00327 };
00328
00329 template<typename A, typename LEN, typename CLASS>
00330 struct Svr_val_ops<Array_ref<A, LEN>, Dir_out, CLASS> :
00331     Svr_noops<Array_ref<typename L4::Types::Remove_const<A>::type, LEN> &>
00332 {
00333     typedef typename L4::Types::Remove_const<A>::type elem_type;
00334     typedef Array_ref<elem_type, LEN> &svr_type;
00335
00336     using Svr_noops<svr_type>::to_svr;
00337     static int to_svr(char *msg, unsigned offset, unsigned limit,
00338                      svr_type a, Dir_out, Cls_data)
00339     {
00340         offset = align_to<LEN>(offset);
00341         if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00342             return -L4_MSGTOO LONG;
00343
00344         offset = align_to<A>(offset + sizeof(LEN));
00345         a.data = reinterpret_cast<elem_type *>(msg + offset);
00346         a.length = (limit - offset) / sizeof(A);
00347         return offset;
00348     }
00349
00350     using Svr_noops<svr_type>::from_svr;
00351     static int from_svr(char *msg, unsigned offset, unsigned limit, long,
00352                        svr_type a, Dir_out, Cls_data)
00353     {
00354         offset = align_to<LEN>(offset);
00355         if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00356             return -L4_MSGTOO LONG;
00357
00358         *reinterpret_cast<LEN *>(msg + offset) = a.length;
00359
00360         offset = align_to<A>(offset + sizeof(LEN));
00361         if (L4_UNLIKELY(!check_size<A>(offset, limit, a.length)))
00362             return -L4_MSGTOO LONG;
00363
00364         return offset + a.length * sizeof(A);
00365     }
00366 };
00367
00368 template<typename A, typename LEN>
00369 struct Svr_xmit<Array<A, LEN> &> : Svr_xmit<Array_ref<A, LEN> &> {};

```

```

00370
00371 // Pointer to array is not implemented.
00372 template<typename A, typename LEN>
00373 struct Is_valid_rpc_type< Array_ref<A, LEN> *> : L4::Types::False {};
00374
00375 // Optional input arrays are not implemented.
00376 template<typename A, typename LEN>
00377 struct Is_valid_rpc_type< Opt<Array_ref<A, LEN> > > : L4::Types::False {};
00378
00379 } // namespace Msg
00380
00381 }}

```

16.444 ipc_basics

```

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 "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 template<typename T>
00037 constexpr unsigned long align_to(unsigned long bytes) noexcept
00038 { return align_to(bytes, __alignof(T)); }
00039
00040 template<typename T>
00041 constexpr bool check_size(unsigned offset, unsigned limit) noexcept
00042 {
00043     return offset + sizeof(T) <= limit;
00044 }
00045
00046 template<typename T, typename CTYPE>
00047 inline bool check_size(unsigned offset, unsigned limit, CTYPE cnt) noexcept
00048 {
00049     if (L4_UNLIKELY(sizeof(CTYPE) <= sizeof(unsigned) &&
00050                     ~0U / sizeof(T) <= static_cast<unsigned>(cnt)))
00051         return false;
00052
00053     if (L4_UNLIKELY(sizeof(CTYPE) > sizeof(unsigned) &&
00054                     static_cast<CTYPE>(~0U / sizeof(T)) <= cnt))
00055         return false;
00056
00057     return sizeof(T) * cnt <= limit - offset;
00058 }
00059
00060 enum
00061 {
00062     Word_bytes = sizeof(l4_umword_t),
00063     Item_words = 2,
00064     Item_bytes = Word_bytes * Item_words,
00065     Mr_words = L4_UTCB_GENERIC_DATA_SIZE,
00066     Mr_bytes = Word_bytes * Mr_words,
00067     Br_bytes = Word_bytes * L4_UTCB_GENERIC_BUFFERS_SIZE,
00068 }
00069
00070 }
00071
00072 }
00073
00074 }

```



```

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
00235 template<typename T> struct Svr_noops
00236 {

```

```

00237     template<typename A, typename B>
00238     static constexpr int from_svr(char *, unsigned offset, unsigned, long, T, A, B) noexcept
00239     { return offset; }
00240
00241     template<typename A, typename B>
00242     static constexpr int to_svr(char *, unsigned offset, unsigned, T, A, B) noexcept
00243     { return offset; }
00244 };
00245
00246 template<typename MTYPE, typename CLASS>
00247 struct Clnt_val_ops<MTYPE, Dir_in, CLASS> : Clnt_noops<MTYPE>
00248 {
00249     using Clnt_noops<MTYPE>::to_msg;
00250     static int to_msg(char *msg, unsigned offset, unsigned limit,
00251                     MTYPE arg, Dir_in, CLASS) noexcept
00252     { return msg_add<MTYPE>(msg, offset, limit, arg); }
00253 };
00254
00255 template<typename MTYPE, typename CLASS>
00256 struct Clnt_val_ops<MTYPE, Dir_out, CLASS> : Clnt_noops<MTYPE>
00257 {
00258     using Clnt_noops<MTYPE>::from_msg;
00259     static int from_msg(char *msg, unsigned offset, unsigned limit, long,
00260                       MTYPE &arg, Dir_out, CLASS) noexcept
00261     { return msg_get<MTYPE>(msg, offset, limit, arg); }
00262 };
00263
00264 template<typename MTYPE, typename DIR, typename CLASS> struct Svr_val_ops;
00265
00266 template<typename MTYPE, typename CLASS>
00267 struct Svr_val_ops<MTYPE, Dir_in, CLASS> : Svr_noops<MTYPE>
00268 {
00269     using Svr_noops<MTYPE>::to_svr;
00270     static int to_svr(char *msg, unsigned offset, unsigned limit,
00271                     MTYPE &arg, Dir_in, CLASS) noexcept
00272     { return msg_get<MTYPE>(msg, offset, limit, arg); }
00273 };
00274
00275 template<typename MTYPE, typename CLASS>
00276 struct Svr_val_ops<MTYPE, Dir_out, CLASS> : Svr_noops<MTYPE>
00277 {
00278     using Svr_noops<MTYPE>::to_svr;
00279     static int to_svr(char *, unsigned offs, unsigned limit,
00280                     MTYPE &, Dir_out, CLASS) noexcept
00281     {
00282         offs = align_to<MTYPE>(offs);
00283         if (L4_UNLIKELY(!check_size<MTYPE>(offs, limit)))
00284             return -L4_MSGTOOLONG;
00285         return offs + sizeof(MTYPE);
00286     }
00287
00288     using Svr_noops<MTYPE>::from_svr;
00289     static int from_svr(char *msg, unsigned offset, unsigned limit, long,
00290                       MTYPE arg, Dir_out, CLASS) noexcept
00291     { return msg_add<MTYPE>(msg, offset, limit, arg); }
00292 };
00293
00294 template<typename T> struct Elem
00295 {
00296     typedef T arg_type;
00297     typedef T svr_type;
00298     typedef T svr_arg_type; // might by const & (depending on the size)
00299
00300     enum { Is_optional = false };
00301 };
00302
00303 template<typename T> struct Elem<T &>
00304 {
00305     typedef T &arg_type;
00306     typedef T svr_type;
00307     typedef T &svr_arg_type;
00308     enum { Is_optional = false };
00309 };
00310
00311 template<typename T> struct Elem<T const &>
00312 {
00313     typedef T const &arg_type;
00314     typedef T svr_type;
00315     // as the RPC uses a const reference we use it here too,
00316     // we could also use pass by value depending on the size
00317     typedef T const &svr_arg_type;
00318     enum { Is_optional = false };
00319 };
00320
00321 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::Ipc::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>
00395 >
00396 struct _Clnt_xmit : CLNT_OPS {};
00397
00398 template<typename T,
00399     typename ELEM = _Elem<T>,
00400     typename SVR_OPS = Svr_val_ops<typename ELEM::svr_type,
00401     typename Direction<T>::type,
00402     typename Class<typename Detail::_Plain<T>::type>::type>
00403 >
00404 struct _Svr_xmit : SVR_OPS {};
00405
00406 } // namespace Detail
00407 template<typename T> struct Clnt_xmit : Detail::_Clnt_xmit<T> {};
00408 template<typename T> struct Svr_xmit : Detail::_Svr_xmit<T> {};
00409
00410 }}} // namespace Msg, Ipc, L4
00411
00412

```

16.445 l4/sys/cxx/ipc_client File Reference

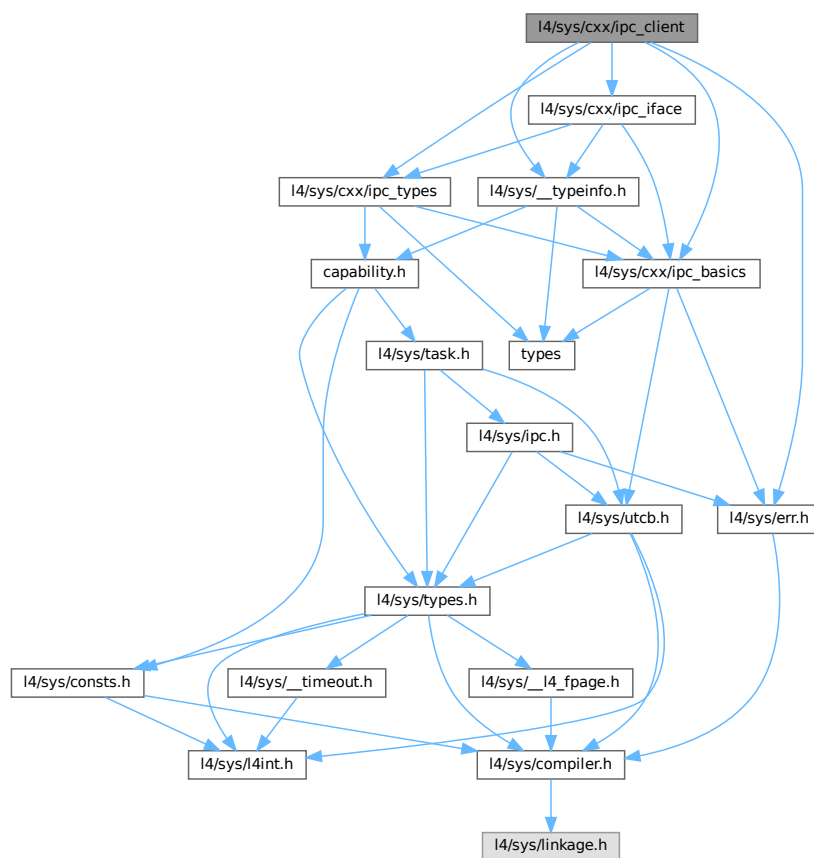
```

#include <l4/sys/cxx/ipc_basics>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/__typeinfo.h>

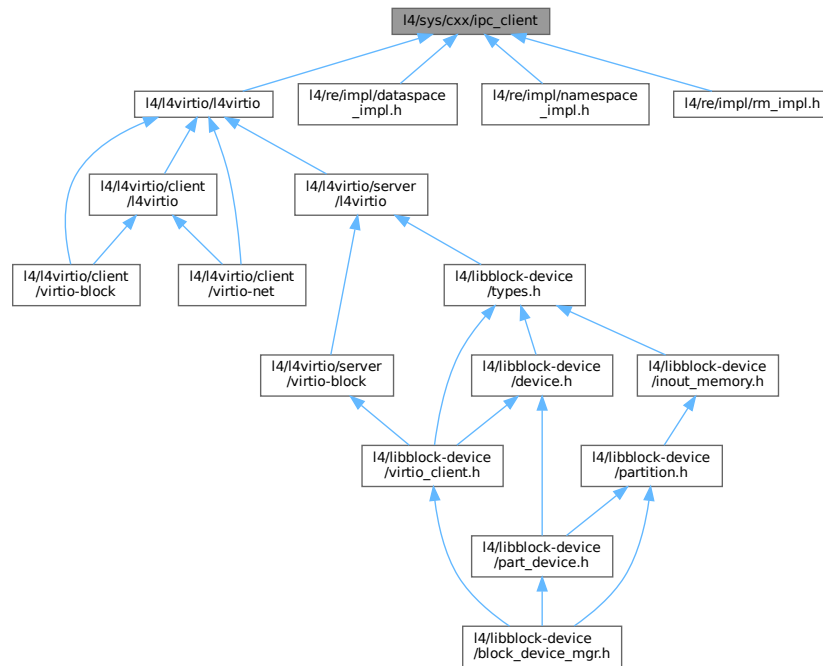
```

```
#include <l4/sys/err.h>
```

Include dependency graph for ipc_client:



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.

16.445.1 Macro Definition Documentation

16.445.1.1 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](#).

16.446 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

```

16.447 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
00040 class Server_iface
00041 {
00042 private:
00043     Server_iface(Server_iface const &);
00044     Server_iface const &operator = (Server_iface const &);
00045
00046 public:
00047     typedef L4::Type_info::Demand Demand;
00048
00049     Server_iface(Server_iface &&) = delete;
00050     Server_iface &operator = (Server_iface &&) = delete;
00051
00052     Server_iface() {}
00053
00054     // Destroy the server interface
00055     virtual ~Server_iface() = 0;
00056
00057     virtual int alloc_buffer_demand(Demand const &demand) = 0;
00058
00059     virtual L4::Cap<void> get_rcv_cap(int index) const = 0;
00060
00061     virtual int realloc_rcv_cap(int index) = 0;
00062
00063     virtual int add_timeout(Timeout *timeout, l4_kernel_clock_t time) = 0;
00064
00065     virtual int remove_timeout(Timeout *timeout) = 0;
00066
00067     template<typename T>
00068     L4::Cap<T> rcv_cap(int index) const
00069     { return L4::cap_cast<T>(get_rcv_cap(index)); }
00070
00071     L4::Cap<void> rcv_cap(int index) const
00072     { return get_rcv_cap(index); }
00073 };
00074
00075 inline Server_iface::~Server_iface() {}
00076
00077 } // namespace Ipc_svr
00078
00079 struct Epiface
00080 {
00081     Epiface(Epiface const &) = delete;
00082     Epiface &operator = (Epiface const &) = delete;
00083
00084     typedef Ipc_svr::Server_iface Server_iface;
00085     typedef Ipc_svr::Server_iface::Demand Demand;
00086
00087     class Stored_cap : public Cap<void>
00088     {
00089     private:
00090         enum { Managed = 0x10 };
00091
00092     public:
00093         Stored_cap() = default;
00094         Stored_cap(Cap<void> const &c, bool managed = false)
00095             : Cap<void>((c.cap() & L4_CAP_MASK) | (managed ? Managed : 0))
00096         {
00097             static_assert (!(L4_CAP_MASK & Managed), "conflicting bits used...");
00098         }
00099     };
00100
00101     Stored_cap cap;
00102 };

```

```

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::Irc> 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>((Msvr *)0, 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, Bool<I::Proto == (long)L4_PROTO_META>());
00487         return _Dispatch<IFACE, typename LIST::type>::f(self, tag, r, utcb);
00488     }
00489 };
00490 };
00491

```

```

00492 template<typename IFACE>
00493 struct Dispatch :
00494     _Dispatch<IFACE, typename L4::Kobject_typeid<IFACE>::Iface_list::type>
00495 {};
00496
00497 } // namespace Detail
00498
00499 template<typename EPIFACE>
00500 struct Dispatch : Detail::Dispatch<typename EPIFACE::Interface>
00501 {};
00502
00503 } // namespace Ipc
00504
00511 template<typename Derived, typename IFACE, typename BASE = L4::Epiface,
00512         bool = cxx::is_polymorphic<BASE>::value>
00513 struct Epiface_t : Epiface_t0<IFACE, BASE>
00514 {
00515     l4_msgtag_t
00516     dispatch(l4_msgtag_t tag, unsigned rights, l4_utcb_t *utcb) final
00517     {
00518         typedef Ipc::Dispatch<Derived> Dispatch;
00519         return Dispatch::f(static_cast<Derived*>(this), tag, rights, utcb);
00520     }
00521 };
00522
00523 template<typename Derived, typename IFACE, typename BASE>
00524 struct Epiface_t<Derived, IFACE, BASE, false> : Epiface_t0<IFACE, BASE>
00525 {
00526     l4_msgtag_t
00527     dispatch(l4_msgtag_t tag, unsigned rights, l4_utcb_t *utcb)
00528     {
00529         typedef Ipc::Dispatch<Derived> Dispatch;
00530         return Dispatch::f(static_cast<Derived*>(this), tag, rights, utcb);
00531     }
00532 };
00533
00539 class Basic_registry
00540 {
00541 public:
00542     typedef Epiface Value;
00543     static Value *find(l4_umword_t label)
00544     { return reinterpret_cast<Value*>(label & ~3UL); }
00545
00564     static l4_msgtag_t dispatch(l4_msgtag_t tag, l4_umword_t label,
00565                                l4_utcb_t *utcb)
00566     {
00567         return find(label)->dispatch(tag, label, utcb);
00568     }
00569 };
00570
00571
00572 } // namespace L4
00573

```

16.448 l4/sys/cxx/ipc_iface File Reference

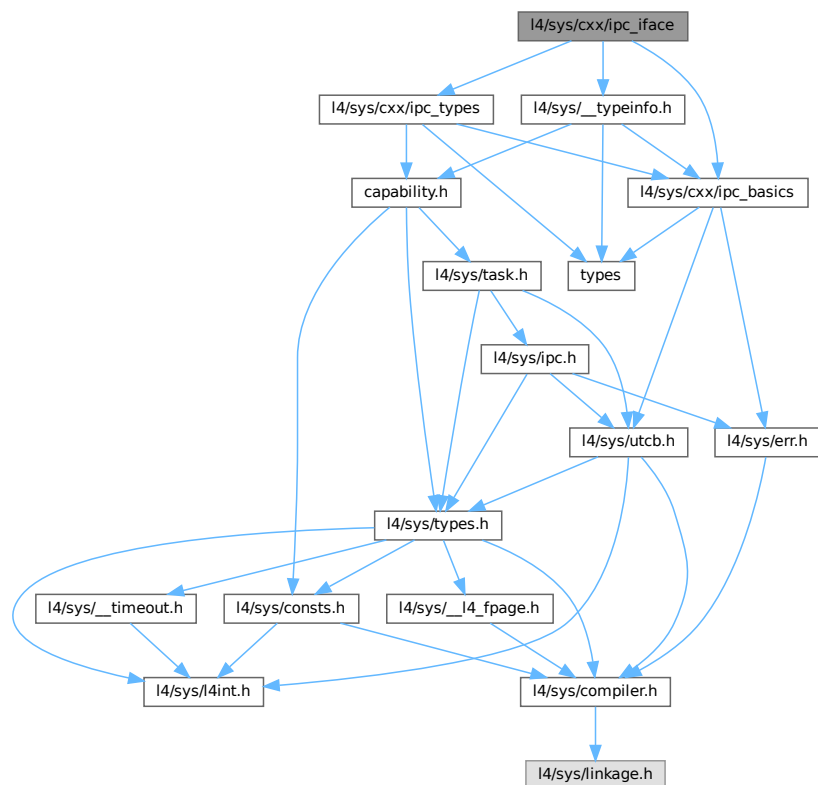
Interface Definition Language.

```

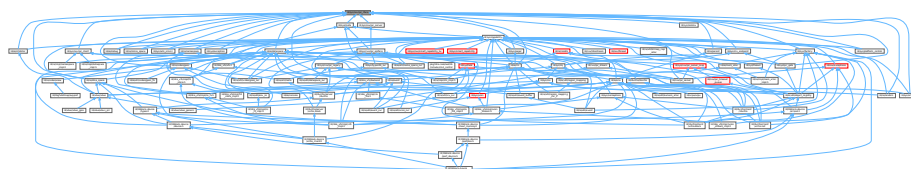
#include <l4/sys/cxx/ipc_basics>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/__typeinfo.h>

```

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).

16.448.1 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](#).

16.448.2 Macro Definition Documentation

16.448.2.1 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 (L4::ipc::Call , L4::ipc::Call_t etc.).

Definition at line 469 of file [ipc_iface](#).

16.448.2.2 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 (L4::ipc::Call , L4::ipc::Call_t 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](#).

16.448.2.3 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 (L4::lpc::Call , L4::lpc::Call_t 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](#).

16.448.2.4 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 (L4::lpc::Call , L4::lpc::Call_t etc.).

Definition at line 484 of file [ipc_iface](#).

16.448.2.5 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 (L4::lpc::Call , L4::lpc::Call_t etc.).

Definition at line 528 of file [ipc_iface](#).

16.448.2.6 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 (L4::ipc::Call , L4::ipc::Call_t etc.).

Definition at line 497 of file [ipc_iface](#).

16.448.2.7 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 (L4::ipc::Call , L4::ipc::Call_t etc.).

Definition at line 512 of file [ipc_iface](#).

16.448.2.8 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 (<code>name_t</code> is used for the type.)
<i>args</i>	The argument list of the RPC function.
<i>attr</i>	Optional RPC attributes (L4::Ipc::Call , L4::Ipc::Call_t etc.).

Definition at line 543 of file [ipc_iface](#).

16.449 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         int r = Data::template write<DIR>((char *)b, offset, limit, m...);
00619         if (L4_LIKELY(r >= offset))
00620             return r - offset;
00621         return r;
00622     }
00623
00624     template<typename DIR>
00625     static int read(void *b, int offset, int limit, long ret,
00626                    typename _Elem<ARGS>::arg_type ...m) noexcept
00627     {
00628         int r = Data::template read<DIR>((char *)b, offset, limit, ret, m...);
00629         if (L4_LIKELY(r >= offset))
00630             return r - offset;
00631         return r;
00632     }
00633 };
00634
00635 template<typename IPC_TYPE, typename OPCODE = void>
00636 struct Part;
00637
00638 // The version without an op-code
00639 template<typename R, typename ...ARGS>
00640 struct Part<R (ARGS...), void> : _Part<ARGS...>
00641 {
00642     typedef Buf<ARGS...> Data;
00643
00644     // write arguments, skipping the dummy opcode
00645     template<typename DIR>
00646     static int write_op(void *b, int offset, int limit,

```

```

00655             int /*placeholder for op*/,
00656             typename _Elem<ARGS>::arg_type ...m) noexcept
00657     {
00658         int r = Data::template write<DIR>((char *)b, offset, limit, m...);
00659         if (L4_LIKELY(r >= offset))
00660             return r - offset;
00661         return r;
00662     }
00663 };
00664
00665 // Message part with additional opcode
00666 template<typename OPCODE, typename R, typename ...ARGS>
00667 struct Part<R (ARGS...), OPCODE> : _Part<ARGS...>
00668 {
00669     typedef OPCODE opcode_type;
00670     typedef Buf<opcode_type, ARGS...> Data;
00671
00672     // write arguments, including the opcode
00673     template<typename DIR>
00674     static int write_op(void *b, int offset, int limit,
00675                         opcode_type op, typename _Elem<ARGS>::arg_type ...m) noexcept
00676     {
00677         int r = Data::template write<DIR>((char *)b, offset, limit, op, m...);
00678         if (L4_LIKELY(r >= offset))
00679             return r - offset;
00680         return r;
00681     }
00682 };
00683 };
00684
00685
00686 } // namespace Detail
00687
00688 //-----
00689 // Implementation of the RPC call
00690 // TODO: Add support for timeout via special RPC argument
00691 // TODO: Add support for passing the UTCB pointer as argument
00692 //
00693 template<typename OP, typename CLASS, typename FLAGS, typename R,
00694         typename ...ARGS>
00695 inline R
00696 Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>::
00697     call(L4::Cap<CLASS> cap,
00698         typename _Elem<ARGS>::arg_type ...a,
00699         l4_utcb_t *utcb) noexcept
00700 {
00701     using namespace Ipc::Msg;
00702
00703     typedef typename Kobject_typeid<CLASS>::Iface::Rpcs Rpcs;
00704     typedef typename Rpcs::template Rpc<OP> Opt;
00705     typedef Detail::Part<ipc_type, typename Rpcs::opcode_type> Args;
00706
00707     l4_msg_regs_t *mrs = l4_utcb_mr_u(utcb);
00708
00709     // handle in-data part of the arguments
00710     int send_bytes =
00711         Args::template write_op<Do_in_data>(mrs->mr, 0, Mr_bytes,
00712                                             Opt::Opcode, a...);
00713
00714     if (L4_UNLIKELY(send_bytes < 0))
00715         return return_err<R>(send_bytes);
00716
00717     send_bytes = align_to<l4_umword_t>(send_bytes);
00718     int const send_words = send_bytes / Word_bytes;
00719     // write the in-items part of the message if there is one
00720     int item_bytes =
00721         Args::template write<Do_in_items>(&mrs->mr[send_words], 0,
00722                                           Mr_bytes - send_bytes, a...);
00723
00724     if (L4_UNLIKELY(item_bytes < 0))
00725         return return_err<R>(item_bytes);
00726
00727     int send_items = item_bytes / Item_bytes;
00728
00729     {
00730         // setup the receive buffers for the RPC call
00731         l4_buf_regs_t *brs = l4_utcb_br_u(utcb);
00732         // XXX: we currently support only one type of receive buffers per call
00733         brs->bdr = 0; // we always start at br[0]
00734
00735         // the limit leaves us at least one register for the zero terminator
00736         // add the buffers given as arguments to the buffer registers
00737         int bytes =
00738             Args::template write<Do_rcv_buffers>(brs->br, 0, Br_bytes - Word_bytes,
00739                                                  a...);
00740
00741         if (L4_UNLIKELY(bytes < 0))
00742             return return_err<R>(bytes);
00743     }

```

```

00743
00744     brs->br[bytes / Word_bytes] = 0;
00745 }
00746
00747
00748 // here we do the actual IPC -----
00749 l4_msgtag_t t;
00750 t = l4_msgtag(CLASS::Protocol, send_words, send_items, 0);
00751 // do the call (Q: do we need support for timeouts?)
00752 if (flags_type::Is_call)
00753     t = l4_ipc_call(cap.cap(), utcb, t, flags_type::timeout());
00754 else
00755 {
00756     t = l4_ipc_send(cap.cap(), utcb, t, flags_type::timeout());
00757     if (L4_UNLIKELY(t.has_error()))
00758         return return_ipc_err<R>(t, utcb);
00759     return return_code<R>(l4_msgtag(0, 0, 0, t.flags()));
00760 }
00761
00762 // unmarshalling starts here -----
00763
00764 // bail out early in the case of an IPC error
00765 if (L4_UNLIKELY(t.has_error()))
00766     return return_ipc_err<R>(t, utcb);
00767 // take the label as return value
00768 long r = t.label();
00769
00770 // bail out on negative error codes too
00771 if (L4_UNLIKELY(r < 0))
00772     return return_err<R>(r);
00773
00774 int const rcv_bytes = t.words() * Word_bytes;
00775
00776 // read the static out-data values to the arguments
00777 int err = Args::template read<Do_out_data>(&mrs->mr, 0, rcv_bytes, r, a...);
00778
00779 int const item_limit = t.items() * Item_bytes;
00780
00781 if (L4_UNLIKELY(err < 0 || item_limit > Mr_bytes))
00782     return return_err<R>(-L4_MSGTOOSHORT);
00783
00784 // read the static out-items to the arguments
00785 err = Args::template read<Do_out_items>(&mrs->mr[t.words()], 0, item_limit,
00786                                         r, a...);
00787
00788 if (L4_UNLIKELY(err < 0))
00789     return return_err<R>(-L4_MSGTOOSHORT);
00790
00791 return return_code<R>(t);
00792 }
00793
00794 } // namespace Msg
00795 } // namespace Ipc
00796 } // namespace L4
00797
00800

```

16.450 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::Rpcs>(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

```

16.451 ipc_ret_array

```

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 "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;
00039         unsigned max;
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 make sure 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 }

```

16.452 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>

```

- 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::Irq_handler_object`
Server object base class for handling IRQ messages.

- namespace **L4**
L4 low-level kernel interface.

Definition in file [ipc_server](#).

16.453 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 }
```

16.454 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         return Base::template get<DIR>((char *)msg, offset, limit);
00161     }
00162
00163     template<typename DIR>
00164     int set(void *msg, unsigned offset, unsigned limit, long ret)
00165     {
00166         return Base::template set<DIR>((char *)msg, offset, limit, ret);
00167     }
00168 }
00169
00170 template<typename IPC_TYPE, typename O, typename ...ARGS>
00171 static l4_msgtag_t
00172 handle_svr_obj_call(O *o, l4_utcb_t *utcb, l4_msgtag_t tag, ARGS ...args)
00173 {
00174     typedef Svr_arg_pack<typename IPC_TYPE::rpc::ipc_type> Pack;
00175     enum
00176     {
00177         Do_reply = IPC_TYPE::rpc::flags_type::Is_call,
00178         Short_err = Do_reply ? -L4_EMSTOOSHORT : -L4_ENOREPLY,
00179     };

```

```

00186     };
00187
00188     // XXX: send a reply or just do not reply in case of a cheating client
00189     if (L4_UNLIKELY(tag.words() + tag.items() * Item_words > Mr_words))
00190         return l4_msgtag(Short_err, 0, 0, 0);
00191
00192     // our whole arguments data structure
00193     Pack pack;
00194     l4_msg_regs_t *mrs = l4_utcb_mr_u(utcb);
00195
00196     int in_pos = Detail::Sizeof<typename IPC_TYPE::opcode_type>::size;
00197
00198     unsigned const in_bytes = tag.words() * Word_bytes;
00199
00200     in_pos = pack.template get<Do_in_data>(&mrs->mr[0], in_pos, in_bytes);
00201
00202     if (L4_UNLIKELY(in_pos < 0))
00203         return l4_msgtag(Short_err, 0, 0, 0);
00204
00205     if (L4_UNLIKELY(pack.template get<Do_out_data>(mrs->mr, 0, Mr_bytes) < 0))
00206         return l4_msgtag(Short_err, 0, 0, 0);
00207
00208
00209     in_pos = pack.template get<Do_in_items>(&mrs->mr[tag.words()], 0,
00210                                           tag.items() * Item_bytes);
00211
00212     if (L4_UNLIKELY(in_pos < 0))
00213         return l4_msgtag(Short_err, 0, 0, 0);
00214
00215     asm volatile ("": "=m" (mrs->mr));
00216
00217     // call the server function
00218     long ret = pack.template obj_call<0, typename IPC_TYPE::rpc, ARGS...>(o, args...);
00219
00220     if (!Do_reply)
00221         return l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00222
00223     // our convention says that negative return value means no
00224     // reply data
00225     if (L4_UNLIKELY(ret < 0))
00226         return l4_msgtag(ret, 0, 0, 0);
00227
00228     // reply with the reply data from the server function
00229     int bytes = pack.template set<Do_out_data>(mrs->mr, 0, Mr_bytes, ret);
00230     if (L4_UNLIKELY(bytes < 0))
00231         return l4_msgtag(-L4_EMSTOOLONG, 0, 0, 0);
00232
00233     unsigned words = (bytes + Word_bytes - 1) / Word_bytes;
00234     bytes = pack.template set<Do_out_items>(&mrs->mr[words], 0,
00235                                           Mr_bytes - words * Word_bytes,
00236                                           ret);
00237     if (L4_UNLIKELY(bytes < 0))
00238         return l4_msgtag(-L4_EMSTOOLONG, 0, 0, 0);
00239
00240     unsigned const items = bytes / Item_bytes;
00241     return l4_msgtag(ret, words, items, 0);
00242 }
00243
00244 //-----
00245
00246 template<typename RPCS, typename OPCODE_TYPE>
00247 struct Dispatch_call;
00248
00249 template<typename CLASS>
00250 struct Dispatch_call<L4::Typeid::Raw_ipc<CLASS>, void>
00251 {
00252     template<typename OBJ, typename ...ARGS>
00253     static l4_msgtag_t
00254     call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, ARGS ...a)
00255     {
00256         return o->op_dispatch(utcb, tag, a...);
00257     }
00258 };
00259
00260 template<typename RPCS>
00261 struct Dispatch_call<RPCS, void>
00262 {
00263     constexpr static unsigned rmask()
00264     { return RPCS::rpc::flags_type::Rights & 3UL; }
00265
00266     template<typename OBJ, typename ...ARGS>
00267     static l4_msgtag_t
00268     call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, unsigned rights, ARGS ...a)
00269     {
00270         if ((rights & rmask()) != rmask())
00271             return l4_msgtag(-L4_EPERM, 0, 0, 0);
00272

```

```

00273     typedef L4::Typeid::Rights<typename RPCS::rpc::class_type> Rights;
00274     return handle_svr_obj_call<RPCS>(o, utcb, tag,
00275                                     Rights(rights), a...);
00276 }
00277 };
00278 };
00279
00280 template<typename RPCS, typename OPCODE_TYPE>
00281 struct Dispatch_call
00282 {
00283     constexpr static unsigned rmask()
00284     { return RPCS::rpc::flags_type::Rights & 3UL; }
00285
00286     template<typename OBJ, typename ...ARGS>
00287     static l4_msgtag_t
00288     _call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, unsigned rights, OPCODE_TYPE op, ARGS ...a)
00289     {
00290         if (L4::Types::Same<typename RPCS::opcode_type, void>::value
00291             || RPCS::Opcode == op)
00292         {
00293             if ((rights & rmask()) != rmask())
00294                 return l4_msgtag(-L4_EPERM, 0, 0, 0);
00295
00296             typedef L4::Typeid::Rights<typename RPCS::rpc::class_type> Rights;
00297             return handle_svr_obj_call<RPCS>(o, utcb, tag,
00298                                             Rights(rights), a...);
00299         }
00300         return Dispatch_call<typename RPCS::next, OPCODE_TYPE>::template
00301             _call<OBJ, ARGS...>(o, utcb, tag, rights, op, a...);
00302     }
00303
00304     template<typename OBJ, typename ...ARGS>
00305     static l4_msgtag_t
00306     call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, unsigned rights, ARGS ...a)
00307     {
00308         OPCODE_TYPE op;
00309         unsigned limit = tag.words() * Word_bytes;
00310         typedef Svr_xmit<OPCODE_TYPE> S;
00311         int err = S::to_svr((char *)l4_utcb_mr_u(utcb)->mr, 0, limit, op,
00312                           Dir_in(), Cls_data());
00313         if (L4_UNLIKELY(err < 0))
00314             return l4_msgtag(-L4_EMSTOOSHORT, 0, 0, 0);
00315
00316         return _call<OBJ, ARGS...>(o, utcb, tag, rights, op, a...);
00317     }
00318 };
00319
00320 template<>
00321 struct Dispatch_call<Typeid::Detail::Rpc_end, void>
00322 {
00323     template<typename OBJ, typename ...ARGS>
00324     static l4_msgtag_t
00325     _call(OBJ *, l4_utcb_t *, l4_msgtag_t, unsigned, int, ARGS ...)
00326     { return l4_msgtag(-L4_ENOSYS, 0, 0, 0); }
00327
00328     template<typename OBJ, typename ...ARGS>
00329     static l4_msgtag_t
00330     call(OBJ *, l4_utcb_t *, l4_msgtag_t, unsigned, ARGS ...)
00331     { return l4_msgtag(-L4_ENOSYS, 0, 0, 0); }
00332 };
00333
00334 template<typename OPCODE_TYPE>
00335 struct Dispatch_call<Typeid::Detail::Rpc_end, OPCODE_TYPE> :
00336     Dispatch_call<Typeid::Detail::Rpc_end, void> {};
00337
00338 template<typename RPCS, typename OBJ, typename ...ARGS>
00339 static l4_msgtag_t
00340 dispatch_call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, unsigned rights, ARGS ...a)
00341 {
00342     return Dispatch_call<typename RPCS::type, typename RPCS::opcode_type>::template
00343         call<OBJ, ARGS...>(o, utcb, tag, rights, a...);
00344 }
00345
00346 } // namespace Msg
00347 } // namespace Ipc
00348 } // namespace L4
00349
00350

```

16.455 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         while(1)
00310             ;
00311     }
00312 #endif
00313 protected:
00315     inline l4_msgtag_t reply_n_wait(l4_msgtag_t reply, l4_umword_t *p, l4_utcb_t *);
00316 };
00317
00318 template< typename L >
00319 inline l4_msgtag_t
00320 Server<L>::reply_n_wait(l4_msgtag_t reply, l4_umword_t *p, l4_utcb_t *utcb)

```

```

00321 {
00322     if (reply.label() != -L4_ENOREPLY)
00323     {
00324         Ipc_svr::Reply_mode m = this->before_reply(reply, utcb);
00325         if (m == Ipc_svr::Reply_compound)
00326         {
00327             this->setup_wait(utcb, m);
00328             return l4_ipc_reply_and_wait(utcb, reply, p, this->timeout());
00329         }
00330         else
00331         {
00332             l4_msgtag_t res = l4_ipc_send(L4_INVALID_CAP | L4_SYSF_REPLY, utcb, reply, this->timeout());
00333             if (res.has_error())
00334                 return res;
00335         }
00336     }
00337     this->setup_wait(utcb, Ipc_svr::Reply_separate);
00338     return l4_ipc_wait(utcb, p, this->timeout());
00339 }
00340
00341 template< typename L >
00342 template< typename DISPATCH >
00343 inline L4_NORETURN void
00344 Server<L>::internal_loop(DISPATCH dispatch, l4_utcb_t *utcb)
00345 {
00346     l4_msgtag_t res;
00347     l4_umword_t p;
00348     l4_msgtag_t r = l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00349
00350     while (true)
00351     {
00352         res = reply_n_wait(r, &p, utcb);
00353         if (res.has_error())
00354         {
00355             this->error(res, utcb);
00356             r = l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00357             continue;
00358         }
00359
00360         r = dispatch(res, p, utcb);
00361     }
00362 }
00363
00364 } // namespace L4

```

16.456 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 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_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 strlength(CHAR *d) { LEN l = 0; while (d[l]) ++l; return l; }
00030         String() {}
00031         String(CHAR *d) : Array<CHAR, LEN>(strlength(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;

```



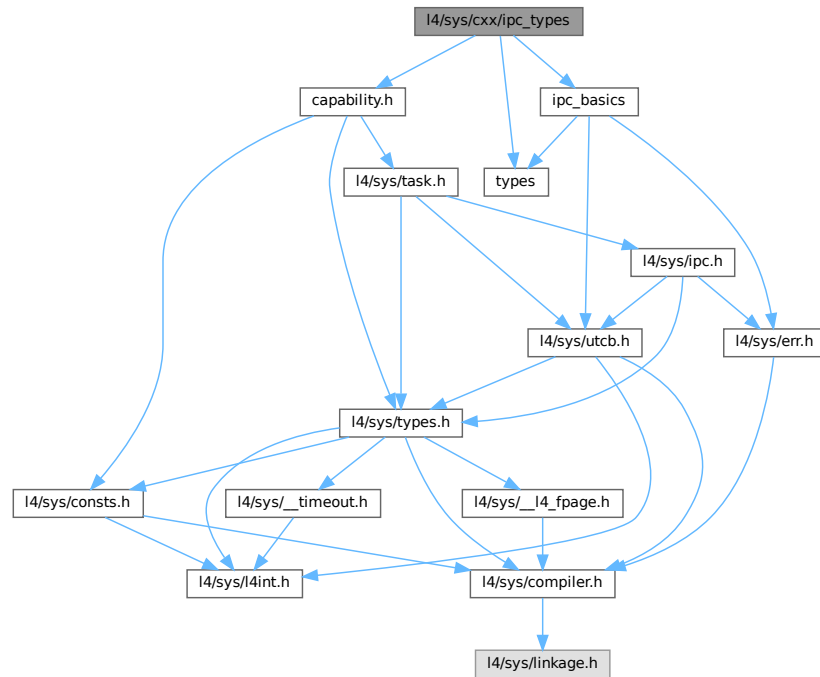
```

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 }

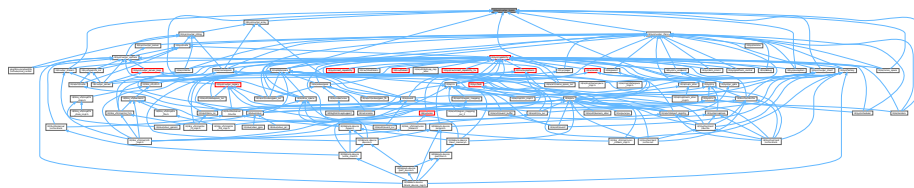
```

16.457 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::Snd_item](#)
RPC wrapper for a send item.
- class [L4::lpc::Buf_item](#)
RPC wrapper for a receive item.
- class [L4::lpc::Gen_fpage< T >](#)
Generic RPC wrapper for L4 flex-pages.
- 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.

Typedefs

- typedef [Gen_fpage< Snd_item >](#) [L4::lpc::Snd_fpage](#)
Send flex-page.
- typedef [Gen_fpage< Buf_item >](#) [L4::lpc::Rcv_fpage](#)
Rcv flex-page.

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.

16.458 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;
00030
00031         template<typename T> struct L4_EXPORT In_out
00032         {
00033             T v;
00034             In_out() {}
00035             In_out(T v) : v(v) {}
00036             operator T () const { return v; }
00037             operator T & () { return v; }
00038         };
00039
00040         namespace Msg {
00041             template<typename A> struct Elem< In_out<A*> > : Elem<A*> {};
00042
00043             template<typename A>
00044             struct Svr_xmit< In_out<A*> > : Svr_xmit<A*>, Svr_xmit<A const*>
00045             {
00046                 using Svr_xmit<A*>::from_svr;
00047                 using Svr_xmit<A const*>::to_svr;
00048             };
00049
00050             template<typename A>
00051             struct Clnt_xmit< In_out<A*> > : Clnt_xmit<A*>, Clnt_xmit<A const*>
00052             {
00053                 using Clnt_xmit<A*>::from_msg;
00054                 using Clnt_xmit<A const*>::to_msg;
00055             };
00056
00057             template<typename A>
00058             struct Is_valid_rpc_type< In_out<A*> > : Is_valid_rpc_type<A*> {};
00059             template<typename A>
00060             struct Is_valid_rpc_type< In_out<A const*> > : L4::Types::False {};
00061
00062 #ifdef CONFIG_ALLOW_REFS
00063             template<typename A> struct Elem< In_out<A&> > : Elem<A&> {};
00064
00065             template<typename A>
00066             struct Svr_xmit< In_out<A&> > : Svr_xmit<A&>, Svr_xmit<A const&>
00067             {
00068                 using Svr_xmit<A&>::from_svr;
00069                 using Svr_xmit<A const&>::to_svr;
00070             };
00071
00072             template<typename A>
00073             struct Clnt_xmit< In_out<A&> > : Clnt_xmit<A&>, Clnt_xmit<A const&>
00074             {
00075                 using Clnt_xmit<A&>::from_msg;
00076                 using Clnt_xmit<A const&>::to_msg;
00077             };
00078
00079             template<typename A>
00080             struct Is_valid_rpc_type< In_out<A&> > : Is_valid_rpc_type<A&> {};
00081             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 Snd_item
00275 {
00276 public:
00277     Snd_item(l4_umword_t base, l4_umword_t data) noexcept : _base(base), _data(data) {}
00278
00279 protected:
00280     l4_umword_t _base;
00281     l4_umword_t _data;
00282 };
00283
00284 class Buf_item
00285 {
00286 public:
00287     Buf_item(l4_umword_t base, l4_umword_t data) noexcept : _base(base), _data(data) {}
00288
00289 protected:
00290     l4_umword_t _base;
00291     l4_umword_t _data;
00292 };
00293
00294 template< typename T >
00295 class L4_EXPORT Gen_fpage : public T
00296 {

```

```

00323 public:
00325     enum Type
00326     {
00327         Special = L4_FPAGE_SPECIAL « 4,
00328         Memory  = L4_FPAGE_MEMORY « 4,
00329         Io      = L4_FPAGE_IO « 4,
00330         Obj     = L4_FPAGE_OBJ « 4
00331     };
00332
00334     enum Map_type
00335     {
00336         Map    = L4_MAP_ITEM_MAP,
00337         Grant  = L4_MAP_ITEM_GRANT,
00338     };
00339
00341     enum Cacheopt
00342     {
00343         None    = 0,
00344         Cached  = L4_FPAGE_CACHEABLE « 4,
00345         Buffered = L4_FPAGE_BUFFERABLE « 4,
00346         Uncached = L4_FPAGE_UNCACHEABLE « 4
00347     };
00348
00349     enum Continue
00350     {
00351         Single  = 0,
00352         Last    = 0,
00353         More    = L4_ITEM_CONT,
00354         Compound = L4_ITEM_CONT,
00355     };
00356
00357 private:
00358     Gen_fpage(l4_umword_t d, l4_umword_t fp) noexcept : T(d, fp) {}
00359
00360     Gen_fpage(Type type, l4_addr_t base, int order,
00361               unsigned char rights,
00362               l4_addr_t snd_base,
00363               Map_type map_type,
00364               Cacheopt cache, Continue cont) noexcept
00365     : T(L4_ITEM_MAP | (snd_base & (~0UL « 10)) | l4_umword_t(map_type) | l4_umword_t(cache)
00366       | l4_umword_t(cont),
00367       base | l4_umword_t(type) | rights | (l4_umword_t(order) « 6))
00368     {}
00369
00370 public:
00371     Gen_fpage() noexcept : T(0, 0) {}
00372     Gen_fpage(l4_fpage_t const &fp, l4_addr_t snd_base = 0,
00373               Map_type map_type = Map,
00374               Cacheopt cache = None, Continue cont = Last) noexcept
00375     : T(L4_ITEM_MAP | (snd_base & (~0UL « 10)) | l4_umword_t(map_type) | l4_umword_t(cache)
00376       | l4_umword_t(cont),
00377       fp.raw)
00378     {}
00379
00380     Gen_fpage(L4::Cap<void> cap, unsigned rights, Map_type map_type = Map) noexcept
00381     : T(L4_ITEM_MAP | l4_umword_t(map_type) | (rights & 0xf0),
00382       cap.fpage(rights).raw)
00383     {}
00384
00385     static Gen_fpage<T> obj(l4_addr_t base, int order,
00386                           unsigned char rights,
00387                           l4_addr_t snd_base = 0,
00388                           Map_type map_type = Map,
00389                           Continue cont = Last) noexcept
00390     {
00391         return Gen_fpage<T>(Obj, base « 12, order, rights, snd_base, map_type, None, cont);
00392     }
00393
00394     static Gen_fpage<T> mem(l4_addr_t base, int order,
00395                           unsigned char rights,
00396                           l4_addr_t snd_base = 0,
00397                           Map_type map_type = Map,
00398                           Cacheopt cache = None, Continue cont = Last) noexcept
00399     {
00400         return Gen_fpage<T>(Memory, base, order, rights, snd_base,
00401                           map_type, cache, cont);
00402     }
00403
00404     static Gen_fpage<T> rmem(l4_addr_t base, int order, l4_addr_t snd_base,
00405                           unsigned char rights, unsigned cap_br) noexcept
00406     {
00407         return Gen_fpage<T>(
00408             L4_ITEM_MAP | (snd_base & (~0UL « 10)) | l4_umword_t(Map)
00409             | l4_umword_t(None) | l4_umword_t(Compound) | (cap_br « 8),
00410             base | l4_umword_t(Memory) | rights | (l4_umword_t(order) « 6));
00411     }
00412

```

```

00413
00414     static Gen_fpage<T> io(l4_addr_t base, int order,
00415                          unsigned char rights,
00416                          l4_addr_t snd_base = 0,
00417                          Map_type map_type = Map,
00418                          Continue cont = Last) noexcept
00419     {
00420         return Gen_fpage<T>(Io, base « 12, order, rights, snd_base, map_type, None, cont);
00421     }
00422
00423     unsigned order() const noexcept { return (T::_data » 6) & 0x3f; }
00424     unsigned snd_order() const noexcept { return (T::_data » 6) & 0x3f; }
00425     unsigned rcv_order() const noexcept { return (T::_base » 6) & 0x3f; }
00426     l4_addr_t base() const noexcept { return T::_data & (~0UL « 12); }
00427     l4_addr_t snd_base() const noexcept { return T::_base & (~0UL « 10); }
00428     void snd_base(l4_addr_t b) noexcept { T::_base = (T::_base & ~(~0UL « 10)) | (b & (~0UL « 10)); }
00429
00431     bool is_valid() const noexcept { return T::_base & L4_ITEM_MAP; }
00442     bool cap_received() const noexcept { return (T::_base & 0x3e) == 0x38; }
00454     bool id_received() const noexcept { return (T::_base & 0x3e) == 0x3c; }
00464     bool local_id_received() const noexcept { return (T::_base & 0x3e) == 0x3e; }
00465
00472     bool is_compound() const noexcept { return T::_base & 1; }
00474     l4_umword_t data() const noexcept { return T::_data; }
00476     l4_umword_t base_x() const noexcept { return T::_base; }
00477 };
00478
00479
00481 typedef Gen_fpage<Snd_item> Snd_fpage;
00483 typedef Gen_fpage<Buf_item> Rcv_fpage;
00484
00485 #ifdef L4_CXX_IPC_SUPPORT_STRINGS
00486 template <typename T, typename B>
00487 class Gen_string : public T
00488 {
00489 public:
00490     Gen_string() noexcept : T(0, 0) {}
00491     Gen_string(B buf, unsigned long size) noexcept
00492         : T(size « 10, l4_umword_t(buf))
00493     {}
00494
00495     unsigned long len() const noexcept { return T::_base » 10; }
00496 };
00497
00498 typedef Gen_string<Snd_item, void const *> Snd_string;
00499 typedef Gen_string<Buf_item, void *> Rcv_string;
00500 #endif
00501
00502
00503 namespace Msg {
00504
00505     // Snd_fpage are out items
00506     template<> struct Class<L4::Ipc::Snd_fpage> : Cls_item {};
00507
00508     // Rcv_fpage are buffer items
00509     template<> struct Class<L4::Ipc::Rcv_fpage> : Cls_buffer {};
00510
00511     // Remove receive buffers from server-side arguments
00512     template<> struct Elem<L4::Ipc::Rcv_fpage>
00513     {
00514         typedef L4::Ipc::Rcv_fpage arg_type;
00515         typedef void svr_type;
00516         typedef void svr_arg_type;
00517         enum { Is_optional = false };
00518     };
00519
00520     // Rcv_fpage are buffer items
00521     template<> struct Class<L4::Ipc::Small_buf> : Cls_buffer {};
00522
00523     // Remove receive buffers from server-side arguments
00524     template<> struct Elem<L4::Ipc::Small_buf>
00525     {
00526         typedef L4::Ipc::Small_buf arg_type;
00527         typedef void svr_type;
00528         typedef void svr_arg_type;
00529         enum { Is_optional = false };
00530     };
00531 } // namespace Msg
00532
00533 // L4::Cap<> handling
00534
00545 template<typename T> class Cap
00546 {
00547     template<typename O> friend class Cap;
00548     l4_umword_t _cap_n_rights;
00549
00550 public:

```



```

00551     enum
00552     {
00553         Rights_mask = 0xff,
00554         Cap_mask    = L4_CAP_MASK
00555     };
00556
00557     template<typename O>
00558     Cap(Cap<O> const &o) noexcept : _cap_n_rights(o._cap_n_rights)
00559     { T *x = (O*)1; (void)x; }
00560
00561     Cap(L4::Cap<T> cap) noexcept
00562     : _cap_n_rights((cap.cap() & Cap_mask) | (cap ? L4_CAP_FPAGE_R : 0))
00563     {}
00564
00565     template<typename O>
00566     Cap(L4::Cap<O> cap) noexcept
00567     : _cap_n_rights((cap.cap() & Cap_mask) | (cap ? L4_CAP_FPAGE_R : 0))
00568     { T *x = (O*)1; (void)x; }
00569
00570     Cap() noexcept : _cap_n_rights(L4_INVALID_CAP) {}
00571
00572     Cap(L4::Cap<T> cap, unsigned char rights) noexcept
00573     : _cap_n_rights((cap.cap() & Cap_mask) | (rights & Rights_mask)) {}
00574
00575     static Cap from_ci(l4_cap_idx_t c) noexcept
00576     { return Cap(L4::Cap<T>(c & Cap_mask), c & Rights_mask); }
00577
00578     L4::Cap<T> cap() const noexcept
00579     { return L4::Cap<T>(_cap_n_rights & Cap_mask); }
00580
00581     unsigned rights() const noexcept
00582     { return _cap_n_rights & Rights_mask; }
00583
00584     L4::Ipc::Snd_fpage fpage() const noexcept
00585     { return L4::Ipc::Snd_fpage(cap(), rights()); }
00586
00587     bool is_valid() const noexcept
00588     { return !(_cap_n_rights & L4_INVALID_CAP_BIT); }
00589 };
00590
00591 template<typename T>
00592 Cap<T> make_cap(L4::Cap<T> cap, unsigned rights) noexcept
00593 { return Cap<T>(cap, rights); }
00594
00595 template<typename T>
00596 Cap<T> make_cap_rw(L4::Cap<T> cap) noexcept
00597 { return Cap<T>(cap, L4_CAP_FPAGE_RW); }
00598
00599 template<typename T>
00600 Cap<T> make_cap_rws(L4::Cap<T> cap) noexcept
00601 { return Cap<T>(cap, L4_CAP_FPAGE_RWS); }
00602
00603 template<typename T>
00604 Cap<T> make_cap_full(L4::Cap<T> cap) noexcept
00605 { return Cap<T>(cap, L4_CAP_FPAGE_RWS | L4_FPAGE_C_OBJ_RIGHTS); }
00606
00607 // caps are special the have an invalid representation
00608 template<typename T> struct L4_EXPORT Opt< Cap<T> >
00609 {
00610     Cap<T> _value;
00611     Opt() noexcept {}
00612     Opt(Cap<T> value) noexcept : _value(value) {}
00613     Opt(L4::Cap<T> value) noexcept : _value(value) {}
00614     Opt &operator = (Cap<T> value) noexcept
00615     { this->_value = value; }
00616     Opt &operator = (L4::Cap<T> value) noexcept
00617     { this->_value = value; }
00618
00619     Cap<T> value() const noexcept { return this->_value; }
00620     bool is_valid() const noexcept { return this->_value.is_valid(); }
00621 };
00622
00623 namespace Msg {
00624     // prohibit L4::Cap as argument
00625     template<typename A>
00626     struct Is_valid_rpc_type< L4::Cap<A> > : L4::Types::False {};
00627
00628     template<typename A> struct Class< Cap<A> > : Cls_item {};
00629     template<typename A> struct Elem< Cap<A> >
00630     {
00631         enum { Is_optional = false };
00632         typedef Cap<A> arg_type;
00633         typedef L4::Ipc::Snd_fpage svr_type;
00634         typedef L4::Ipc::Snd_fpage svr_arg_type;
00635     };
00636 }

```

```

00699
00700
00701 template<typename A, typename CLASS>
00702 struct Svr_val_ops<Cap<A>, Dir_in, CLASS> :
00703     Svr_val_ops<L4::Ipc::Snd_fpage, Dir_in, CLASS>
00704 {};
00705
00706 template<typename A, typename CLASS>
00707 struct Clnt_val_ops<Cap<A>, Dir_in, CLASS> :
00708     Clnt_noops< Cap<A> >
00709 {
00710     using Clnt_noops< Cap<A> >::to_msg;
00711
00712     static int to_msg(char *msg, unsigned offset, unsigned limit,
00713                     Cap<A> arg, Dir_in, Cls_item) noexcept
00714     {
00715         // passing an invalid cap as mandatory argument is an error
00716         // XXX: This checks for a client calling error, we could
00717         //      also just ignore this for performance reasons and
00718         //      let the client fail badly (Alex: I'd prefer this)
00719         if (L4_UNLIKELY(!arg.is_valid()))
00720             return -L4_MSGMISSARG;
00721
00722         return msg_add(msg, offset, limit, arg.fpage());
00723     }
00724 };
00725
00726 template<typename A>
00727 struct Elem<Out<L4::Cap<A> > >
00728 {
00729     enum { Is_optional = false };
00730     typedef L4::Cap<A> arg_type;
00731     typedef Ipc::Cap<A> svr_type;
00732     typedef svr_type &svr_arg_type;
00733 };
00734
00735 template<typename A> struct Direction< Out< L4::Cap<A> > > : Dir_out {};
00736 template<typename A> struct Class< Out< L4::Cap<A> > > : Cls_item {};
00737
00738 template<typename A>
00739 struct Clnt_val_ops< L4::Cap<A>, Dir_out, Cls_item > :
00740     Clnt_noops< L4::Cap<A> >
00741 {
00742     using Clnt_noops< L4::Cap<A> >::to_msg;
00743     static int to_msg(char *msg, unsigned offset, unsigned limit,
00744                     L4::Cap<A> arg, Dir_in, Cls_buffer) noexcept
00745     {
00746         if (L4_UNLIKELY(!arg.is_valid()))
00747             return -L4_MSGMISSARG; // no buffer inserted
00748         return msg_add(msg, offset, limit, Small_buf(arg));
00749     }
00750 };
00751
00752 template<typename A>
00753 struct Svr_val_ops< L4::Ipc::Cap<A>, Dir_out, Cls_item > :
00754     Svr_noops<Cap<A> &>
00755 {
00756     using Svr_noops<Cap<A> &>::from_svr;
00757     static int from_svr(char *msg, unsigned offset, unsigned limit, long,
00758                     Cap<A> arg, Dir_out, Cls_item) noexcept
00759     {
00760         if (L4_UNLIKELY(!arg.is_valid()))
00761             // do not map anything
00762             return msg_add(msg, offset, limit, L4::Ipc::Snd_fpage(arg.cap(), 0));
00763
00764         return msg_add(msg, offset, limit, arg.fpage());
00765     }
00766 };
00767
00768 // prohibit a UTCTB pointer as normal RPC argument
00769 template<> struct Is_valid_rpc_type<l4_utcb_t *> : L4::Types::False {};
00770
00771 } // namespace Msg
00772 } // namespace Ipc
00773 } // namespace L4
00774

```

16.459 ipc_varg

```

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 "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() { return (L4_varg_type)(Id & ~L4_VARG_TYPE_SIGN); }
00042     static L4_varg_type signed_id() { return (L4_varg_type)(Id | L4_VARG_TYPE_SIGN); }
00043     static L4_varg_type id() { return (L4_varg_type)Id; }
00044 };
00045
00046 template< typename T > struct Va_type_id;
00047 template<> struct Va_type_id<l4_umword_t> { enum { Id = L4_VARG_TYPE_UMWORD }; };
00048 template<> struct Va_type_id<l4_mword_t> { enum { Id = L4_VARG_TYPE_MWORD }; };
00049 template<> struct Va_type_id<l4_fpage_t> { enum { Id = L4_VARG_TYPE_FPAGE }; };
00050 template<> struct Va_type_id<void> { enum { Id = L4_VARG_TYPE_NIL }; };
00051 template<> struct Va_type_id<char const *> { enum { Id = L4_VARG_TYPE_STRING }; };
00052
00053 template< typename T > struct Va_type;
00054
00055 template<> struct Va_type<l4_umword_t> : Generic_va_type<l4_umword_t, Va_type_id> {};
00056 template<> struct Va_type<l4_mword_t> : Generic_va_type<l4_mword_t, Va_type_id> {};
00057 template<> struct Va_type<l4_fpage_t> : Generic_va_type<l4_fpage_t, Va_type_id> {};
00058
00059 template<> struct Va_type<void>
00060 {
00061     typedef void Ret_value;
00062     typedef void Value;
00063
00064     static void const *addr_of(void) { return 0; }
00065
00066     static void value(void const *) {}
00067     static L4_varg_type id() { return L4_VARG_TYPE_NIL; }
00068     static unsigned size(void const *) { return 0; }
00069 };
00070
00071 template<> struct Va_type<char const *>
00072 {
00073     typedef char const *Ret_value;
00074     typedef char const *Value;
00075
00076     static void const *addr_of(Value v) { return v; }
00077
00078     static L4_varg_type id() { return L4_VARG_TYPE_STRING; }
00079     static unsigned size(void const *s)
00080     {
00081         char const *_s = reinterpret_cast<char const *>(s);
00082         int l = 1;
00083         while (*_s)
00084         {
00085             ++_s; ++l;
00086         }
00087         return l;
00088     }
00089
00090     static Ret_value value(void const *d) { return (char const *)d; }
00091 };

```

```

00092
00096 class Varg
00097 {
00098 private:
00099     enum { Direct_data = 0x8000 };
00100     l4_umword_t _tag;
00101     char const *_d;
00102
00103 public:
00104
00106     typedef l4_umword_t Tag;
00107
00109     L4_varg_type type() const { return (L4_varg_type)(_tag & 0xff); }
00114     unsigned length() const { return _tag » 16; }
00116     Tag tag() const { return _tag & ~Direct_data; }
00118     void tag(Tag tag) { _tag = tag; }
00120     void data(char const *d) { _d = d; }
00121
00123     char const *data() const
00124     {
00125         if (_tag & Direct_data)
00126         {
00127             union T { char const *d; char v[sizeof(char const *)]; };
00128             return reinterpret_cast<T const *>(&_d)->v;
00129         }
00130         return _d;
00131     }
00132
00134 #if __cplusplus >= 201103L
00135     Varg() = default;
00136 #else
00137     Varg() {}
00138 #endif
00139
00141     Varg(L4_varg_type t, void const *v, int len)
00142     : _tag(t | ((l4_mword_t)len « 16)), _d((char const *)v)
00143     {}
00144
00145     static Varg nil() { return Varg(L4_VARG_TYPE_NIL, 0, 0); }
00146
00153     template< typename V >
00154     typename Va_type<V>::Ret_value value() const
00155     {
00156         if (_tag & Direct_data)
00157         {
00158             union X { char const *d; V v; };
00159             return reinterpret_cast<X const &>(&_d).v;
00160         }
00161
00162         return Va_type<V>::value(_d);
00163     }
00164
00165
00167     template< typename T >
00168     bool is_of() const { return Va_type<T>::id() == type(); }
00169
00171     bool is_nil() const { return is_of<void>(); }
00172
00174     bool is_of_int() const
00175     { return (type() & ~L4_VARG_TYPE_SIGN) == L4_VARG_TYPE_UMWORD; }
00176
00183     template< typename T >
00184     bool get_value(typename Va_type<T>::Value *v) const
00185     {
00186         if (!is_of<T>())
00187             return false;
00188
00189         *v = this->value<T>();
00190         return true;
00191     }
00192
00194     template< typename T >
00195     void set_value(void const *d)
00196     {
00197         typedef Va_type<T> Vt;
00198         _tag = Vt::id() | (Vt::size(d) « 16);
00199         _d = (char const *)d;
00200     }
00201
00203     template<typename T>
00204     void set_direct_value(T val, typename L4::Types::Enable_if<sizeof(T) <= sizeof(char const *)>,
00205 bool>::type = true)
00206     {
00207         static_assert(sizeof(T) <= sizeof(char const *), "direct Varg value too big");
00208         typedef Va_type<T> Vt;
00209         _tag = Vt::id() | (sizeof(T) « 16) | Direct_data;
00210         union X { char const *d; T v; };

```

```

00210     reinterpret_cast<X &>(_d).v = val;
00211 }
00212
00214 template<typename T> explicit
00215 Varg(T const *data) { set_value<T>(data); }
00217 Varg(char const *data) { set_value<char const *>(data); }
00218
00220 template<typename T> explicit
00221 Varg(T data, typename L4::Types::Enable_if<sizeof(T) <= sizeof(char const *), bool>::type = true)
00222 { set_direct_value<T>(data); }
00223 };
00224
00225
00226 template<typename T>
00227 class Varg_t : public Varg
00228 {
00229 public:
00230     typedef typename Va_type<T>::Value Value;
00231     explicit Varg_t(Value v) : Varg()
00232     { _data = v; set_value<T>(Va_type<T>::addr_of(_data)); }
00233
00234 private:
00235     Value _data;
00236 };
00237
00238 template<unsigned MAX = L4_UTCB_GENERIC_DATA_SIZE>
00239 class Varg_list;
00240
00252 class Varg_list_ref
00253 {
00254 private:
00255     template<unsigned T>
00256     friend class Varg_list;
00257
00259     class Iter_state
00260     {
00261     private:
00262         using M = l4_umword_t;
00263         using Mp = M const *;
00264         Mp _c;
00265         Mp _e;
00266
00268         Mp next_arg(Varg const &a) const
00269         {
00270             return _c + 1 + (Msg::align_to<M>(a.length()) / sizeof(M));
00271         }
00272
00273     public:
00275         Iter_state() : _c(nullptr) {}
00276
00278         Iter_state(Mp c, Mp e) : _c(c), _e(e)
00279         {}
00280
00282         bool valid() const
00283         { return _c && _c < _e; }
00284
00286         Mp begin() const { return _c; }
00287
00289         Mp end() const { return _e; }
00290
00295         Varg pop()
00296         {
00297             if (!valid())
00298                 return Varg::nil();
00299
00300             Varg a;
00301             a.tag(_c[0]);
00302             a.data(reinterpret_cast<char const *>(&_c[1]));
00303             _c = next_arg(a);
00304             if (_c > _e)
00305                 return Varg::nil();
00306
00307             return a;
00308         }
00309
00311         bool operator == (Iter_state const &o) const
00312         { return _c == o._c; }
00313
00315         bool operator != (Iter_state const &o) const
00316         { return _c != o._c; }
00317     };
00318
00319     Iter_state _s;
00320
00321     public:
00323     Varg_list_ref() = default;
00324

```

```

00331 Varg_list_ref(void const *start, void const *end)
00332 : _s(reinterpret_cast<l4_umword_t const *>(start),
00333     reinterpret_cast<l4_umword_t const *>(end))
00334 {}
00335
00337 class Iterator
00338 {
00339 private:
00340     Iter_state _s;
00341     Varg _a;
00342
00343 public:
00345     Iterator(Iter_state const &s)
00346     : _s(s)
00347     {
00348         _a = _s.pop();
00349     }
00350
00352     explicit operator bool () const
00353     { return !_a.is_nil(); }
00354
00356     Iterator &operator ++ ()
00357     {
00358         if (!_a.is_nil())
00359             _a = _s.pop();
00360
00361         return *this;
00362     }
00363
00365     Varg operator * () const
00366     { return _a; }
00367
00369     bool equals(Iterator const &o) const
00370     {
00371         if (_a.is_nil() && o._a.is_nil())
00372             return true;
00373
00374         return _s == o._s;
00375     }
00376
00377     bool operator == (Iterator const &o) const
00378     { return equals(o); }
00379
00380     bool operator != (Iterator const &o) const
00381     { return !equals(o); }
00382 };
00383
00385 Varg pop_front()
00386 { return _s.pop(); }
00387
00389 Varg next()
00390     L4_DEPRECATED("Use range for or pop_front.")
00391 { return _s.pop(); }
00392
00394 Iterator begin() const
00395 { return Iterator(_s); }
00396
00398 Iterator end() const
00399 { return Iterator(Iter_state()); }
00400 };
00401
00409 template<unsigned MAX>
00410 class Varg_list : public Varg_list_ref
00411 {
00412     l4_umword_t data[MAX];
00413     Varg_list(Varg_list const &);
00414
00415 public:
00417     Varg_list(Varg_list_ref const &r)
00418     {
00419         if (!r._s.valid())
00420             return;
00421
00422         l4_umword_t const *rs = r._s.begin();
00423         unsigned c = r._s.end() - rs;
00424         for (unsigned i = 0; i < c; ++i)
00425             data[i] = rs[i];
00426
00427         this->_s = Iter_state(data, data + c);
00428     }
00429 };
00430
00431 namespace Msg {
00432     template<> struct Elem<Varg const *>
00433     {
00434         typedef Varg const *arg_type;

```

```

00436     typedef Varg_list_ref svr_type;
00437     typedef Varg_list_ref svr_arg_type;
00438     enum { Is_optional = false };
00439 };
00440
00441 template<> struct Is_valid_rpc_type<Varg> : L4::Types::False {};
00442 template<> struct Is_valid_rpc_type<Varg *> : L4::Types::False {};
00443 template<> struct Is_valid_rpc_type<Varg &> : L4::Types::False {};
00444 template<> struct Is_valid_rpc_type<Varg const &> : L4::Types::False {};
00445
00446 template<> struct Direction<Varg const *> : Dir_in {};
00447 template<> struct Class<Varg const *> : Cls_data {};
00448
00449 template<typename DIR, typename CLASS>
00450 struct Clnt_val_ops<Varg, DIR, CLASS>;
00451
00452 template<>
00453 struct Clnt_val_ops<Varg, Dir_in, Cls_data> :
00454     Clnt_noops<Varg const &>
00455 {
00456     using Clnt_noops<Varg const &>::to_msg;
00457     static int to_msg(char *msg, unsigned offs, unsigned limit,
00458                     Varg const &a, Dir_in, Cls_data)
00459     {
00460         for (Varg const *i = &a; i->tag(); ++i)
00461         {
00462             offs = align_to<l4_umword_t>(offs);
00463             if (L4_UNLIKELY(!check_size<l4_umword_t>(offs, limit)))
00464                 return -L4_MSGTOOLONG;
00465             *reinterpret_cast<l4_umword_t*>(msg + offs) = i->tag();
00466             offs += sizeof(l4_umword_t);
00467             if (L4_UNLIKELY(!check_size<char>(offs, limit, i->length())))
00468                 return -L4_MSGTOOLONG;
00469             char const *d = i->data();
00470             for (unsigned x = 0; x < i->length(); ++x)
00471                 msg[offs++] = *d++;
00472         }
00473         return offs;
00474     }
00475 };
00476
00477
00478 template<>
00479 struct Svr_val_ops<Varg_list_ref, Dir_in, Cls_data> :
00480     Svr_noops<Varg_list_ref>
00481 {
00482     using Svr_noops<Varg_list_ref>::to_svr;
00483     static int to_svr(char *msg, unsigned offset, unsigned limit,
00484                     Varg_list_ref &a, Dir_in, Cls_data)
00485     {
00486         unsigned start = align_to<l4_umword_t>(offset);
00487         unsigned offs;
00488         for (offs = start; offs < limit;)
00489         {
00490             unsigned noffs = align_to<l4_umword_t>(offs);
00491             if (L4_UNLIKELY(!check_size<l4_umword_t>(noffs, limit)))
00492                 break;
00493
00494             offs = noffs;
00495             Varg arg;
00496             arg.tag(*reinterpret_cast<l4_umword_t*>(msg + offs));
00497
00498             if (!arg.tag())
00499                 break;
00500
00501             offs += sizeof(l4_umword_t);
00502
00503             if (L4_UNLIKELY(!check_size<char>(offs, limit, arg.length())))
00504                 return -L4_MSGTOOLONG;
00505             offs += arg.length();
00506         }
00507
00508         a = Varg_list_ref(msg + start, msg + align_to<l4_umword_t>(offs));
00509         return offs;
00510     }
00511 };
00512
00513 }

```


Namespaces

- namespace [L4](#)

[L4](#) low-level kernel interface.

16.461 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     { T* __t = ((O*)100); (void)__t; }
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         T* __t = ((O*)100); (void)__t;
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     { T* __t = ((O*)100); (void)__t; }
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         T* __t = ((O*)100); (void)__t;
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()))

```

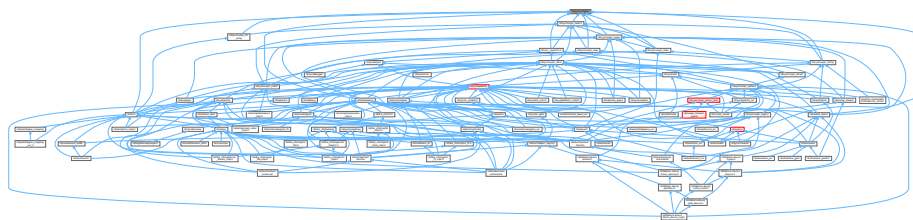
```

00163     {}
00164
00165     template<typename O>
00166     Shared_cap_impl(Shared_cap_impl<O, IMPL> const &o) noexcept
00167     : Base(IMPL::copy(o))
00168     { T* __t = ((O*)100); (void)__t; }
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         T* __t = ((O*)100); (void)__t;
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

```

16.462 I4/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.

16.462.1 Macro Definition Documentation

16.462.1.1 L4_TYPES_FLAGS_OPS_DEF

```
#define L4_TYPES_FLAGS_OPS_DEF (
    T )
```

Value:

```
friend constexpr T operator ~ (T f)
{
    return T(~((typename L4::Types::Int_for_type<T>::type)f));
}

friend constexpr T operator | (T l, T r)
{
    return T(((typename L4::Types::Int_for_type<T>::type)l)
              | ((typename L4::Types::Int_for_type<T>::type)r));
}

friend constexpr T operator & (T l, T r)
{
    return T(((typename L4::Types::Int_for_type<T>::type)l)
              & ((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 207 of file [types](#).

16.463 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         struct Private_bool;
00075         value_type _v;
00076         explicit Flags(value_type v) : _v(v) {}
00077
00078     public:
00080         enum None_type { None };
00081
00090         Flags(None_type) : _v(0) {}
00091
00093         Flags() : _v(0) {}
00094
00103         Flags(BITS_ENUM e) : _v(((value_type)1) << e) {}
00104
00110         static type from_raw(value_type v) { return type(v); }
00111
00113         operator Private_bool * () const
00114         { return _v != 0 ? (Private_bool *)1 : 0; }
00115
00117         bool operator ! () const { return _v == 0; }
00118
00120         friend type operator | (type lhs, type rhs)
00121         { return type(lhs._v | rhs._v); }
00122
00124         friend type operator | (type lhs, bits_enum_type rhs)
00125         { return lhs | type(rhs); }
00126
00128         friend type operator & (type lhs, type rhs)
00129         { return type(lhs._v & rhs._v); }
00130
00132         friend type operator & (type lhs, bits_enum_type rhs)
00133         { return lhs & type(rhs); }
00134
00136         type &operator |= (type rhs) { _v |= rhs._v; return *this; }
00138         type &operator |= (bits_enum_type rhs) { return operator |= (type(rhs)); }
00139
00141         type &operator &= (type rhs) { _v &= rhs._v; return *this; }
00143         type &operator &= (bits_enum_type rhs) { return operator &= (type(rhs)); }
00144
00146         type operator ~ () const { return type(~_v); }
00147
00154         type &clear(bits_enum_type flag) { return operator &= (~type(flag)); }
00155
00157         value_type as_value() const { return _v; }
00158     };
00159
00165     template<unsigned SIZE, bool = true> struct Int_for_size;
00166
00167     template<> struct Int_for_size<sizeof(unsigned char), true>
00168     { typedef unsigned char type; };
00169
00170     template<> struct Int_for_size<sizeof(unsigned short),
00171                                     (sizeof(unsigned short) > sizeof(unsigned char))>
00172     { typedef unsigned short type; };
00173
00174     template<> struct Int_for_size<sizeof(unsigned),
00175                                     (sizeof(unsigned) > sizeof(unsigned short))>
00176     { typedef unsigned type; };
00177
00178     template<> struct Int_for_size<sizeof(unsigned long),
00179                                     (sizeof(unsigned long) > sizeof(unsigned))>
00180     { typedef unsigned long type; };
00181
00182     template<> struct Int_for_size<sizeof(unsigned long long),
00183                                     (sizeof(unsigned long long) > sizeof(unsigned long))>
00184     { typedef unsigned long long type; };
00185

```

```

00192 template<typename T> struct Int_for_type
00193 {
00197     typedef typename Int_for_size<sizeof(T)>::type type;
00198 };
00199
00207 #define L4_TYPES_FLAGS_OPS_DEF(T)
00208     friend constexpr T operator ~ (T f)
00209     {
00210         return T(~((typename L4::Types::Int_for_type<T>::type)f));
00211     }
00212
00213     friend constexpr T operator | (T l, T r)
00214     {
00215         return T(((typename L4::Types::Int_for_type<T>::type)l)
00216             | ((typename L4::Types::Int_for_type<T>::type)r));
00217     }
00218
00219     friend constexpr T operator & (T l, T r)
00220     {
00221         return T(((typename L4::Types::Int_for_type<T>::type)l)
00222             & ((typename L4::Types::Int_for_type<T>::type)r));
00223     }
00224
00231 template<typename DT>
00232 struct Flags_ops_t
00233 {
00235     friend constexpr DT operator | (DT l, DT r)
00236     { return DT(l.raw | r.raw); }
00237
00239     friend constexpr DT operator & (DT l, DT r)
00240     { return DT(l.raw & r.raw); }
00241
00243     friend constexpr bool operator == (DT l, DT r)
00244     { return l.raw == r.raw; }
00245
00247     friend constexpr bool operator != (DT l, DT r)
00248     { return l.raw != r.raw; }
00249
00251     DT operator |= (DT r)
00252     {
00253         static_cast<DT *>(this)->raw |= r.raw;
00254         return *static_cast<DT *>(this);
00255     }
00256
00258     DT operator &= (DT r)
00259     {
00260         static_cast<DT *>(this)->raw &= r.raw;
00261         return *static_cast<DT *>(this);
00262     }
00263
00265     explicit constexpr operator bool () const
00266     {
00267         return static_cast<DT const *>(this)->raw != 0;
00268     }
00269
00271     constexpr DT operator ~ () const
00272     { return DT(~static_cast<DT const *>(this)->raw); }
00273 };
00274
00283 template<typename DT, typename T>
00284 struct Flags_t : Flags_ops_t<Flags_t<DT, T>
00285 {
00287     T raw;
00289     Flags_t() = default;
00291     explicit constexpr Flags_t(T f) : raw(f) {}
00292 };
00293
00294
00300 template< bool V > struct Bool
00301 {
00302     typedef Bool<V> type;
00303     enum { value = V };
00304 };
00305
00308 struct False : Bool<false> {};
00309
00312 struct True : Bool<true> {};
00313
00314 /*****/
00323 template<typename A, typename B>
00324 struct Same : False {};
00325
00326 template<typename A>
00327 struct Same<A, A> : True {};
00328
00329 template<bool EXP, typename T = void> struct Enable_if {};
00330 template<typename T> struct Enable_if<true, T> { typedef T type; };

```

```

00331
00332 template<typename T1, typename T2, typename T = void>
00333 struct Enable_if_same : Enable_if<Same<T1, T2>::value, T> {};
00334
00335 template<typename T> struct Remove_const { typedef T type; };
00336 template<typename T> struct Remove_const<T const> { typedef T type; };
00337 template<typename T> struct Remove_volatile { typedef T type; };
00338 template<typename T> struct Remove_volatile<T volatile> { typedef T type; };
00339 template<typename T> struct Remove_cv
00340 { typedef typename Remove_const<typename Remove_volatile<T>::type>::type type; };
00341
00342 template<typename T> struct Remove_pointer { typedef T type; };
00343 template<typename T> struct Remove_pointer<T*> { typedef T type; };
00344 template<typename T> struct Remove_reference { typedef T type; };
00345 template<typename T> struct Remove_reference<T&> { typedef T type; };
00346 template<typename T> struct Remove_pr { typedef T type; };
00347 template<typename T> struct Remove_pr<T&> { typedef T type; };
00348 template<typename T> struct Remove_pr<T*> { typedef T type; };
00349 } // Types
00350 } // L4

```

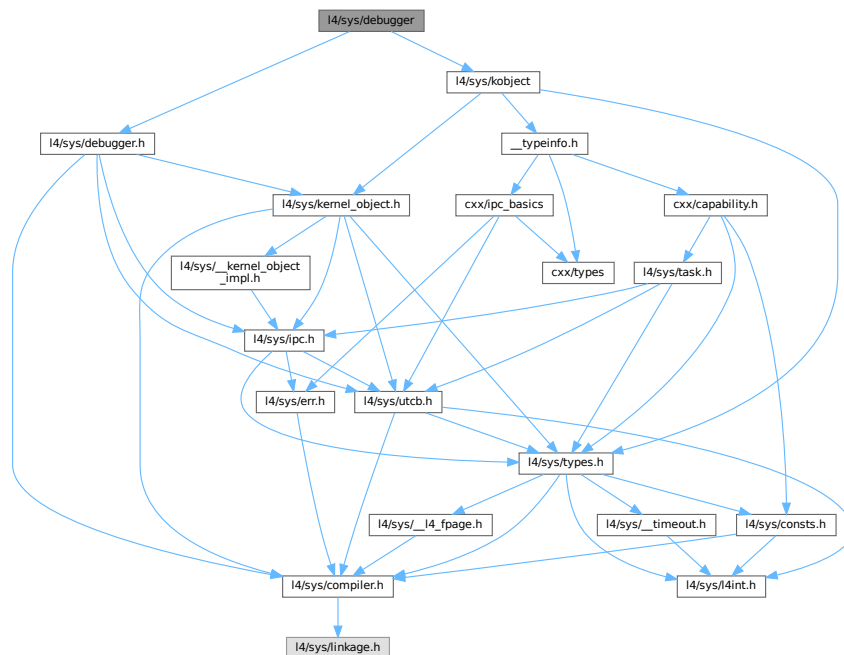
16.464 I4/sys/debugger File Reference

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.

16.464.1 Detailed Description

The debugger interface specifies common debugging related definitions.

Definition in file [debugger](#).

16.465 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 };
00163 }

```

16.466 l4/sys/debugger.h File Reference

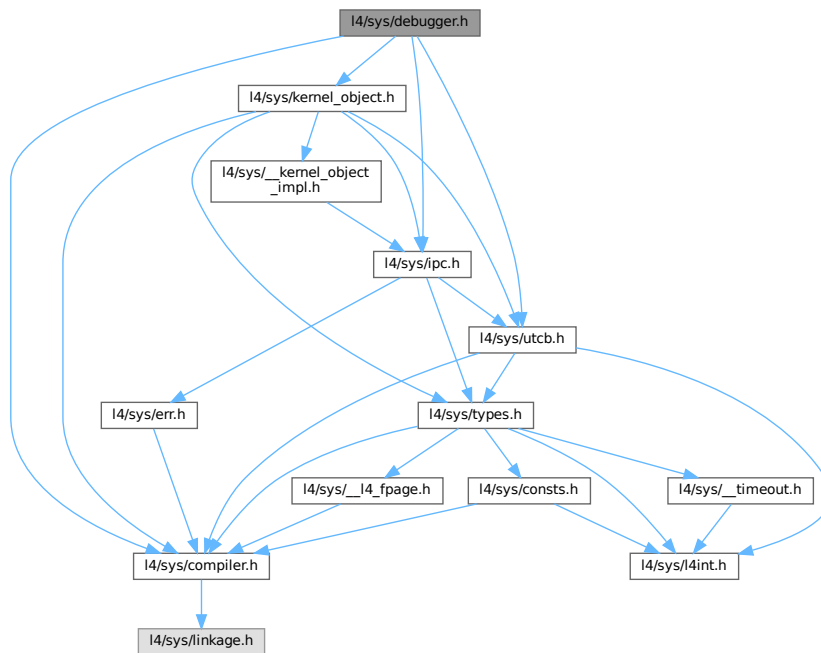
Debugger related definitions.

```

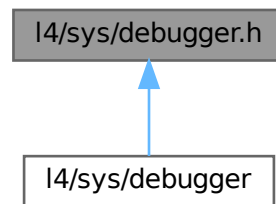
#include <l4/sys/compiler.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
#include <l4/sys/kernel_object.h>

```

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.

16.466.1 Detailed Description

Debugger related definitions.

Definition in file [debugger.h](#).

16.467 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
00201 enum
00202 {
00203     L4_DEBUGGER_NAME_SET_OP          = 0UL,
00204     L4_DEBUGGER_GLOBAL_ID_OP        = 1UL,
00205     L4_DEBUGGER_KOBJ_TO_ID_OP       = 2UL,
00206     L4_DEBUGGER_QUERY_LOG_TYPEID_OP = 3UL,
00207     L4_DEBUGGER_SWITCH_LOG_OP       = 4UL,
00208     L4_DEBUGGER_NAME_GET_OP         = 5UL,
00209     L4_DEBUGGER_QUERY_LOG_NAME_OP   = 6UL,
00210 };
00211
00212 enum
00213 {
00214     L4_DEBUGGER_SWITCH_LOG_ON = 1,
00215     L4_DEBUGGER_SWITCH_LOG_OFF = 0,
00216 };
00217
00218 /* IMPLEMENTATION ----- */

```

```

00219
00220 #include <l4/sys/kernel_object.h>
00221
00235 L4_INLINE unsigned
00236 __strcpy_maxlen(char *dst, char const *src, unsigned maxlen)
00237 {
00238     unsigned i;
00239     if (!maxlen)
00240         return 0;
00241
00242     for (i = 0; i < maxlen - 1 && src[i]; ++i)
00243         dst[i] = src[i];
00244     dst[i] = '\0';
00245
00246     return i + 1;
00247 }
00248
00249 L4_INLINE l4_msgtag_t
00250 l4_debugger_set_object_name_u(l4_cap_idx_t cap,
00251                               const char *name, l4_utcb_t *utcb) L4_NOTHROW
00252 {
00253     unsigned i;
00254     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_NAME_SET_OP;
00255     i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[1], name,
00256                        (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(l4_umword_t));
00257     i = l4_bytes_to_mwords(i);
00258     return l4_invoke_debugger(cap, l4_msgtag(0, 1 + i, 0, 0), utcb);
00259 }
00260
00261 L4_INLINE unsigned long
00262 l4_debugger_global_id_u(l4_cap_idx_t cap, l4_utcb_t *utcb) L4_NOTHROW
00263 {
00264     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_GLOBAL_ID_OP;
00265     if (l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 1, 0, 0), utcb), utcb))
00266         return ~0UL;
00267     return l4_utcb_mr_u(utcb)->mr[0];
00268 }
00269
00270 L4_INLINE unsigned long
00271 l4_debugger_kobj_to_id_u(l4_cap_idx_t cap, l4_addr_t kobjp, l4_utcb_t *utcb) L4_NOTHROW
00272 {
00273     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_KOBJ_TO_ID_OP;
00274     l4_utcb_mr_u(utcb)->mr[1] = kobjp;
00275     if (l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb), utcb))
00276         return ~0UL;
00277     return l4_utcb_mr_u(utcb)->mr[0];
00278 }
00279
00280 L4_INLINE long
00281 l4_debugger_query_log_typeid_u(l4_cap_idx_t cap, const char *name,
00282                                unsigned idx,
00283                                l4_utcb_t *utcb) L4_NOTHROW
00284 {
00285     unsigned i;
00286     long e;
00287     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_QUERY_LOG_TYPEID_OP;
00288     l4_utcb_mr_u(utcb)->mr[1] = idx;
00289     i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[2], name, 32);
00290     i = l4_bytes_to_mwords(i);
00291     e = l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2 + i, 0, 0), utcb), utcb);
00292     if (e < 0)
00293         return e;
00294     return l4_utcb_mr_u(utcb)->mr[0];
00295 }
00296
00297 L4_INLINE long
00298 l4_debugger_query_log_name_u(l4_cap_idx_t cap, unsigned idx,
00299                              char *name, unsigned namelen,
00300                              char *shortname, unsigned shortnamelen,
00301                              l4_utcb_t *utcb) L4_NOTHROW
00302 {
00303     long e;
00304     char const *n;
00305     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_QUERY_LOG_NAME_OP;
00306     l4_utcb_mr_u(utcb)->mr[1] = idx;
00307     e = l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb), utcb);
00308     if (e < 0)
00309         return e;
00310     n = (char const *)&l4_utcb_mr_u(utcb)->mr[0];
00311     __strcpy_maxlen(name, n, namelen);
00312     __strcpy_maxlen(shortname, n + __builtin_strlen(n) + 1, shortnamelen);
00313     return 0;
00314 }
00315
00316
00317 L4_INLINE l4_msgtag_t
00318 l4_debugger_switch_log_u(l4_cap_idx_t cap, const char *name, int on_off,

```

```

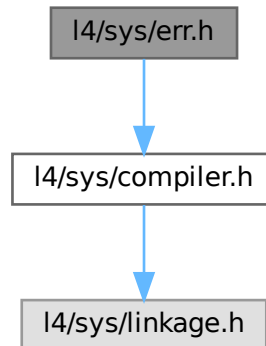
00319             l4_utcb_t *utcb) L4_NOTHROW
00320 {
00321     unsigned i;
00322     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_SWITCH_LOG_OP;
00323     l4_utcb_mr_u(utcb)->mr[1] = on_off;
00324     i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[2], name, 32);
00325     i = l4_bytes_to_mwords(i);
00326     return l4_invoke_debugger(cap, l4_msgtag(0, 2 + i, 0, 0), utcb);
00327 }
00328
00329 L4_INLINE l4_msgtag_t
00330 l4_debugger_get_object_name_u(l4_cap_idx_t cap, unsigned id,
00331                               char *name, unsigned size,
00332                               l4_utcb_t *utcb) L4_NOTHROW
00333 {
00334     l4_msgtag_t t;
00335     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_NAME_GET_OP;
00336     l4_utcb_mr_u(utcb)->mr[1] = id;
00337     t = l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb);
00338     __strcpy_maxlen(name, (char const *)&l4_utcb_mr_u(utcb)->mr[0], size);
00339     return t;
00340 }
00341
00342
00343 L4_INLINE l4_msgtag_t
00344 l4_debugger_set_object_name(l4_cap_idx_t cap,
00345                             const char *name) L4_NOTHROW
00346 {
00347     return l4_debugger_set_object_name_u(cap, name, l4_utcb());
00348 }
00349
00350 L4_INLINE unsigned long
00351 l4_debugger_global_id(l4_cap_idx_t cap) L4_NOTHROW
00352 {
00353     return l4_debugger_global_id_u(cap, l4_utcb());
00354 }
00355
00356 L4_INLINE unsigned long
00357 l4_debugger_kobj_to_id(l4_cap_idx_t cap, l4_addr_t kobjp) L4_NOTHROW
00358 {
00359     return l4_debugger_kobj_to_id_u(cap, kobjp, l4_utcb());
00360 }
00361
00362 L4_INLINE long
00363 l4_debugger_query_log_typeid(l4_cap_idx_t cap, const char *name,
00364                             unsigned idx) L4_NOTHROW
00365 {
00366     return l4_debugger_query_log_typeid_u(cap, name, idx, l4_utcb());
00367 }
00368
00369 L4_INLINE long
00370 l4_debugger_query_log_name(l4_cap_idx_t cap, unsigned idx,
00371                            char *name, unsigned namelen,
00372                            char *shortname, unsigned shortnamelen) L4_NOTHROW
00373 {
00374     return l4_debugger_query_log_name_u(cap, idx, name, namelen,
00375                                         shortname, shortnamelen, l4_utcb());
00376 }
00377
00378 L4_INLINE l4_msgtag_t
00379 l4_debugger_switch_log(l4_cap_idx_t cap, const char *name,
00380                       int on_off) L4_NOTHROW
00381 {
00382     return l4_debugger_switch_log_u(cap, name, on_off, l4_utcb());
00383 }
00384
00385 L4_INLINE l4_msgtag_t
00386 l4_debugger_get_object_name(l4_cap_idx_t cap, unsigned id,
00387                             char *name, unsigned size) L4_NOTHROW
00388 {
00389     return l4_debugger_get_object_name_u(cap, id, name, size, l4_utcb());
00390 }

```

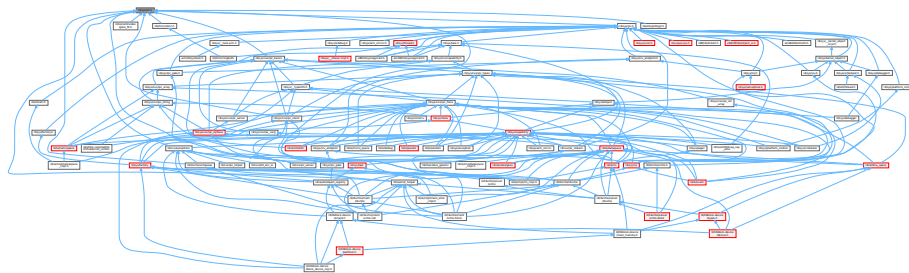
16.468 l4/sys/err.h File Reference

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_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.

16.468.1 Detailed Description

Error codes.

Definition in file [err.h](#).

16.469 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_EINVAL            = 22,
00057     L4_ENOSPC             = 28,
00058     L4_ERANGE             = 34,
00059     L4_ENAMETOOLONG       = 36,
00060     L4_ENOSYS             = 38,
00061     L4_EBADPROTO          = 39,
00062     L4_EADDRNOTAVAIL      = 99,
00063     L4_ERRNOMAX           = 100,
00065     L4_ENOREPLY           = 1000,
00066     L4_EMSGTOOSHORT       = 1001,
00067     L4_EMSGTOOLONG        = 1002,
00068     L4_EMMSGMISSARG       = 1003,
00070     L4_EIPC_LO            = 2000,
00071     L4_EIPC_HI            = 2000 + 0x1f,
00072 };
00073
00074 __BEGIN_DECLS
00075 L4_CV char const *l4sys_errtostr(long err) L4_NOTHROW;
00076 __END_DECLS
00077
00078

```

16.470 l4/sys/exception File Reference

Exception C++ interface.

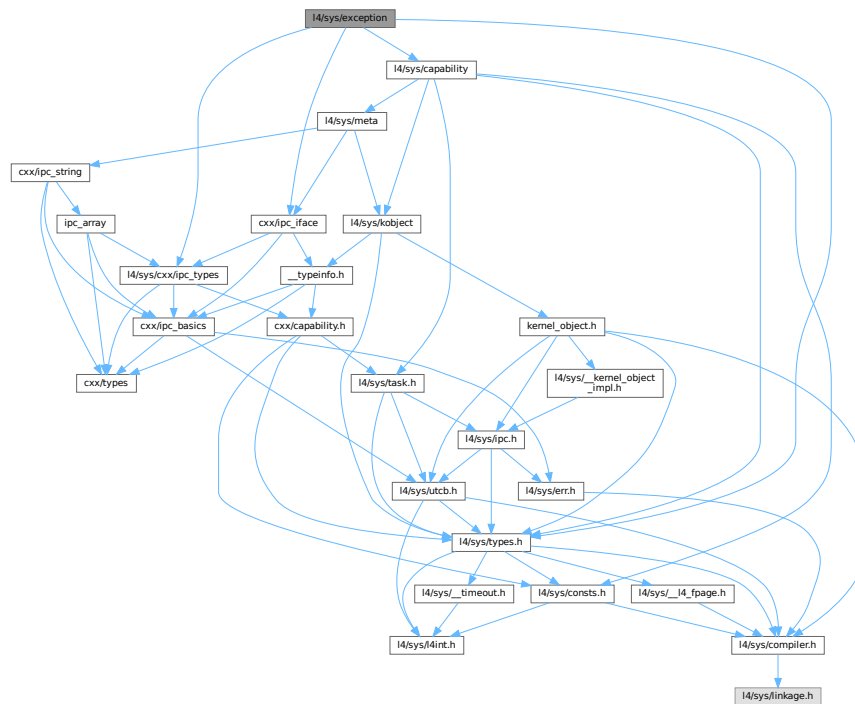
```

#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 exception:



Data Structures

- class [L4::Exception](#)
Exception interface.

Namespaces

- namespace [L4](#)
L4 low-level kernel interface.

16.470.1 Detailed Description

Exception C++ interface.

Definition in file [exception](#).

16.471 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 }
```

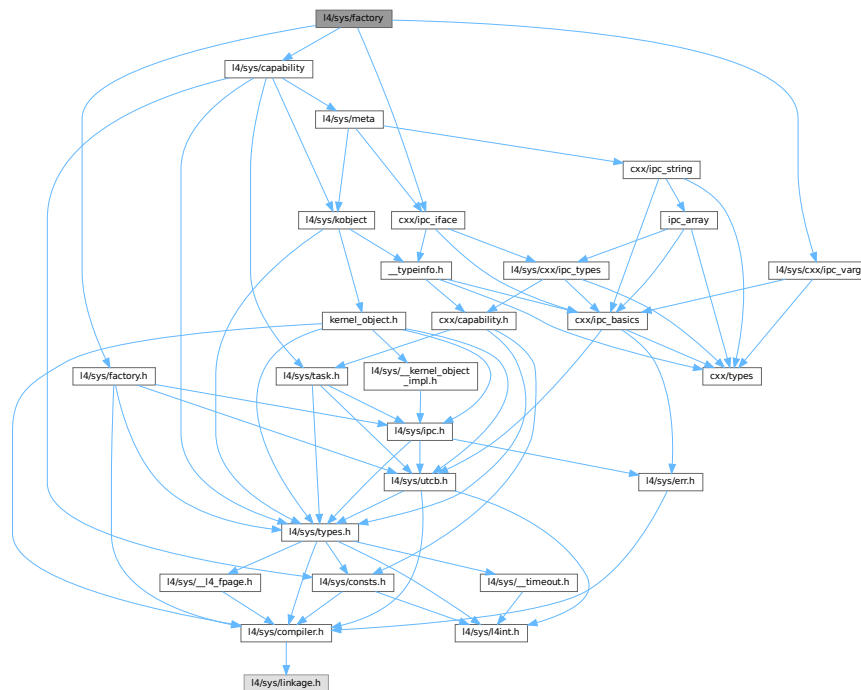
16.472 l4/sys/factory File Reference

Common factory related definitions.

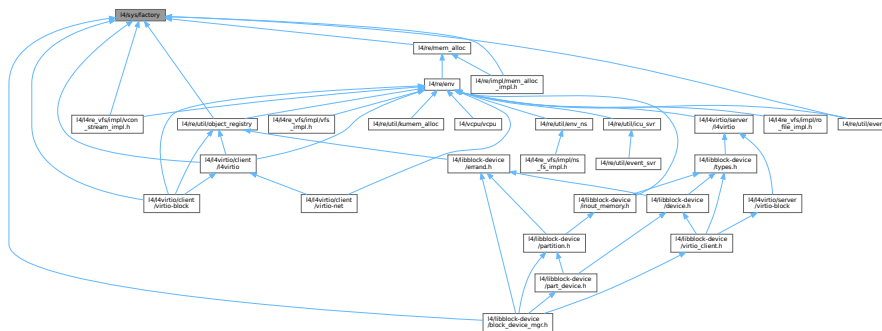
```

#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.

16.472.1 Detailed Description

Common factory related definitions.

Definition in file [factory](#).

16.473 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(Nil) & 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
00344 l4_msgtag_t create_task(Cap<Task> const & target_cap,
00345                        l4_fpage_t const & utcb_area,
00346                        l4_utcb_t *utcb = l4_utcb()) noexcept
00347 { return l4_factory_create_task_u(cap(), target_cap.cap(), utcb_area, utcb); }
00348
00377 l4_msgtag_t create_factory(Cap<Factory> const & target_cap,
00378                           unsigned long limit,
00379                           l4_utcb_t *utcb = l4_utcb()) noexcept
00380 { return l4_factory_create_factory_u(cap(), target_cap.cap(), limit, utcb); }
00381
00410 l4_msgtag_t create_gate(Cap<void> const & target_cap,
00411                        Cap<Thread> const & thread_cap, l4_umword_t label,
00412                        l4_utcb_t *utcb = l4_utcb()) noexcept
00413 { return l4_factory_create_gate_u(cap(), target_cap.cap(), thread_cap.cap(), label, utcb); }
00414
00415 typedef L4::Typeid::Rpc_nocode<create_t> Rpcs;
00416 };
00417
00418 }

```

16.474 l4/sys/factory.h File Reference

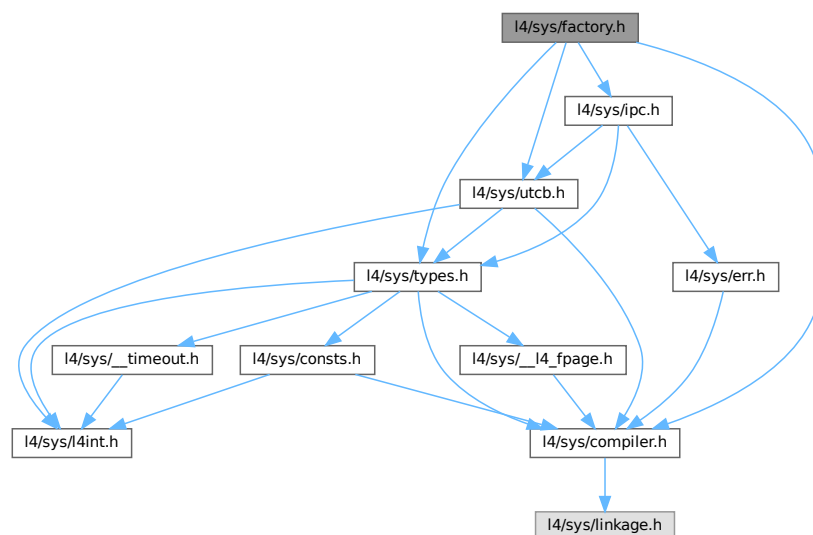
Common factory related definitions.

```

#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:




```

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
00091 L4_INLINE l4_msgtag_t
00092 l4_factory_create_task(l4_cap_idx_t factory,
00093                      l4_cap_idx_t target_cap, l4_fpage_t utcb_area) L4_NOTHROW;
00094
00099 L4_INLINE l4_msgtag_t
00100 l4_factory_create_task_u(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
00101                       l4_fpage_t utcb_area, l4_utcb_t *utcb) L4_NOTHROW;
00102
00120 L4_INLINE l4_msgtag_t
00121 l4_factory_create_thread(l4_cap_idx_t factory,
00122                        l4_cap_idx_t target_cap) L4_NOTHROW;
00123
00128 L4_INLINE l4_msgtag_t
00129 l4_factory_create_thread_u(l4_cap_idx_t factory,
00130                          l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW;
00131
00156 L4_INLINE l4_msgtag_t
00157 l4_factory_create_factory(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
00158                        unsigned long limit) L4_NOTHROW;
00159
00164 L4_INLINE l4_msgtag_t
00165 l4_factory_create_factory_u(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
00166                          unsigned long limit, l4_utcb_t *utcb) L4_NOTHROW;
00167
00193 L4_INLINE l4_msgtag_t
00194 l4_factory_create_gate(l4_cap_idx_t factory,
00195                      l4_cap_idx_t target_cap,
00196                      l4_cap_idx_t thread_cap, l4_umword_t label) L4_NOTHROW;
00197
00202 L4_INLINE l4_msgtag_t
00203 l4_factory_create_gate_u(l4_cap_idx_t factory,
00204                        l4_cap_idx_t target_cap,
00205                        l4_cap_idx_t thread_cap, l4_umword_t label,
00206                        l4_utcb_t *utcb) L4_NOTHROW;
00207
00223 L4_INLINE l4_msgtag_t
00224 l4_factory_create_irq(l4_cap_idx_t factory,
00225                     l4_cap_idx_t target_cap) L4_NOTHROW;
00226
00231 L4_INLINE l4_msgtag_t
00232 l4_factory_create_irq_u(l4_cap_idx_t factory,
00233                      l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW;
00234
00252 L4_INLINE l4_msgtag_t
00253 l4_factory_create_vm(l4_cap_idx_t factory,
00254                   l4_cap_idx_t target_cap) L4_NOTHROW;
00255
00260 L4_INLINE l4_msgtag_t
00261 l4_factory_create_vm_u(l4_cap_idx_t factory,
00262                      l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW;
00263
00268 L4_INLINE l4_msgtag_t
00269 l4_factory_create_start_u(long obj, l4_cap_idx_t target,
00270                        l4_utcb_t *utcb) L4_NOTHROW;
00271
00276 L4_INLINE int
00277 l4_factory_create_add_fpage_u(l4_fpage_t d, l4_msgtag_t *tag,
00278                             l4_utcb_t *utcb) L4_NOTHROW;
00279
00284 L4_INLINE int
00285 l4_factory_create_add_int_u(l4_mword_t d, l4_msgtag_t *tag,
00286                          l4_utcb_t *utcb) L4_NOTHROW;
00287
00292 L4_INLINE int
00293 l4_factory_create_add_uint_u(l4_umword_t d, l4_msgtag_t *tag,
00294                          l4_utcb_t *utcb) L4_NOTHROW;

```

```

00295
00300 L4_INLINE int
00301 l4_factory_create_add_str_u(char const *s, l4_msgtag_t *tag,
00302                             l4_utcb_t *utcb) L4_NOTHROW;
00303
00308 L4_INLINE int
00309 l4_factory_create_add_lstr_u(char const *s, unsigned len, l4_msgtag_t *tag,
00310                             l4_utcb_t *utcb) L4_NOTHROW;
00311
00316 L4_INLINE int
00317 l4_factory_create_add_nil_u(l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW;
00318
00323 L4_INLINE l4_msgtag_t
00324 l4_factory_create_commit_u(l4_cap_idx_t factory, l4_msgtag_t tag,
00325                             l4_utcb_t *utcb) L4_NOTHROW;
00326
00331 L4_INLINE l4_msgtag_t
00332 l4_factory_create_u(l4_cap_idx_t factory, long obj, l4_cap_idx_t target,
00333                    l4_utcb_t *utcb) L4_NOTHROW;
00334
00335
00352 L4_INLINE l4_msgtag_t
00353 l4_factory_create(l4_cap_idx_t factory, long obj,
00354                  l4_cap_idx_t target) L4_NOTHROW;
00355
00356 /* IMPLEMENTATION -----*/
00357
00358 #include <l4/sys/ipc.h>
00359
00360 L4_INLINE l4_msgtag_t
00361 l4_factory_create_task_u(l4_cap_idx_t factory,
00362                          l4_cap_idx_t target_cap, l4_fpage_t utcb_area,
00363                          l4_utcb_t *u) L4_NOTHROW
00364 {
00365     l4_msgtag_t t;
00366     t = l4_factory_create_start_u(L4_PROTO_TASK, target_cap, u);
00367     l4_factory_create_add_fpage_u(utcb_area, &t, u);
00368     return l4_factory_create_commit_u(factory, t, u);
00369 }
00370
00371 L4_INLINE l4_msgtag_t
00372 l4_factory_create_thread_u(l4_cap_idx_t factory,
00373                            l4_cap_idx_t target_cap, l4_utcb_t *u) L4_NOTHROW
00374 {
00375     return l4_factory_create_u(factory, L4_PROTO_THREAD, target_cap, u);
00376 }
00377
00378 L4_INLINE l4_msgtag_t
00379 l4_factory_create_factory_u(l4_cap_idx_t factory,
00380                             l4_cap_idx_t target_cap, unsigned long limit,
00381                             l4_utcb_t *u) L4_NOTHROW
00382 {
00383     l4_msgtag_t t;
00384     t = l4_factory_create_start_u(L4_PROTO_FACTORY, target_cap, u);
00385     l4_factory_create_add_uint_u(limit, &t, u);
00386     return l4_factory_create_commit_u(factory, t, u);
00387 }
00388
00389 L4_INLINE l4_msgtag_t
00390 l4_factory_create_gate_u(l4_cap_idx_t factory,
00391                          l4_cap_idx_t target_cap,
00392                          l4_cap_idx_t thread_cap, l4_umword_t label,
00393                          l4_utcb_t *u) L4_NOTHROW
00394 {
00395     l4_msgtag_t t;
00396     l4_msg_regs_t *v;
00397     int items = 0;
00398     t = l4_factory_create_start_u(0, target_cap, u);
00399     l4_factory_create_add_uint_u(label, &t, u);
00400     v = l4_utcb_mr_u(u);
00401     if (!(thread_cap & L4_INVALID_CAP_BIT))
00402     {
00403         items = 1;
00404         v->mr[3] = l4_map_obj_control(0,0);
00405         v->mr[4] = l4_obj_fpage(thread_cap, 0, L4_CAP_FPAGE_RWS).raw;
00406     }
00407     t = l4_msgtag(l4_msgtag_label(t), l4_msgtag_words(t), items, l4_msgtag_flags(t));
00408     return l4_factory_create_commit_u(factory, t, u);
00409 }
00410
00411 L4_INLINE l4_msgtag_t
00412 l4_factory_create_irq_u(l4_cap_idx_t factory,
00413                        l4_cap_idx_t target_cap, l4_utcb_t *u) L4_NOTHROW
00414 {
00415     return l4_factory_create_u(factory, L4_PROTO_IRQ_SENDER, target_cap, u);
00416 }
00417

```



```

00418 L4_INLINE l4_msgtag_t
00419 l4_factory_create_vm_u(l4_cap_idx_t factory,
00420                        l4_cap_idx_t target_cap,
00421                        l4_utcb_t *u) L4_NOTHROW
00422 {
00423     return l4_factory_create_u(factory, L4_PROTO_VM, target_cap, u);
00424 }
00425
00426
00427
00428
00429
00430 L4_INLINE l4_msgtag_t
00431 l4_factory_create_task(l4_cap_idx_t factory,
00432                       l4_cap_idx_t target_cap, l4_fpage_t utcb_area) L4_NOTHROW
00433 {
00434     return l4_factory_create_task_u(factory, target_cap, utcb_area, l4_utcb());
00435 }
00436
00437 L4_INLINE l4_msgtag_t
00438 l4_factory_create_thread(l4_cap_idx_t factory,
00439                         l4_cap_idx_t target_cap) L4_NOTHROW
00440 {
00441     return l4_factory_create_thread_u(factory, target_cap, l4_utcb());
00442 }
00443
00444 L4_INLINE l4_msgtag_t
00445 l4_factory_create_factory(l4_cap_idx_t factory,
00446                          l4_cap_idx_t target_cap, unsigned long limit) L4_NOTHROW
00447 {
00448     return l4_factory_create_factory_u(factory, target_cap, limit, l4_utcb());
00449 }
00450
00451
00452 L4_INLINE l4_msgtag_t
00453 l4_factory_create_gate(l4_cap_idx_t factory,
00454                      l4_cap_idx_t target_cap,
00455                      l4_cap_idx_t thread_cap, l4_umword_t label) L4_NOTHROW
00456 {
00457     return l4_factory_create_gate_u(factory, target_cap, thread_cap, label, l4_utcb());
00458 }
00459
00460 L4_INLINE l4_msgtag_t
00461 l4_factory_create_irq(l4_cap_idx_t factory,
00462                     l4_cap_idx_t target_cap) L4_NOTHROW
00463 {
00464     return l4_factory_create_irq_u(factory, target_cap, l4_utcb());
00465 }
00466
00467 L4_INLINE l4_msgtag_t
00468 l4_factory_create_vm(l4_cap_idx_t factory,
00469                    l4_cap_idx_t target_cap) L4_NOTHROW
00470 {
00471     return l4_factory_create_vm_u(factory, target_cap, l4_utcb());
00472 }
00473
00474 L4_INLINE l4_msgtag_t
00475 l4_factory_create_start_u(long obj, l4_cap_idx_t target_cap,
00476                          l4_utcb_t *u) L4_NOTHROW
00477 {
00478     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00479     l4_buf_regs_t *b = l4_utcb_br_u(u);
00480     v->mr[0] = obj;
00481     b->bdr = 0;
00482     b->br[0] = target_cap | L4_RCV_ITEM_SINGLE_CAP;
00483     return l4_msgtag(L4_PROTO_FACTORY, 1, 0, 0);
00484 }
00485
00486 L4_INLINE int
00487 l4_factory_create_add_fpage_u(l4_fpage_t d, l4_msgtag_t *tag,
00488                             l4_utcb_t *u) L4_NOTHROW
00489 {
00490     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00491     int w = l4_msgtag_words(*tag);
00492     if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00493         return 0;
00494     v->mr[w] = L4_VARG_TYPE_FPAGE | (sizeof(l4_fpage_t) << 16);
00495     v->mr[w + 1] = d.raw;
00496     w += 2;
00497     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00498     return 1;
00499 }
00500
00501 L4_INLINE int
00502 l4_factory_create_add_int_u(l4_mword_t d, l4_msgtag_t *tag,
00503                          l4_utcb_t *u) L4_NOTHROW
00504 {

```

```

00505     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00506     int w = l4_msgtag_words(*tag);
00507     if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00508         return 0;
00509     v->mr[w] = L4_VARG_TYPE_MWORD | (sizeof(l4_mword_t) << 16);
00510     v->mr[w + 1] = d;
00511     w += 2;
00512     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00513     return 1;
00514 }
00515
00516 L4_INLINE int
00517 l4_factory_create_add_uint_u(l4_umword_t d, l4_msgtag_t *tag,
00518                             l4_utcb_t *u) L4_NOTHROW
00519 {
00520     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00521     int w = l4_msgtag_words(*tag);
00522     if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00523         return 0;
00524     v->mr[w] = L4_VARG_TYPE_UMWORD | (sizeof(l4_umword_t) << 16);
00525     v->mr[w + 1] = d;
00526     w += 2;
00527     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00528     return 1;
00529 }
00530
00531 L4_INLINE int
00532 l4_factory_create_add_str_u(char const *s, l4_msgtag_t *tag,
00533                             l4_utcb_t *u) L4_NOTHROW
00534 {
00535     return l4_factory_create_add_lstr_u(s, __builtin_strlen(s) + 1, tag, u);
00536 }
00537
00538 L4_INLINE int
00539 l4_factory_create_add_lstr_u(char const *s, unsigned len, l4_msgtag_t *tag,
00540                             l4_utcb_t *u) L4_NOTHROW
00541 {
00542     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00543     unsigned w = l4_msgtag_words(*tag);
00544     char *c;
00545     unsigned i;
00546     if (w + 1 + l4_bytes_to_mwords(len) > L4_UTCB_GENERIC_DATA_SIZE)
00547         return 0;
00548     v->mr[w] = L4_VARG_TYPE_STRING | (len << 16);
00549     c = (char*)&v->mr[w + 1];
00550     for (i = 0; i < len; ++i)
00551         *c++ = *s++;
00552     w = w + 1 + l4_bytes_to_mwords(len);
00553     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00554     return 1;
00555 }
00556
00557 L4_INLINE int
00558 l4_factory_create_add_nil_u(l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW
00559 {
00560     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00561     int w = l4_msgtag_words(*tag);
00562     v->mr[w] = L4_VARG_TYPE_NIL;
00563     ++w;
00564     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00565     return 1;
00566 }
00567
00568 L4_INLINE l4_msgtag_t
00569 l4_factory_create_commit_u(l4_cap_idx_t factory, l4_msgtag_t tag,
00570                             l4_utcb_t *u) L4_NOTHROW
00571 {
00572     return l4_ipc_call(factory, u, tag, L4_IPC_NEVER);
00573 }
00574
00575 L4_INLINE l4_msgtag_t
00576 l4_factory_create_u(l4_cap_idx_t factory, long obj, l4_cap_idx_t target,
00577                     l4_utcb_t *utcb) L4_NOTHROW
00578 {
00579     l4_msgtag_t t = l4_factory_create_start_u(obj, target, utcb);
00580     return l4_factory_create_commit_u(factory, t, utcb);
00581 }
00582
00583 L4_INLINE l4_msgtag_t
00584 l4_factory_create(l4_cap_idx_t factory, long obj,

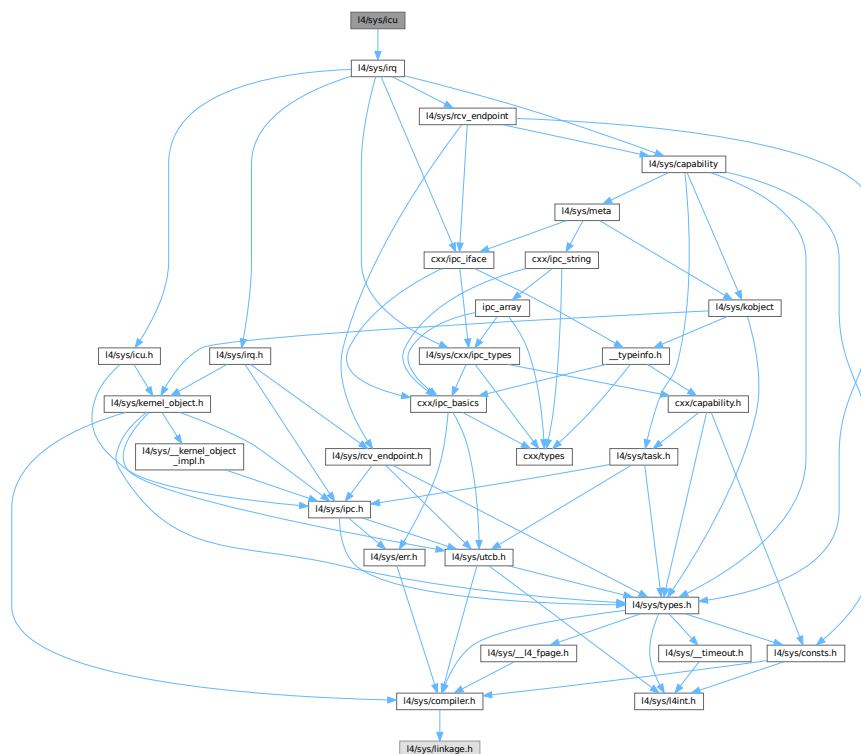
```

```
00592         14_cap_idx_t target) L4_NOTHROW
00593 {
00594     return 14_factory_create_u(factory, obj, target, 14_utcb());
00595 }
```

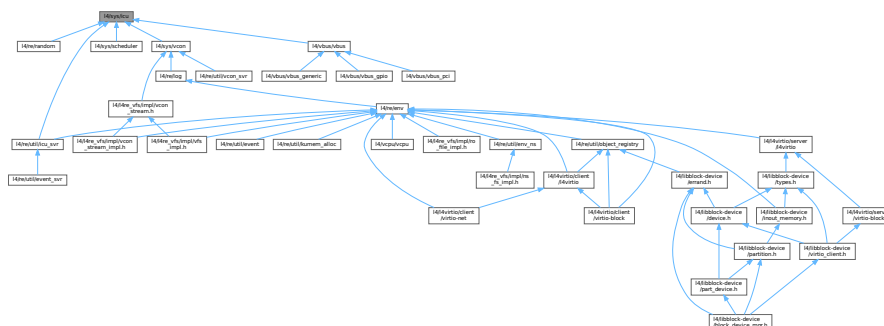
16.476 I4/sys/icu File Reference

Interrupt controller.

```
#include <linux/sys/irq>
```



This graph shows which files directly or indirectly include this file:



16.476.1 Detailed Description

Interrupt controller.

Definition in file [icu](#).

16.477 icu

[Go to the documentation of this file.](#)

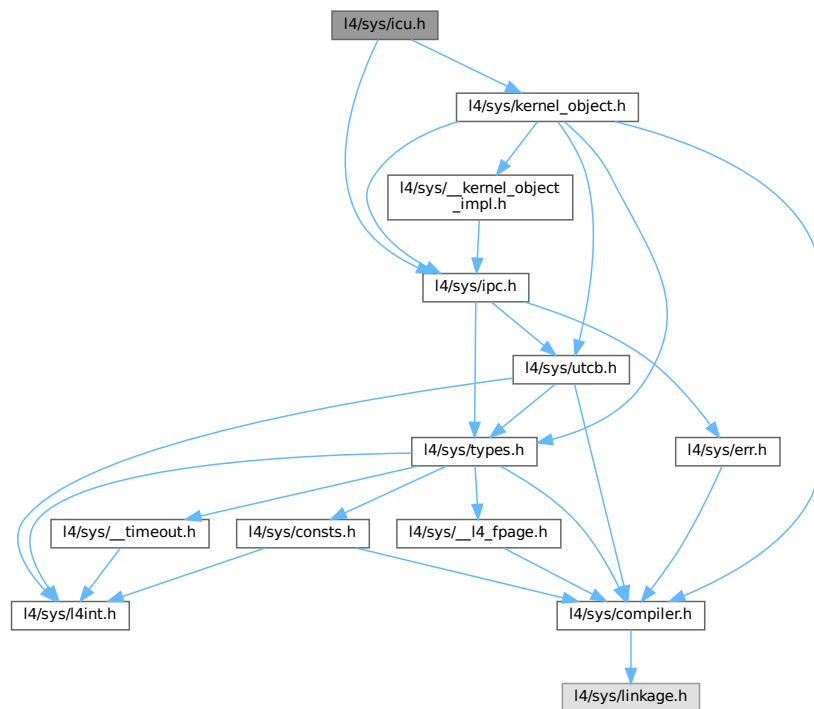
```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00007 /*
00008  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public 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>
```

16.478 l4/sys/icu.h File Reference

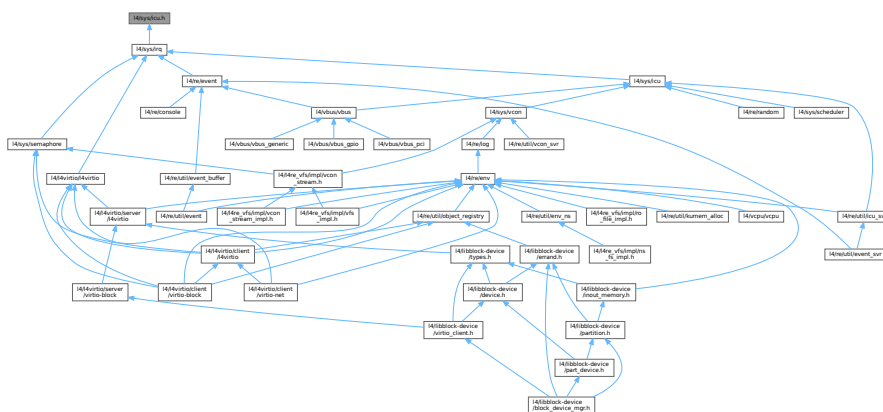
Interrupt controller.

```
#include <l4/sys/kernel_object.h>
#include <l4/sys/ipc.h>
```

Include dependency graph for icu.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `l4_icu_info_t`
Info structure for an ICU.
- struct `l4_icu_msi_info_t`
Info to use for a specific MSI.

Typedefs

- typedef struct [l4_icu_info_t](#) [l4_icu_info_t](#)
Info structure for an ICU.
- typedef struct [l4_icu_msi_info_t](#) [l4_icu_msi_info_t](#)
Info to use for a specific MSI.

Enumerations

- enum [L4_icu_flags](#) { [L4_ICU_FLAG_MSI](#) }
Flags for IRQ numbers used for the ICU.
- 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.
- 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.

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.

16.478.1 Detailed Description

Interrupt controller.

Definition in file `icu.h`.

16.479 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 *m = l4_utcb_mr_u(utcb);
00469     m->mr[0] = L4_ICU_OP_SET_MODE;
00470     m->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 }

```

16.480 iommu

```

00001 // vi:set ft=cpp: -*- 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 }

```

16.481 ipc.h

```

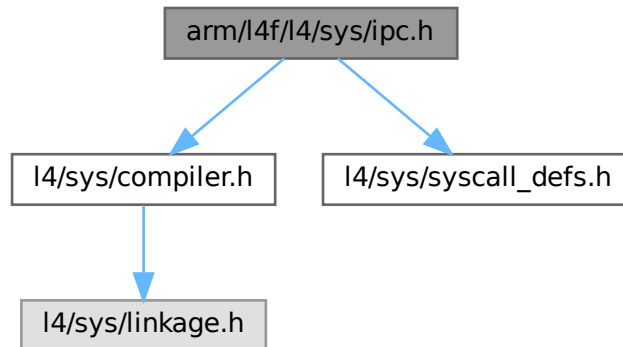
00001
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00005  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00006  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00007  *      economic rights: Technische 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 #ifndef __L4SYS__INCLUDE__ARCH_AMD64__L4API_L4F__IPC_H__
00023 #define __L4SYS__INCLUDE__ARCH_AMD64__L4API_L4F__IPC_H__
00024
00025 #include_next <l4/sys/ipc.h>
00026
00027 L4_INLINE l4_msgtag_t
00028 l4_ipc(l4_cap_idx_t dest, l4_utcb_t *utcb,
00029        l4_umword_t flags,
00030        l4_umword_t slabel,
00031        l4_msgtag_t tag,
00032        l4_umword_t *rlabel,
00033        l4_timeout_t timeout) L4_NOTHROW
00034 {
00035     l4_umword_t dummy, dummy2;
00036     register l4_umword_t to __asm__("r8") = timeout.raw;
00037
00038     (void)utcb;
00039
00040     __asm__ __volatile__
00041     ("syscall"
00042      :
00043      "=d" (dummy2),
00044      "=S" (slabel),
00045      "=D" (dummy),
00046      "=a" (tag.raw)
00047      :
00048      "S" (slabel),
00049      "r" (to),
00050      "a" (tag.raw),
00051      "d" (dest | flags)
00052      :
00053      "memory", "cc", "rcx", "r11", "r15"
00054      );
00055
00056     if (rlabel)
00057         *rlabel = slabel;
00058
00059     return tag;
00060 }
00061
00062 #endif /* ! __L4SYS__INCLUDE__ARCH_AMD64__L4API_L4F__IPC_H__ */

```

16.482 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.

16.482.1 Detailed Description

[L4 IPC System Calls](#), ARM.

Definition in file [ipc.h](#).

16.483 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 lr, pc\n"
00049      "mvn pc, %[sc]\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", "lr");
00059
00060     if (rlabel)
00061         *rlabel = _label;
00062     tag.raw = _tag;
00063
00064     return tag;
00065 }
00066
00067 #endif // __GNUC__

```

16.484 l4/sys/ipc.h File Reference

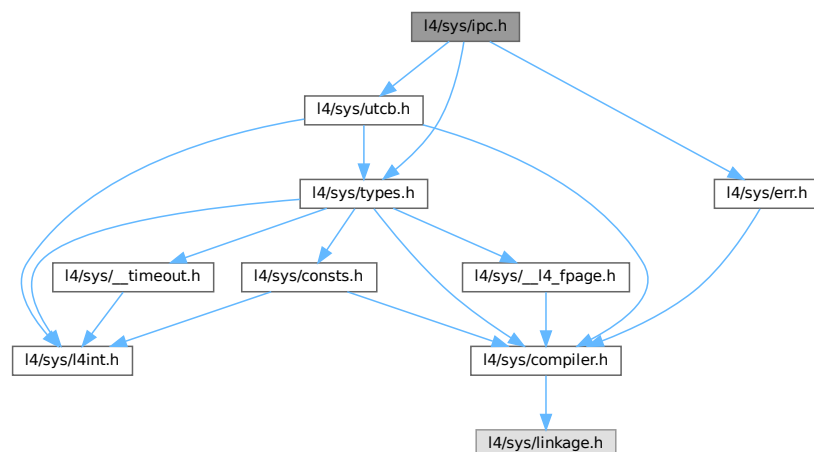
Common IPC interface.

```

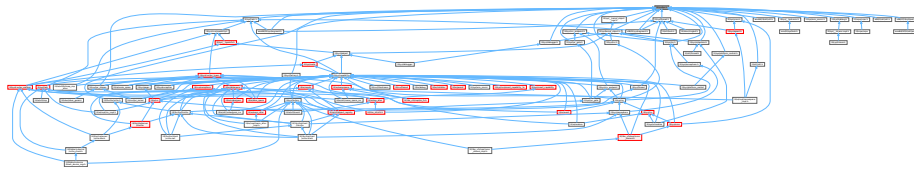
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/err.h>

```

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](#) (unsigned ms) [L4_NOTHROW](#)

Sleep for a certain amount of milliseconds.

- [l4_msgtag_t](#) [l4_ipc_sleep_us](#) (unsigned 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.

16.484.1 Detailed Description

Common IPC interface.

Definition in file [ipc.h](#).

16.484.2 Function Documentation

16.484.2.1 [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

ipc_error_code	IPC error code as delivered by the kernel. (or returned by the l4_ipc_error_code() function).
--------------------------------	---

Returns

negative error code in the range of [L4_EIPC_LO](#) to [L4_EIPC_HI](#).

Definition at line 546 of file [ipc.h](#).

References [L4_EIPC_LO](#).

16.485 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
00056 /*****
00057 *** IPC result checking
00058 *****/
00059
00075 enum l4_ipc_tcr_error_t
00076 {
00077     L4_IPC_ERROR_MASK           = 0x1F,
00078     L4_IPC_SND_ERR_MASK        = 0x01,
00080     L4_IPC_ENOT_EXISTENT       = 0x04,
00083     L4_IPC_RETIMEOUT           = 0x03,
00086     L4_IPC_SETIMEOUT           = 0x02,
00089     L4_IPC_RECANCELED          = 0x07,
00092     L4_IPC_SECANCELED          = 0x06,
00095     L4_IPC_REMAPFAILED         = 0x11,
00099     L4_IPC_SEMAPFAILED         = 0x10,
00102     L4_IPC_RESNDPFTO           = 0x0b,
00106     L4_IPC_SESNDPFTO           = 0x0a,
00110     L4_IPC_RERCVPFTO           = 0x0d,
00114     L4_IPC_SERCVPFTO           = 0x0c,
00118     L4_IPC_REABORTED           = 0x0f,
00121     L4_IPC_SEABORTED           = 0x0e,
00130     L4_IPC_REMSGCUT            = 0x09,
00131
00137     L4_IPC_SEMSGCUT            = 0x08,
00138 };
00139
00140
00151 L4_INLINE l4_umword_t
00152 l4_ipc_error(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;
00153
00154
00171 L4_INLINE long
00172 l4_error(l4_msgtag_t tag) L4_NOTHROW;
00173
00174 L4_INLINE long
00175 l4_error_u(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;
00176
00177 /*****
00178 *** IPC results
00179 *****/
00180
00190 L4_INLINE int l4_ipc_is_snd_error(l4_utcb_t *utcb) L4_NOTHROW;
00191
00201 L4_INLINE int l4_ipc_is_rcv_error(l4_utcb_t *utcb) L4_NOTHROW;
00202
00212 L4_INLINE int l4_ipc_error_code(l4_utcb_t *utcb) L4_NOTHROW;
00213
00220 L4_INLINE long l4_ipc_to_errno(unsigned long ipc_error_code) L4_NOTHROW;
00221
00222
00223 /*****
00224 *** IPC calls
00225 *****/
00226
00255 L4_INLINE l4_msgtag_t
00256 l4_ipc_send(l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag,
00257             l4_timeout_t timeout) L4_NOTHROW;
00258
00259
00286 L4_INLINE l4_msgtag_t
00287 l4_ipc_wait(l4_utcb_t *utcb, l4_umword_t *label,
00288             l4_timeout_t timeout) L4_NOTHROW;
00289
00290

```



```

00315 L4_INLINE l4_msgtag_t
00316 l4_ipc_receive(l4_cap_idx_t object, l4_utcb_t *utcb,
00317               l4_timeout_t timeout) L4_NOTHROW;
00318
00340 L4_INLINE l4_msgtag_t
00341 l4_ipc_call(l4_cap_idx_t object, l4_utcb_t *utcb, l4_msgtag_t tag,
00342            l4_timeout_t timeout) L4_NOTHROW;
00343
00344
00370 L4_INLINE l4_msgtag_t
00371 l4_ipc_reply_and_wait(l4_utcb_t *utcb, l4_msgtag_t tag,
00372                      l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW;
00373
00402 L4_INLINE l4_msgtag_t
00403 l4_ipc_send_and_wait(l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag,
00404                     l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW;
00405
00412 #if 0
00423 L4_INLINE l4_msgtag_t
00424 l4_ipc_wait_next_period(l4_utcb_t *utcb,
00425                         l4_umword_t *label,
00426                         l4_timeout_t timeout);
00427
00428 #endif
00429
00449 L4_ALWAYS_INLINE l4_msgtag_t
00450 l4_ipc(l4_cap_idx_t dest,
00451        l4_utcb_t *utcb,
00452        l4_umword_t flags,
00453        l4_umword_t slabel,
00454        l4_msgtag_t tag,
00455        l4_umword_t *rlabel,
00456        l4_timeout_t timeout) L4_NOTHROW;
00457
00474 L4_INLINE l4_msgtag_t
00475 l4_ipc_sleep(l4_timeout_t timeout) L4_NOTHROW;
00476
00493 L4_INLINE l4_msgtag_t
00494 l4_ipc_sleep_ms(unsigned ms) L4_NOTHROW;
00495
00512 L4_INLINE l4_msgtag_t
00513 l4_ipc_sleep_us(unsigned us) L4_NOTHROW;
00514
00529 L4_INLINE int
00530 l4_sndfpage_add(l4_fpage_t const snd_fpage, unsigned long snd_base,
00531                l4_msgtag_t *tag) L4_NOTHROW;
00532
00533 /*
00534  * \internal
00535  * \ingroup l4_ipc_api
00536  */
00537 L4_INLINE int
00538 l4_sndfpage_add_u(l4_fpage_t const snd_fpage, unsigned long snd_base,
00539                  l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW;
00540
00541
00542 /*****
00543  * Implementations
00544  *****/
00545
00546 L4_INLINE long l4_ipc_to_errno(unsigned long ipc_error_code) L4_NOTHROW
00547 { return -(L4_EIPC_LO + ipc_error_code); }
00548
00549 L4_INLINE l4_msgtag_t
00550 l4_ipc_call(l4_cap_idx_t dest, l4_utcb_t *utcb,
00551            l4_msgtag_t tag,
00552            l4_timeout_t timeout) L4_NOTHROW
00553 {
00554     return l4_ipc(dest, utcb, L4_SYSF_CALL, 0, tag, 0, timeout);
00555 }
00556
00557 L4_INLINE l4_msgtag_t
00558 l4_ipc_reply_and_wait(l4_utcb_t *utcb, l4_msgtag_t tag,
00559                      l4_umword_t *label,
00560                      l4_timeout_t timeout) L4_NOTHROW
00561 {
00562     return l4_ipc(L4_INVALID_CAP, utcb, L4_SYSF_REPLY_AND_WAIT, 0, tag, label, timeout);
00563 }
00564
00565 L4_INLINE l4_msgtag_t
00566 l4_ipc_send_and_wait(l4_cap_idx_t dest, l4_utcb_t *utcb,
00567                     l4_msgtag_t tag,
00568                     l4_umword_t *src,
00569                     l4_timeout_t timeout) L4_NOTHROW
00570 {
00571     return l4_ipc(dest, utcb, L4_SYSF_SEND_AND_WAIT, 0, tag, src, timeout);
00572 }

```

```

00573
00574 L4_INLINE l4_msgtag_t
00575 l4_ipc_send(l4_cap_idx_t dest, l4_utcb_t *utcb,
00576             l4_msgtag_t tag,
00577             l4_timeout_t timeout) L4_NOTHROW
00578 {
00579     return l4_ipc(dest, utcb, L4_SYSF_SEND, 0, tag, 0, timeout);
00580 }
00581
00582 L4_INLINE l4_msgtag_t
00583 l4_ipc_wait(l4_utcb_t *utcb, l4_umword_t *src,
00584            l4_timeout_t timeout) L4_NOTHROW
00585 {
00586     l4_msgtag_t t;
00587     t.raw = 0;
00588     return l4_ipc(L4_INVALID_CAP, utcb, L4_SYSF_WAIT, 0, t, src, timeout);
00589 }
00590
00591 L4_INLINE l4_msgtag_t
00592 l4_ipc_receive(l4_cap_idx_t src, l4_utcb_t *utcb,
00593              l4_timeout_t timeout) L4_NOTHROW
00594 {
00595     l4_msgtag_t t;
00596     t.raw = 0;
00597     return l4_ipc(src, utcb, L4_SYSF_RECV, 0, t, 0, timeout);
00598 }
00599
00600 L4_INLINE l4_msgtag_t
00601 l4_ipc_sleep(l4_timeout_t timeout) L4_NOTHROW
00602 { return l4_ipc_receive(L4_INVALID_CAP, NULL, timeout); }
00603
00604 L4_INLINE l4_msgtag_t
00605 l4_ipc_sleep_ms(unsigned ms) L4_NOTHROW
00606 {
00607     return l4_ipc_sleep(l4_timeout(L4_IPC_TIMEOUT_NEVER,
00608                                   l4_timeout_from_us(ms * 1000)));
00609 }
00610
00611 L4_INLINE l4_msgtag_t
00612 l4_ipc_sleep_us(unsigned us) L4_NOTHROW
00613 {
00614     return l4_ipc_sleep(l4_timeout(L4_IPC_TIMEOUT_NEVER,
00615                                   l4_timeout_from_us(us)));
00616 }
00617
00618 L4_INLINE l4_umword_t
00619 l4_ipc_error(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW
00620 {
00621     if (!l4_msgtag_has_error(tag))
00622         return 0;
00623     return l4_utcb_tcr_u(utcb)->error & L4_IPC_ERROR_MASK;
00624 }
00625
00626 L4_INLINE long
00627 l4_error_u(l4_msgtag_t tag, l4_utcb_t *u) L4_NOTHROW
00628 {
00629     if (l4_msgtag_has_error(tag))
00630         return l4_ipc_to_errno(l4_utcb_tcr_u(u)->error & L4_IPC_ERROR_MASK);
00631     return l4_msgtag_label(tag);
00632 }
00633
00634 L4_INLINE long
00635 l4_error(l4_msgtag_t tag) L4_NOTHROW
00636 {
00637     return l4_error_u(tag, l4_utcb());
00638 }
00639
00640
00641
00642 L4_INLINE int l4_ipc_is_snd_error(l4_utcb_t *u) L4_NOTHROW
00643 { return (l4_utcb_tcr_u(u)->error & 1) == 0; }
00644
00645 L4_INLINE int l4_ipc_is_rcv_error(l4_utcb_t *u) L4_NOTHROW
00646 { return l4_utcb_tcr_u(u)->error & 1; }
00647
00648 L4_INLINE int l4_ipc_error_code(l4_utcb_t *u) L4_NOTHROW
00649 { return l4_utcb_tcr_u(u)->error & L4_IPC_ERROR_MASK; }
00650
00651
00652 /*
00653  * \internal
00654  * \ingroup l4_ipc_api
00655  */
00656 L4_INLINE int
00657 l4_sndfpage_add_u(l4_fpage_t const snd_fpage, unsigned long snd_base,
00658                  l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW
00659 {

```

```

00660  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00661  int i = l4_msgtag_words(*tag) + 2 * l4_msgtag_items(*tag);
00662
00663  if (i >= L4_UTCB_GENERIC_DATA_SIZE - 1)
00664      return -L4_ENOMEM;
00665
00666  v->mr[i] = snd_base | L4_ITEM_MAP | L4_ITEM_CONT;
00667  v->mr[i + 1] = snd_fpage.raw;
00668
00669  *tag = l4_msgtag(l4_msgtag_label(*tag), l4_msgtag_words(*tag),
00670                  l4_msgtag_items(*tag) + 1, l4_msgtag_flags(*tag));
00671  return 0;
00672 }
00673
00674 L4_INLINE int
00675 l4_sndfpage_add(l4_fpage_t const snd_fpage, unsigned long snd_base,
00676                l4_msgtag_t *tag) L4_NOTHROW
00677 {
00678     return l4_sndfpage_add_u(snd_fpage, snd_base, tag, l4_utcb());
00679 }
00680
00681
00682 #endif /* ! __L4SYS__INCLUDE__L4API_FIASCO__IPC_H__ */

```

16.486 x86/l4f/l4/sys/ipc.h File Reference

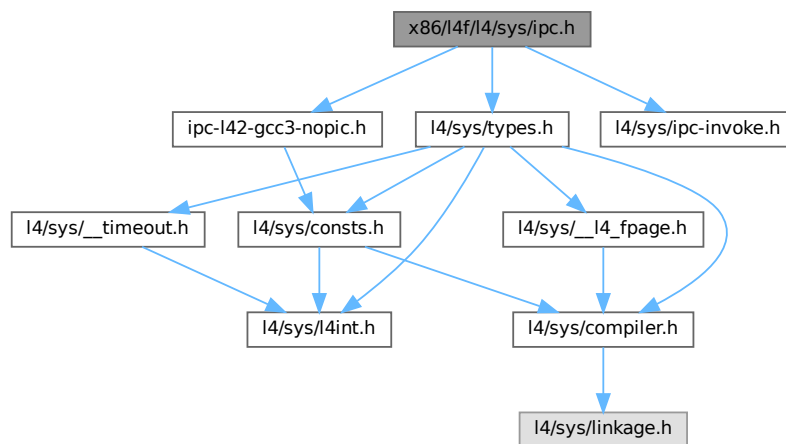
[L4](#) IPC System Calls, x86.

```

#include <l4/sys/types.h>
#include <l4/sys/ipc-invoke.h>
#include "ipc-l42-gcc3-nopic.h"

```

Include dependency graph for ipc.h:



16.486.1 Detailed Description

[L4](#) IPC System Calls, x86.

Definition in file [ipc.h](#).

16.487 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__ */

```

16.488 l4/sys/ipc_gate File Reference

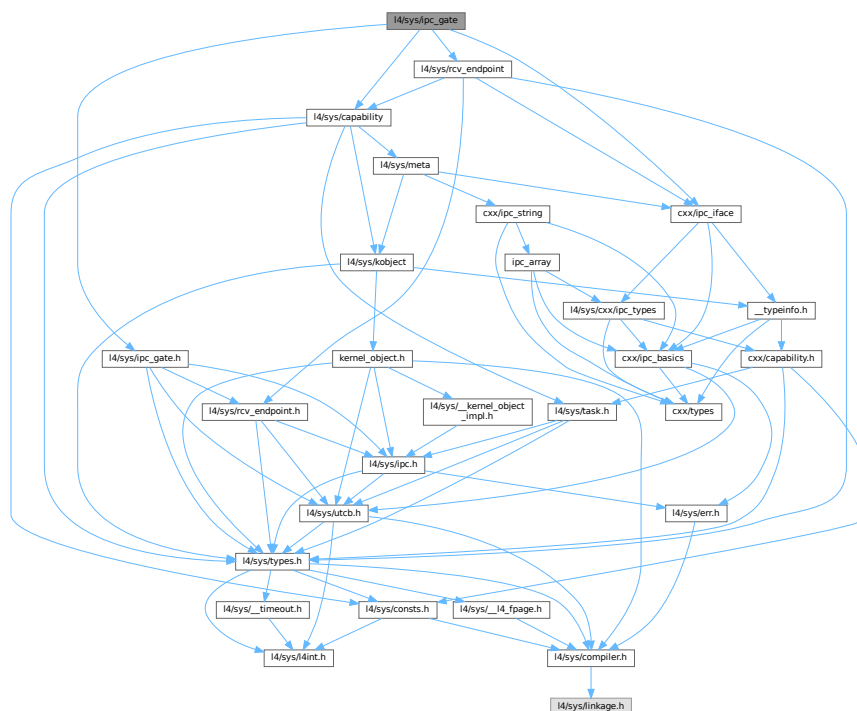
The C++ IPC gate interface.

```

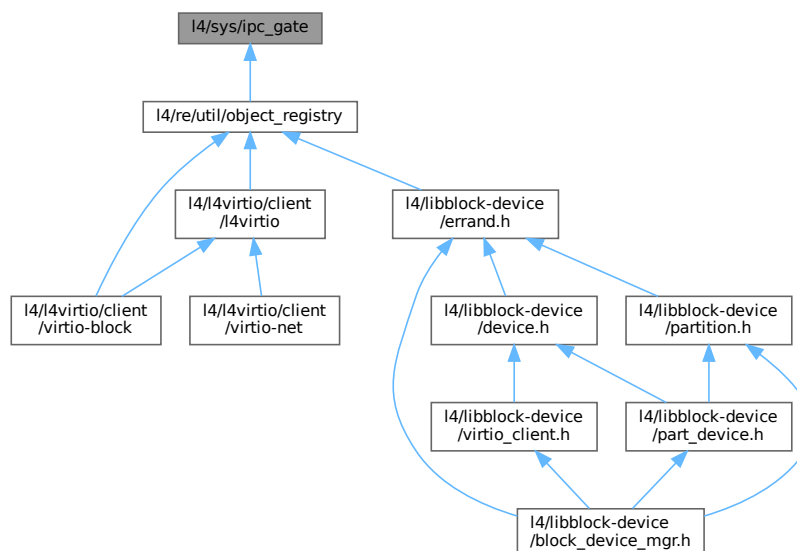
#include <l4/sys/ipc_gate.h>
#include <l4/sys/capability>
#include <l4/sys/rcv_endpoint>
#include <l4/sys/cxx/ipc_iface>

```

Include dependency graph for ipc_gate:



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.

16.488.1 Detailed Description

The C++ IPC gate interface.

Definition in file [ipc_gate](#).

16.489 ipc_gate

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2009-2010 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 #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::Rpcs_sys<bind_thread_t, get_infos_t> Rpcs;
00116 };
00117
00118 }
```

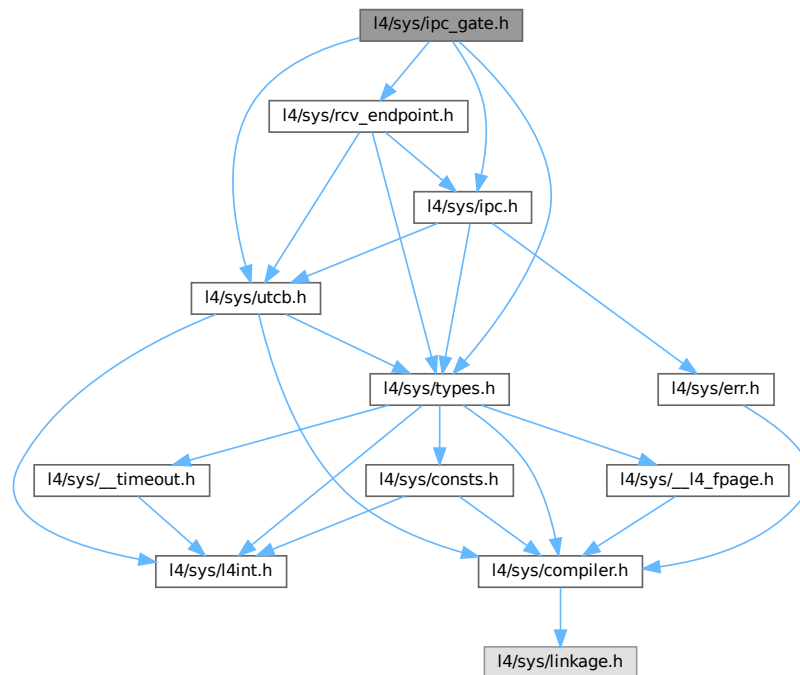
16.490 l4/sys/ipc_gate.h File Reference

The C IPC gate interface, see [L4::ipc_gate](#) for the C++ interface.

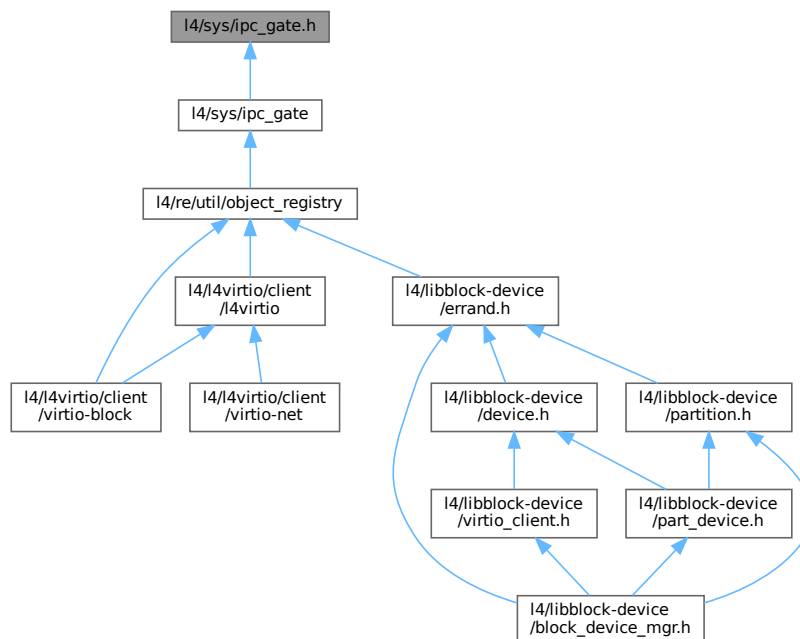
```
#include <l4/sys/utcb.h>
#include <l4/sys/types.h>
#include <l4/sys/rcv_endpoint.h>
```

```
#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.*

16.490.1 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](#).

16.491 ipc_gate.h

[Go to the documentation of this file.](#)

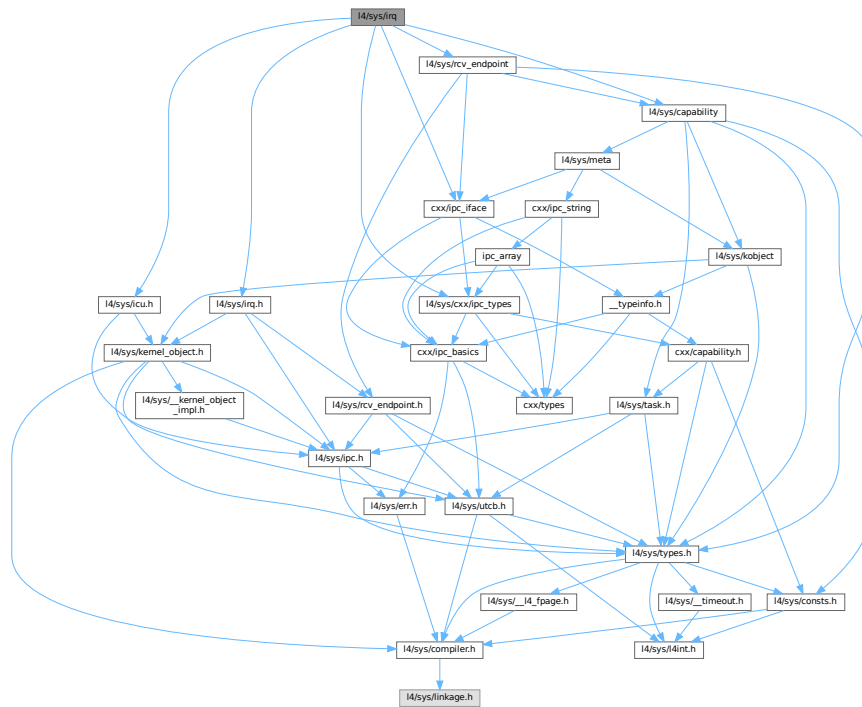
```

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 }

```

C++ Irq interface.

Include dependency graph for irq:

[illegible]

Data Structures

- class [L4::Irq_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.

16.492.1 Detailed Description

C++ Irq interface.

Definition in file [irq](#).

16.493 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);
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
00195     l4_msgtag_t unmask(l4_utcb_t *utcb = l4_utcb()) noexcept
00196     { return unmask(-1, 0, L4_IPC_NEVER, utcb); }
00197 };
00198
00214 struct Irq_mux : Kobject_t<Irq_mux, Triggerable, L4_PROTO_IRQ_MUX>
00215 {
00229     l4_msgtag_t chain(Cap<Triggerable> const &slave,
00230                     l4_utcb_t *utcb = l4_utcb()) noexcept
00231     { return l4_irq_mux_chain_u(cap(), slave.cap(), utcb); }
00232 };
00233
00234
00259 class Icu :
00260     public Kobject_t<Icu, Irq_eoi, L4_PROTO_IRQ,
00261                    Type_info::Demand_t<1> >
00262 {
00263 public:
00264     enum Mode
00265     {
00266         F_none           = L4_IRQ_F_NONE,
00267         F_level_high     = L4_IRQ_F_LEVEL_HIGH,
00268         F_level_low      = L4_IRQ_F_LEVEL_LOW,
00269         F_pos_edge       = L4_IRQ_F_POS_EDGE,
00270         F_neg_edge       = L4_IRQ_F_NEG_EDGE,
00271         F_both_edge      = L4_IRQ_F_BOTH_EDGE,
00272         F_mask           = L4_IRQ_F_MASK,
00273
00274         F_set_wakeup     = L4_IRQ_F_SET_WAKEUP,
00275         F_clear_wakeup   = L4_IRQ_F_CLEAR_WAKEUP,
00276     };
00277
00278     enum Flags
00279     {
00280         F_msi = L4_ICU_FLAG_MSI
00281     };
00282
00286     class Info : public l4_icu_info_t
00287     {
00288 public:
00290         bool supports_msi() const noexcept { return features & F_msi; }
00291     };
00292
00318     l4_msgtag_t bind(unsigned irqnum, L4::Cap<Triggerable> irq,
00319                    l4_utcb_t *utcb = l4_utcb()) noexcept
00320     { return l4_icu_bind_u(cap(), irqnum, irq.cap(), utcb); }
00321
00322     L4_RPC_NF_OP(
00323         L4_ICU_OP_BIND,
00324         l4_msgtag_t, bind, (l4_umword_t irqnum, Ipc::Cap<Irq> irq)
00325     );
00326
00336     l4_msgtag_t unbind(unsigned irqnum, L4::Cap<Triggerable> irq,
00337                      l4_utcb_t *utcb = l4_utcb()) noexcept
00338     { return l4_icu_unbind_u(cap(), irqnum, irq.cap(), utcb); }
00339

```

```

00340 L4_RPC_NF_OP(
00341     L4_ICU_OP_UNBIND,
00342     l4_msgtag_t, unbind, (l4_umword_t irqnum, l4::Cap<Irq> irq)
00343 );
00344
00353 l4_msgtag_t info(l4_icu_info_t *info, l4_utcb_t *utcb = l4_utcb()) noexcept
00354 { return l4_icu_info_u(cap(), info, utcb); }
00355
00356 struct _Info { l4_umword_t features, nr_irqs, nr_msis; };
00357 L4_RPC_NF_OP(L4_ICU_OP_INFO, l4_msgtag_t, info, (_Info *info));
00358
00371 L4_INLINE_RPC_OP(L4_ICU_OP_MSI_INFO,
00372     l4_msgtag_t, msi_info, (l4_umword_t irqnum, l4_uint64_t source,
00373     l4_icu_msi_info_t *msi_info));
00374
00378 l4_msgtag_t control(unsigned irqnum, unsigned op, l4_umword_t *label,
00379     l4_timeout_t to, l4_utcb_t *utcb = l4_utcb()) noexcept
00380 { return l4_icu_control_u(cap(), irqnum, op, label, to, utcb); }
00381
00401 l4_msgtag_t mask(unsigned irqnum,
00402     l4_umword_t *label = 0,
00403     l4_timeout_t to = L4_IPC_NEVER,
00404     l4_utcb_t *utcb = l4_utcb()) noexcept
00405 { return l4_icu_mask_u(cap(), irqnum, label, to, utcb); }
00406
00407 L4_RPC_NF_OP(
00408     L4_ICU_OP_MASK,
00409     l4_msgtag_t, mask, (l4_umword_t irqnum),
00410     L4::lpc::Send_only
00411 );
00412
00413
00414 L4_RPC_NF_OP(
00415     L4_ICU_OP_UNMASK,
00416     l4_msgtag_t, unmask, (l4_umword_t irqnum),
00417     L4::lpc::Send_only
00418 );
00419
00429 l4_msgtag_t set_mode(unsigned irqnum, l4_umword_t mode,
00430     l4_utcb_t *utcb = l4_utcb()) noexcept
00431 { return l4_icu_set_mode_u(cap(), irqnum, mode, utcb); }
00432
00433 L4_RPC_NF_OP(
00434     L4_ICU_OP_SET_MODE,
00435     l4_msgtag_t, set_mode, (l4_umword_t irqnum, l4_umword_t mode)
00436 );
00437
00438 typedef L4::Typeid::Rpcsys<
00439     bind_t, unbind_t, info_t, msi_info_t, unmask_t, mask_t, set_mode_t
00440     > Rpcsys;
00441 };
00442
00443 }

```

16.494 l4/sys/kdebug.h File Reference

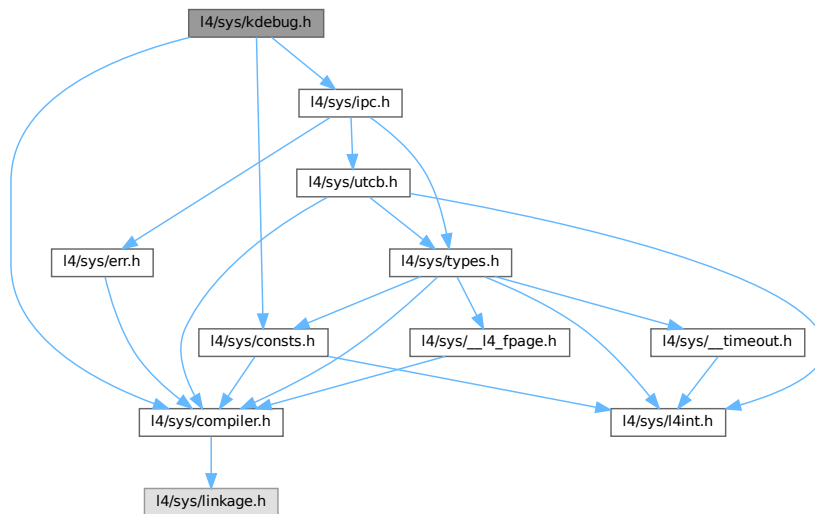
Functionality for invoking the kernel debugger.

```

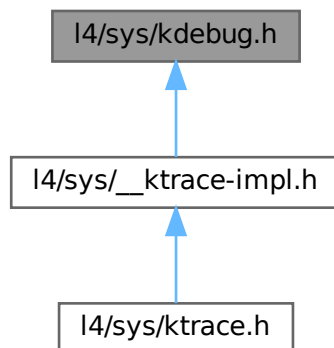
#include <l4/sys/compiler.h>
#include <l4/sys/consts.h>
#include <l4/sys/ipc.h>

```

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.

16.494.1 Detailed Description

Functionality for invoking the kernel debugger.

Definition in file [kdebug.h](#).

16.494.2 Enumeration Type Documentation

16.494.2.1 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](#).

16.494.3 Function Documentation

16.494.3.1 __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 l4_kdebug_ops_t or a value above 0x200 used by the kernel trace buffer implementation (__ktrace-impl.h).
<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

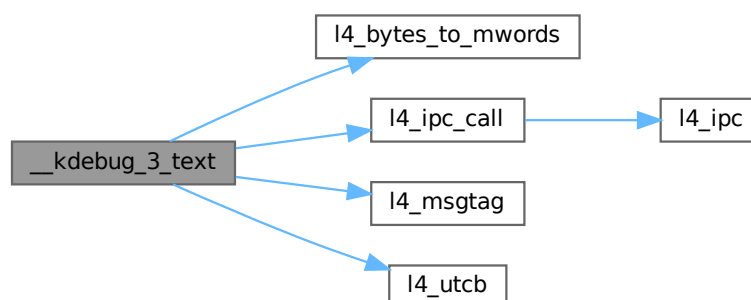
<i>Message</i>	tag returned from the IPC on the base debugger capability.
----------------	--

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:



16.494.3.2 __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 l4_kdebug_ops_t or a value above 0x200 used by the kernel trace buffer implementation (__ktrace-impl.h).
-----------	--

Return values

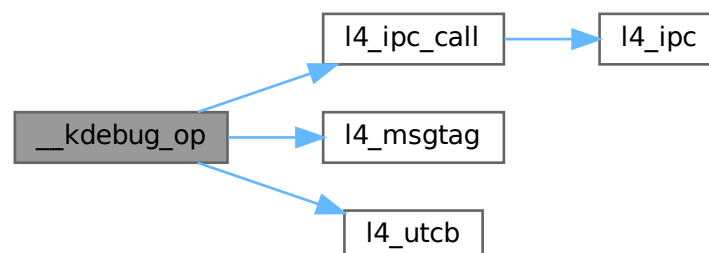
<i>Message</i>	tag returned from the IPC on the base debugger capability.
----------------	--

Definition at line 66 of file [kdebug.h](#).

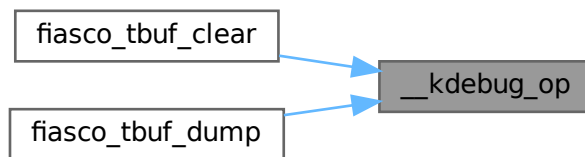
References [L4_BASE_DEBUGGER_CAP](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_PROTO_DEBUGGER](#), [l4_utcb\(\)](#), and [l4_msg_regs_t::mr](#).

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:



16.494.3.3 __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 l4_kdebug_ops_t or a value above 0x200 used by the kernel trace buffer implementation (__ktrace-impl.h).
<i>val</i>	Machine word argument to the unary operation.

Return values

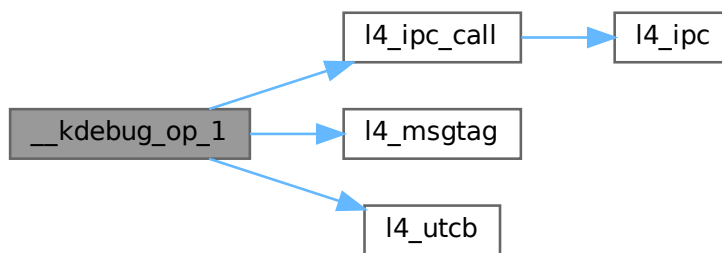
<i>Message</i>	tag returned from the IPC on the base debugger capability.
----------------	--

Definition at line 174 of file [kdebug.h](#).

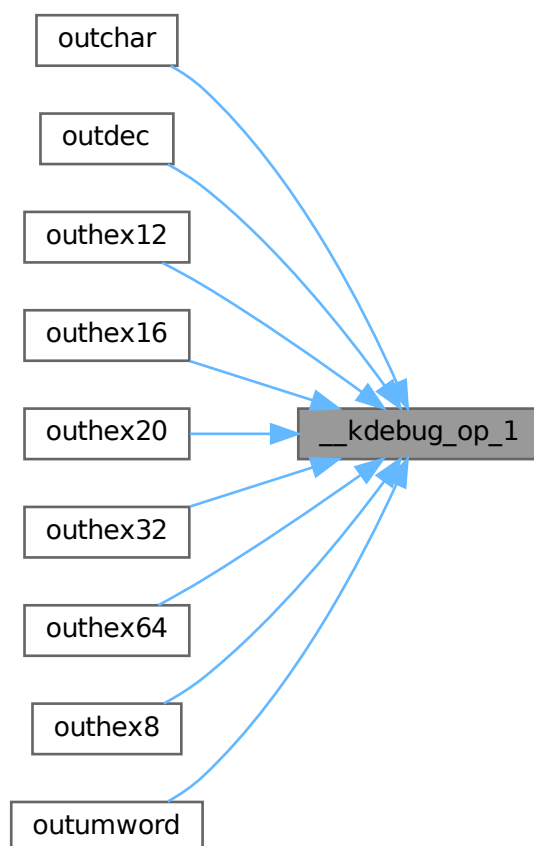
References [L4_BASE_DEBUGGER_CAP](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_PROTO_DEBUGGER](#), [l4_utcb\(\)](#), and [l4_msg_regs_t::mr](#).

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:



16.494.3.4 `__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 l4_kdebug_ops_t or a value above 0x200 used by the kernel trace buffer implementation (__ktrace-impl.h).
<i>text</i>	Output string.
<i>len</i>	Length of the output string. The maximum length is limited to L4_UTCB_GENERIC_DATA_SIZE - 2 machine words. Output strings longer than this limit will be cropped.

Return values

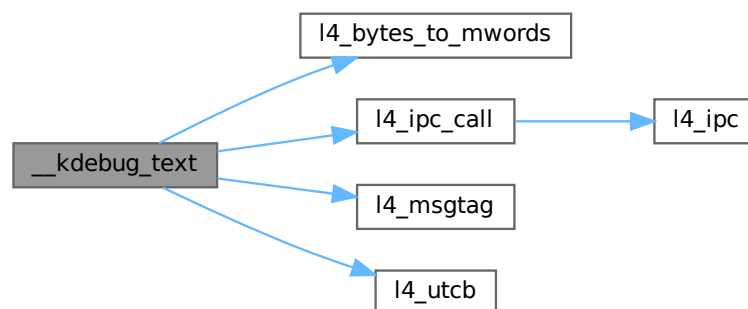
<i>Message</i>	tag returned from the IPC on the base debugger capability.
----------------	--

Definition at line 96 of file [kdebug.h](#).

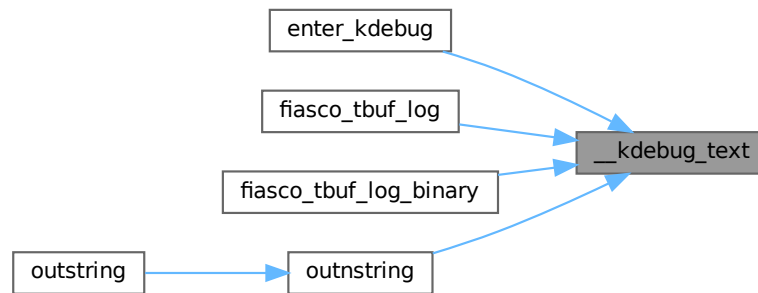
References [L4_BASE_DEBUGGER_CAP](#), [l4_bytes_to_mwords\(\)](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_PROTO_DEBUGGER](#), [l4_utcb\(\)](#), and [l4_msg_regs_t::mr](#).

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:



16.494.3.5 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.

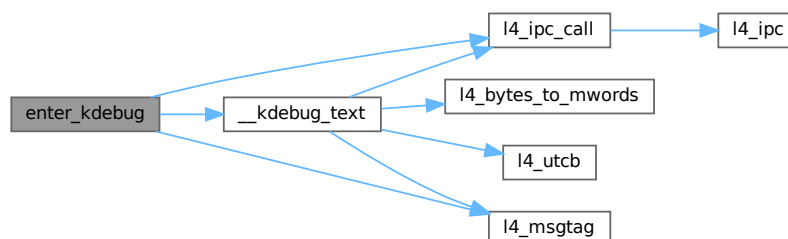
Examples

[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:



16.494.3.6 outchar()

```
void outchar (
    char c ) [inline]
```

Output a single character via the kernel debugger.

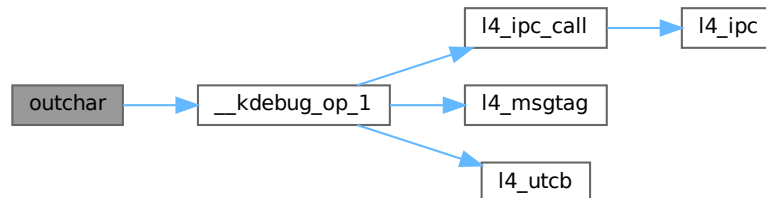
Parameters

<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:



16.494.3.7 outdec()

```
void outdec (
    l4_mword_t number ) [inline]
```

Output a decimal unsigned machine word via the kernel debugger.

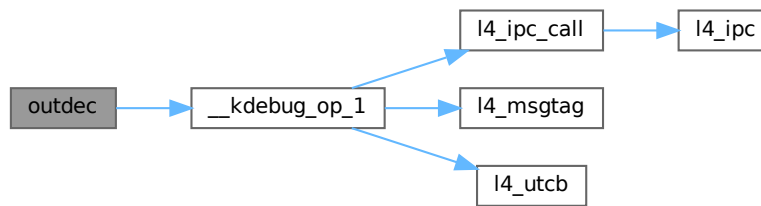
Parameters

<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:



16.494.3.8 outhex12()

```
void outhex12 (
    l4_uint16_t number ) [inline]
```

Output a 12-bit unsigned hexadecimal number via the kernel debugger.

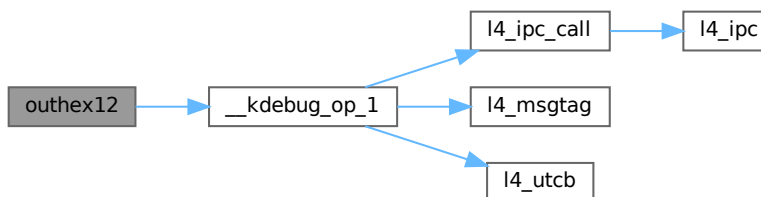
Parameters

<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:



16.494.3.9 outhex16()

```
void outhex16 (
    l4_uint16_t number ) [inline]
```

Output a 16-bit unsigned hexadecimal number via the kernel debugger.

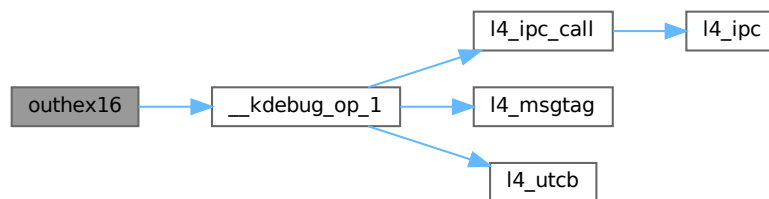
Parameters

<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:



16.494.3.10 outhex20()

```
void outhex20 (
    14_uint32_t number ) [inline]
```

Output a 20-bit unsigned hexadecimal number via the kernel debugger.

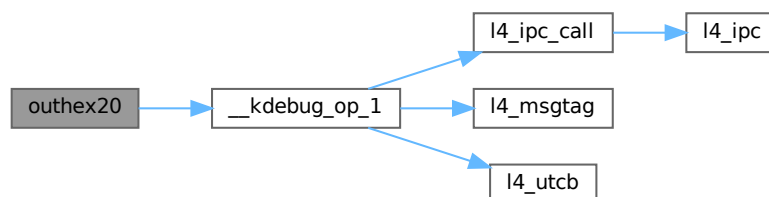
Parameters

<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:



16.494.3.11 outhex32()

```
void outhex32 (
    l4_uint32_t number ) [inline]
```

Output a 32-bit unsigned hexadecimal number via the kernel debugger.

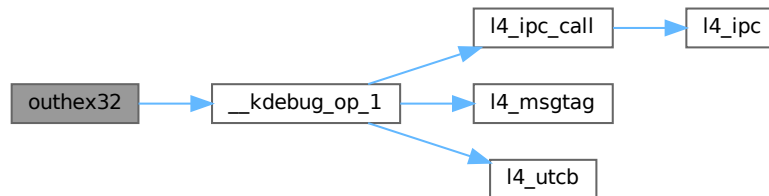
Parameters

<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:



16.494.3.12 outhex64()

```
void outhex64 (
    l4_uint64_t number ) [inline]
```

Output a 64-bit unsigned hexadecimal number via the kernel debugger.

Parameters

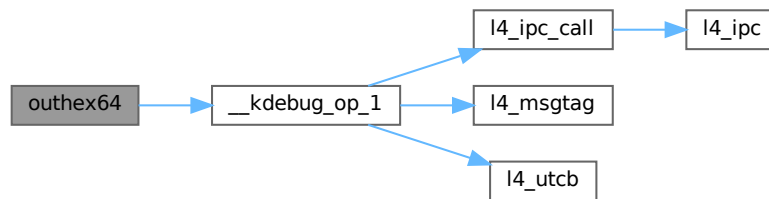
<i>number</i>	Output 64-bit number.
---------------	-----------------------

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:



16.494.3.13 outhex8()

```
void outhex8 (
    l4_uint8_t number ) [inline]
```

Output an 8-bit unsigned hexadecimal number via the kernel debugger.

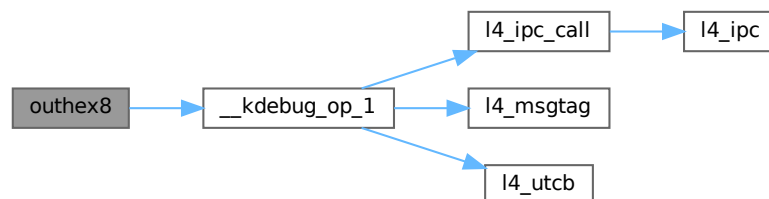
Parameters

<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:



16.494.3.14 outnstring()

```
void outnstring (
    char const * text,
    unsigned len ) [inline]
```

Output a fixed-length string via the kernel debugger.

Parameters

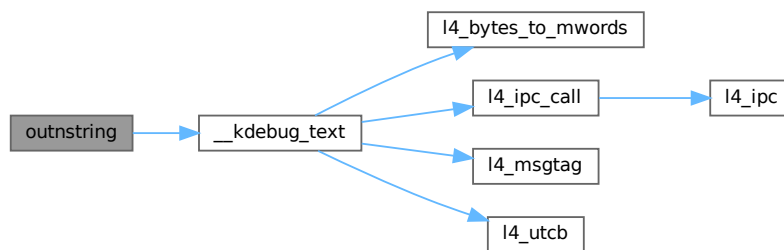
<i>text</i>	Beginning of the output string.
<i>len</i>	Length of the output string. The maximum length is limited to L4_UTCB_GENERIC_DATA_SIZE - 2 machine words. Output strings longer than this limit will be cropped.

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:



16.494.3.15 outstring()

```
void outstring (
    char const * text ) [inline]
```

Output a string via the kernel debugger.

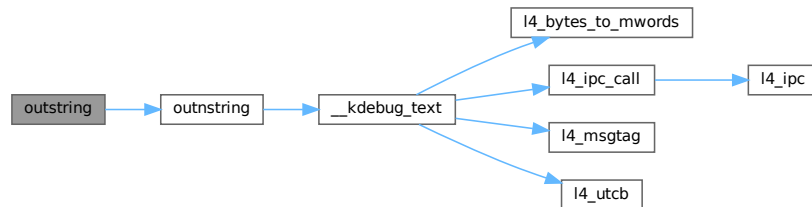
Parameters

<i>text</i>	Beginning of the output string. The maximum length of the output string is limited to L4_UTCB_GENERIC_DATA_SIZE - 2 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:



16.494.3.16 outumword()

```
void outumword (
    l4_umword_t number ) [inline]
```

Output a hexadecimal unsigned machine word via the kernel debugger.

Parameters

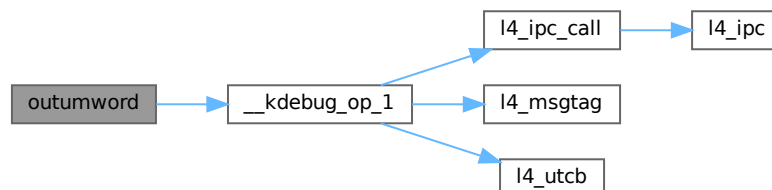
<i>number</i>	Output machine word.
---------------	----------------------

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:



16.495 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
00021 #ifndef __KDEBUG_H__
00022 #define __KDEBUG_H__
00023
00024 #include <l4/sys/compiler.h>
00025 #include <l4/sys/consts.h>
00026 #include <l4/sys/ipc.h>
00027
00028 L4_INLINE void
00029 enter_kdebug(char const *text) L4_NOTHROW;
00030
00031 enum l4_kdebug_ops_t
00032 {
00033     L4_KDEBUG_ENTER      = 0,
00034     L4_KDEBUG_OUTCHAR    = 1,
00035     L4_KDEBUG_OUTSTRING  = 2,
00036     L4_KDEBUG_OUTHEX32   = 3,
00037     L4_KDEBUG_OUTHEX20   = 4,
00038     L4_KDEBUG_OUTHEX16   = 5,
00039     L4_KDEBUG_OUTHEX12   = 6,
00040     L4_KDEBUG_OUTHEX8    = 7,
00041     L4_KDEBUG_OUTDEC     = 8,
00042 };
00043
00044 L4_INLINE l4_msgtag_t
00045 __kdebug_op(unsigned op) L4_NOTHROW
00046 {
00047     l4_msgtag_t res;
00048     l4_utcb_t *u = l4_utcb();
00049     l4_msg_regs_t *mr = l4_utcb_mr_u(u);
00050     l4_umword_t mr0 = mr->mr[0];
00051
00052     mr->mr[0] = op;
00053     res = l4_ipc_call(L4_BASE_DEBUGGER_CAP, u,
00054                     l4_msgtag(L4_PROTO_DEBUGGER, 1, 0, 0),
00055                     L4_IPC_NEVER);
00056     mr->mr[0] = mr0;
00057     return res;
00058 }
00059
00060 L4_INLINE l4_msgtag_t
00061 __kdebug_text(unsigned op, char const *text, unsigned len) L4_NOTHROW
00062 {
00063     l4_msg_regs_t store;
00064     l4_msgtag_t res;
00065     l4_utcb_t *u = l4_utcb();
00066     l4_msg_regs_t *mr = l4_utcb_mr_u(u);
00067
00068     if (len > (sizeof(store) - (2 * sizeof(l4_umword_t))))
00069         len = sizeof(store) - (2 * sizeof(l4_umword_t));
00070
00071     __builtin_memcpy(&store, mr, sizeof(store));
00072     mr->mr[0] = op;
00073     mr->mr[1] = len;
00074     __builtin_memcpy(&mr->mr[2], text, len);
00075     res = l4_ipc_call(L4_BASE_DEBUGGER_CAP, u,
00076                     l4_msgtag(L4_PROTO_DEBUGGER,
00077                               l4_bytes_to_mwords(len) + 2, 0, 0),
00078                     L4_IPC_NEVER);
00079     __builtin_memcpy(mr, &store, sizeof(*mr));
00080     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__

```

16.496 l4/sys/kernel_object.h File Reference

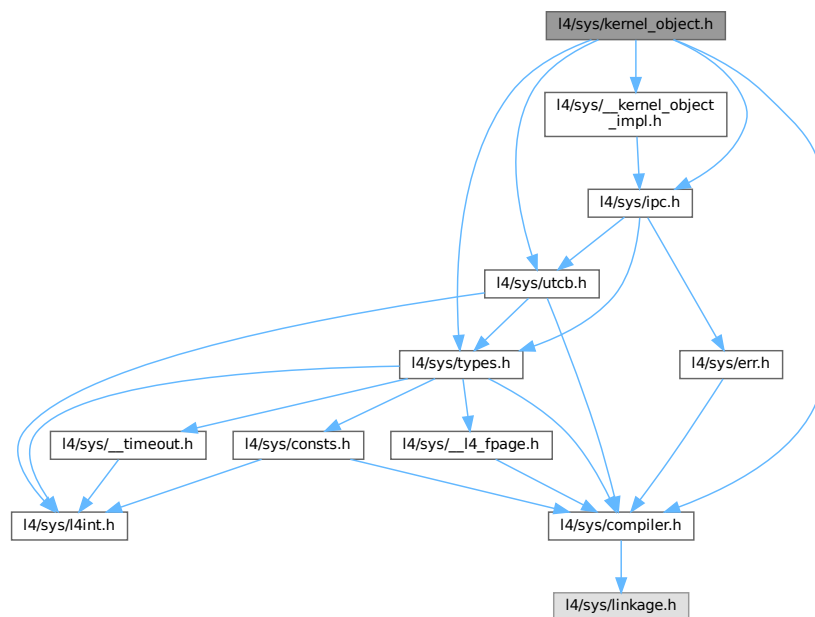
Kernel object system calls.

```

#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:



16.496.1 Detailed Description

Kernel object system calls.

Definition in file [kernel_object.h](#).

16.497 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;

```

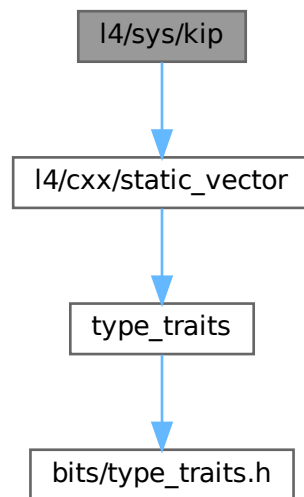


```
00077     return l4_ipc_call(obj, u, l4_msgtag(L4_PROTO_KOBJECT, 2, 0, 0), L4_IPC_NEVER);
00078 }
00079
00080 L4_INLINE l4_msgtag_t
00081 l4_kobject_dec_refcnt(l4_cap_idx_t obj, l4_mword_t diff) L4_NOTHROW
00082 {
00083     return l4_kobject_dec_refcnt_u(obj, diff, l4_utcb());
00084 }
00085
00086 #endif /* ! __L4SYS__KERNEL_OBJECT_H__ */
```

16.498 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.

16.498.1 Detailed Description

L4::Kip class, memory descriptors.

Author

Alexander Warg alexander.warg@os.inf.tu-dresden.de

Definition in file [kip](#).

16.499 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 *((unsigned long *)kip + 21); }
00094
00095             static unsigned long memory_info(void const *kip) noexcept
00096             { return *((unsigned long const *)kip + 21); }
00097
00098         public:
00106             static Mem_desc *first(void *kip) noexcept
00107             {
00108                 return (Mem_desc *)((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                 return (Mem_desc const *)((char const *)kip
00115                     + (memory_info(kip) » ((sizeof(unsigned long) / 2) * 8)));
00116             }
00117
00125             static unsigned long count(void const *kip) noexcept
00126             {
00127                 return memory_info(kip)
00128                     & ((1UL « ((sizeof(unsigned long) / 2) * 8)) - 1);
00128             }
00129         };
00130     }
00131 }
```

```

00129     }
00130
00137     static void count(void *kip, unsigned count) noexcept
00138     {
00139         unsigned long &mi = memory_info(kip);
00140         mi = (mi & ~(1UL << ((sizeof(unsigned long) / 2) * 8)) - 1) | count;
00141     }
00142
00148     static inline cxx::static_vector<Mem_desc const> all(void const *kip)
00149     {
00150         return cxx::static_vector<Mem_desc const>(Mem_desc::first(kip),
00151                                                    Mem_desc::count(kip));
00152     }
00153
00159     static inline cxx::static_vector<Mem_desc> all(void *kip)
00160     {
00161         return cxx::static_vector<Mem_desc>(Mem_desc::first(kip),
00162                                              Mem_desc::count(kip));
00163     }
00164
00175     Mem_desc(unsigned long start, unsigned long end,
00176              Mem_type t, unsigned char st = 0, bool virt = false) noexcept
00177     : _l((start & ~0x3ffUL) | (t & 0x0f) | ((st << 4) & 0x0f0)
00178         | (virt ? 0x0200 : 0x0)), _h(end | 0x3ffUL)
00179     {}
00180
00186     unsigned long start() const noexcept { return _l & ~0x3ffUL; }
00187
00193     unsigned long end() const noexcept { return _h | 0x3ffUL; }
00194
00200     unsigned long size() const noexcept { return end() + 1 - start(); }
00201
00207     Mem_type type() const noexcept { return (Mem_type)(_l & 0x0f); }
00208
00214     unsigned char sub_type() const noexcept { return (_l >> 4) & 0x0f; }
00215
00222     unsigned is_virtual() const noexcept { return _l & 0x200; }
00223
00233     void set(unsigned long start, unsigned long end,
00234             Mem_type t, unsigned char st = 0, bool virt = false) noexcept
00235     {
00236         _l = (start & ~0x3ffUL) | (t & 0x0f) | ((st << 4) & 0x0f0)
00237             | (virt?0x0200:0x0);
00238
00239         _h = end | 0x3ffUL;
00240     }
00241
00242 };
00243 };
00244 };
00245
00246 #endif

```

16.500 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
00059     struct __Kobject_typeid
00060     {
00061         typedef Type_info::Demand_t<> Demand;
00062         static Type_info const _m;
00063     };
00064
00065     l4_cap_idx_t cap() const noexcept { return _c(); }
00066
00067 private:
00068     l4_cap_idx_t _c() const noexcept
00069     { return reinterpret_cast<l4_cap_idx_t>(this) & L4_CAP_MASK; }
00070
00071 public:
00072     l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
00073     { return l4_kobject_dec_refcnt_u(cap(), diff, utcb); }
00074 };
00075
00076 template<typename Derived, long PROTO = L4::PROTO_ANY,
00077         typename S_DEMAND = Type_info::Demand_t<> >
00078 struct Kobject_0t : Kobject_t<Derived, L4::Kobject, PROTO, S_DEMAND> {};
00079
00080 }
00081
00082

```

16.501 l4/sys/ktrace.h File Reference

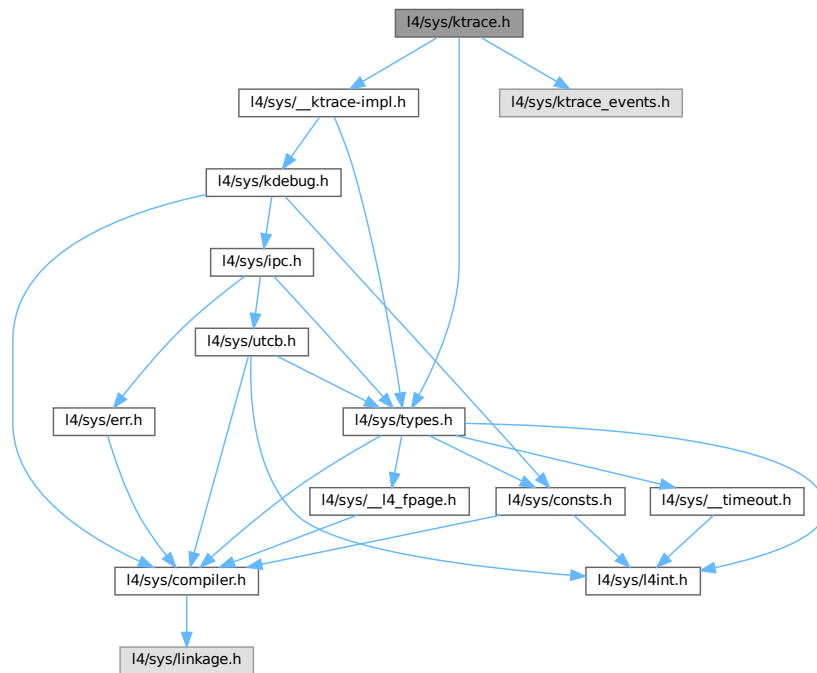
L4 kernel event tracing.

```

#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.

16.501.1 Detailed Description

[L4](#) kernel event tracing.

Definition in file [ktrace.h](#).

16.502 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

```

16.503 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.

16.503.1 Detailed Description

Fixed sized integer types, amd64 version.

Definition in file [l4int.h](#).

16.504 l4int.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 #pragma once
00026
00027 #include_next <l4/sys/l4int.h>
00028
00034 #define L4_MWORD_BITS      64
00036 typedef unsigned long      l4_size_t;
00037 typedef signed long       l4_ssize_t;
```

16.505 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.

16.505.1 Detailed Description

Fixed sized integer types, arm version.

Definition in file [l4int.h](#).

16.506 I4int.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;

```

16.507 I4/sys/I4int.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 **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.

16.507.1 Detailed Description

Fixed sized integer types, generic version.

Definition in file [l4int.h](#).

16.508 l4int.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__ */

```

16.509 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.

16.509.1 Detailed Description

Fixed sized integer types, x86 version.

Definition in file [l4int.h](#).

16.510 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;

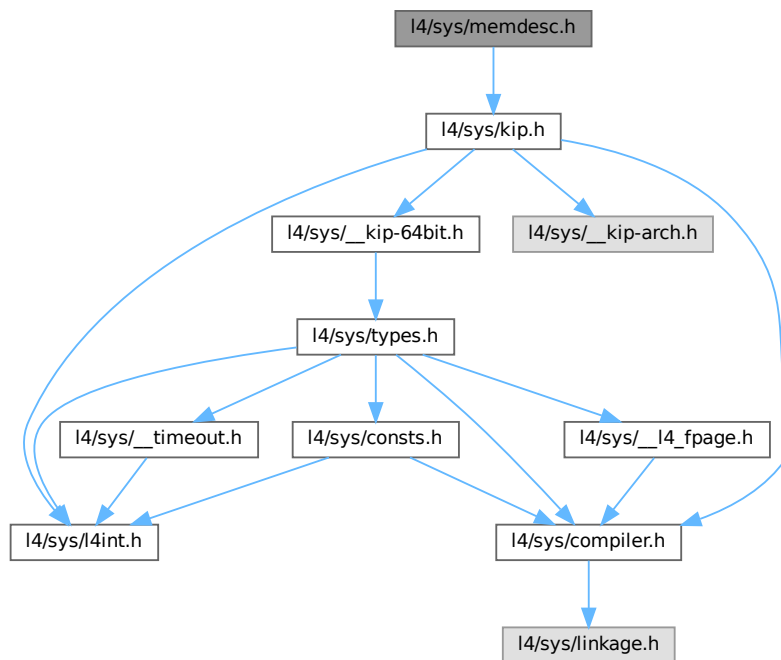
```

16.511 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.

16.511.1 Detailed Description

Memory description functions.

Definition in file [memdesc.h](#).

16.512 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
00091
00092
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) & 0x0f0)
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__ */

```

16.513 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 }}

```

16.514 l4/sys/meta File Reference

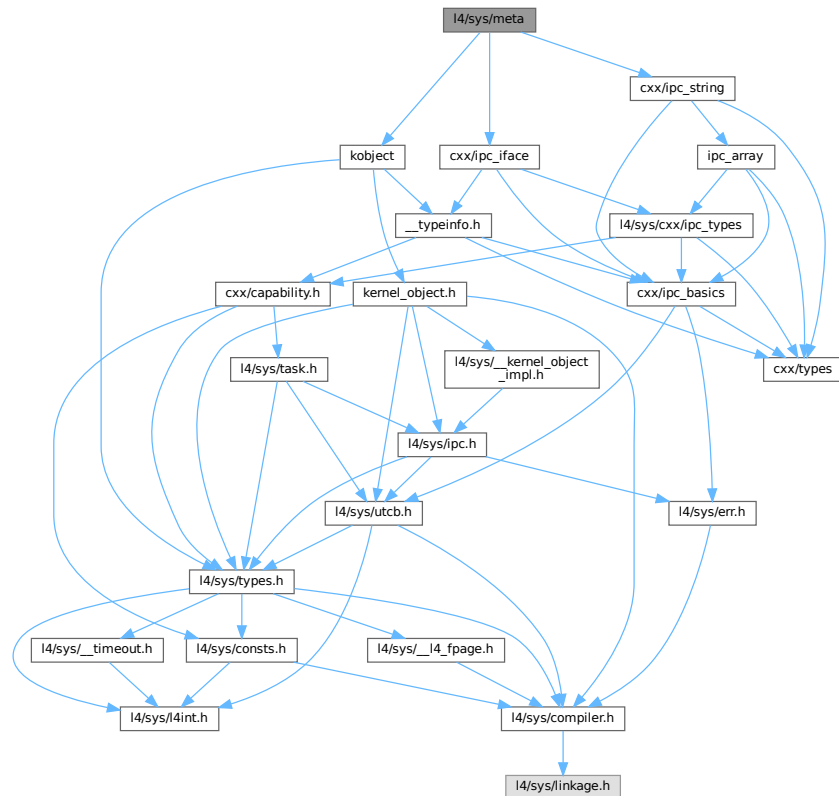
Meta interface for getting dynamic type information about objects behind capabilities.

```

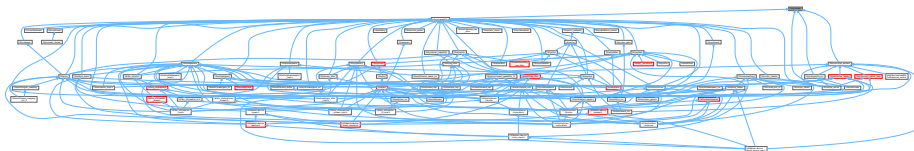
#include "kobject"
#include "cxx/ipc_iface"
#include "cxx/ipc_string"

```

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.

16.514.1 Detailed Description

Meta interface for getting dynamic type information about objects behind capabilities.

Definition in file [meta](#).

16.515 meta

[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
00025 #pragma once
00026
00027 #include "kobject"
00028 #include "cxx/ipc_iface"
00029 #include "cxx/ipc_string"
00030
00031 namespace L4 {
00032
00037 class Meta : public Kobject_t<Meta, Kobject, L4_PROTO_META>
00038 {
00039 public:
00046     L4_INLINE_RPC(l4_msgtag_t, num_interfaces, ());
00047
00061     L4_INLINE_RPC(l4_msgtag_t, interface, (l4_umword_t idx, long *proto,
00062                                           L4::Ipc::String<char> *name));
00063
00075     L4_INLINE_RPC(l4_msgtag_t, supports, (l4_mword_t protocol));
00076
00077     typedef L4::Typeid::Rpc<num_interfaces_t, interface_t, supports_t> Rpc;
00078 };
00079
00080 }
```

16.516 l4/sys/pager File Reference

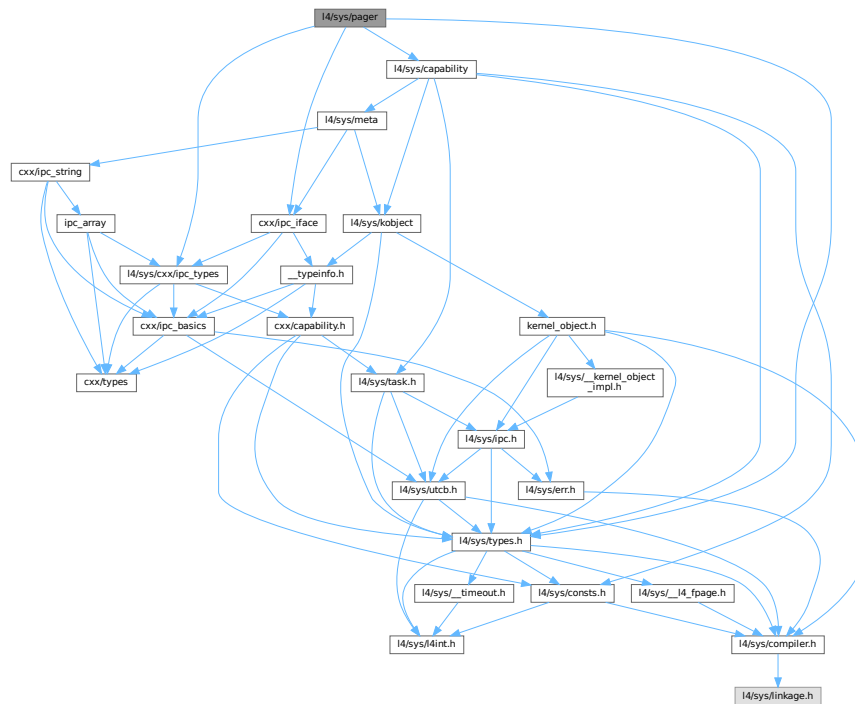
Pager and lo_pager C++ interface.

```
#include <l4/sys/capability>
#include <l4/sys/types.h>
#include <l4/sys/cxx/ipc_types>
```

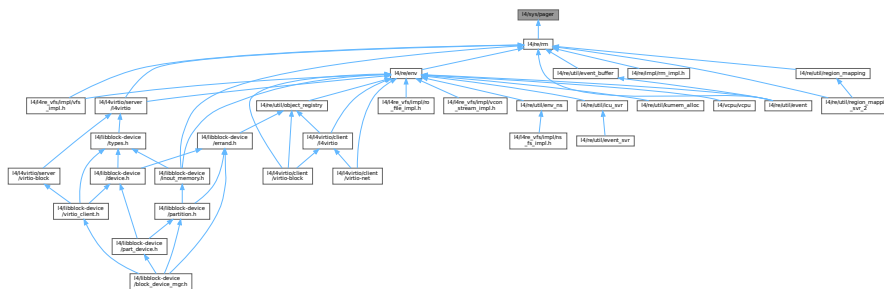


```
#include <l4/sys/cxx/ipc_iface>
```

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.

16.516.1 Detailed Description

Pager and Io_pager C++ interface.

Definition in file [pager](#).

16.517 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> Rpc;
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> Rpc;
00140 };
00141
00142 }
```

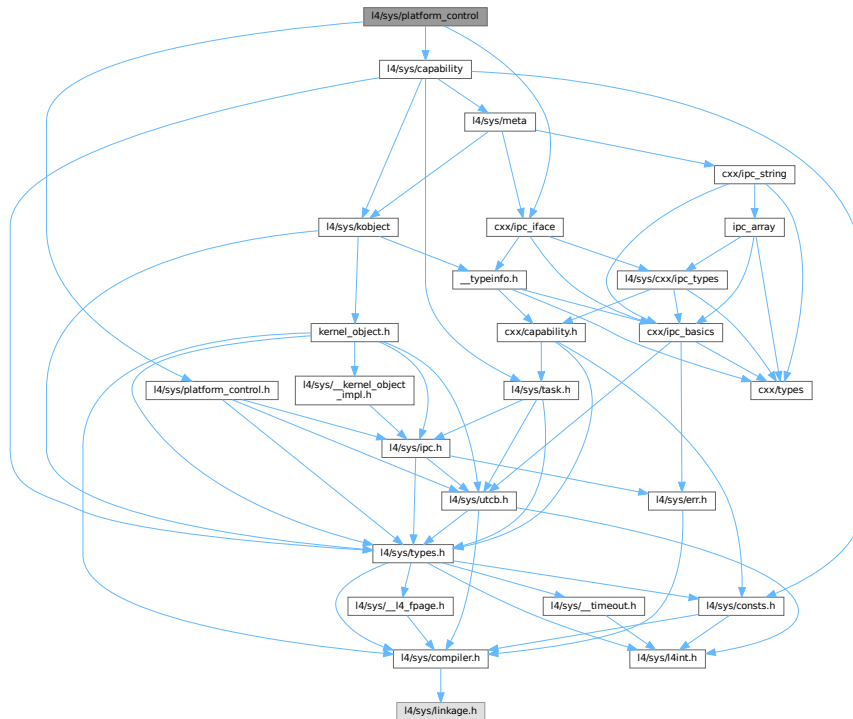
16.518 l4/sys/platform_control File Reference

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.

16.518.1 Detailed Description

Platform control object.

Definition in file [platform_control](#).

16.519 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:
00061     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_SYS_SUSPEND_OP,
00062                     l4_msgtag_t, system_suspend, (l4_umword_t extras));
00063
00069     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_SYS_SHUTDOWN_OP,
00070                     l4_msgtag_t, system_shutdown, (l4_umword_t reboot));
00071
00081     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP,
00082                     l4_msgtag_t, cpu_allow_shutdown,
00083                     (l4_umword_t phys_id, l4_umword_t enable));
00084
00094     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_ENABLE_OP,
00095                     l4_msgtag_t, cpu_enable, (l4_umword_t phys_id));
00096
00106     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_DISABLE_OP,
00107                     l4_msgtag_t, cpu_disable, (l4_umword_t phys_id));
00108
00109     typedef L4::Typeid::Rpcsys<system_suspend_t, system_shutdown_t,
00110                               cpu_allow_shutdown_t, cpu_enable_t,
00111                               cpu_disable_t> Rpcsys;
00112 };
00113
00114 }
00115

```

16.520 l4/sys/platform_control.h File Reference

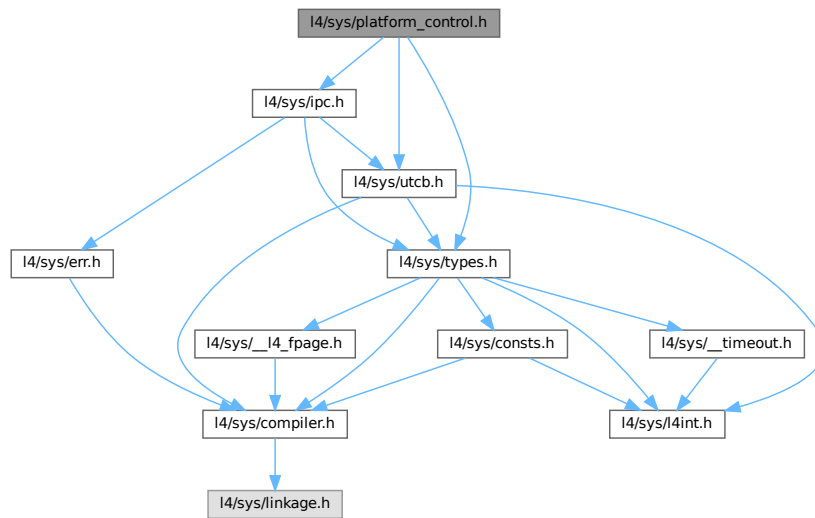
Platform control object.

```

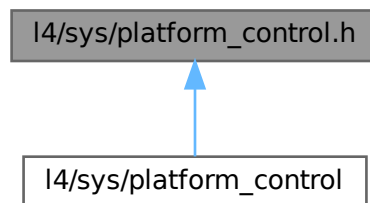
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>

```

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 }
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.

16.520.1 Detailed Description

Platform control object.

Definition in file [platform_control.h](#).

16.521 [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
00056 L4_INLINE l4_msgtag_t
00057 l4_platform_ctl_system_suspend(l4_cap_idx_t pfc,
00058                               l4_umword_t extras) L4_NOTHROW;
00059
00063 L4_INLINE l4_msgtag_t
00064 l4_platform_ctl_system_suspend_u(l4_cap_idx_t pfc,
00065                                  l4_umword_t extras,
00066                                  l4_utcb_t *utcb) L4_NOTHROW;
00067
00068
00077 L4_INLINE l4_msgtag_t
00078 l4_platform_ctl_system_shutdown(l4_cap_idx_t pfc,
00079                                 l4_umword_t reboot) L4_NOTHROW;
00080
00084 L4_INLINE l4_msgtag_t
00085 l4_platform_ctl_system_shutdown_u(l4_cap_idx_t pfc,
00086                                   l4_umword_t reboot,
00087                                   l4_utcb_t *utcb) L4_NOTHROW;
00088
00098 L4_INLINE l4_msgtag_t
00099 l4_platform_ctl_cpu_allow_shutdown(l4_cap_idx_t pfc,
00100                                   l4_umword_t phys_id,
00101                                   l4_umword_t enable) L4_NOTHROW;
00102
00106 L4_INLINE l4_msgtag_t
00107 l4_platform_ctl_cpu_allow_shutdown_u(l4_cap_idx_t pfc,

```

```

00108             l4_umword_t phys_id,
00109             l4_umword_t enable,
00110             l4_utcb_t *utcb) L4_NOTHROW;
00121 L4_INLINE l4_msgtag_t
00122 l4_platform_ctl_cpu_enable(l4_cap_idx_t pfc,
00123             l4_umword_t phys_id) L4_NOTHROW;
00124
00128 L4_INLINE l4_msgtag_t
00129 l4_platform_ctl_cpu_enable_u(l4_cap_idx_t pfc,
00130             l4_umword_t phys_id,
00131             l4_utcb_t *utcb) L4_NOTHROW;
00132
00143 L4_INLINE l4_msgtag_t
00144 l4_platform_ctl_cpu_disable(l4_cap_idx_t pfc,
00145             l4_umword_t phys_id) L4_NOTHROW;
00146
00150 L4_INLINE l4_msgtag_t
00151 l4_platform_ctl_cpu_disable_u(l4_cap_idx_t pfc,
00152             l4_umword_t phys_id,
00153             l4_utcb_t *utcb) L4_NOTHROW;
00154
00156 /* ends l4_platform_control_api group */ 00156
00157
00166 enum L4_platform_ctl_ops
00167 {
00168     L4_PLATFORM_CTL_SYS_SUSPEND_OP      = 0UL,
00169     L4_PLATFORM_CTL_SYS_SHUTDOWN_OP     = 1UL,
00170     L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP = 2UL,
00171     L4_PLATFORM_CTL_CPU_ENABLE_OP       = 3UL,
00172     L4_PLATFORM_CTL_CPU_DISABLE_OP      = 4UL,
00173 };
00174
00179 enum L4_platform_ctl_proto
00180 {
00181     L4_PROTO_PLATFORM_CTL = 0
00182 };
00183
00184 /* IMPLEMENTATION -----*/
00185
00186 #include <l4/sys/ipc.h>
00187
00188 L4_INLINE l4_msgtag_t
00189 l4_platform_ctl_system_suspend_u(l4_cap_idx_t pfc,
00190             l4_umword_t extras,
00191             l4_utcb_t *utcb) L4_NOTHROW
00192 {
00193     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00194     v->mr[0] = L4_PLATFORM_CTL_SYS_SUSPEND_OP;
00195     v->mr[1] = extras;
00196     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00197             L4_IPC_NEVER);
00198 }
00199
00200 L4_INLINE l4_msgtag_t
00201 l4_platform_ctl_system_shutdown_u(l4_cap_idx_t pfc,
00202             l4_umword_t reboot,
00203             l4_utcb_t *utcb) L4_NOTHROW
00204 {
00205     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00206     v->mr[0] = L4_PLATFORM_CTL_SYS_SHUTDOWN_OP;
00207     v->mr[1] = reboot;
00208     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00209             L4_IPC_NEVER);
00210 }
00211
00212 L4_INLINE l4_msgtag_t
00213 l4_platform_ctl_system_suspend(l4_cap_idx_t pfc,
00214             l4_umword_t extras) L4_NOTHROW
00215 {
00216     return l4_platform_ctl_system_suspend_u(pfc, extras, l4_utcb());
00217 }
00218
00219 L4_INLINE l4_msgtag_t
00220 l4_platform_ctl_system_shutdown(l4_cap_idx_t pfc,
00221             l4_umword_t reboot) L4_NOTHROW
00222 {
00223     return l4_platform_ctl_system_shutdown_u(pfc, reboot, l4_utcb());
00224 }
00225
00226 L4_INLINE l4_msgtag_t
00227 l4_platform_ctl_cpu_allow_shutdown_u(l4_cap_idx_t pfc,
00228             l4_umword_t phys_id,
00229             l4_umword_t enable,
00230             l4_utcb_t *utcb) L4_NOTHROW
00231 {
00232     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);

```

```

00239 v->mr[0] = L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP;
00240 v->mr[1] = phys_id;
00241 v->mr[2] = enable;
00242 return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 3, 0, 0),
00243                     L4_IPC_NEVER);
00244 }
00245
00246 L4_INLINE l4_msgtag_t
00247 l4_platform_ctl_cpu_allow_shutdown(l4_cap_idx_t pfc,
00248                                   l4_umword_t phys_id,
00249                                   l4_umword_t enable) L4_NOTHROW
00250 {
00251     return l4_platform_ctl_cpu_allow_shutdown_u(pfc, phys_id, enable, l4_utcb());
00252 }
00253
00254 L4_INLINE l4_msgtag_t
00255 l4_platform_ctl_cpu_enable_u(l4_cap_idx_t pfc,
00256                             l4_umword_t phys_id,
00257                             l4_utcb_t *utcb) L4_NOTHROW
00258 {
00259     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00260     v->mr[0] = L4_PLATFORM_CTL_CPU_ENABLE_OP;
00261     v->mr[1] = phys_id;
00262     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00263                     L4_IPC_NEVER);
00264 }
00265
00266 L4_INLINE l4_msgtag_t
00267 l4_platform_ctl_cpu_disable_u(l4_cap_idx_t pfc,
00268                              l4_umword_t phys_id,
00269                              l4_utcb_t *utcb) L4_NOTHROW
00270 {
00271     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00272     v->mr[0] = L4_PLATFORM_CTL_CPU_DISABLE_OP;
00273     v->mr[1] = phys_id;
00274     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00275                     L4_IPC_NEVER);
00276 }
00277
00278 L4_INLINE l4_msgtag_t
00279 l4_platform_ctl_cpu_enable(l4_cap_idx_t pfc,
00280                           l4_umword_t phys_id) L4_NOTHROW
00281 {
00282     return l4_platform_ctl_cpu_enable_u(pfc, phys_id, l4_utcb());
00283 }
00284
00285 L4_INLINE l4_msgtag_t
00286 l4_platform_ctl_cpu_disable(l4_cap_idx_t pfc,
00287                             l4_umword_t phys_id) L4_NOTHROW
00288 {
00289     return l4_platform_ctl_cpu_disable_u(pfc, phys_id, l4_utcb());
00290 }

```

16.522 l4/sys/rcv_endpoint File Reference

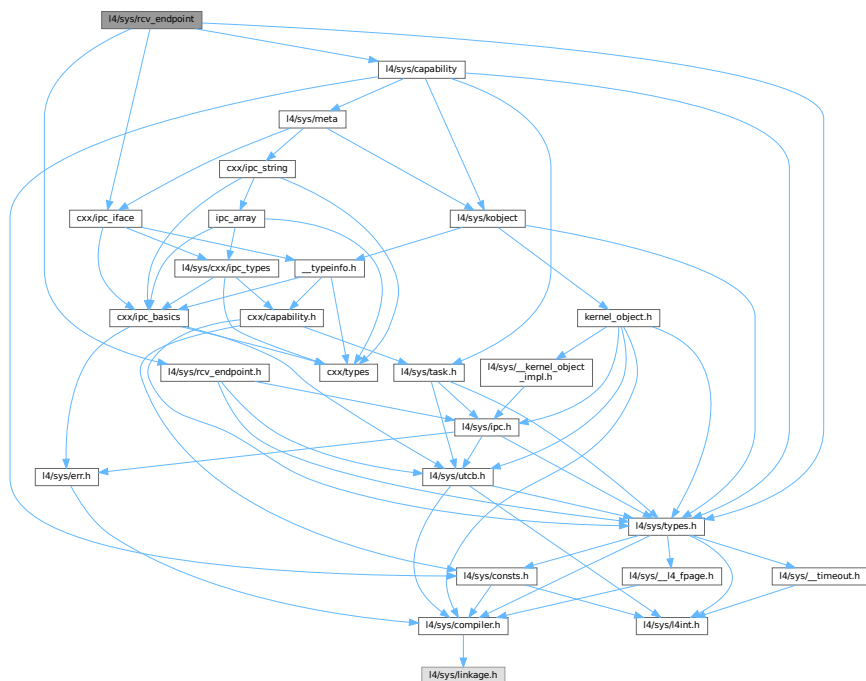
The C++ Receive endpoint interface.

```

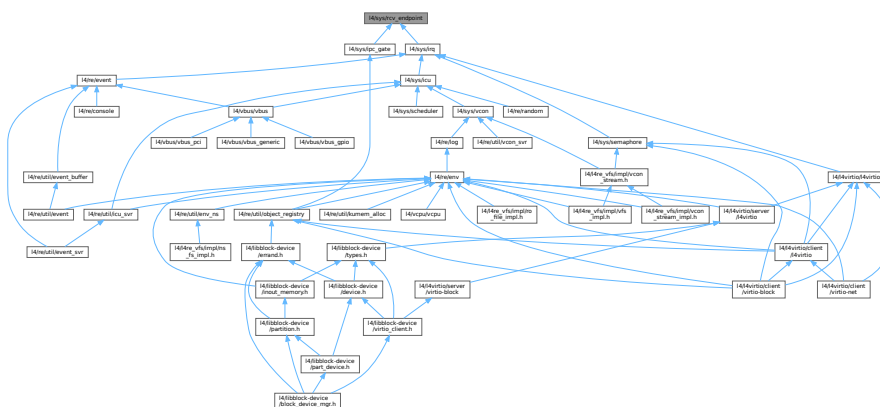
#include <l4/sys/rcv_endpoint.h>
#include <l4/sys/types.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>

```


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.

16.522.1 Detailed Description

The C++ Receive endpoint interface.

Definition in file [rcv_endpoint](#).

16.523 rcv_endpoint

[Go to the documentation of this file.](#)

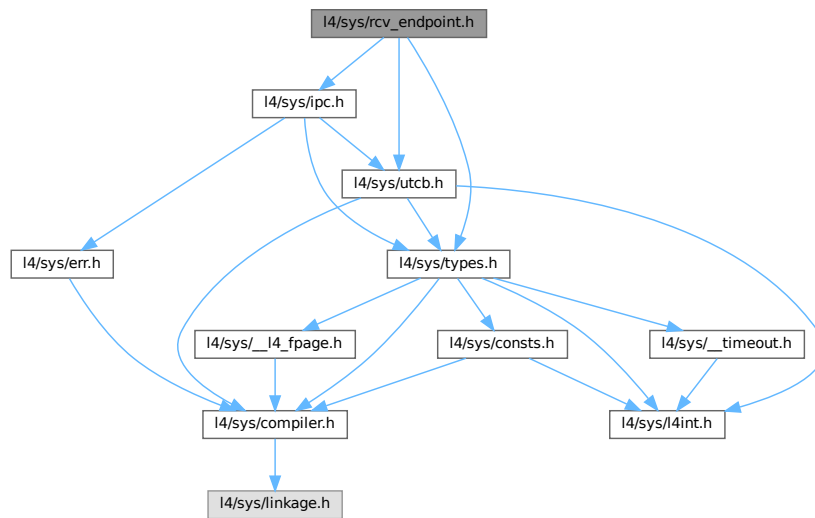
```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
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_sys<bind_thread_t> Rpcsys;
00076 };
00077
00078 }
```

16.524 l4/sys/rcv_endpoint.h File Reference

Receive endpoint C interface.

```
#include <l4/sys/utcb.h>
#include <l4/sys/types.h>
```

Include dependency graph for rcv_endpoint.h:

[illegible]

- enum L4_rcv_ep_ops { L4_RCV_EP_BIND_OP = 0x10 }

Functions

- Bind the IPC receive endpoint to a thread.*

16.524.1 Detailed Description

Receive endpoint C interface.

Definition in file [rcv_endpoint.h](#).

16.524.2 Enumeration Type Documentation

16.524.2.1 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](#).

16.525 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,

```

16.526 I4/sys/scheduler File Reference

```
#include <l4/sys/icu>
#include <l4/sys/scheduler.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>
Include dependency graph for scheduler:
```



- C++ interface of the [Scheduler](#) kernel object, see [Scheduler](#) for the C interface.

Namespaces

- namespace [L4](#)
[L4](#) *low-level kernel interface.*

16.526.1 Detailed Description

Scheduler object functions.

Definition in file [scheduler](#).

16.527 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<1> >
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) *cpu_max = max;
00094         if (sched_classes) *sched_classes = sc;
00095         return t;
00096     }
00097
00121     L4_INLINE_RPC_OP(L4_SCHEDULER_RUN_THREAD_OP,
00122         l4_msgtag_t, run_thread, (Ipc::Cap<Thread> thread, l4_sched_param_t const &sp));
00123
00150     L4_INLINE_RPC_OP(L4_SCHEDULER_IDLE_TIME_OP,
00151         l4_msgtag_t, idle_time, (l4_sched_cpu_set_t const &cpus,
00152             l4_kernel_clock_t *us));
00153
00163     bool is_online(l4_umword_t cpu, l4_utcb_t *utcb = l4_utcb()) const noexcept
00164     { return l4_scheduler_is_online_u(cap(), cpu, utcb); }
00165
00166     typedef L4::Typeid::Rpc_sys<info_t, run_thread_t, idle_time_t> Rpc_sys;
00167 };
00168 }
```

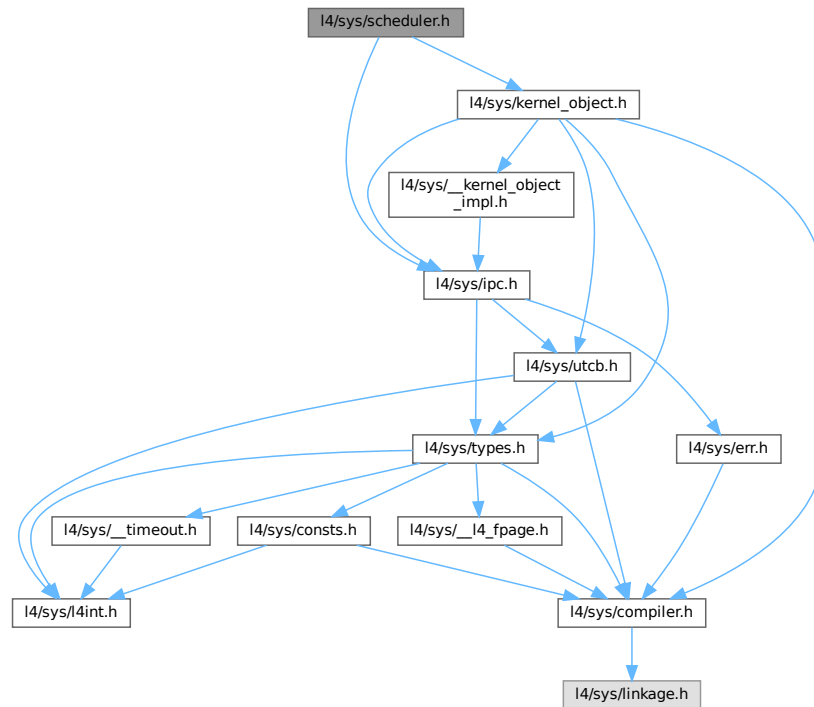
16.528 l4/sys/scheduler.h File Reference

Scheduler object functions.

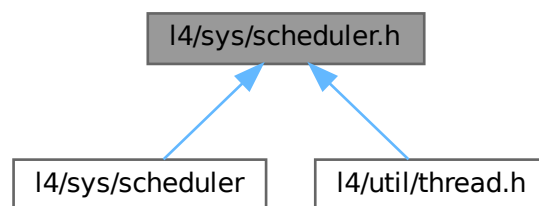
```
#include <l4/sys/kernel_object.h>
```

```
#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 }
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 μ 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.

16.528.1 Detailed Description

Scheduler object functions.

Definition in file [scheduler.h](#).

16.529 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     { gran_offset = ((l4_umword_t)granularity « 24) | (offset & 0x00ffffff); }
00103 #endif
00104 } l4_sched_cpu_set_t;
00105
00116 L4_INLINE l4_sched_cpu_set_t
00117 l4_sched_cpu_set(l4_umword_t offset, unsigned char granularity,
00118                 l4_umword_t map L4_DEFAULT_PARAM(1)) L4_NOTHROW;
00119
00136 L4_INLINE l4_msgtag_t
00137 l4_scheduler_info(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00138                 l4_sched_cpu_set_t *cpus) L4_NOTHROW;
00139
00161 L4_INLINE l4_msgtag_t
00162 l4_scheduler_info_with_classes(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00163                               l4_sched_cpu_set_t *cpus,
00164                               l4_umword_t *sched_classes) L4_NOTHROW;
00165
00169 L4_INLINE l4_msgtag_t
00170 l4_scheduler_info_u(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00171                   l4_sched_cpu_set_t *cpus, l4_umword_t *sched_classes,
00172                   l4_utcb_t *utcb) L4_NOTHROW;
00173
00174
00179 typedef struct l4_sched_param_t
00180 {
00181     l4_sched_cpu_set_t affinity;
00182     l4_umword_t prio;
00183     l4_umword_t quantum;
00184 } l4_sched_param_t;
00185
00192 L4_INLINE l4_sched_param_t
00193 l4_sched_param(unsigned prio,
00194               l4_umword_t quantum L4_DEFAULT_PARAM(0)) L4_NOTHROW;
00195
00203 L4_INLINE l4_msgtag_t
00204 l4_scheduler_run_thread(l4_cap_idx_t scheduler,
00205                       l4_cap_idx_t thread, l4_sched_param_t const *sp) L4_NOTHROW;
00206
00210 L4_INLINE l4_msgtag_t
00211 l4_scheduler_run_thread_u(l4_cap_idx_t scheduler, l4_cap_idx_t thread,
00212                          l4_sched_param_t const *sp, l4_utcb_t *utcb) L4_NOTHROW;
00213

```

```

00221 L4_INLINE l4_msgtag_t
00222 l4_scheduler_idle_time(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00223                        l4_kernel_clock_t *us) L4_NOTHROW;
00224
00228 L4_INLINE l4_msgtag_t
00229 l4_scheduler_idle_time_u(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00230                          l4_kernel_clock_t *us, l4_utcb_t *utcb) L4_NOTHROW;
00231
00232
00233
00244 L4_INLINE int
00245 l4_scheduler_is_online(l4_cap_idx_t scheduler, l4_umword_t cpu) L4_NOTHROW;
00246
00250 L4_INLINE int
00251 l4_scheduler_is_online_u(l4_cap_idx_t scheduler, l4_umword_t cpu,
00252                          l4_utcb_t *utcb) L4_NOTHROW;
00253
00254
00255
00262 enum L4_scheduler_ops
00263 {
00264     L4_SCHEDULER_INFO_OP      = 0UL,
00265     L4_SCHEDULER_RUN_THREAD_OP = 1UL,
00266     L4_SCHEDULER_IDLE_TIME_OP = 2UL,
00267 };
00268
00269 /***** Implementations *****/
00270
00271 L4_INLINE l4_sched_cpu_set_t
00272 l4_sched_cpu_set(l4_umword_t offset, unsigned char granularity,
00273                  l4_umword_t map) L4_NOTHROW
00274 {
00275     l4_sched_cpu_set_t cs;
00276     cs.gran_offset = ((l4_umword_t)granularity << 24) | (offset & 0x00ffffff);
00277     cs.map         = map;
00278     return cs;
00279 }
00280
00281 L4_INLINE l4_sched_param_t
00282 l4_sched_param(unsigned prio, l4_umword_t quantum) L4_NOTHROW
00283 {
00284     l4_sched_param_t sp;
00285     sp.prio          = prio;
00286     sp.quantum       = quantum;
00287     sp.affinity      = l4_sched_cpu_set(0, ~0, 1);
00288     return sp;
00289 }
00290
00291
00292 L4_INLINE l4_msgtag_t
00293 l4_scheduler_info_u(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00294                    l4_sched_cpu_set_t *cpus, l4_umword_t *sched_classes,
00295                    l4_utcb_t *utcb) L4_NOTHROW
00296 {
00297     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00298     l4_msgtag_t res;
00299
00300     m->mr[0] = L4_SCHEDULER_INFO_OP;
00301     m->mr[1] = cpus->gran_offset;
00302
00303     res = l4_ipc_call(scheduler, utcb, l4_msgtag(L4_PROTO_SCHEDULER, 2, 0, 0), L4_IPC_NEVER);
00304
00305     if (l4_msgtag_has_error(res))
00306         return res;
00307
00308     cpus->map = m->mr[0];
00309
00310     if (cpu_max)
00311         *cpu_max = m->mr[1];
00312
00313     if (sched_classes)
00314         *sched_classes = m->mr[2];
00315
00316     return res;
00317 }
00318
00319 L4_INLINE l4_msgtag_t
00320 l4_scheduler_run_thread_u(l4_cap_idx_t scheduler, l4_cap_idx_t thread,
00321                           l4_sched_param_t const *sp, l4_utcb_t *utcb) L4_NOTHROW
00322 {
00323     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00324     m->mr[0] = L4_SCHEDULER_RUN_THREAD_OP;
00325     m->mr[1] = sp->affinity.gran_offset;
00326     m->mr[2] = sp->affinity.map;
00327     m->mr[3] = sp->prio;
00328     m->mr[4] = sp->quantum;
00329     m->mr[5] = l4_map_obj_control(0, 0);

```

```

00330 m->mr[6] = l4_obj_fpage(thread, 0, L4_CAP_FPAGE_RWS).raw;
00331
00332 return l4_ipc_call(scheduler, utcb, l4_msgtag(L4_PROTO_SCHEDULER, 5, 1, 0), L4_IPC_NEVER);
00333 }
00334
00335 L4_INLINE l4_msgtag_t
00336 l4_scheduler_idle_time_u(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00337                         l4_kernel_clock_t *us, l4_utcb_t *utcb) L4_NOTHROW
00338 {
00339     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00340     l4_msgtag_t res;
00341
00342     v->mr[0] = L4_SCHEDULER_IDLE_TIME_OP;
00343     v->mr[1] = cpus->gran_offset;
00344     v->mr[2] = cpus->map;
00345
00346     res = l4_ipc_call(scheduler, utcb,
00347                      l4_msgtag(L4_PROTO_SCHEDULER, 3, 0, 0), L4_IPC_NEVER);
00348
00349     if (l4_msgtag_has_error(res))
00350         return res;
00351
00352     *us = v->mr64[l4_utcb_mr64_idx(0)];
00353
00354     return res;
00355 }
00356
00357
00358 L4_INLINE int
00359 l4_scheduler_is_online_u(l4_cap_idx_t scheduler, l4_umword_t cpu,
00360                         l4_utcb_t *utcb) L4_NOTHROW
00361 {
00362     l4_sched_cpu_set_t s;
00363     l4_msgtag_t r;
00364     s.gran_offset = cpu;
00365     r = l4_scheduler_info_u(scheduler, NULL, &s, NULL, utcb);
00366     if (l4_msgtag_has_error(r) || l4_msgtag_label(r) < 0)
00367         return 0;
00368
00369     return s.map & 1;
00370 }
00371
00372
00373 L4_INLINE l4_msgtag_t
00374 l4_scheduler_info(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00375                  l4_sched_cpu_set_t *cpus) L4_NOTHROW
00376 {
00377     return l4_scheduler_info_u(scheduler, cpu_max, cpus, NULL, l4_utcb());
00378 }
00379
00380 L4_INLINE l4_msgtag_t
00381 l4_scheduler_info_with_classes(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00382                               l4_sched_cpu_set_t *cpus,
00383                               l4_umword_t *sched_classes) L4_NOTHROW
00384 {
00385     return l4_scheduler_info_u(scheduler, cpu_max, cpus, sched_classes, l4_utcb());
00386 }
00387
00388 L4_INLINE l4_msgtag_t
00389 l4_scheduler_run_thread(l4_cap_idx_t scheduler,
00390                       l4_cap_idx_t thread, l4_sched_param_t const *sp) L4_NOTHROW
00391 {
00392     return l4_scheduler_run_thread_u(scheduler, thread, sp, l4_utcb());
00393 }
00394
00395 L4_INLINE l4_msgtag_t
00396 l4_scheduler_idle_time(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00397                       l4_kernel_clock_t *us) L4_NOTHROW
00398 {
00399     return l4_scheduler_idle_time_u(scheduler, cpus, us, l4_utcb());
00400 }
00401
00402 L4_INLINE int
00403 l4_scheduler_is_online(l4_cap_idx_t scheduler, l4_umword_t cpu) L4_NOTHROW
00404 {
00405     return l4_scheduler_is_online_u(scheduler, cpu, l4_utcb());
00406 }

```

16.530 l4/sys/semaphore File Reference

Semaphore class definition.

16.530.1 Detailed Description

Semaphore class definition.

Definition in file [semaphore](#).

16.531 semaphore

[Go to the documentation of this file.](#)

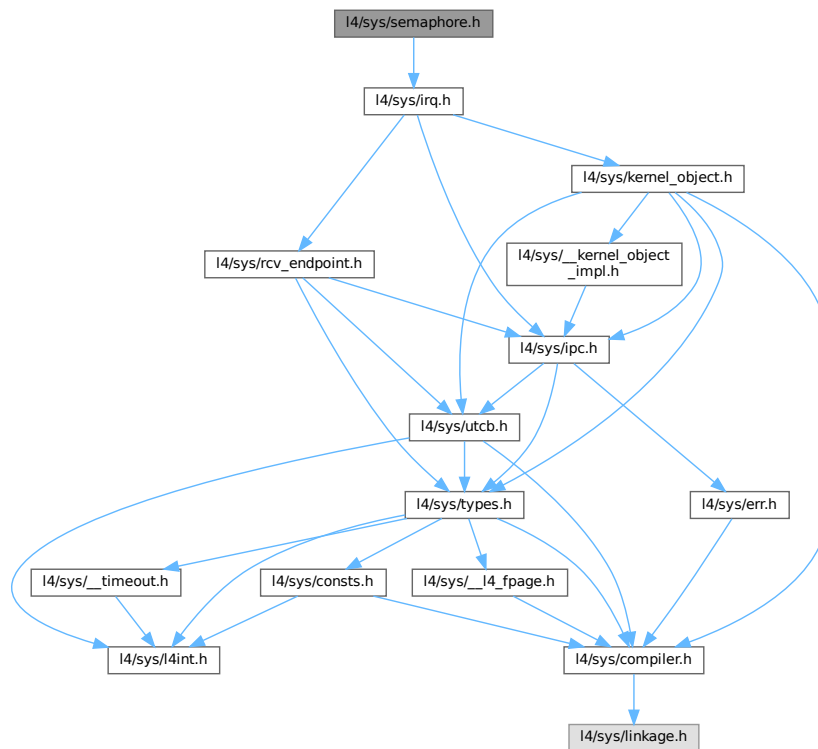
```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2015 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
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
00052 struct Semaphore : Kobject_t<Semaphore, Triggerable, L4_PROTO_SEMAPHORE>
00053 {
00068     l4_msgtag_t up(l4_utcb_t *utcb = l4_utcb()) noexcept
00069     { return trigger(utcb); }
00070
00088     l4_msgtag_t down(l4_timeout_t timeout = L4_IPC_NEVER,
00089                     l4_utcb_t *utcb = l4_utcb()) noexcept
00090     { return l4_semaphore_down_u(cap(), timeout, utcb); }
00091 };
00092
00093 }
```

16.532 l4/sys/semaphore.h File Reference

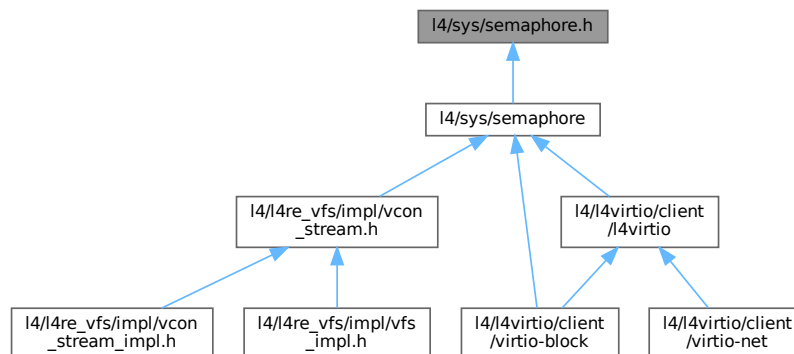
C semaphore interface.

```
#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.

16.532.1 Detailed Description

C semaphore interface.

Definition in file [semaphore.h](#).

16.533 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

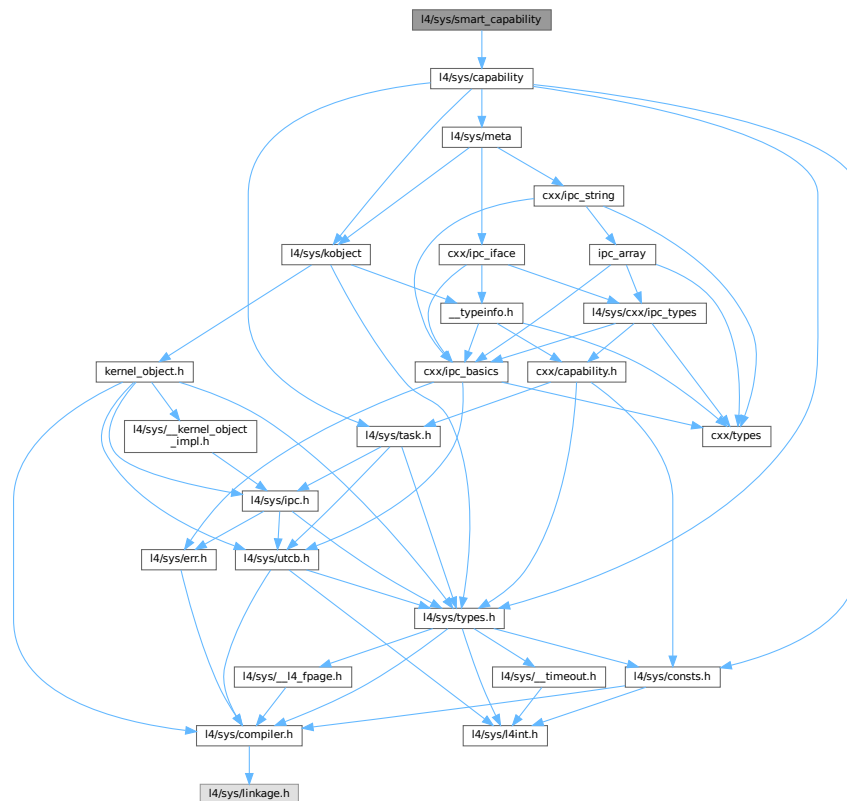
```

16.534 I4/sys/smart_capability File Reference

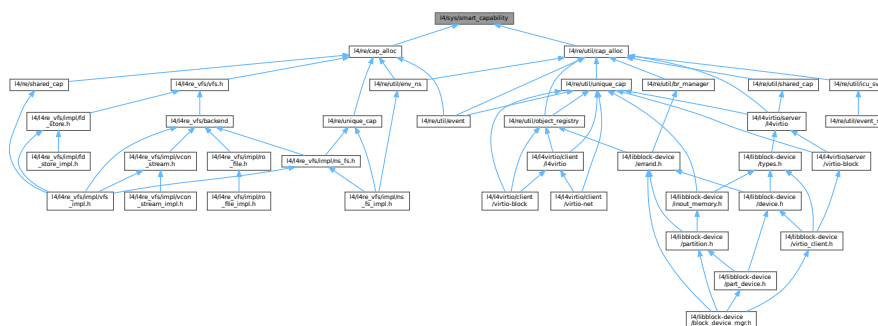
L4::Capability class.

```
#include <l4/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.

16.534.1 Detailed Description

L4::Capability class.

Author

Alexander Warg alexander.warg@os.inf.tu-dresden.de

Definition in file [smart_capability](#).

16.535 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 }
```

```

00047   Cap<T> release() const noexcept
00048   {
00049       l4_cap_idx_t r = cap();
00050       SMART::invalidate(const_cast<Smart_cap<T, SMART>&>(*this));
00051
00052       return Cap<T>(r);
00053   }
00054
00055   void reset() noexcept
00056   {
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   { T* __t = ((O*)100); (void)__t; }
00075
00076   template< typename O >
00077   Smart_cap(Cap<O> const &p, SMART const &smart) noexcept
00078   : Cap_base(p.cap()), SMART(smart)
00079   { T* __t = ((O*)100); (void)__t; }
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   { T* __t = ((O*)100); (void)__t; }
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   { T* __t = ((O*)100); (void)__t; }
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   { T* __t = ((O*)100); (void)__t; }
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     { T* __t = ((O*)100); (void)__t; }
00170
00171     template< typename O, typename S >
00172     Weak_cap(Smart_cap<O, S> const &c) noexcept : Cap_base(c.cap())
00173     { T* __t = ((O*)100); (void)__t; }
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     { T* __t = ((O*)100); (void)__t; }
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     (void)static_cast<T const *>(reinterpret_cast<F const *>(100));
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 }
00229

```

16.536 l4/sys/task File Reference

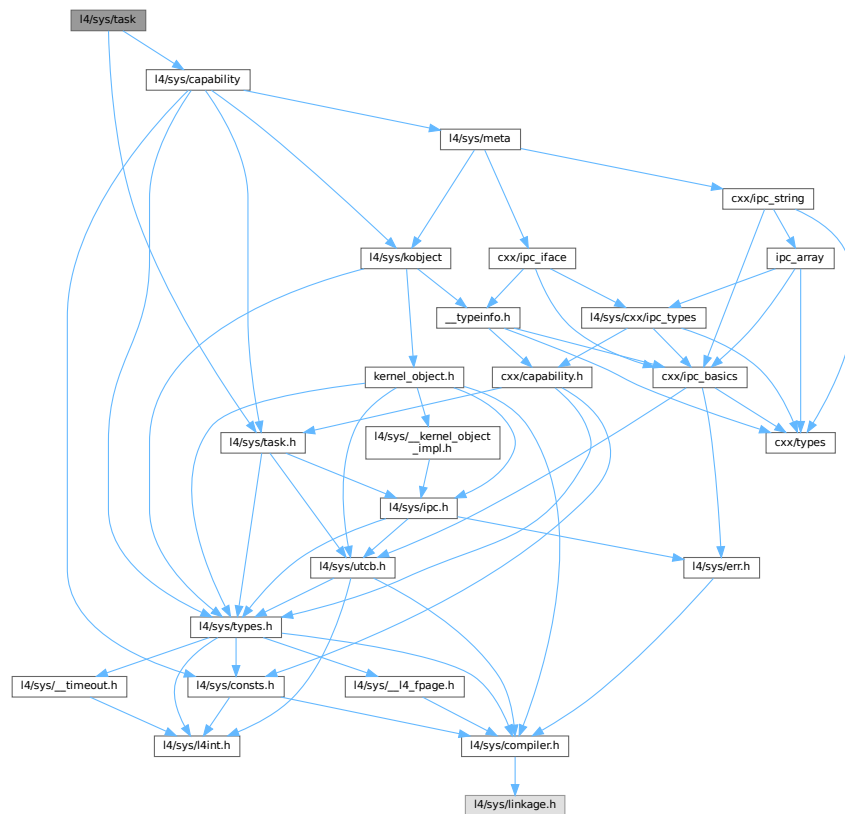
Common task related definitions.

```

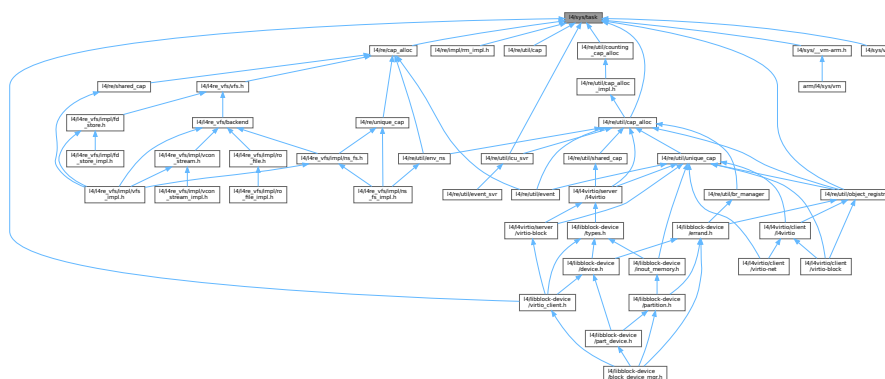
#include <l4/sys/task.h>
#include <l4/sys/capability>

```

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.

16.536.1 Detailed Description

Common task related definitions.

Definition in file [task](#).

16.537 task

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=c++: -- 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<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
00253     l4_msgtag_t add_ku_mem(l4_fpage_t const &fpage,
00254         l4_utcb_t *utcb = l4_utcb()) noexcept
00255     { return l4_task_add_ku_mem_u(cap(), fpage, utcb); }
00256
00257 };
00258 }
00259
00260
```

16.538 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>

```

16.539 l4/sys/task.h File Reference

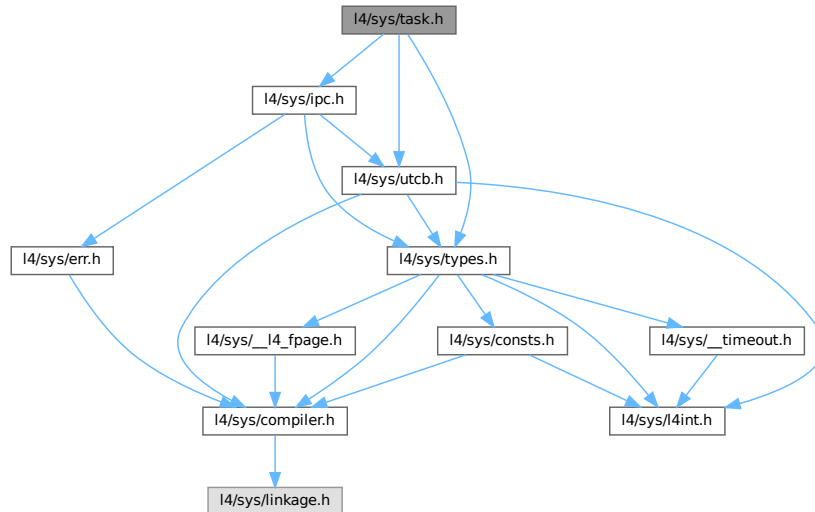
Common task related definitions.

```

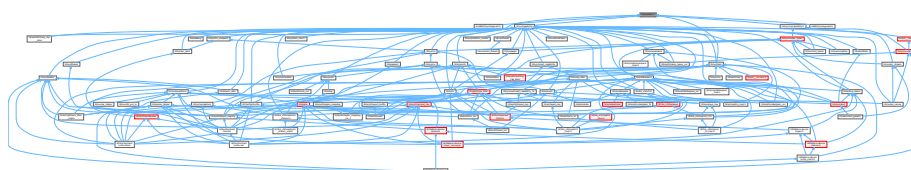
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>

```

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.

16.539.1 Detailed Description

Common task related definitions.

Definition in file [task.h](#).

16.540 task.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  * 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
```

```

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
00295  L4_INLINE l4_msgtag_t
00296  l4_task_add_ku_mem(l4_cap_idx_t task, l4_fpage_t ku_mem) L4_NOTHROW;
00297
00298
00302  L4_INLINE l4_msgtag_t
00303  l4_task_cap_equal_u(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00304                     l4_cap_idx_t cap_b, l4_utcb_t *utcb) L4_NOTHROW;
00305
00310  enum L4_task_ops
00311  {
00312      L4_TASK_MAP_OP          = 0UL,
00313      L4_TASK_UNMAP_OP       = 1UL,
00314      L4_TASK_CAP_INFO_OP    = 2UL,
00315      L4_TASK_ADD_KU_MEM_OP   = 3UL,
00316      L4_TASK_LDT_SET_X86_OP  = 0x11UL,
00317      L4_TASK_MAP_VGICC_ARM_OP = 0x12UL,
00318  };
00319
00320
00321  /* IMPLEMENTATION ----- */
00322
00323  #include <l4/sys/ipc.h>
00324
00325
00326  L4_INLINE l4_msgtag_t
00327  l4_task_map_u(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,

```



```

00328         l4_fpage_t snd_fpage, l4_umword_t snd_base, l4_utcb_t *u) L4_NOTHROW
00329 {
00330     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00331     v->mr[0] = L4_TASK_MAP_OP;
00332     v->mr[3] = l4_map_obj_control(0,0);
00333     v->mr[4] = l4_obj_fpage(src_task, 0, L4_CAP_FPAGE_RWS).raw;
00334     v->mr[1] = snd_base;
00335     v->mr[2] = snd_fpage.raw;
00336     return l4_ipc_call(dst_task, u, l4_msgtag(L4_PROTO_TASK, 3, 1, 0), L4_IPC_NEVER);
00337 }
00338
00339 L4_INLINE l4_msgtag_t
00340 l4_task_unmap_u(l4_cap_idx_t task, l4_fpage_t fpage,
00341                l4_umword_t map_mask, l4_utcb_t *u) L4_NOTHROW
00342 {
00343     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00344     v->mr[0] = L4_TASK_UNMAP_OP;
00345     v->mr[1] = map_mask;
00346     v->mr[2] = fpage.raw;
00347     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 3, 0, 0), L4_IPC_NEVER);
00348 }
00349
00350 L4_INLINE l4_msgtag_t
00351 l4_task_unmap_batch_u(l4_cap_idx_t task, l4_fpage_t const *fpages,
00352                      unsigned num_fpages, l4_umword_t map_mask,
00353                      l4_utcb_t *u) L4_NOTHROW
00354 {
00355     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00356     v->mr[0] = L4_TASK_UNMAP_OP;
00357     v->mr[1] = map_mask;
00358     __builtin_memcpy(&v->mr[2], fpages, num_fpages * sizeof(l4_fpage_t));
00359     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2 + num_fpages, 0, 0), L4_IPC_NEVER);
00360 }
00361
00362 L4_INLINE l4_msgtag_t
00363 l4_task_cap_valid_u(l4_cap_idx_t task, l4_cap_idx_t cap, l4_utcb_t *u) L4_NOTHROW
00364 {
00365     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00366     v->mr[0] = L4_TASK_CAP_INFO_OP;
00367     v->mr[1] = cap & ~1UL;
00368     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0), L4_IPC_NEVER);
00369 }
00370
00371 L4_INLINE l4_msgtag_t
00372 l4_task_cap_equal_u(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00373                    l4_cap_idx_t cap_b, l4_utcb_t *u) L4_NOTHROW
00374 {
00375     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00376     v->mr[0] = L4_TASK_CAP_INFO_OP;
00377     v->mr[1] = cap_a;
00378     v->mr[2] = cap_b;
00379     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 3, 0, 0), L4_IPC_NEVER);
00380 }
00381
00382 L4_INLINE l4_msgtag_t
00383 l4_task_add_ku_mem_u(l4_cap_idx_t task, l4_fpage_t ku_mem,
00384                     l4_utcb_t *u) L4_NOTHROW
00385 {
00386     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00387     v->mr[0] = L4_TASK_ADD_KU_MEM_OP;
00388     v->mr[1] = ku_mem.raw;
00389     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0), L4_IPC_NEVER);
00390 }
00391
00392
00393
00394 L4_INLINE l4_msgtag_t
00395 l4_task_map(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00396             l4_fpage_t snd_fpage, l4_umword_t snd_base) L4_NOTHROW
00397 {
00398     return l4_task_map_u(dst_task, src_task, snd_fpage, snd_base, l4_utcb());
00399 }
00400
00401 L4_INLINE l4_msgtag_t
00402 l4_task_unmap(l4_cap_idx_t task, l4_fpage_t fpage,
00403               l4_umword_t map_mask) L4_NOTHROW
00404 {
00405     return l4_task_unmap_u(task, fpage, map_mask, l4_utcb());
00406 }
00407
00408 L4_INLINE l4_msgtag_t
00409 l4_task_unmap_batch(l4_cap_idx_t task, l4_fpage_t const *fpages,
00410                    unsigned num_fpages, l4_umword_t map_mask) L4_NOTHROW
00411 {
00412     return l4_task_unmap_batch_u(task, fpages, num_fpages, map_mask,
00413                                  l4_utcb());
00414 }

```

```

00415
00416 L4_INLINE l4_msgtag_t
00417 l4_task_delete_obj_u(l4_cap_idx_t task, l4_cap_idx_t obj,
00418                      l4_utcb_t *u) L4_NOTHROW
00419 {
00420     return l4_task_unmap_u(task, l4_obj_fpage(obj, 0, L4_CAP_FPAGE_RWSD),
00421                            L4_FP_DELETE_OBJ, u);
00422 }
00423
00424 L4_INLINE l4_msgtag_t
00425 l4_task_delete_obj(l4_cap_idx_t task, l4_cap_idx_t obj) L4_NOTHROW
00426 {
00427     return l4_task_delete_obj_u(task, obj, l4_utcb());
00428 }
00429
00430
00431 L4_INLINE l4_msgtag_t
00432 l4_task_release_cap_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00433                      l4_utcb_t *u) L4_NOTHROW
00434 {
00435     return l4_task_unmap_u(task, l4_obj_fpage(cap, 0, L4_CAP_FPAGE_RWSD),
00436                            L4_FP_ALL_SPACES, u);
00437 }
00438
00439 L4_INLINE l4_msgtag_t
00440 l4_task_release_cap(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW
00441 {
00442     return l4_task_release_cap_u(task, cap, l4_utcb());
00443 }
00444
00445 L4_INLINE l4_msgtag_t
00446 l4_task_cap_valid(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW
00447 {
00448     return l4_task_cap_valid_u(task, cap, l4_utcb());
00449 }
00450
00451 L4_INLINE l4_msgtag_t
00452 l4_task_cap_equal(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00453                  l4_cap_idx_t cap_b) L4_NOTHROW
00454 {
00455     return l4_task_cap_equal_u(task, cap_a, cap_b, l4_utcb());
00456 }
00457
00458 L4_INLINE l4_msgtag_t
00459 l4_task_add_ku_mem(l4_cap_idx_t task, l4_fpage_t ku_mem) L4_NOTHROW
00460 {
00461     return l4_task_add_ku_mem_u(task, ku_mem, l4_utcb());
00462 }

```

16.541 l4/cxx/thread File Reference

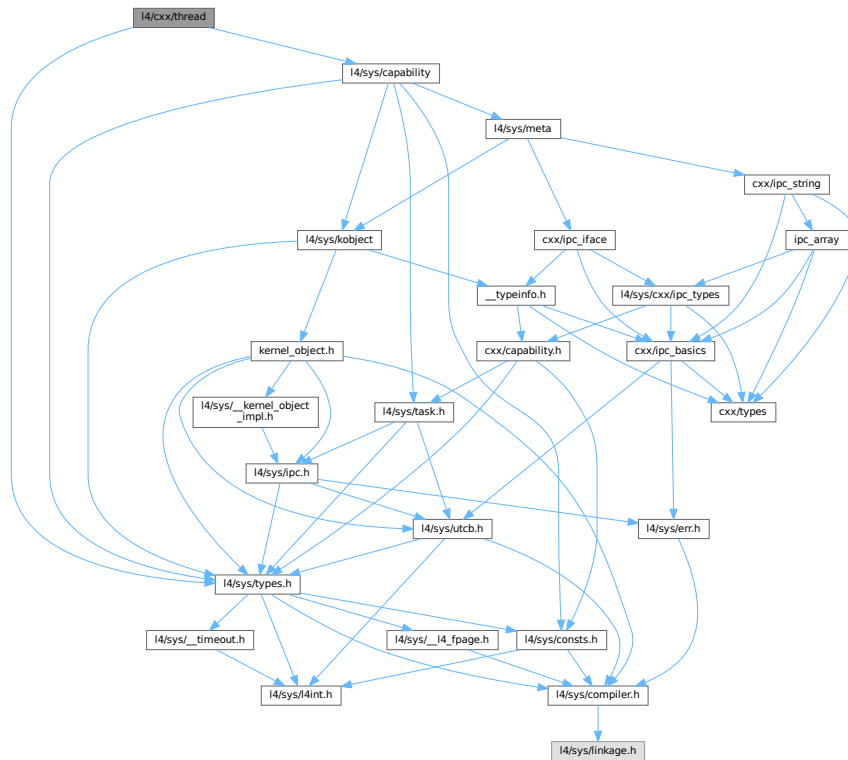
Thread implementation.

```

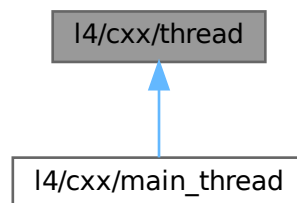
#include <l4/sys/capability>
#include <l4/sys/types.h>

```

Include dependency graph for thread:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [cxx](#)
Our C++ library.

16.541.1 Detailed Description

Thread implementation.

Definition in file [thread](#).

16.542 thread

[Go to the documentation of this file.](#)

```

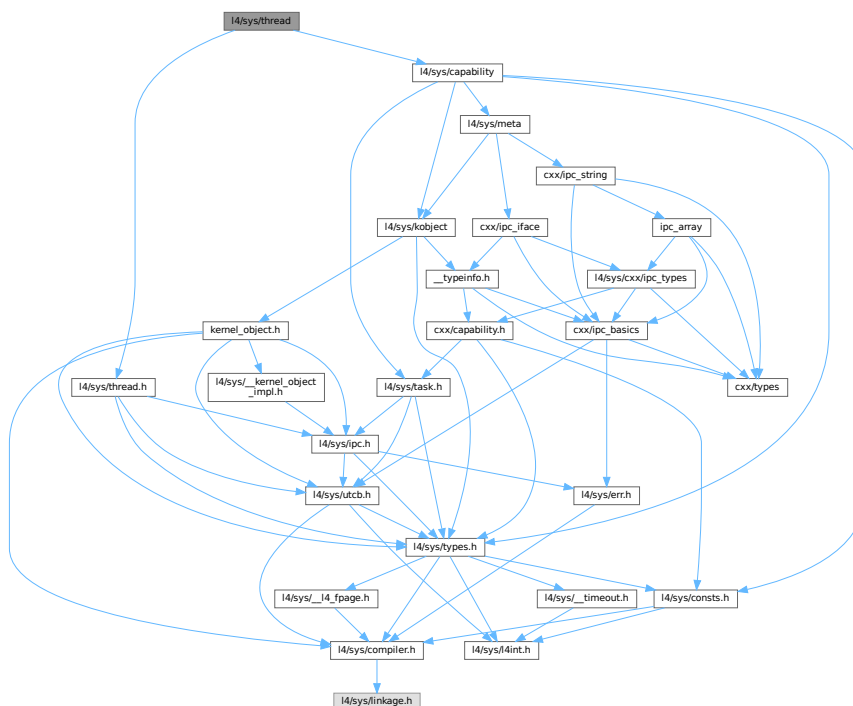
00001
00005 /*
00006  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012
00013 #ifndef CXX_THREAD_H__
00014 #define CXX_THREAD_H__
00015
00016 #include <l4/sys/capability>
00017 #include <l4/sys/types.h>
00018
00019 namespace cxx {
00020
00021     class Thread
00022     {
00023     public:
00024
00025         enum State
00026         {
00027             Dead    = 0,
00028             Running = 1,
00029             Stopped = 2,
00030         };
00031
00032         Thread(bool initiate);
00033         Thread(void *stack);
00034         Thread(void *stack, L4::Cap<L4::Thread> const &cap);
00035         virtual ~Thread();
00036         void execute() asm ("L4_Thread_execute");
00037         virtual void run() = 0;
00038         virtual void shutdown() asm ("L4_Thread_shutdown");
00039         void start();
00040         void stop();
00041
00042         L4::Cap<L4::Thread> self() const throw()
00043         { return _cap; }
00044
00045         State state() const
00046         { return _state; }
00047
00048         static void start_cxx_thread(Thread *_this)
00049             asm ("L4_Thread_start_cxx_thread");
00050
00051         static void kill_cxx_thread(Thread *_this)
00052             asm ("L4_Thread_kill_cxx_thread");
00053
00054         static void set_pager(L4::Cap<void> const &p) throw()
00055         { _pager = p; }
00056
00057     private:
00058         int create();
00059
00060         L4::Cap<L4::Thread> _cap;
00061         State _state;
00062
00063     protected:
00064         void *_stack;
00065
00066     private:
00067         static L4::Cap<void> _pager;
00068         static L4::Cap<void> _master;
00069     };
00070
00071 };
00072
00073 #endif /* CXX_THREAD_H__ */
00074

```

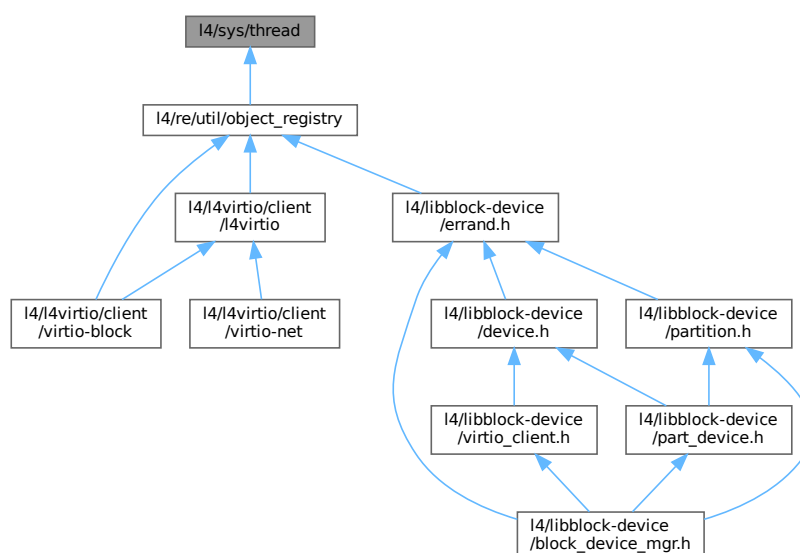
16.543 l4/sys/thread File Reference

Common thread related definitions.

```
#include <linux/capability.h>
#include <linux/thread.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.

16.543.1 Detailed Description

Common thread related definitions.

Definition in file [thread](#).

16.544 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
00498     l4_msgtag_t modify_senders(Modify_senders const &todo) noexcept
00499     {
00500         return l4_ipc_call(cap(), todo.utcb, l4_msgtag(L4_PROTO_THREAD, todo.cnt, 0, 0), L4_IPC_NEVER);

```

```

00501     }
00502 };
00503 }

```

16.545 I4/sys/typeinfo_svr File Reference

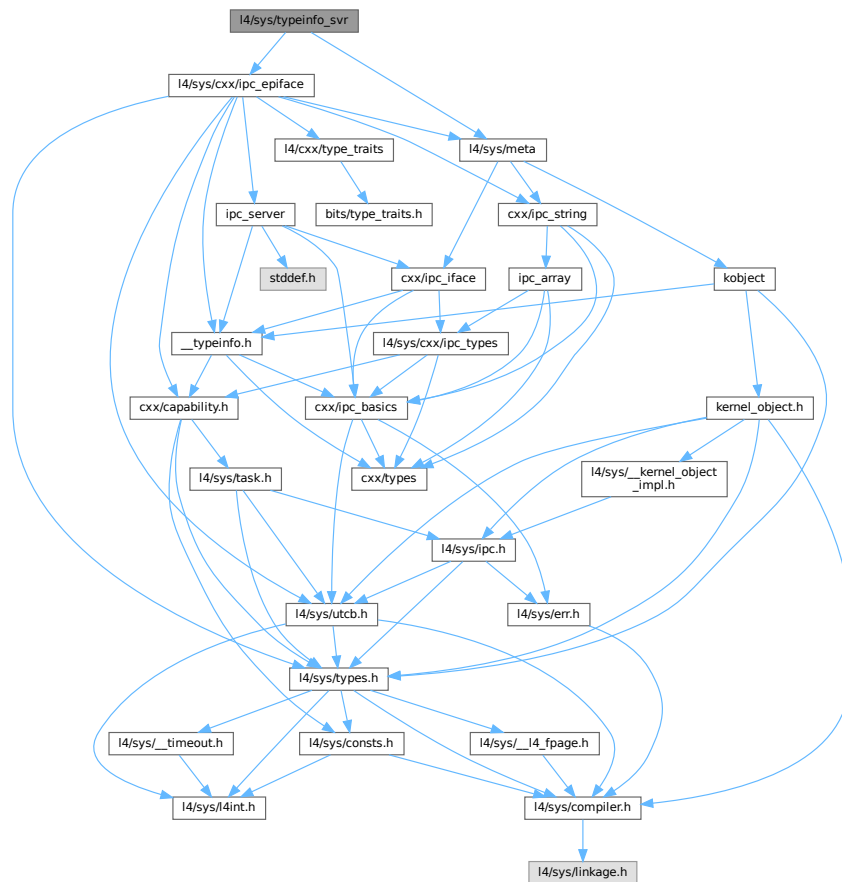
Type information server template.

```

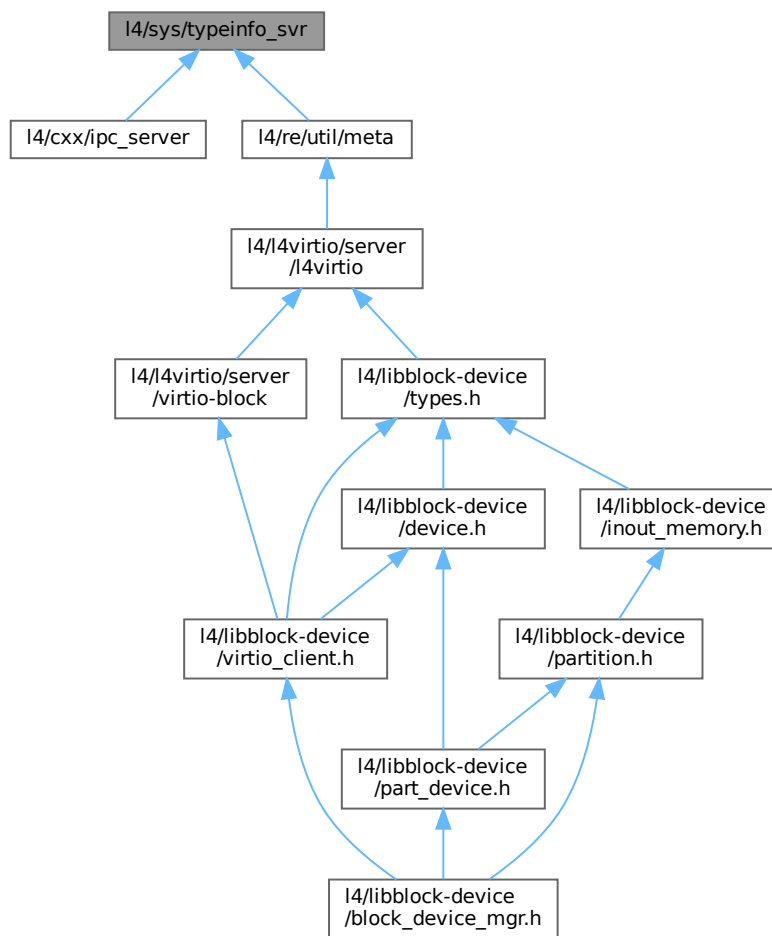
#include <l4/sys/meta>
#include <l4/sys/cxx/ipc_epiface>

```

Include dependency graph for typeinfo_svr:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [L4](#)
L4 low-level kernel interface.

16.545.1 Detailed Description

Type information server template.

Definition in file [typeinfo_svr](#).

16.546 typeinfo_svr

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 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/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     l4_msgtag_t tag = dispatch_call<L4::Meta::Rpcs>((Msvr *)0, ios.utcb(),
00037                                                    ios.tag(), 0);
00038     ios.set_ipc_params(tag);
00039     return tag.label();
00040 }
00041
00042 }}

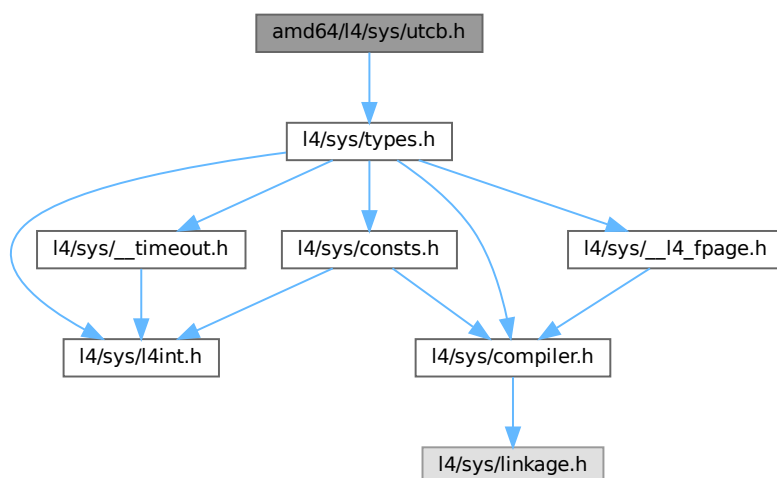
```

16.547 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\(\)](#).

16.547.1 Detailed Description

UTCB definitions for amd64.

Definition in file [utcb.h](#).

16.548 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__ */

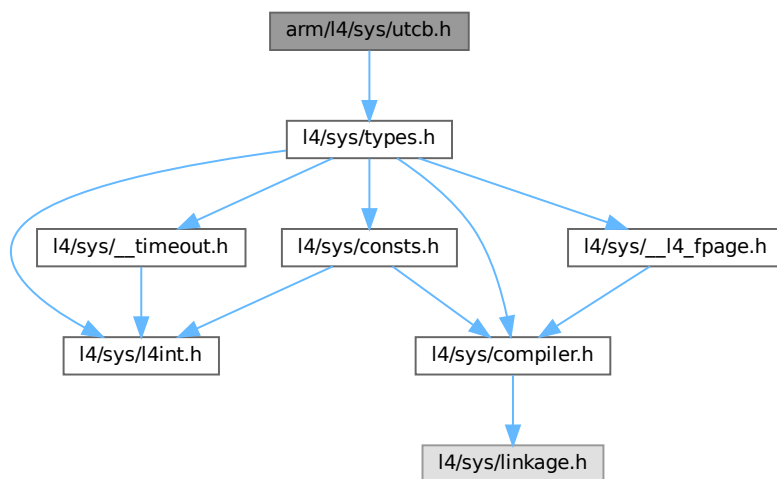
```

16.549 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\(\)](#).

16.549.1 Detailed Description

UTCB definitions for ARM.

Definition in file [utcb.h](#).

16.550 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>
```

```

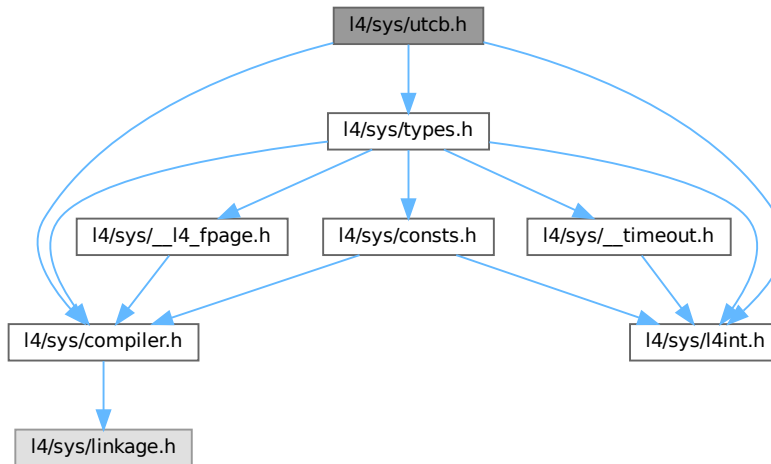
00028
00038 typedef struct l4_exc_regs_t
00039 {
00040     l4_umword_t pfa;
00041     l4_umword_t err;
00043     l4_umword_t r[13];
00044     l4_umword_t sp;
00045     l4_umword_t ulr;
00046     l4_umword_t _dummy1;
00047     l4_umword_t pc;
00048     l4_umword_t cpsr;
00049     l4_umword_t tpidruro;
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     register l4_utcb_t *utcb __asm__ ("r0");
00084     __asm__ ("mov lr, pc\n"
00085             "mvn pc, #0xff\n" // write 0xfffff00 to pc
00086             : "=r"(utcb) : : "lr");
00087     return utcb;
00088 }
00089 #endif
00090
00091 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW
00092 {
00093     return u->pc;
00094 }
00095
00096 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW
00097 {
00098     u->pc = pc;
00099 }
00100
00101 L4_INLINE l4_umword_t l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW
00102 {
00103     return u->err >> 26;
00104 }
00105
00106 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW
00107 {
00108     return ((u->err >> 26) & 0x30) == 0x20;
00109 }
00110
00111 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW
00112 {
00113     return (u->pfa & ~7UL) | ((u->err >> 5) & 2);
00114 }
00115
00116 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW
00117 {
00118     return l4_utcb_exc_typeval(u) == 0x3e;
00119 }
00120
00121 #endif /* ! __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__ */

```

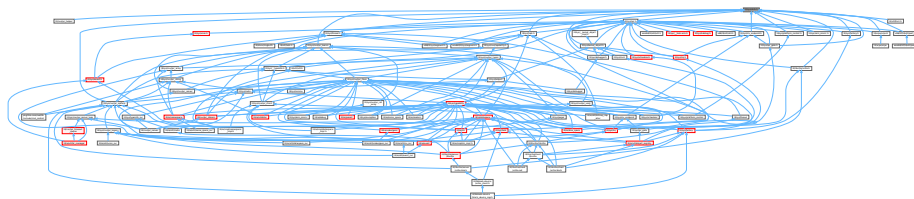
16.551 l4/sys/utcb.h File Reference

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.

16.551.1 Detailed Description

UTCB definitions.

Definition in file [utcb.h](#).

16.552 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_CV 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

```

```

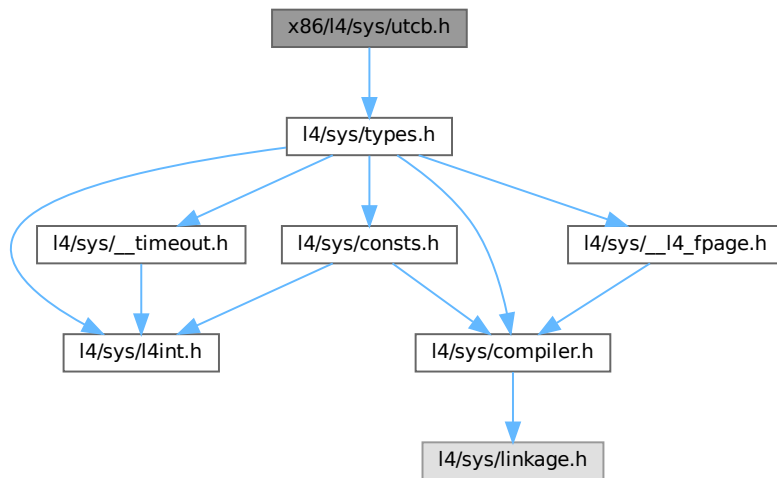
00318  *****/
00319
00320 L4_INLINE l4_msg_regs_t *l4_utcb_mr_u(l4_utcb_t *u) L4_NOTHROW
00321 { return (l4_msg_regs_t*)((char*)u + L4_UTCB_MSG_REGS_OFFSET); }
00322
00323 L4_INLINE l4_buf_regs_t *l4_utcb_br_u(l4_utcb_t *u) L4_NOTHROW
00324 { return (l4_buf_regs_t*)((char*)u + L4_UTCB_BUF_REGS_OFFSET); }
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 */

```

16.553 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()`.

16.553.1 Detailed Description

UTCB definitions for X86.

Definition in file [utcb.h](#).

16.554 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 *      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__INCLUDE__ARCH_X86__UTCB_H__
00027 #define __L4_SYS__INCLUDE__ARCH_X86__UTCB_H__
00028
00029 #include <l4/sys/types.h>
00030
00041 enum l4_utcb_consts_x86
00042 {
00044     L4_UTCB_EXCEPTION_REGS_SIZE    = 19,
00045
00047     L4_UTCB_GENERIC_DATA_SIZE      = 63,
00048
00050     L4_UTCB_GENERIC_BUFFERS_SIZE   = 58,
00051
00053     L4_UTCB_MSG_REGS_OFFSET        = 0,
00054
00056     L4_UTCB_BUF_REGS_OFFSET        = 64 * sizeof(l4_umword_t),
00057
00059     L4_UTCB_THREAD_REGS_OFFSET     = 123 * sizeof(l4_umword_t),
00060
00062     L4_UTCB_INHERIT_FPU            = 1UL << 24,
00063
00065     L4_UTCB_OFFSET                 = 512,
00066 };
00067
00072 typedef struct l4_exc_regs_t
00073 {
00074     l4_umword_t es;
00075     l4_umword_t ds;
00076     l4_umword_t gs;
00077     l4_umword_t fs;
00079     l4_umword_t edi;
00080     l4_umword_t esi;
00081     l4_umword_t ebp;
00082     l4_umword_t pfa;
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__ */

```

16.555 l4/sys/vcon File Reference

C++ Virtual console interface.

```

#include <l4/sys/icu>
#include <l4/sys/vcon.h>
#include <l4/sys/capability>

```

[illegible][illegible]

- class `L4::Vcon`
C++ L4 Vcon interface, see [Virtual Console](#) for the C interface.

- namespace **L4**
L4 low-level kernel interface.

16.555.1 Detailed Description

C++ Virtual console interface.

Definition in file [vcon](#).

16.556 vcon

[Go to the documentation of this file.](#)

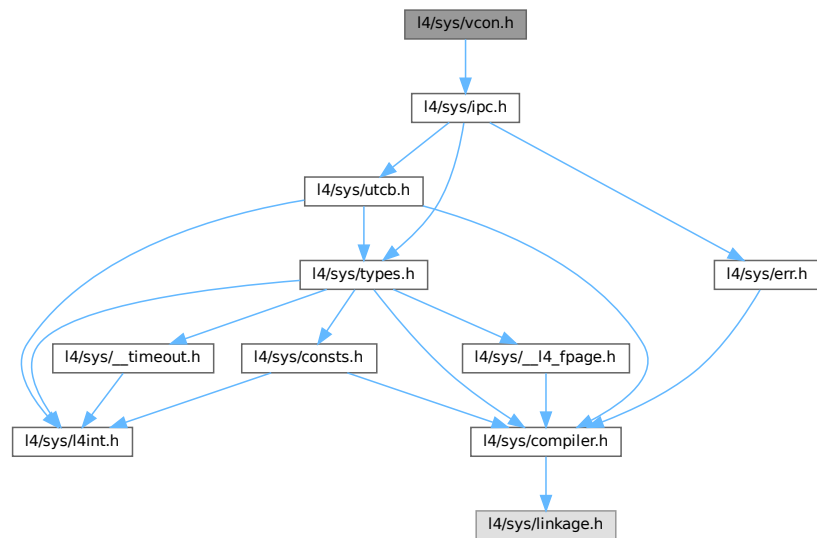
```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (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 }
```

16.557 l4/sys/vcon.h File Reference

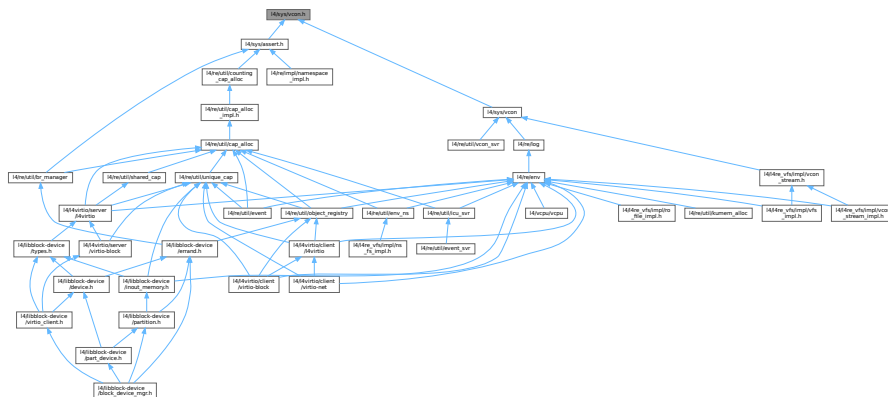
Virtual console interface.


```
#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.

16.557.1 Detailed Description

Virtual console interface.

Definition in file [vcon.h](#).

16.557.2 Enumeration Type Documentation

16.557.2.1 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](#).

16.558 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

```

16.559 l4/sys/vhw.h File Reference

Descriptors for virtual hardware (under UX).

```

#include <l4/sys/types.h>
#include <l4/sys/kip.h>

```


16.560 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 */
```

16.561 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>
```

16.562 l4/sys/vm File Reference

Virtualization interface.

```
#include <l4/sys/vm.h>
#include <l4/sys/task>
```


16.563 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 };
```

16.564 l4/sys/assert.h File Reference

Low-level assert implementation.

```
#include <l4/sys/compiler.h>
#include <l4/sys/thread.h>
#include <l4/sys/vcon.h>
```


16.564.1 Detailed Description

Low-level assert implementation.

Definition in file [assert.h](#).

16.564.2 Macro Definition Documentation

16.564.2.1 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](#).

16.565 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 NDEBUG
00021 #define l4_assert(x) do { } while (0)
00022 #define l4_check(x) do { (void)(x); } while (0)
00023 #else
00024 #include <l4/sys/compiler.h>
00025 #include <l4/sys/thread.h>
00026 #include <l4/sys/vcon.h>
00027 #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(bool 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(bool 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 */

```

16.566 l4/util/assert.h File Reference

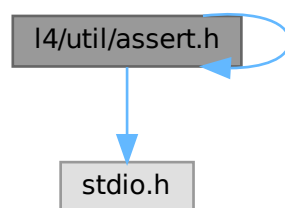
Some useful assert-style macros.

```

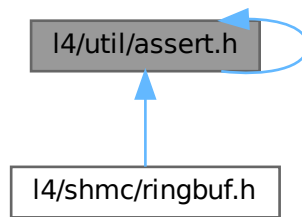
#include <stdio.h>
#include <assert.h>

```

Include dependency graph for assert.h:



This graph shows which files directly or indirectly include this file:



16.566.1 Detailed Description

Some useful assert-style macros.

Date

09/2009

Author

Bjoern Doebel doebel@tudos.org

Definition in file [assert.h](#).

16.567 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

```

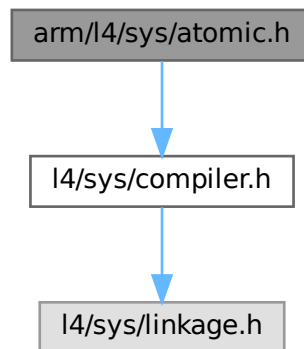


```
00123 #endif // NDEBUG
```

16.568 arm/l4/sys/atomic.h File Reference

Atomic memory modifications.

```
#include <l4/sys/compiler.h>
Include dependency graph for atomic.h:
```



16.568.1 Detailed Description

Atomic memory modifications.

Definition in file [atomic.h](#).

16.569 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;

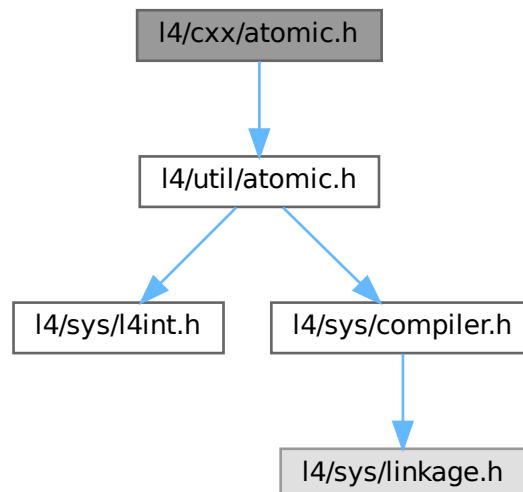
```

16.570 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.

16.570.1 Detailed Description

Atomic template.

Definition in file [atomic.h](#).

16.571 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/util/atomic.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 }

```

16.572 l4/util/atomic.h File Reference

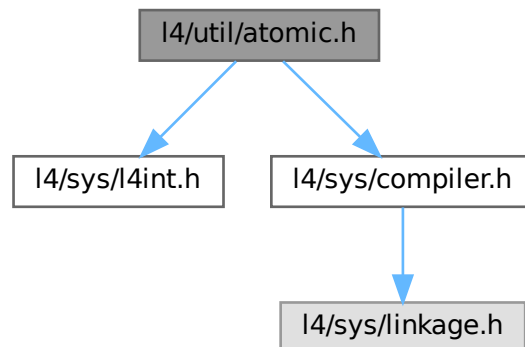
atomic operations header and generic implementations

```

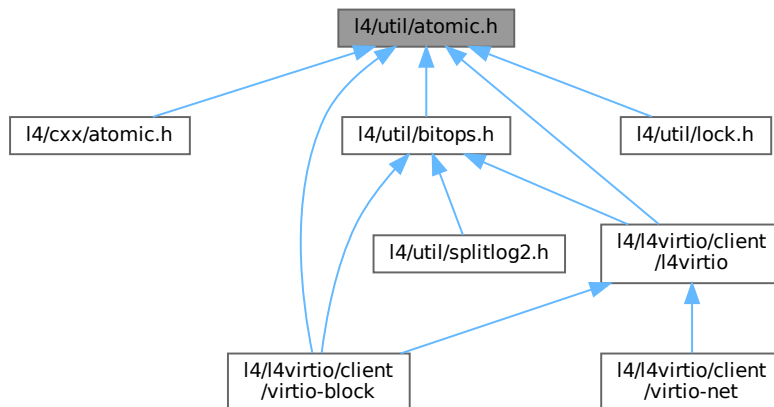
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for atomic.h:



This graph shows which files directly or indirectly include this file:



Functions

- `int l4util_cmpxchg64 (volatile l4_uint64_t *dest, l4_uint64_t cmp_val, l4_uint64_t new_val)`
Atomic compare and exchange (64 bit version)
- `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)

16.572.1 Detailed Description

atomic operations header and generic implementations

Date

10/20/2000

Author

Lars Reuther reuther@os.inf.tu-dresden.de, Jork Loeser jork@os.inf.tu-dresden.de ↵

Definition in file [atomic.h](#).

16.573 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
00049 L4_INLINE int
00050 l4util_cmpxchg64(volatile l4_uint64_t * dest,
00051                 l4_uint64_t cmp_val, l4_uint64_t new_val);
00052
00066 L4_INLINE int
00067 l4util_cmpxchg32(volatile l4_uint32_t * dest,
00068                 l4_uint32_t cmp_val, l4_uint32_t new_val);
00069
00083 L4_INLINE int
00084 l4util_cmpxchg16(volatile l4_uint16_t * dest,
00085                 l4_uint16_t cmp_val, l4_uint16_t new_val);
00086
00100 L4_INLINE int
00101 l4util_cmpxchg8(volatile l4_uint8_t * dest,
00102                 l4_uint8_t cmp_val, l4_uint8_t new_val);
00103
00117 L4_INLINE int
00118 l4util_cmpxchg(volatile l4_umword_t * dest,
00119                l4_umword_t cmp_val, l4_umword_t new_val);
00120
00130 L4_INLINE l4_uint32_t
00131 l4util_xchg32(volatile l4_uint32_t * dest, l4_uint32_t val);
00132
00142 L4_INLINE l4_uint16_t
00143 l4util_xchg16(volatile l4_uint16_t * dest, l4_uint16_t val);
00144
00154 L4_INLINE l4_uint8_t
00155 l4util_xchg8(volatile l4_uint8_t * dest, l4_uint8_t val);
00156
00166 L4_INLINE l4_umword_t
00167 l4util_xchg(volatile l4_umword_t * dest, l4_umword_t val);
00168
```

```
00170
00176 L4_INLINE void
00177 l4util_add8(volatile l4_uint8_t *dest, l4_uint8_t val);
00179 L4_INLINE void
00180 l4util_add16(volatile l4_uint16_t *dest, l4_uint16_t val);
00182 L4_INLINE void
00183 l4util_add32(volatile l4_uint32_t *dest, l4_uint32_t val);
00185 L4_INLINE void
00186 l4util_sub8(volatile l4_uint8_t *dest, l4_uint8_t val);
00188 L4_INLINE void
00189 l4util_sub16(volatile l4_uint16_t *dest, l4_uint16_t val);
00191 L4_INLINE void
00192 l4util_sub32(volatile l4_uint32_t *dest, l4_uint32_t val);
00194 L4_INLINE void
00195 l4util_and8(volatile l4_uint8_t *dest, l4_uint8_t val);
00197 L4_INLINE void
00198 l4util_and16(volatile l4_uint16_t *dest, l4_uint16_t val);
00200 L4_INLINE void
00201 l4util_and32(volatile l4_uint32_t *dest, l4_uint32_t val);
00203 L4_INLINE void
00204 l4util_or8(volatile l4_uint8_t *dest, l4_uint8_t val);
00206 L4_INLINE void
00207 l4util_or16(volatile l4_uint16_t *dest, l4_uint16_t val);
00209 L4_INLINE void
00210 l4util_or32(volatile l4_uint32_t *dest, l4_uint32_t val);
00212
00214
00221 L4_INLINE l4_uint8_t
00222 l4util_add8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00224 L4_INLINE l4_uint16_t
00225 l4util_add16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00227 L4_INLINE l4_uint32_t
00228 l4util_add32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00230 L4_INLINE l4_uint8_t
00231 l4util_sub8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00233 L4_INLINE l4_uint16_t
00234 l4util_sub16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00236 L4_INLINE l4_uint32_t
00237 l4util_sub32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00239 L4_INLINE l4_uint8_t
00240 l4util_and8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00242 L4_INLINE l4_uint16_t
00243 l4util_and16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00245 L4_INLINE l4_uint32_t
00246 l4util_and32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00248 L4_INLINE l4_uint8_t
00249 l4util_or8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00251 L4_INLINE l4_uint16_t
00252 l4util_or16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00254 L4_INLINE l4_uint32_t
00255 l4util_or32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00257
00259
00264 L4_INLINE void
00265 l4util_inc8(volatile l4_uint8_t *dest);
00267 L4_INLINE void
00268 l4util_inc16(volatile l4_uint16_t *dest);
00270 L4_INLINE void
00271 l4util_inc32(volatile l4_uint32_t *dest);
00273 L4_INLINE void
00274 l4util_dec8(volatile l4_uint8_t *dest);
00276 L4_INLINE void
00277 l4util_dec16(volatile l4_uint16_t *dest);
00279 L4_INLINE void
00280 l4util_dec32(volatile l4_uint32_t *dest);
00282
00284
00290 L4_INLINE l4_uint8_t
00291 l4util_inc8_res(volatile l4_uint8_t *dest);
00293 L4_INLINE l4_uint16_t
00294 l4util_inc16_res(volatile l4_uint16_t *dest);
00296 L4_INLINE l4_uint32_t
00297 l4util_inc32_res(volatile l4_uint32_t *dest);
00299 L4_INLINE l4_uint8_t
00300 l4util_dec8_res(volatile l4_uint8_t *dest);
00302 L4_INLINE l4_uint16_t
00303 l4util_dec16_res(volatile l4_uint16_t *dest);
00305 L4_INLINE l4_uint32_t
00306 l4util_dec32_res(volatile l4_uint32_t *dest);
00308
00316 L4_INLINE void
00317 l4util_atomic_add(volatile long *dest, long val);
00318
00325 L4_INLINE void
00326 l4util_atomic_inc(volatile long *dest);
00327
00328 EXTERN_C_END
```

```

00329
00330 /*****
00331  * IMPLEMENTAION *
00332  *****/
00333
00334 L4_INLINE int
00335 l4util_cmpxchg64(volatile l4_uint64_t * dest,
00336                  l4_uint64_t cmp_val, l4_uint64_t new_val)
00337 {
00338     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00339                                         __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00340 }
00341
00342 L4_INLINE int
00343 l4util_cmpxchg32(volatile l4_uint32_t * dest,
00344                  l4_uint32_t cmp_val, l4_uint32_t new_val)
00345 {
00346     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00347                                         __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00348 }
00349
00350 L4_INLINE int
00351 l4util_cmpxchg16(volatile l4_uint16_t * dest,
00352                  l4_uint16_t cmp_val, l4_uint16_t new_val)
00353 {
00354     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00355                                         __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00356 }
00357
00358 L4_INLINE int
00359 l4util_cmpxchg8(volatile l4_uint8_t * dest,
00360                 l4_uint8_t cmp_val, l4_uint8_t new_val)
00361 {
00362     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00363                                         __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00364 }
00365
00366 L4_INLINE int
00367 l4util_cmpxchg(volatile l4_umword_t * dest,
00368                l4_umword_t cmp_val, l4_umword_t new_val)
00369 {
00370     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00371                                         __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00372 }
00373
00374 L4_INLINE l4_uint32_t
00375 l4util_xchg32(volatile l4_uint32_t * dest, l4_uint32_t val)
00376 {
00377     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00378 }
00379
00380 L4_INLINE l4_uint16_t
00381 l4util_xchg16(volatile l4_uint16_t * dest, l4_uint16_t val)
00382 {
00383     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00384 }
00385
00386 L4_INLINE l4_uint8_t
00387 l4util_xchg8(volatile l4_uint8_t * dest, l4_uint8_t val)
00388 {
00389     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00390 }
00391
00392 L4_INLINE l4_umword_t
00393 l4util_xchg(volatile l4_umword_t * dest, l4_umword_t val)
00394 {
00395     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00396 }
00397
00398 L4_INLINE void
00399 l4util_inc8(volatile l4_uint8_t *dest)
00400 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00401
00402 L4_INLINE void
00403 l4util_inc16(volatile l4_uint16_t *dest)
00404 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00405
00406 L4_INLINE void
00407 l4util_inc32(volatile l4_uint32_t *dest)
00408 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00409
00410 L4_INLINE void
00411 l4util_atomic_inc(volatile long *dest)
00412 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00413
00414 L4_INLINE void
00415 l4util_dec8(volatile l4_uint8_t *dest)

```



```

00416 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00417
00418 L4_INLINE void
00419 l4util_dec16(volatile l4_uint16_t *dest)
00420 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00421
00422 L4_INLINE void
00423 l4util_dec32(volatile l4_uint32_t *dest)
00424 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00425
00426
00427 L4_INLINE l4_uint8_t
00428 l4util_inc8_res(volatile l4_uint8_t *dest)
00429 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00430
00431 L4_INLINE l4_uint16_t
00432 l4util_inc16_res(volatile l4_uint16_t *dest)
00433 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00434
00435 L4_INLINE l4_uint32_t
00436 l4util_inc32_res(volatile l4_uint32_t *dest)
00437 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00438
00439 L4_INLINE l4_uint8_t
00440 l4util_dec8_res(volatile l4_uint8_t *dest)
00441 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00442
00443 L4_INLINE l4_uint16_t
00444 l4util_dec16_res(volatile l4_uint16_t *dest)
00445 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00446
00447 L4_INLINE l4_uint32_t
00448 l4util_dec32_res(volatile l4_uint32_t *dest)
00449 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00450
00451 L4_INLINE l4_umword_t
00452 l4util_dec_res(volatile l4_umword_t *dest)
00453 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00454
00455 L4_INLINE void
00456 l4util_add8(volatile l4_uint8_t *dest, l4_uint8_t val)
00457 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00458
00459 L4_INLINE void
00460 l4util_add16(volatile l4_uint16_t *dest, l4_uint16_t val)
00461 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00462
00463 L4_INLINE void
00464 l4util_add32(volatile l4_uint32_t *dest, l4_uint32_t val)
00465 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00466
00467 L4_INLINE void
00468 l4util_atomic_add(volatile long *dest, long val)
00469 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00470
00471 L4_INLINE void
00472 l4util_sub8(volatile l4_uint8_t *dest, l4_uint8_t val)
00473 { __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }
00474
00475 L4_INLINE void
00476 l4util_sub16(volatile l4_uint16_t *dest, l4_uint16_t val)
00477 { __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }
00478
00479 L4_INLINE void
00480 l4util_sub32(volatile l4_uint32_t *dest, l4_uint32_t val)
00481 { __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }
00482
00483 L4_INLINE void
00484 l4util_and8(volatile l4_uint8_t *dest, l4_uint8_t val)
00485 { __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }
00486
00487 L4_INLINE void
00488 l4util_and16(volatile l4_uint16_t *dest, l4_uint16_t val)
00489 { __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }
00490
00491 L4_INLINE void
00492 l4util_and32(volatile l4_uint32_t *dest, l4_uint32_t val)
00493 { __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }
00494
00495 L4_INLINE void
00496 l4util_or8(volatile l4_uint8_t *dest, l4_uint8_t val)
00497 { __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }
00498
00499 L4_INLINE void
00500 l4util_or16(volatile l4_uint16_t *dest, l4_uint16_t val)
00501 { __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }
00502

```

```

00503 L4_INLINE void
00504 l4util_or32(volatile l4_uint32_t *dest, l4_uint32_t val)
00505 { __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }
00506
00507 L4_INLINE l4_uint8_t
00508 l4util_add8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00509 { return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }
00510
00511 L4_INLINE l4_uint16_t
00512 l4util_add16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00513 { return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }
00514
00515 L4_INLINE l4_uint32_t
00516 l4util_add32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00517 { return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }
00518
00519 L4_INLINE l4_uint8_t
00520 l4util_sub8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00521 { return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }
00522
00523 L4_INLINE l4_uint16_t
00524 l4util_sub16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00525 { return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }
00526
00527 L4_INLINE l4_uint32_t
00528 l4util_sub32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00529 { return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }
00530
00531 L4_INLINE l4_uint8_t
00532 l4util_and8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00533 { return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }
00534
00535 L4_INLINE l4_uint16_t
00536 l4util_and16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00537 { return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }
00538
00539 L4_INLINE l4_uint32_t
00540 l4util_and32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00541 { return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }
00542
00543 L4_INLINE l4_uint8_t
00544 l4util_or8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00545 { return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }
00546
00547 L4_INLINE l4_uint16_t
00548 l4util_or16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00549 { return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }
00550
00551 L4_INLINE l4_uint32_t
00552 l4util_or32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00553 { return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }
00554
00555 #endif /* ! __L4UTIL__INCLUDE__ATOMIC_H__ */

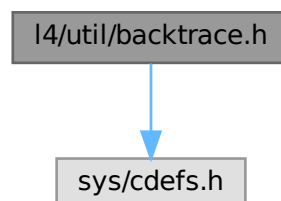
```

16.574 I4/util/backtrace.h File Reference

Backtrace.

```
#include <sys/cdefs.h>
```

Include dependency graph for backtrace.h:



16.575 backtrace.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 #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

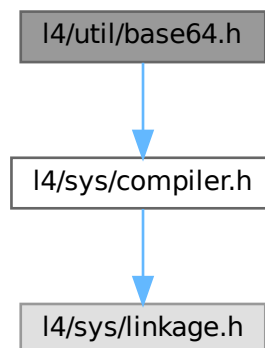
```

16.576 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

16.576.1 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

Definition in file [base64.h](#).

16.577 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
```

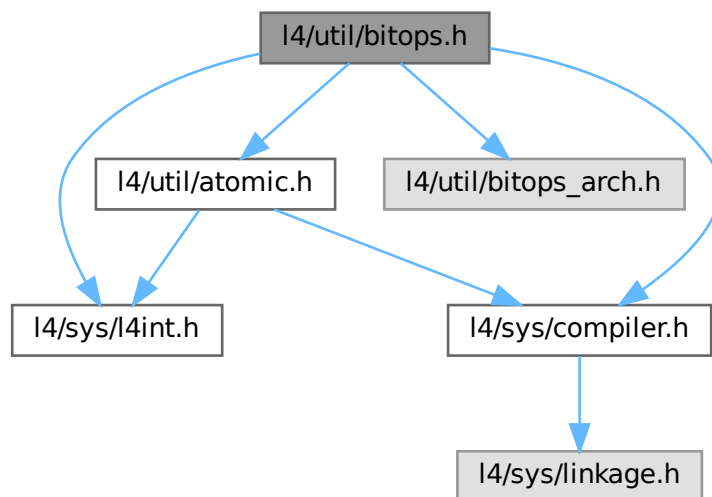
16.578 l4/util/bitops.h File Reference

bit manipulation functions

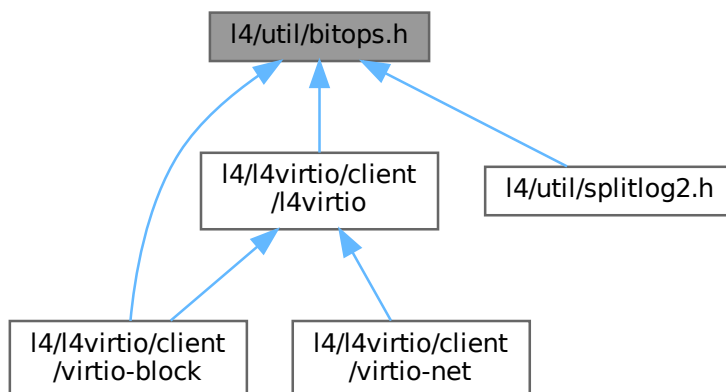
```
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
#include <l4/util/bitops_arch.h>
```

```
#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.

16.578.1 Detailed Description

bit manipulation functions

Date

07/03/2001

Author

Lars Reuther reuther@os.inf.tu-dresden.de

Definition in file [bitops.h](#).

16.579 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     return -1;
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     return -1;
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__ */

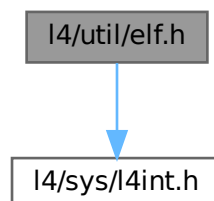
```

16.580 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 }
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_NUM](#) = 192 }

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 }
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](#).

16.580.1 Detailed Description

ELF definition.

Date

08/18/2000

Author

Frank Mehnert fm3@os.inf.tu-dresden.de Alexander Warg aw11@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](#).

16.581 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 <linux/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 #else
00105 # warning elf.h: Unsupported build architecture!
00106 #endif
00107
00108
00113 enum
00114 {
00115     EI_NIDENT                = 16,
00116 };
00117
00121 typedef struct
00122 {
00123     unsigned char e_ident[EI_NIDENT];
00124     Elf32_Half    e_type;
00125     Elf32_Half    e_machine;
00126     Elf32_Word    e_version;
00127     Elf32_Addr    e_entry;
00128     Elf32_Off     e_phoff;
00129     Elf32_Off     e_shoff;
00130     Elf32_Word    e_flags;
00131     Elf32_Half    e_ehsize;
00132     Elf32_Half    e_phentsize;
00133     Elf32_Half    e_phnum;
00134     Elf32_Half    e_shentsize;
00135     Elf32_Half    e_shnum;
00136     Elf32_Half    e_shstrndx;
00137 } Elf32_Ehdr;
00138
00142 typedef struct
00143 {
00144     unsigned char e_ident[EI_NIDENT];
00145     Elf64_Half    e_type;
00146     Elf64_Half    e_machine;
00147     Elf64_Word    e_version;
00148     Elf64_Addr    e_entry;
00149     Elf64_Off     e_phoff;
00150     Elf64_Off     e_shoff;
00151     Elf64_Word    e_flags;
00152     Elf64_Half    e_ehsize;
00153     Elf64_Half    e_phentsize;
00154     Elf64_Half    e_phnum;
00155     Elf64_Half    e_shentsize;
00156     Elf64_Half    e_shnum;
00157     Elf64_Half    e_shstrndx;
00158 } Elf64_Ehdr;
00159
00164 enum Elf_ETs
00165 {
00166     ET_NONE          = 0,
00167     ET_REL           = 1,
00168     ET_EXEC          = 2,
00169     ET_DYN           = 3,
00170     ET_CORE          = 4,
00171     ET_LOPROC        = 0xff00,
00172     ET_HIPROC        = 0xffff,
00173 };
00174
00179 enum Elf_EMs
00180 {
00181     EM_NONE          = 0,
00182     EM_M32           = 1,
00183     EM_SPARC         = 2,
00184     EM_386           = 3,
00185     EM_68K           = 4,
00186     EM_88K           = 5,

```

```

00187 EM_860 = 7,
00188 EM_MIPS = 8,
00189 EM_MIPS_RS4_BE = 10,
00190 EM_SPARC64 = 11,
00191 EM_PARISC = 15,
00192 EM_VPP500 = 17,
00193 EM_SPARC32PLUS = 18,
00194 EM_960 = 19,
00195 EM_PPC = 20,
00196 EM_V800 = 36,
00197 EM_FR20 = 37,
00198 EM_RH32 = 38,
00199 EM_RCE = 39,
00200 EM_ARM = 40,
00201 EM_ALPHA = 41,
00202 EM_SH = 42,
00203 EM_SPARCV9 = 43,
00204 EM_TRICORE = 44,
00205 EM_ARC = 45,
00206 EM_H8_300 = 46,
00207 EM_H8_300H = 47,
00208 EM_H8S = 48,
00209 EM_H8_500 = 49,
00210 EM_IA_64 = 50,
00211 EM_MIPS_X = 51,
00212 EM_COLDFIRE = 52,
00213 EM_68HC12 = 53,
00214 EM_X86_64 = 62,
00215 EM_PDSP = 63,
00216 EM_FX66 = 66,
00217 EM_ST9PLUS = 67,
00218 EM_ST7 = 68,
00219 EM_68HC16 = 69,
00220 EM_68HC11 = 70,
00221 EM_68HC08 = 71,
00222 EM_68HC05 = 72,
00223 EM_SVX = 73,
00224 EM_ST19 = 74,
00225 EM_VAX = 75,
00226 EM_CRIS = 76,
00227 EM_JAVELIN = 77,
00228 EM_FIREPATH = 78,
00229 EM_ZSP = 79,
00230 EM_MMIX = 80,
00231 EM_HUANY = 81,
00232 EM_PRISM = 82,
00233 EM_AVR = 83,
00234 EM_FR30 = 84,
00235 EM_D10V = 85,
00236 EM_D30V = 86,
00237 EM_V850 = 87,
00238 EM_M32R = 88,
00239 EM_MN10300 = 89,
00240 EM_MN10200 = 90,
00241 EM_PJ = 91,
00242 EM_OPENRISC = 92,
00243 EM_ARC_A5 = 93,
00244 EM_XTENSA = 94,
00245 EM_ALTERA_NIOS2 = 113,
00246 EM_AARCH64 = 183,
00247 EM_TILEPRO = 188,
00248 EM_MICROBLAZE = 189,
00249 EM_TILEGX = 191,
00250 EM_NUM = 192,
00251 };
00252
00253 #if 0
00254 #define EM_ALPHA 0x9026 /* interim value used by Linux until the
00255                          committee comes up with a final number */
00256 #define EM_S390 0xA390 /* interim value used for IBM S390 */
00257 #endif
00258
00261 enum Elf_EVs
00262 {
00263     EV_NONE = 0,
00264     EV_CURRENT = 1,
00265 };
00266
00269 enum Elf_EIs
00270 {
00271     EI_MAG0 = 0,
00272     EI_MAG1 = 1,
00273     EI_MAG2 = 2,
00274     EI_MAG3 = 3,
00275     EI_CLASS = 4,
00276     EI_DATA = 5,
00277     EI_VERSION = 6,

```

```

00278 EI_OSABI           = 7,
00279 EI_ABIVERSION       = 8,
00280 EI_PAD              = 9,
00281 };
00282
00284 enum Elf_MAGs
00285 {
00286     ELFMAG0           = 0x7f,
00287     ELFMAG1           = 'E',
00288     ELFMAG2           = 'L',
00289     ELFMAG3           = 'F',
00290 };
00291
00293 enum Elf_CLASSs
00294 {
00295     ELFCLASSNONE       = 0,
00296     ELFCLASS32         = 1,
00297     ELFCLASS64         = 2,
00298     ELFCLASSNUM        = 3,
00299 };
00300
00302 enum Elf_DATAs
00303 {
00304     ELFDATANONE        = 0,
00305     ELFDATA2LSB        = 1,
00306     ELFDATA2MSB        = 2,
00307     ELFDATANUM         = 3,
00308 };
00309
00311 enum Elf_OSABIs
00312 {
00313     ELFOSABI_NONE      = 0,
00314     ELFOSABI_SYSV      = 0,
00315     ELFOSABI_HPUX      = 1,
00316     ELFOSABI_NETBSD    = 2,
00317     ELFOSABI_LINUX     = 3,
00318     ELFOSABI_SOLARIS   = 6,
00319     ELFOSABI_AIX       = 7,
00320     ELFOSABI_IRIX      = 8,
00321     ELFOSABI_FREEBSD   = 9,
00322     ELFOSABI_TRU64     = 10,
00323     ELFOSABI_MODESTO   = 11,
00324     ELFOSABI_OPENBSD   = 12,
00325     ELFOSABI_ARM       = 97,
00326     ELFOSABI_STANDALONE = 255,
00327 };
00328
00330 enum Elf_SHNs
00331 {
00332     SHN_UNDEF          = 0,
00333     SHN_LORESERVE      = 0xff00,
00334     SHN_LOPROC         = 0xff00,
00335     SHN_HIPROC         = 0xff1f,
00336     SHN_ABS            = 0xffff1,
00337     SHN_COMMON         = 0xffff2,
00338     SHN_HIRESERVE      = 0xffff,
00339 };
00340
00342 typedef struct
00343 {
00344     Elf32_Word      sh_name;
00345     Elf32_Word      sh_type;
00346     Elf32_Word      sh_flags;
00347     Elf32_Addr      sh_addr;
00348     Elf32_Off       sh_offset;
00349     Elf32_Word      sh_size;
00350     Elf32_Word      sh_link;
00351     Elf32_Word      sh_info;
00352     Elf32_Word      sh_addralign;
00353     Elf32_Word      sh_entsize;
00354 } Elf32_Shdr;
00355
00357 typedef struct
00358 {
00359     Elf64_Word      sh_name;
00360     Elf64_Word      sh_type;
00361     Elf64_Xword     sh_flags;
00362     Elf64_Addr      sh_addr;
00363     Elf64_Off       sh_offset;
00364     Elf64_Xword     sh_size;
00365     Elf64_Word      sh_link;
00366     Elf64_Word      sh_info;
00367     Elf64_Xword     sh_addralign;
00368     Elf64_Xword     sh_entsize;
00369 } Elf64_Shdr;
00370
00372 enum Elf_SHTs

```

```

00373 {
00374     SHT_NULL                = 0,
00375     SHT_PROGBITS            = 1,
00376     SHT_SYMTAB              = 2,
00377     SHT_STRTAB              = 3,
00378     SHT_RELA                = 4,
00379     SHT_HASH                = 5,
00380     SHT_DYNAMIC             = 6,
00381     SHT_NOTE                = 7,
00382     SHT_NOBITS              = 8,
00383     SHT_REL                 = 9,
00384     SHT_SHLIB               = 10,
00385     SHT_DYNSYM              = 11,
00386     SHT_INIT_ARRAY          = 14,
00387     SHT_FINI_ARRAY          = 15,
00388     SHT_PREINIT_ARRAY       = 16,
00389     SHT_GROUP                = 17,
00390     SHT_SYMTAB_SHNDX        = 18,
00391     SHT_NUM                  = 19,
00392     SHT_LOOS                 = 0x60000000,
00393     SHT_HIOS                 = 0xffffffff,
00394     SHT_LOPROC               = 0x70000000,
00395     SHT_HIPROC               = 0x7fffffff,
00396     SHT_LOUSER               = 0x80000000,
00397     SHT_HIUSER               = 0xffffffff,
00398 };
00399
00401 enum Elf_SHFs
00402 {
00403     SHF_WRITE                = 0x1,
00404     SHF_ALLOC                = 0x2,
00405     SHF_EXECINSTR            = 0x4,
00406     SHF_MERGE                = 0x10,
00407     SHF_STRINGS              = 0x20,
00408     SHF_INFO_LINK            = 0x40,
00409     SHF_LINK_ORDER           = 0x80,
00410     SHF_OS_NONCONFORMING     = 0x100,
00411     SHF_GROUP                = 0x200,
00412     SHF_TLS                  = 0x400,
00413     SHF_MASKOS               = 0x0ff00000,
00414     SHF_MASKPROC             = 0xf0000000,
00415 };
00416
00417
00418
00420 typedef struct
00421 {
00422     Elf32_Word    p_type;
00423     Elf32_Off     p_offset;
00424     Elf32_Addr    p_vaddr;
00425     Elf32_Addr    p_paddr;
00426     Elf32_Word    p_filesz;
00427     Elf32_Word    p_memsz;
00428     Elf32_Word    p_flags;
00429     Elf32_Word    p_align;
00430 } Elf32_Phdr;
00431
00433 typedef struct
00434 {
00435     Elf64_Word    p_type;
00436     Elf64_Word    p_flags;
00437     Elf64_Off     p_offset;
00438     Elf64_Addr    p_vaddr;
00439     Elf64_Addr    p_paddr;
00440     Elf64_Xword   p_filesz;
00441     Elf64_Xword   p_memsz;
00442     Elf64_Xword   p_align;
00443 } Elf64_Phdr;
00444
00446 enum Elf_PTs
00447 {
00448     PT_NULL                = 0,
00449     PT_LOAD                = 1,
00450     PT_DYNAMIC             = 2,
00451     PT_INTERP              = 3,
00452     PT_NOTE                = 4,
00453     PT_SHLIB               = 5,
00454     PT_PHDR                = 6,
00455     PT_TLS                 = 7,
00456     PT_NUM                  = 8,
00457     PT_LOOS                 = 0x60000000,
00458     PT_HIOS                 = 0xffffffff,
00459     PT_LOPROC               = 0x70000000,
00460     PT_HIPROC               = 0x7fffffff,
00461     PT_GNU_EH_FRAME         = PT_LOOS + 0x474e550,
00462     PT_GNU_STACK            = PT_LOOS + 0x474e551,
00463     PT_GNU_RELRO            = PT_LOOS + 0x474e552,
00464     PT_L4_STACK             = PT_LOOS + 0x12,

```

```

00467 PT_L4_KIP          = PT_LOOS + 0x13,
00468 PT_L4_AUX          = PT_LOOS + 0x14,
00469 };
00470
00472 enum ELF_PFs
00473 {
00474     PF_X              = 0x1,
00475     PF_W              = 0x2,
00476     PF_R              = 0x4,
00477     PF_MASKOS         = 0x0ff00000,
00478     PF_MASKPROC       = 0x7fffffff,
00479 };
00480
00482 enum Elf_NTs_core
00483 {
00484     NT_PRSTATUS       = 1,
00485     NT_FPREGSET       = 2,
00486     NT_PRPSINFO       = 3,
00487     NT_PRXREG         = 4,
00488     NT_TASKSTRUCT     = 4,
00489     NT_PLATFORM       = 5,
00490     NT_AUXV           = 6,
00491     NT_GWINDOWS       = 7,
00492     NT_ASRS           = 8,
00493     NT_PSTATUS        = 10,
00494     NT_PSINFO         = 13,
00495     NT_PRCRED         = 14,
00496     NT_UTSNAME        = 15,
00497     NT_LWPSTATUS      = 16,
00498     NT_LWPSINFO       = 17,
00499     NT_PRFPXREG       = 20,
00500 };
00501
00503 enum Elf_NTs_obj
00504 {
00505     NT_VERSION         = 1,
00506 };
00507
00509 typedef struct
00510 {
00511     Elf32_Sword  d_tag;
00512     union
00513     {
00514         Elf32_Word  d_val;
00515         Elf32_Addr  d_ptr;
00516     } d_un;
00517 } Elf32_Dyn;
00518
00520 typedef struct
00521 {
00522     Elf64_Sxword d_tag;
00523     union
00524     {
00525         Elf64_Xword d_val;
00526         Elf64_Addr  d_ptr;
00527     } d_un;
00528 } Elf64_Dyn;
00529
00531 enum Elf_DTs
00532 {
00533     DT_NULL           = 0,
00534     DT_NEEDED         = 1,
00535     DT_PLTRELSZ       = 2,
00536     DT_PLTGOT         = 3,
00537     DT_HASH           = 4,
00538     DT_STRTAB         = 5,
00539     DT_SYMTAB         = 6,
00540     DT_RELA           = 7,
00541     DT_RELASZ         = 8,
00542     DT_RELAENT        = 9,
00543     DT_STRSZ          = 10,
00544     DT_SYMENT         = 11,
00545     DT_INIT           = 12,
00546     DT_FINI           = 13,
00547     DT_SONAME         = 14,
00548     DT_RPATH          = 15,
00549     DT_SYMBOLIC       = 16,
00550     DT_REL            = 17,
00551     DT_RELSZ          = 18,
00552     DT_RELENT         = 19,
00553     DT_PTRREL         = 20,
00554     DT_DEBUG          = 21,
00555     DT_TEXTREL        = 22,
00556     DT_JMPREL         = 23,
00557     DT_BIND_NOW       = 24,
00558     DT_INIT_ARRAY     = 25,
00559     DT_FINI_ARRAY     = 26,

```

```

00560 DT_INIT_ARRAYSZ      = 27,
00561 DT_FINI_ARRAYSZ      = 28,
00562 DT_RUNPATH           = 29,
00563 DT_FLAGS              = 30,
00564 DT_ENCODING          = 32,
00565 DT_PREINIT_ARRAY     = 32,
00566 DT_PREINIT_ARRAYSZ   = 33,
00567 DT_NUM                = 34,
00568 DT_LOOS              = 0x6000000d,
00569 DT_HIOS              = 0x6ffff000,
00570 DT_LOPROC            = 0x70000000,
00571 DT_HIPROC            = 0x7fffffff,
00572 };
00573
00577 enum Elf_DFs
00578 {
00579     DF_ORIGIN          = 0x00000001,
00580     DF_SYMBOLIC        = 0x00000002,
00581     DF_TEXTREL         = 0x00000004,
00582     DF_BIND_NOW        = 0x00000008,
00583     DF_STATIC_TLS      = 0x00000010,
00584 };
00585
00590 enum Elf_DF_1s
00591 {
00592     DF_1_NOW           = 0x00000001,
00593     DF_1_GLOBAL        = 0x00000002,
00594     DF_1_GROUP         = 0x00000004,
00595     DF_1_NODELETE      = 0x00000008,
00596     DF_1_LOADFLTR     = 0x00000010,
00597     DF_1_INITFIRST    = 0x00000020,
00598     DF_1_NOOPEN        = 0x00000040,
00599     DF_1_ORIGIN        = 0x00000080,
00600     DF_1_DIRECT        = 0x00000100,
00601     DF_1_TRANS         = 0x00000200,
00602     DF_1_INTERPOSE     = 0x00000400,
00603     DF_1_NODEFLIB     = 0x00000800,
00604     DF_1_NODUMP        = 0x00001000,
00605     DF_1_CONFALT       = 0x00002000,
00606     DF_1_ENDFILTEE     = 0x00004000,
00607     DF_1_DISPRELDNE    = 0x00008000,
00608     DF_1_DISPRELPND    = 0x00010000,
00609 };
00610
00612 enum Elf_DTF_1s
00613 {
00614     DTF_1_PARINIT      = 0x00000001,
00615     DTF_1_CONFEXP      = 0x00000002,
00616 };
00617
00619 enum Elf_DF_P1s
00620 {
00621     DF_P1_LAZYLOAD     = 0x00000001,
00622     DF_P1_GROUPEPERM   = 0x00000002,
00623 };
00624
00625
00627 typedef struct
00628 {
00629     Elf32_Addr          r_offset;
00630     Elf32_Word          r_info;
00631 } Elf32_Rel;
00632
00634 typedef struct
00635 {
00636     Elf32_Addr          r_offset;
00637     Elf32_Word          r_info;
00638     Elf32_Sword         r_addend;
00639 } Elf32_Rela;
00640
00642 typedef struct
00643 {
00644     Elf64_Addr          r_offset;
00645     Elf64_Xword         r_info;
00646 } Elf64_Rel;
00647
00649 typedef struct
00650 {
00651     Elf64_Addr          r_offset;
00652     Elf64_Xword         r_info;
00653     Elf64_Sxword        r_addend;
00654 } Elf64_Rela;
00655
00657 #define ELF32_R_SYM(i)    ((i)>>8)
00659 #define ELF32_R_TYPE(i)  ((unsigned char)(i))
00661 #define ELF32_R_INFO(s,t) (((s)>>8)+(unsigned char)(t))
00662
00664 #define ELF64_R_SYM(i)    ((i)>>32)

```

```

00665
00667 #define ELF64_R_TYPE(i)      ((i)&0xffffffffL)
00668
00670 #define ELF64_R_INFO(s,t)    (((s)<<32)+(t)&0xffffffffL)
00671
00673 enum Elf_R_386_s
00674 {
00675     R_386_NONE                = 0,
00676     R_386_32                  = 1,
00677     R_386_PC32                 = 2,
00678     R_386_GOT32                = 3,
00679     R_386_PLT32                = 4,
00680     R_386_COPY                 = 5,
00681     R_386_GLOB_DAT             = 6,
00682     R_386_JMP_SLOT             = 7,
00683     R_386_RELATIVE             = 8,
00684     R_386_GOTOFF               = 9,
00685     R_386_GOTPC                = 10,
00686     R_386_32PLT               = 11,
00687     R_386_TLS_TPOFF            = 14,
00688     R_386_TLS_IE               = 15,
00689     R_386_TLS_GOTIE           = 16,
00691     R_386_TLS_LE               = 17,
00692     R_386_TLS_GD               = 18,
00694     R_386_TLS_LDM              = 19,
00696     R_386_16                   = 20,
00697     R_386_PC16                 = 21,
00698     R_386_8                    = 22,
00699     R_386_PC8                  = 23,
00700     R_386_TLS_GD_32            = 24,
00702     R_386_TLS_GD_PUSH          = 25,
00703     R_386_TLS_GD_CALL          = 26,
00705     R_386_TLS_GD_POP           = 27,
00706     R_386_TLS_LDM_32           = 28,
00708     R_386_TLS_LDM_PUSH         = 29,
00709     R_386_TLS_LDM_CALL         = 30,
00711     R_386_TLS_LDM_POP          = 31,
00712     R_386_TLS_LDO_32           = 32,
00713     R_386_TLS_IE_32            = 33,
00715     R_386_TLS_LE_32            = 34,
00717     R_386_TLS_DTPMOD32         = 35,
00718     R_386_TLS_DTPOFF32         = 36,
00719     R_386_TLS_TPOFF32          = 37,
00720     R_386_NUM                   = 38,
00721 };
00722
00726 enum Elf_EF_ARM_s
00727 {
00728     EF_ARM_RELEXEC              = 0x01,
00729     EF_ARM_HASENTRY             = 0x02,
00730     EF_ARM_INTERWORK            = 0x04,
00731     EF_ARM_APCS_26              = 0x08,
00732     EF_ARM_APCS_FLOAT           = 0x10,
00733     EF_ARM_PIC                  = 0x20,
00734     EF_ARM_ALIGN8               = 0x40,
00735     EF_ARM_NEW_ABI              = 0x80,
00736     EF_ARM_OLD_ABI              = 0x100,
00737
00738     /* Other constants defined in the ARM ELF spec. version B-01. */
00739     /* NB. These conflict with values defined above. */
00740     EF_ARM_SYMSARESORTED        = 0x04,
00741     EF_ARM_DYNSYMSUSESEGIDX      = 0x08,
00742     EF_ARM_MAPSYMSFIRST         = 0x10,
00743     EF_ARM_EABIMASK             = 0xFF000000,
00744
00745     #define EF_ARM_EABI_VERSION(flags) ((flags) & EF_ARM_EABIMASK)
00746     EF_ARM_EABI_UNKNOWN          = 0x00000000,
00747     EF_ARM_EABI_VER1             = 0x01000000,
00748     EF_ARM_EABI_VER2            = 0x02000000,
00749 };
00750
00752 enum Elf_STT_ARM_s
00753 {
00754     STT_ARM_TFUNC               = 0xd,
00755 };
00756
00758 enum Elf_SHF_s_ARM
00759 {
00760     SHF_ARM_ENTRYSECT           = 0x10000000,
00761     SHF_ARM_COMDEF              = 0x80000000,
00763 };
00764
00766 enum Elf_ARM_SBs
00767 {
00768     PF_ARM_SB                   = 0x10000000,
00770 };
00771

```



```

00773 enum Elf_R_ARM_s
00774 {
00775     R_ARM_NONE                = 0,
00776     R_ARM_PC24                = 1,
00777     R_ARM_ABS32               = 2,
00778     R_ARM_REL32               = 3,
00779     R_ARM_PC13                = 4,
00780     R_ARM_ABS16               = 5,
00781     R_ARM_ABS12               = 6,
00782     R_ARM_THM_ABS5            = 7,
00783     R_ARM_ABS8                = 8,
00784     R_ARM_SBREL32             = 9,
00785     R_ARM_THM_PC22            = 10,
00786     R_ARM_THM_PC8             = 11,
00787     R_ARM AMP_VCALL9          = 12,
00788     R_ARM_SWI24               = 13,
00789     R_ARM_THM_SWI8            = 14,
00790     R_ARM_XPC25               = 15,
00791     R_ARM_THM_XPC22           = 16,
00792     R_ARM_COPY                = 20,
00793     R_ARM_GLOB_DAT            = 21,
00794     R_ARM_JUMP_SLOT           = 22,
00795     R_ARM_RELATIVE            = 23,
00796     R_ARM_GOTOFF              = 24,
00797     R_ARM_GOTPC               = 25,
00798     R_ARM_GOT32               = 26,
00799     R_ARM_PLT32               = 27,
00800     R_ARM_ALU_PCREL_7_0       = 32,
00801     R_ARM_ALU_PCREL_15_8      = 33,
00802     R_ARM_ALU_PCREL_23_15     = 34,
00803     R_ARM_LDR_SBREL_11_0      = 35,
00804     R_ARM_ALU_SBREL_19_12     = 36,
00805     R_ARM_ALU_SBREL_27_20     = 37,
00806     R_ARM_GNU_VTENTRY         = 100,
00807     R_ARM_GNU_VTINHERIT       = 101,
00808     R_ARM_THM_PC11            = 102,
00809     R_ARM_THM_PC9             = 103,
00810     R_ARM_RXPC25              = 249,
00811     R_ARM_RSBREL32            = 250,
00812     R_ARM_THM_RPC22           = 251,
00813     R_ARM_RREL32              = 252,
00814     R_ARM_RABS22              = 253,
00815     R_ARM_RPC24               = 254,
00816     R_ARM_RBASE               = 255,
00817     R_ARM_NUM                 = 256,
00818 };
00819
00821 enum Elf_R_AARCH64_s
00822 {
00823     R_AARCH64_NONE            = 0,
00824     R_AARCH64_RELATIVE        = 1027,
00825 };
00826
00828 enum Elf_R_X86_64_s
00829 {
00830     R_X86_64_NONE             = 0,
00831     R_X86_64_64               = 1,
00832     R_X86_64_PC32             = 2,
00833     R_X86_64_GOT32            = 3,
00834     R_X86_64_PLT32            = 4,
00835     R_X86_64_COPY             = 5,
00836     R_X86_64_GLOB_DAT         = 6,
00837     R_X86_64_JUMP_SLOT        = 7,
00838     R_X86_64_RELATIVE         = 8,
00839     R_X86_64_GOTPCREL         = 9,
00840     R_X86_64_32               = 10,
00841     R_X86_64_32S              = 11,
00842     R_X86_64_16               = 12,
00843     R_X86_64_PC16             = 13,
00844     R_X86_64_8                = 14,
00845     R_X86_64_PC8              = 15,
00846     R_X86_64_DTPOFF64         = 16,
00847     R_X86_64_DTPOFF64         = 17,
00848     R_X86_64_TPOFF64          = 18,
00849     R_X86_64_TLSD             = 19,
00851     R_X86_64_TLSD             = 20,
00853     R_X86_64_DTPOFF32         = 21,
00854     R_X86_64_GOTTPOFF         = 22,
00856     R_X86_64_TPOFF32          = 23,
00857     R_X86_64_NUM              = 24,
00858 };
00859
00861 enum Elf_STNs
00862 {
00863     STN_UNDEF                 = 0,
00864 };
00865

```

```

00867 typedef struct
00868 {
00869     Elf32_Word      st_name;
00870     Elf32_Addr      st_value;
00871     Elf32_Word      st_size;
00872     unsigned char    st_info;
00873     unsigned char    st_other;
00874     Elf32_Half      st_shndx;
00875 } Elf32_Sym;
00876
00877 typedef struct
00878 {
00879     Elf64_Word      st_name;
00880     unsigned char    st_info;
00881     unsigned char    st_other;
00882     Elf64_Half      st_shndx;
00883     Elf64_Addr      st_value;
00884     Elf64_Xword      st_size;
00885 } Elf64_Sym;
00886
00887 #define ELF32_ST_BIND(i)    ((i)>4)
00888
00889 #define ELF32_ST_TYPE(i)    ((i)&0xf)
00890
00891 #define ELF32_ST_INFO(b,t)  (((b)<4)+((t)&0xf))
00892
00893 #define ELF64_ST_BIND(i)    ((i)>4)
00894
00895 #define ELF64_ST_TYPE(i)    ((i)&0xf)
00896
00897 #define ELF64_ST_INFO(b,t)  (((b)<4)+((t)&0xf))
00898
00899 enum Elf_STBs
00900 {
00901     STB_LOCAL      = 0,
00902     STB_GLOBAL      = 1,
00903     STB_WEAK        = 2,
00904     STB_LOOS        = 10,
00905     STB_HIOS        = 12,
00906     STB_LOPROC      = 13,
00907     STB_HIPROC      = 15,
00908 };
00909
00910 enum Elf_STTs
00911 {
00912     STT_NOTYPE      = 0,
00913     STT_OBJECT      = 1,
00914     STT_FUNC        = 2,
00915     STT_SECTION     = 3,
00916     STT_FILE        = 4,
00917     STT_LOOS        = 10,
00918     STT_HIOS        = 12,
00919     STT_LOPROC      = 13,
00920     STT_HIPROC      = 15,
00921 };
00922
00923 enum Elf_ATs
00924 {
00925     AT_NULL          = 0,
00926     AT_IGNORE        = 1,
00927     AT_EXECD         = 2,
00928     AT_PHDR          = 3,
00929     AT_PHENT         = 4,
00930     AT_PHNUM         = 5,
00931     AT_PAGESZ        = 6,
00932     AT_BASE          = 7,
00933     AT_FLAGS         = 8,
00934     AT_ENTRY         = 9,
00935     AT_NOTELF        = 10,
00936     AT_UID           = 11,
00937     AT_EUID          = 12,
00938     AT_GID           = 13,
00939     AT_EGID          = 14,
00940     AT_L4_AUX        = 0xf0,
00941     AT_L4_ENV         = 0xf1,
00942 };
00943
00944 typedef struct Elf32_Auxv
00945 {
00946     Elf32_Word atype;
00947     Elf32_Word avalue;
00948 } Elf32_Auxv;
00949
00950 typedef struct Elf64_Auxv
00951 {
00952     Elf64_Word atype;
00953     Elf64_Word avalue;
00954 } Elf64_Auxv;

```

```

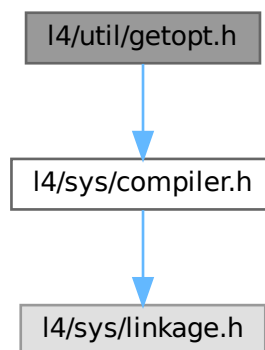
00969 } Elf64_Auxv;
00970
00978 static inline int l4util_elf_check_magic(ElfW(Ehdr) const *hdr);
00979
00987 static inline int l4util_elf_check_arch(ElfW(Ehdr) const *hdr);
00988
00995 static inline ElfW(Phdr) *l4util_elf_phdr(ElfW(Ehdr) const *hdr);
00996
00997
00998 /* Implementations */
00999
01000 static inline
01001 int l4util_elf_check_magic(ElfW(Ehdr) const *hdr)
01002 {
01003     return    hdr->e_ident[EI_MAG0] == ELFMAG0
01004             && hdr->e_ident[EI_MAG1] == ELFMAG1
01005             && hdr->e_ident[EI_MAG2] == ELFMAG2
01006             && hdr->e_ident[EI_MAG3] == ELFMAG3;
01007 }
01008
01009 static inline
01010 int l4util_elf_check_arch(ElfW(Ehdr) const *hdr)
01011 {
01012     return    hdr->e_ident[EI_CLASS] == L4_ARCH_EI_CLASS
01013             && hdr->e_ident[EI_DATA]  == L4_ARCH_EI_DATA
01014             && hdr->e_machine        == L4_ARCH_E_MACHINE;
01015 }
01016
01017 static inline
01018 ElfW(Phdr) *l4util_elf_phdr(ElfW(Ehdr) const *hdr)
01019 {
01020     return (ElfW(Phdr) *) ((char *)hdr + hdr->e_phoff);
01021 }

```

16.582 l4/util/getopt.h File Reference

getopt

#include <l4/sys/compiler.h>
 Include dependency graph for getopt.h:



16.582.1 Detailed Description

getopt

Definition in file [getopt.h](#).

16.583 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 */

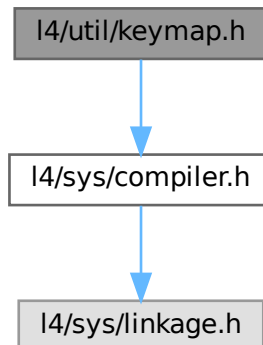
```

16.584 l4/util/keymap.h File Reference

Event to ASCII key mapping.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for keymap.h:



16.584.1 Detailed Description

Event to ASCII key mapping.

Definition in file [keymap.h](#).

16.585 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__ */

```

16.586 l4/sys/kip.h File Reference

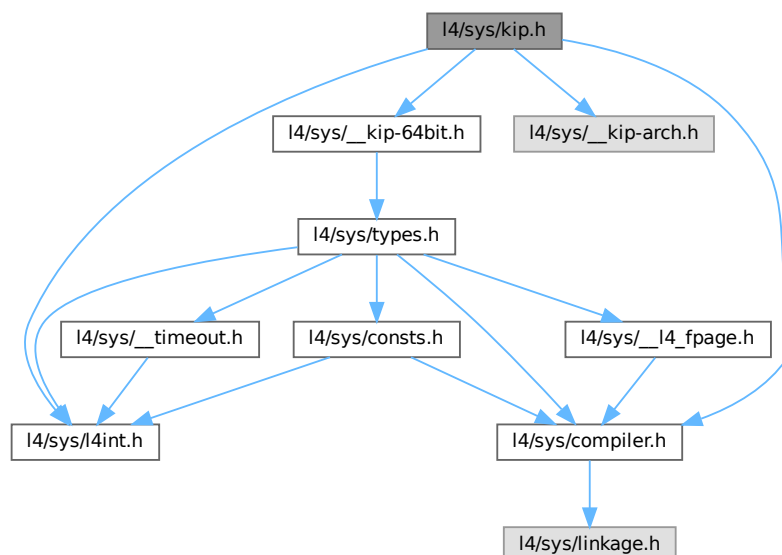
Kernel Info Page access functions.

```

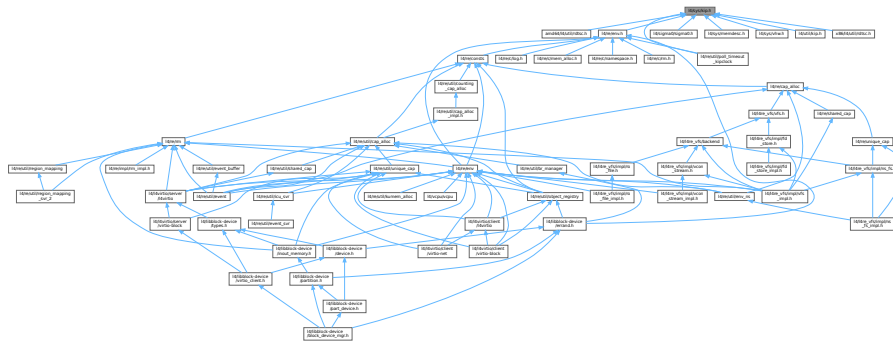
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/__kip-arch.h>
#include <l4/sys/__kip-64bit.h>

```

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.

16.586.1 Detailed Description

Kernel Info Page access functions.

Definition in file [kip.h](#).

16.586.2 Macro Definition Documentation

16.586.2.1 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 252 of file [kip.h](#).

16.586.3 Function Documentation

16.586.3.1 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

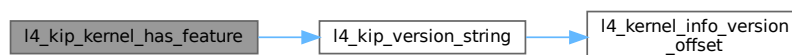
1 if the kernel supports the feature, 0 if not.

Checks the feature field in the KIP for the given string.

Definition at line 266 of file [kip.h](#).

References [l4_kip_for_each_feature](#), and [l4_kip_version_string\(\)](#).

Here is the call graph for this function:



16.587 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
00034 struct l4_kip_platform_info
00035 {
00036     char                name[16];
00037     l4_uint32_t          is_mp;
00038     struct l4_kip_platform_info_arch arch;
00039 };
00040
00041 #if L4_MWORD_BITS == 32
00042 #   include <l4/sys/__kip-32bit.h>
00043 #else
00044 #   include <l4/sys/__kip-64bit.h>
00045 #endif
00046
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_OFFSET_READ_US      = 0x900,
00075
00085     L4_KIP_OFFSET_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
00166 L4_INLINE l4_umword_t
00167 l4_kip_clock_lw(l4_kernel_info_t const *kip) L4_NOTHROW;
00168
00182 L4_INLINE l4_uint64_t
00183 l4_kip_clock_ns(l4_kernel_info_t const *kip) L4_NOTHROW;
00184
00187 /*****
00188  * Implementations
00189  *****/
00190
00191 L4_INLINE l4_kernel_info_t const*
00192 l4_kip(void) L4_NOTHROW

```

```

00193 {
00194 #ifdef __PIC__
00195     return l4_global_kip;
00196 #else
00197     extern char __L4_KIP_ADDR__[];
00198     return (l4_kernel_info_t const *)__L4_KIP_ADDR__;
00199 #endif
00200 }
00201
00202 L4_INLINE l4_umword_t
00203 l4_kip_version(l4_kernel_info_t const *kip) L4_NOTHROW
00204 { return kip->version & L4_KIP_VERSION_FIASCO_MASK; }
00205
00206 L4_INLINE const char*
00207 l4_kip_version_string(l4_kernel_info_t const *k) L4_NOTHROW
00208 { return (const char *)k + l4_kernel_info_version_offset(k); }
00209
00210 L4_INLINE int
00211 l4_kernel_info_version_offset(l4_kernel_info_t const *kip) L4_NOTHROW
00212 { return kip->offset_version_strings « 4; }
00213
00214 L4_INLINE l4_cpu_time_t
00215 l4_kip_clock(l4_kernel_info_t const *kip) L4_NOTHROW
00216 {
00217     // Use kernel-provided code to determine the current clock.
00218     typedef l4_uint64_t (*kip_time_fn_read_us)(void);
00219     kip_time_fn_read_us read_us =
00220         (kip_time_fn_read_us)((l4_uint8_t*)kip + L4_KIP_OFFS_READ_US);
00221     return read_us();
00222 }
00223
00224 L4_INLINE l4_cpu_time_t
00225 l4_kip_clock_ns(l4_kernel_info_t const *kip) L4_NOTHROW
00226 {
00227     typedef l4_uint64_t (*kip_time_fn_read_ns)(void);
00228     kip_time_fn_read_ns read_ns =
00229         (kip_time_fn_read_ns)((l4_uint8_t*)kip + L4_KIP_OFFS_READ_NS);
00230     return read_ns();
00231 }
00232
00233 L4_INLINE l4_umword_t
00234 l4_kip_clock_lw(l4_kernel_info_t const *kip) L4_NOTHROW
00235 {
00236     union Clock_field
00237     {
00238         l4_cpu_time_t t;
00239         unsigned long l;
00240     };
00241     union Clock_field c = { kip->_clock_val };
00242     l4_mb();
00243     return c.l;
00244 }
00245
00252 #define l4_kip_for_each_feature(s) \
00253     for (s += __builtin_strlen(s) + 1; *s; s += __builtin_strlen(s) + 1)
00254
00265 L4_INLINE int
00266 l4_kip_kernel_has_feature(l4_kernel_info_t const *kip, char const *str)
00267 {
00268     const char *s = l4_kip_version_string(kip);
00269     if (!s)
00270         return 0;
00271
00272     l4_kip_for_each_feature(s)
00273     {
00274         if (__builtin_strcmp(s, str) == 0)
00275             return 1;
00276     }
00277     return 0;
00278 }
00279 }

```

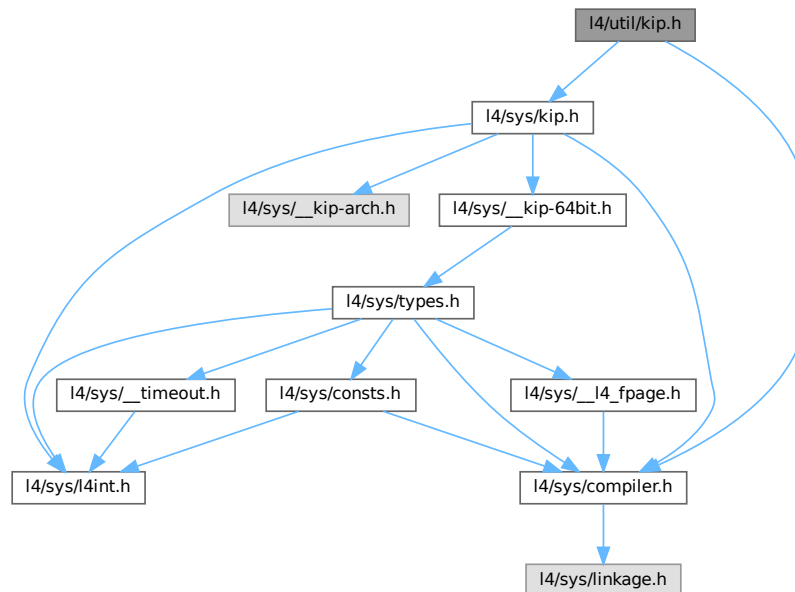
16.588 l4/util/kip.h File Reference

```

#include <l4/sys/kip.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for kip.h:



Macros

- `#define I4util_kip_for_each_feature(s) I4_kip_for_each_feature(s)`
Cycle through kernel features given in the KIP.

Functions

- `int I4util_kip_kernel_is_ux (I4_kernel_info_t const *k)`
Return whether the kernel is running natively or under UX.
- `int I4util_kip_kernel_has_feature (I4_kernel_info_t const *k, char const *str)`
Check if kernel supports a feature.
- `unsigned long I4util_kip_kernel_abi_version (I4_kernel_info_t const *k)`
Return kernel ABI version.

16.589 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>

```

```

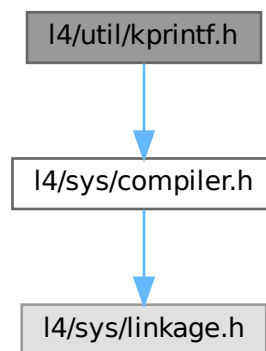
00017 #include <l4/sys/compiler.h>
00018
00026 EXTERN_C_BEGIN
00027
00034 L4_CV int l4util_kip_kernel_is_ux(l4_kernel_info_t const *k);
00035
00048 L4_CV int l4util_kip_kernel_has_feature(l4_kernel_info_t const *k, char const *str);
00049
00056 L4_CV unsigned long l4util_kip_kernel_abi_version(l4_kernel_info_t const *k);
00057
00058 EXTERN_C_END
00059
00068 #define l4util_kip_for_each_feature(s) l4_kip_for_each_feature(s)
00069

```

16.590 l4/util/kprintf.h File Reference

printf using the kernel debugger

```
#include <l4/sys/compiler.h>
Include dependency graph for kprintf.h:
```



16.590.1 Detailed Description

printf using the kernel debugger

Date

04/05/2007

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de,

Definition in file [kprintf.h](#).

16.591 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__ */

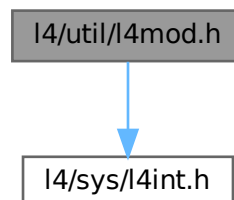
```

16.592 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.

16.592.1 Detailed Description

L4mod structures and constants.

Definition in file [l4mod.h](#).

16.592.2 Enumeration Type Documentation

16.592.2.1 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](#).

16.593 l4mod.h

[Go to the documentation of this file.](#)

```

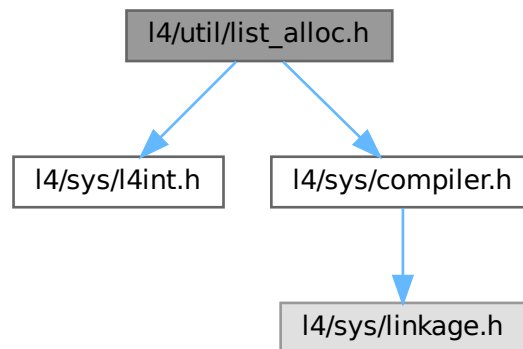
00001 /* SPDX-License-Identifier: GPL-2.0-only or License-Ref-kk-custom */
00002 /*
00003  * Copyright (C) 2021-2022 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@l4re.org>
00005  */
00006
00011 #pragma once
00012
00013 #include <l4/sys/l4int.h>
00014
00016 enum l4util_l4mod_mod_info_flag
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;

```

16.594 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.

16.594.1 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

Definition in file [list_alloc.h](#).

16.594.2 Function Documentation

16.594.2.1 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

16.594.2.2 l4la_avail()

```
l4_size_t l4la_avail (
    l4la_free_t ** first )
```

Show available memory in pool.

Parameters

<i>first</i>	list identifier
--------------	-----------------

16.594.2.3 l4la_dump()

```
void l4la_dump (
    l4la_free_t ** first )
```

Show all list members.

Parameters

<i>first</i>	list identifier
--------------	-----------------

16.594.2.4 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

16.594.2.5 l4la_init()

```
void l4la_init (
    l4la_free_t ** first )
```

Init memory pool.

Parameters

<i>first</i>	list identifier
--------------	-----------------

16.595 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
```

16.596 llulc.h

```
00001 /*
```

```

00002  * (c) 2011 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  * Note, do _NOT_ use this lock unless you got advised to do so.
00020  */
00021 #pragma once
00022
00023 #include <l4/sys/utcb.h>
00024 #include <stddef.h>
00025
00026 __BEGIN_DECLS
00027
00028 struct l4lllock_struct_t;
00029 typedef struct l4ullulock_struct_t l4ullulock_t;
00030
00031 #ifdef __cplusplus
00032 #define DEFAULT_UTCB = l4_utcb()
00033 #else
00034 #define DEFAULT_UTCB
00035 #endif
00036
00037 int l4ullulock_init(l4ullulock_t **t,
00038                    void *(*mem_alloc)(size_t x),
00039                    void (*mem_free)(void *p),
00040                    l4_cap_idx_t (*cap_alloc)(void),
00041                    void (*cap_free)(l4_cap_idx_t c),
00042                    l4_cap_idx_t factory);
00043 int l4ullulock_deinit(l4ullulock_t *t);
00044 int l4ullulock_lock(l4ullulock_t *t, l4_utcb_t *u DEFAULT_UTCB);
00045 int l4ullulock_unlock(l4ullulock_t *t, l4_utcb_t *u DEFAULT_UTCB);
00046
00047 __END_DECLS

```

16.597 l4/util/lock.h File Reference

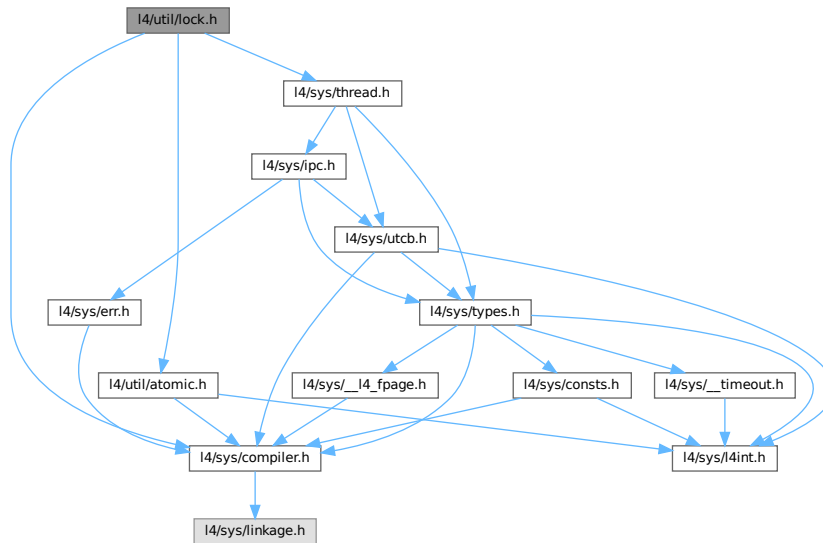
Simple lock implementation.

```

#include <l4/sys/thread.h>
#include <l4/sys/compiler.h>
#include <l4/util/atomic.h>

```

Include dependency graph for lock.h:



16.597.1 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

Definition in file [lock.h](#).

16.598 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

```

16.599 l4/util/mb_info.h File Reference

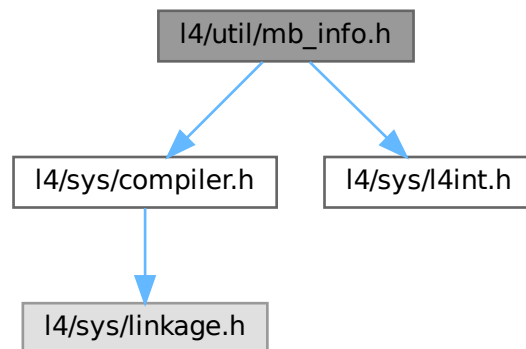
Multiboot info structure as defined by GRUB.

```

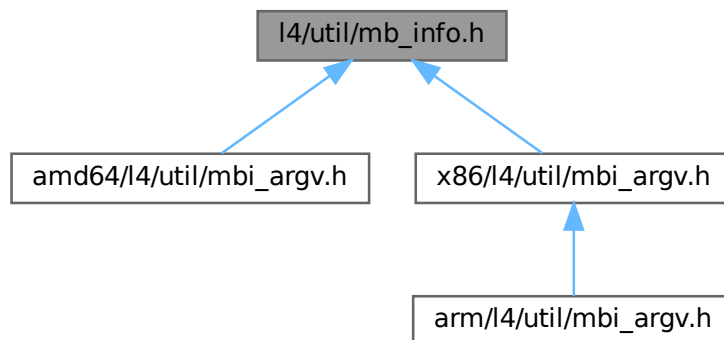
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>

```

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.

16.599.1 Detailed Description

Multiboot info structure as defined by GRUB.

Definition in file [mb_info.h](#).

16.599.2 Macro Definition Documentation

16.599.2.1 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 l4util_mb_addr_range_t * that is consecutively assigned pointers to the entries of the memory map.
<i>mbi</i>	Pointer to the l4util_mb_info_t where the memory map can be found.

Definition at line 334 of file [mb_info.h](#).

16.599.2.2 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](#).

16.599.2.3 MB_ARD_MEMORY

```
#define MB_ARD_MEMORY 1
```

usable memory "Type", all others are reserved.

Definition at line 60 of file [mb_info.h](#).

16.599.2.4 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](#).

16.600 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_ARC_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

```

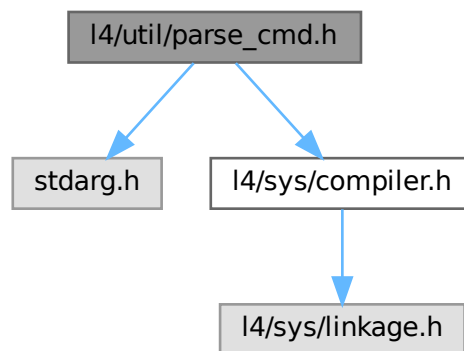
16.601 l4/util/parse_cmd.h File Reference

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.

16.601.1 Detailed Description

comfortable command-line parsing

Date

2002

Author

Jork Loeser jork.loeser@inf.tu-dresden.de

Definition in file [parse_cmd.h](#).

16.602 parse_cmd.h

[Go to the documentation of this file.](#)

```

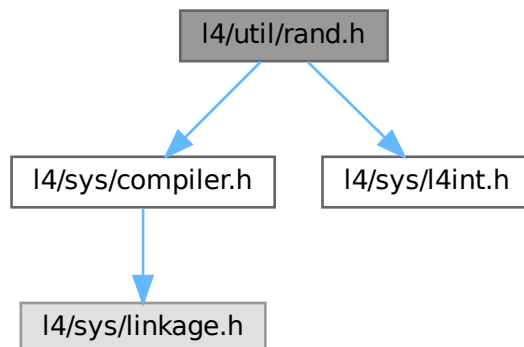
00001
00009 /*
00010  * (c) 2003-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU Lesser General Public License 2.1.
00014  * Please see the COPYING-LGPL-2.1 file for details.
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

```

16.603 I4/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.

16.603.1 Detailed Description

Simple Pseudo-Random Number Generator.

Date

1998

Author

Lars Reuther reuther@os.inf.tu-dresden.de

Definition in file [rand.h](#).

16.604 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 */

```

16.605 I4/util/splitlog2.h File Reference

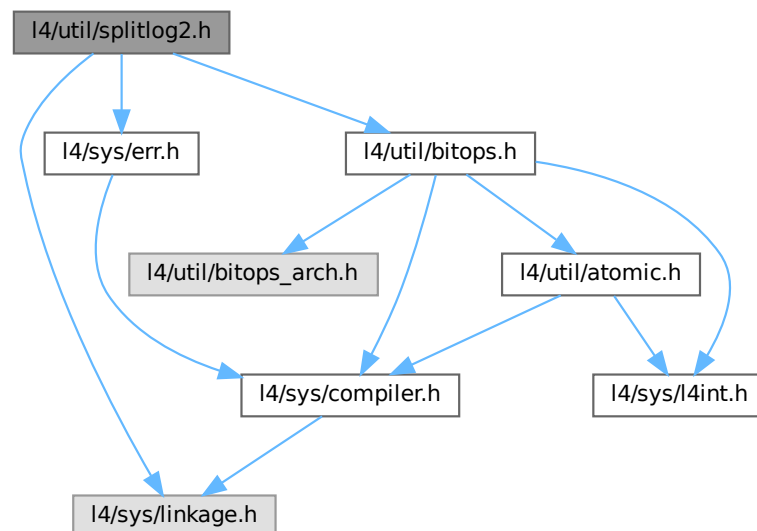
Split a range in log2 aligned and size-aligned chunks.

```

#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.

16.605.1 Detailed Description

Split a range in log2 aligned and size-aligned chunks.

Definition in file [splitlog2.h](#).

16.606 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
00033 L4_INLINE long
00034 l4util_splitlog2_hdl(l4_addr_t start, l4_addr_t end,
00035                     long (*handler)(l4_addr_t s, l4_addr_t e, int log2size));
00036
00045 L4_INLINE l4_addr_t
00046 l4util_splitlog2_size(l4_addr_t start, l4_addr_t end);
00047
00048 EXTERN_C_END
00049
00050 /* Implementation */
00051
00052 L4_INLINE long
00053 l4util_splitlog2_hdl(l4_addr_t start, l4_addr_t end,
00054                     long (*handler)(l4_addr_t s, l4_addr_t e, int log2size))
00055 {
00056     if (end < start)
00057         return -L4_EINVAL;
00058     while (start <= end)
00059     {
00060         long retval;
00061         int len2 = l4util_splitlog2_size(start, end);
00062         l4_addr_t len = 1UL << len2;
00063         if ((retval = handler(start, start + len - 1, len2)))
00064             return retval;
00065         start += len;
00066     }
00067     return 0;
00068 }
00069
00070 L4_INLINE l4_addr_t
00071 l4util_splitlog2_size(l4_addr_t start, l4_addr_t end)
00072 {
00073     int start_bits = l4util_bsf(start);
00074     int len_bits = l4util_bsr(end - start + 1);
00075     if (start_bits != -1 && len_bits > start_bits)
00076         len_bits = start_bits;
00077     return len_bits;
00078 }
00079
00080 }
00081
00082 #endif /* ! __L4UTIL__INCLUDE__SPLITLOG2_H__ */

```


16.607 arm/l4/sys/thread.h File Reference

ARM-specific thread related definitions.

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.

16.607.1 Detailed Description

ARM-specific thread related definitions.

Definition in file [thread.h](#).

16.608 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
00045 /* IMPLEMENTATION ----- */
00046
00047 L4_INLINE l4_msgtag_t
00048 l4_thread_arm_set_tpidruro_u(l4_cap_idx_t thread, l4_addr_t tpidruro,
00049                             l4_utcb_t *utcb) L4_NOTHROW
00050 {
00051     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00052     v->mr[0] = L4_THREAD_ARM_TPIDRURO_OP;
00053     v->mr[1] = tpidruro;
00054     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0),
00055                      L4_IPC_NEVER);
00056 }
00057
00058 L4_INLINE l4_msgtag_t
00059 l4_thread_arm_set_tpidruro(l4_cap_idx_t thread, l4_addr_t tpidruro) L4_NOTHROW
00060 {
00061     return l4_thread_arm_set_tpidruro_u(thread, tpidruro, l4_utcb());
00062 }

```

16.609 I4/sys/thread.h File Reference

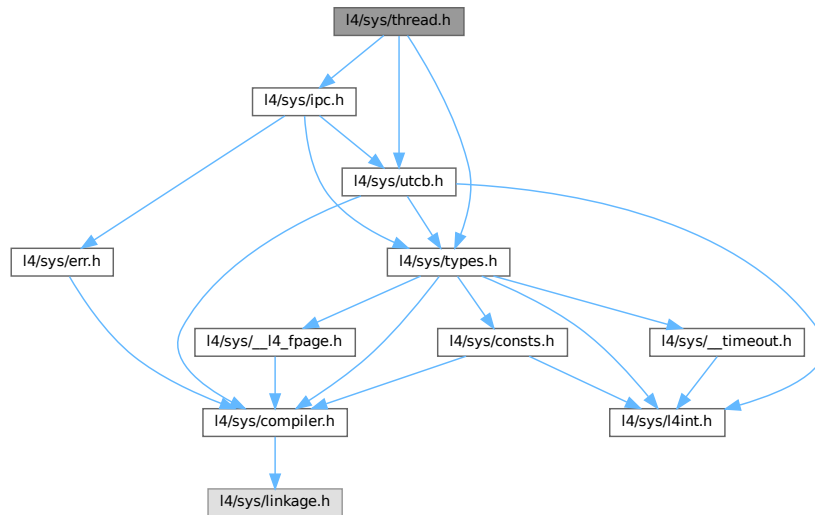
Common thread related definitions.

```
#include <l4/sys/types.h>
```

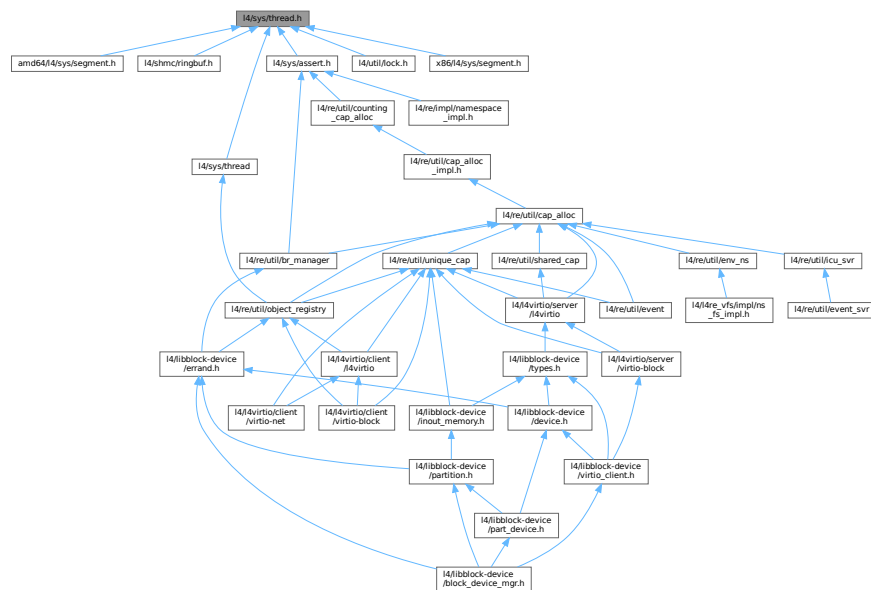
```
#include <l4/sys/utcb.h>
```

```
#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 }
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 μ 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.

16.609.1 Detailed Description

Common thread related definitions.

Definition in file [thread.h](#).

16.610 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
00673 L4_INLINE l4_msgtag_t
00674 l4_thread_modify_sender_commit(l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW;
00675
00680 L4_INLINE l4_msgtag_t
00681 l4_thread_modify_sender_commit_u(l4_cap_idx_t thread, l4_msgtag_t tag,
00682                                  l4_utcb_t *u) L4_NOTHROW;
00683
00690 enum l4_thread_ops
00691 {
00692     L4_THREAD_CONTROL_OP          = 0UL,
00693     L4_THREAD_EX_REGS_OP         = 1UL,
00694     L4_THREAD_SWITCH_OP          = 2UL,
00695     L4_THREAD_STATS_OP           = 3UL,
00696     L4_THREAD_VCPU_RESUME_OP     = 4UL,
00697     L4_THREAD_REGISTER_DELETE_IRQ_OP = 5UL,
00698     L4_THREAD_MODIFY_SENDER_OP   = 6UL,
00699     L4_THREAD_VCPU_CONTROL_OP    = 7UL,

```

```

00700 L4_THREAD_VCPU_CONTROL_EXT_OP      = L4_THREAD_VCPU_CONTROL_OP | 0x10000,
00701 L4_THREAD_X86_GDT_OP               = 0x10UL,
00702 L4_THREAD_ARM_TPIDRURO_OP          = 0x10UL,
00703 L4_THREAD_AMD64_SET_SEGMENT_BASE_OP = 0x12UL,
00704 L4_THREAD_AMD64_GET_SEGMENT_INFO_OP = 0x13UL,
00705 L4_THREAD_OPCODE_MASK              = 0xffff,
00706 };
00707
00718 enum L4_thread_control_flags
00719 {
00721 L4_THREAD_CONTROL_SET_PAGER        = 0x0010000,
00723 L4_THREAD_CONTROL_BIND_TASK        = 0x0200000,
00725 L4_THREAD_CONTROL_ALIEN            = 0x0400000,
00727 L4_THREAD_CONTROL_UX_NATIVE        = 0x0800000,
00729 L4_THREAD_CONTROL_SET_EXC_HANDLER = 0x1000000,
00730 };
00731
00741 enum L4_thread_control_mr_indices
00742 {
00743 L4_THREAD_CONTROL_MR_IDX_FLAGS      = 0,
00744 L4_THREAD_CONTROL_MR_IDX_PAGER      = 1,
00745 L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER = 2,
00746 L4_THREAD_CONTROL_MR_IDX_FLAG_VALS  = 4,
00747 L4_THREAD_CONTROL_MR_IDX_BIND_UTCB  = 5,
00748 L4_THREAD_CONTROL_MR_IDX_BIND_TASK  = 6,
00749 };
00750
00756 enum L4_thread_ex_regs_flags
00757 {
00758 L4_THREAD_EX_REGS_CANCEL            = 0x10000UL,
00759 L4_THREAD_EX_REGS_TRIGGER_EXCEPTION = 0x20000UL,
00760 };
00761
00762
00763 /* IMPLEMENTATION ----- */
00764
00765 #include <l4/sys/ipc.h>
00766 #include <l4/sys/types.h>
00767
00768 L4_INLINE l4_msgtag_t
00769 l4_thread_ex_regs_u(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
00770                    l4_umword_t flags, l4_utcb_t *utcb) L4_NOTHROW
00771 {
00772     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00773     v->mr[0] = L4_THREAD_EX_REGS_OP | flags;
00774     v->mr[1] = ip;
00775     v->mr[2] = sp;
00776     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 3, 0, 0), L4_IPC_NEVER);
00777 }
00778
00779 L4_INLINE l4_msgtag_t
00780 l4_thread_ex_regs_ret_u(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
00781                        l4_umword_t *flags, l4_utcb_t *utcb) L4_NOTHROW
00782 {
00783     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00784     l4_msgtag_t ret = l4_thread_ex_regs_u(thread, *ip, *sp, *flags, utcb);
00785     if (l4_error_u(ret, utcb))
00786         return ret;
00787
00788     *flags = v->mr[0];
00789     *ip    = v->mr[1];
00790     *sp    = v->mr[2];
00791     return ret;
00792 }
00793
00794 L4_INLINE void
00795 l4_thread_control_start_u(l4_utcb_t *utcb) L4_NOTHROW
00796 {
00797     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00798     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] = L4_THREAD_CONTROL_OP;
00799 }
00800
00801 L4_INLINE void
00802 l4_thread_control_pager_u(l4_cap_idx_t pager, l4_utcb_t *utcb) L4_NOTHROW
00803 {
00804     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00805     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_SET_PAGER;
00806     v->mr[L4_THREAD_CONTROL_MR_IDX_PAGER] = pager;
00807 }
00808
00809 L4_INLINE void
00810 l4_thread_control_exc_handler_u(l4_cap_idx_t exc_handler,
00811                                l4_utcb_t *utcb) L4_NOTHROW
00812 {
00813     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00814     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_SET_EXC_HANDLER;
00815     v->mr[L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER] = exc_handler;

```

```

00816 }
00817
00818 L4_INLINE void
00819 l4_thread_control_bind_u(l4_utcb_t *thread_utcb, l4_cap_idx_t task,
00820                          l4_utcb_t *utcb) L4_NOTHROW
00821 {
00822     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00823     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_BIND_TASK;
00824     v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_UTCB] = (l4_addr_t)thread_utcb;
00825     v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_TASK] = L4_ITEM_MAP;
00826     v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_TASK + 1] = l4_obj_fpage(task, 0, L4_CAP_FPAGE_RWS).raw;
00827 }
00828
00829 L4_INLINE void
00830 l4_thread_control_alien_u(l4_utcb_t *utcb, int on) L4_NOTHROW
00831 {
00832     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00833     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_ALIEN;
00834     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAG_VALS] |= on ? L4_THREAD_CONTROL_ALIEN : 0;
00835 }
00836
00837 L4_INLINE void
00838 l4_thread_control_ux_host_syscall_u(l4_utcb_t *utcb, int on) L4_NOTHROW
00839 {
00840     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00841     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_UX_NATIVE;
00842     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAG_VALS] |= on ? L4_THREAD_CONTROL_UX_NATIVE : 0;
00843 }
00844
00845 L4_INLINE l4_msgtag_t
00846 l4_thread_control_commit_u(l4_cap_idx_t thread, l4_utcb_t *utcb) L4_NOTHROW
00847 {
00848     int items = 0;
00849     if (l4_utcb_mr_u(utcb)->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] & L4_THREAD_CONTROL_BIND_TASK)
00850         items = 1;
00851     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 6, items, 0), L4_IPC_NEVER);
00852 }
00853
00854
00855 L4_INLINE l4_msgtag_t
00856 l4_thread_yield(void) L4_NOTHROW
00857 {
00858     l4_ipc_receive(L4_INVALID_CAP, NULL, L4_IPC_BOTH_TIMEOUT_0);
00859     return l4_msgtag(0, 0, 0, 0);
00860 }
00861
00862 /* Preliminary, to be changed */
00863 L4_INLINE l4_msgtag_t
00864 l4_thread_switch_u(l4_cap_idx_t to_thread, l4_utcb_t *utcb) L4_NOTHROW
00865 {
00866     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00867     v->mr[0] = L4_THREAD_SWITCH_OP;
00868     return l4_ipc_call(to_thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER);
00869 }
00870
00871
00872 L4_INLINE l4_msgtag_t
00873 l4_thread_stats_time_u(l4_cap_idx_t thread, l4_kernel_clock_t *us,
00874                        l4_utcb_t *utcb) L4_NOTHROW
00875 {
00876     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00877     l4_msgtag_t res;
00878
00879     v->mr[0] = L4_THREAD_STATS_OP;
00880
00881     res = l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER);
00882
00883     if (l4_msgtag_has_error(res))
00884         return res;
00885
00886     *us = v->mr64[l4_utcb_mr64_idx(0)];
00887
00888     return res;
00889 }
00890
00891 L4_INLINE l4_msgtag_t
00892 l4_thread_vcpu_resume_start_u(l4_utcb_t *utcb) L4_NOTHROW
00893 {
00894     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00895     v->mr[0] = L4_THREAD_VCPU_RESUME_OP;
00896     return l4_msgtag(L4_PROTO_THREAD, 1, 0, 0);
00897 }
00898
00899 L4_INLINE l4_msgtag_t
00900 l4_thread_vcpu_resume_commit_u(l4_cap_idx_t thread,
00901                                l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW
00902 {

```



```

00903     return l4_ipc_call(thread, utcb, tag, L4_IPC_NEVER);
00904 }
00905
00906 L4_INLINE l4_msgtag_t
00907 l4_thread_ex_regs(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
00908                  l4_umword_t flags) L4_NOTHROW
00909 {
00910     return l4_thread_ex_regs_u(thread, ip, sp, flags, l4_utcb());
00911 }
00912
00913 L4_INLINE l4_msgtag_t
00914 l4_thread_ex_regs_ret(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
00915                      l4_umword_t *flags) L4_NOTHROW
00916 {
00917     return l4_thread_ex_regs_ret_u(thread, ip, sp, flags, l4_utcb());
00918 }
00919
00920 L4_INLINE void
00921 l4_thread_control_start(void) L4_NOTHROW
00922 {
00923     l4_thread_control_start_u(l4_utcb());
00924 }
00925
00926 L4_INLINE void
00927 l4_thread_control_pager(l4_cap_idx_t pager) L4_NOTHROW
00928 {
00929     l4_thread_control_pager_u(pager, l4_utcb());
00930 }
00931
00932 L4_INLINE void
00933 l4_thread_control_exc_handler(l4_cap_idx_t exc_handler) L4_NOTHROW
00934 {
00935     l4_thread_control_exc_handler_u(exc_handler, l4_utcb());
00936 }
00937
00938
00939 L4_INLINE void
00940 l4_thread_control_bind(l4_utcb_t *thread_utcb, l4_cap_idx_t task) L4_NOTHROW
00941 {
00942     l4_thread_control_bind_u(thread_utcb, task, l4_utcb());
00943 }
00944
00945 L4_INLINE void
00946 l4_thread_control_alien(int on) L4_NOTHROW
00947 {
00948     l4_thread_control_alien_u(l4_utcb(), on);
00949 }
00950
00951 L4_INLINE void
00952 l4_thread_control_ux_host_syscall(int on) L4_NOTHROW
00953 {
00954     l4_thread_control_ux_host_syscall_u(l4_utcb(), on);
00955 }
00956
00957 L4_INLINE l4_msgtag_t
00958 l4_thread_control_commit(l4_cap_idx_t thread) L4_NOTHROW
00959 {
00960     return l4_thread_control_commit_u(thread, l4_utcb());
00961 }
00962
00963
00964
00965
00966 L4_INLINE l4_msgtag_t
00967 l4_thread_switch(l4_cap_idx_t to_thread) L4_NOTHROW
00968 {
00969     return l4_thread_switch_u(to_thread, l4_utcb());
00970 }
00971
00972
00973
00974
00975 L4_INLINE l4_msgtag_t
00976 l4_thread_stats_time(l4_cap_idx_t thread, l4_kernel_clock_t *us) L4_NOTHROW
00977 {
00978     return l4_thread_stats_time_u(thread, us, l4_utcb());
00979 }
00980
00981 L4_INLINE l4_msgtag_t
00982 l4_thread_vcpu_resume_start(void) L4_NOTHROW
00983 {
00984     return l4_thread_vcpu_resume_start_u(l4_utcb());
00985 }
00986
00987 L4_INLINE l4_msgtag_t
00988 l4_thread_vcpu_resume_commit(l4_cap_idx_t thread,
00989                              l4_msgtag_t tag) L4_NOTHROW

```

```

00990 {
00991     return l4_thread_vcpu_resume_commit_u(thread, tag, l4_utcb());
00992 }
00993
00994
00995 L4_INLINE l4_msgtag_t
00996 l4_thread_register_del_irq_u(l4_cap_idx_t thread, l4_cap_idx_t irq,
00997                             l4_utcb_t *u) L4_NOTHROW
00998 {
00999     l4_msg_regs_t *m = l4_utcb_mr_u(u);
01000     m->mr[0] = L4_THREAD_REGISTER_DELETE_IRQ_OP;
01001     m->mr[1] = l4_map_obj_control(0,0);
01002     m->mr[2] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).raw;
01003     return l4_ipc_call(thread, u, l4_msgtag(L4_PROTO_THREAD, 1, 1, 0), L4_IPC_NEVER);
01004 }
01005
01006
01007 L4_INLINE l4_msgtag_t
01008 l4_thread_register_del_irq(l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW
01009 {
01010     return l4_thread_register_del_irq_u(thread, irq, l4_utcb());
01011 }
01012
01013
01014 L4_INLINE l4_msgtag_t
01015 l4_thread_vcpu_control_u(l4_cap_idx_t thread, l4_addr_t vcpu_state,
01016                         l4_utcb_t *utcb) L4_NOTHROW
01017 {
01018     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
01019     v->mr[0] = L4_THREAD_VCPU_CONTROL_OP;
01020     v->mr[1] = vcpu_state;
01021     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER);
01022 }
01023
01024 L4_INLINE l4_msgtag_t
01025 l4_thread_vcpu_control(l4_cap_idx_t thread, l4_addr_t vcpu_state) L4_NOTHROW
01026 { return l4_thread_vcpu_control_u(thread, vcpu_state, l4_utcb()); }
01027
01028
01029 L4_INLINE l4_msgtag_t
01030 l4_thread_vcpu_control_ext_u(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state,
01031                             l4_utcb_t *utcb) L4_NOTHROW
01032 {
01033     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
01034     v->mr[0] = L4_THREAD_VCPU_CONTROL_EXT_OP;
01035     v->mr[1] = ext_vcpu_state;
01036     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER);
01037 }
01038
01039 L4_INLINE l4_msgtag_t
01040 l4_thread_vcpu_control_ext(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state) L4_NOTHROW
01041 { return l4_thread_vcpu_control_ext_u(thread, ext_vcpu_state, l4_utcb()); }
01042
01043 L4_INLINE l4_msgtag_t
01044 l4_thread_modify_sender_start_u(l4_utcb_t *u) L4_NOTHROW
01045 {
01046     l4_msg_regs_t *m = l4_utcb_mr_u(u);
01047     m->mr[0] = L4_THREAD_MODIFY_SENDER_OP;
01048     return l4_msgtag(L4_PROTO_THREAD, 1, 0, 0);
01049 }
01050
01051 L4_INLINE int
01052 l4_thread_modify_sender_add_u(l4_umword_t match_mask,
01053                              l4_umword_t match,
01054                              l4_umword_t del_bits,
01055                              l4_umword_t add_bits,
01056                              l4_msgtag_t *tag, l4_utcb_t *u) L4_NOTHROW
01057 {
01058     l4_msg_regs_t *m = l4_utcb_mr_u(u);
01059     unsigned w = l4_msgtag_words(*tag);
01060     if (w >= L4_UTCB_GENERIC_DATA_SIZE - 4)
01061         return -L4_ENOMEM;
01062     m->mr[w] = match_mask;
01063     m->mr[w+1] = match;
01064     m->mr[w+2] = del_bits;
01065     m->mr[w+3] = add_bits;
01066     *tag = l4_msgtag(l4_msgtag_label(*tag), w + 4, 0, 0);
01067     return 0;
01068 }
01069
01070
01071 L4_INLINE l4_msgtag_t
01072 l4_thread_modify_sender_commit_u(l4_cap_idx_t thread, l4_msgtag_t tag,
01073                                 l4_utcb_t *u) L4_NOTHROW
01074 {

```

```

01077     return l4_ipc_call(thread, u, tag, L4_IPC_NEVER);
01078 }
01079
01080 L4_INLINE l4_msgtag_t
01081 l4_thread_modify_sender_start(void) L4_NOTHROW
01082 {
01083     return l4_thread_modify_sender_start_u(l4_utcb());
01084 }
01085
01086 L4_INLINE int
01087 l4_thread_modify_sender_add(l4_umword_t match_mask,
01088                             l4_umword_t match,
01089                             l4_umword_t del_bits,
01090                             l4_umword_t add_bits,
01091                             l4_msgtag_t *tag) L4_NOTHROW
01092 {
01093     return l4_thread_modify_sender_add_u(match_mask, match,
01094                                           del_bits, add_bits, tag, l4_utcb());
01095 }
01096
01097 L4_INLINE l4_msgtag_t
01098 l4_thread_modify_sender_commit(l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW
01099 {
01100     return l4_thread_modify_sender_commit_u(thread, tag, l4_utcb());
01101 }

```

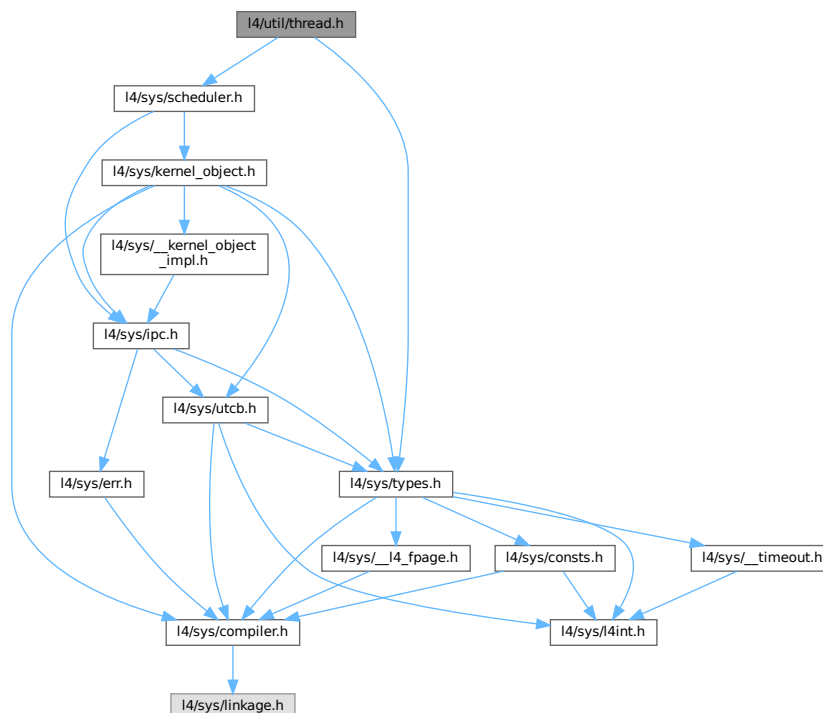
16.611 l4/util/thread.h File Reference

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.

16.611.1 Detailed Description

Low-level Thread Functions.

Date

1997

Author

Sebastian Schönberg

Definition in file [thread.h](#).

16.611.2 Macro Definition Documentation

16.611.2.1 __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](#).

16.612 thread.h

[Go to the documentation of this file.](#)

```
00001
00002 /*
00003  * (c) 2003-2009 Author(s)
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 __L4_THREAD_H
00011 #define __L4_THREAD_H
00012
00013 #include <l4/sys/types.h>
00014 #include <l4/sys/scheduler.h>
00015
00016 EXTERN_C_BEGIN
00017
00018 L4_CV long
00019 l4util_create_thread(l4_cap_idx_t id, l4_utcb_t *thread_utcb,
00020                     l4_cap_idx_t factory,
00021                     l4_umword_t pc, l4_umword_t sp, l4_cap_idx_t pager,
00022                     l4_cap_idx_t task,
00023                     l4_cap_idx_t scheduler, l4_sched_param_t scp) L4_NOTHROW;
00024
00025 EXTERN_C_END
00026
00027 #ifndef L4UTIL_THREAD_FUNC
00028 #define __L4UTIL_THREAD_FUNC(name) void L4_NORETURN name(void)
00029 #define L4UTIL_THREAD_FUNC(name) __L4UTIL_THREAD_FUNC(name)
00030 #define __L4UTIL_THREAD_STATIC_FUNC(name) static L4_NORETURN void name(void)
00031 #define L4UTIL_THREAD_STATIC_FUNC(name) __L4UTIL_THREAD_STATIC_FUNC(name)
00032 #endif
00033
00034 #endif /* __L4_THREAD_H */
```

16.613 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(int 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

```

16.614 I4/vbus/vbus.h File Reference

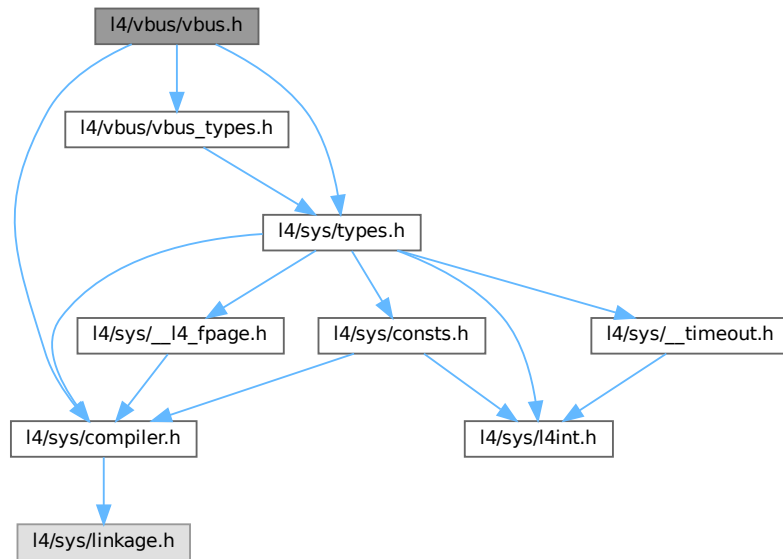
Description of the vbus C API.

```

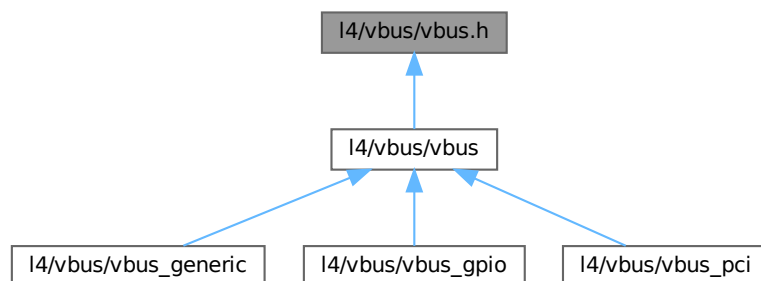
#include <l4/sys/compiler.h>
#include <l4/vbus/vbus_types.h>
#include <l4/sys/types.h>

```

Include dependency graph for vbus.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum { `L4VBUS_NULL` = (`l4_mword_t`)-1 , `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, int 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.

16.614.1 Detailed Description

Description of the vbus C API.

Definition in file [vbus.h](#).

16.614.2 Enumeration Type Documentation

16.614.2.1 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](#).

16.614.2.2 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`.

16.615 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 = (l4_mword_t)-1,
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                    int 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

```

16.616 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 }

```

16.617 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 }

```

16.618 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 };

```

16.619 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

```

16.620 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

```

16.621 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

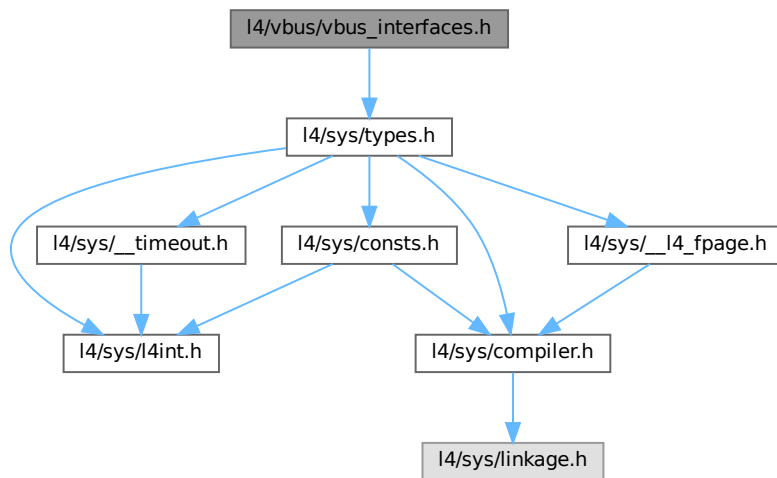
```

16.622 l4/vbus/vbus_interfaces.h File Reference

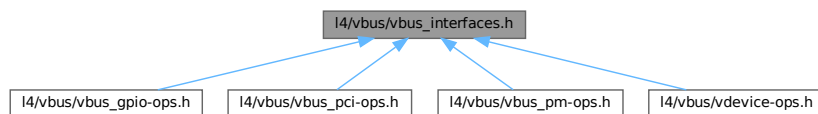
This header contains the definition of VBUS sub-interfaces and convenience functions to work with the interface IDs.

```
#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.

16.622.1 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](#).

16.622.2 Typedef Documentation

16.622.2.1 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.

16.622.3 Enumeration Type Documentation

16.622.3.1 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](#).

16.622.3.2 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](#).

16.622.4 Function Documentation

16.622.4.1 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 l4vbus_device_t .
<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](#).

16.623 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 {
00051     L4VBUS_IFACE_SHIFT = 26
00052 };
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 }

```

16.624 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

```

16.625 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 }

```

16.626 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 };

```

16.627 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_pciddev_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_pciddev_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_pciddev_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

```

16.628 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

```

16.629 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

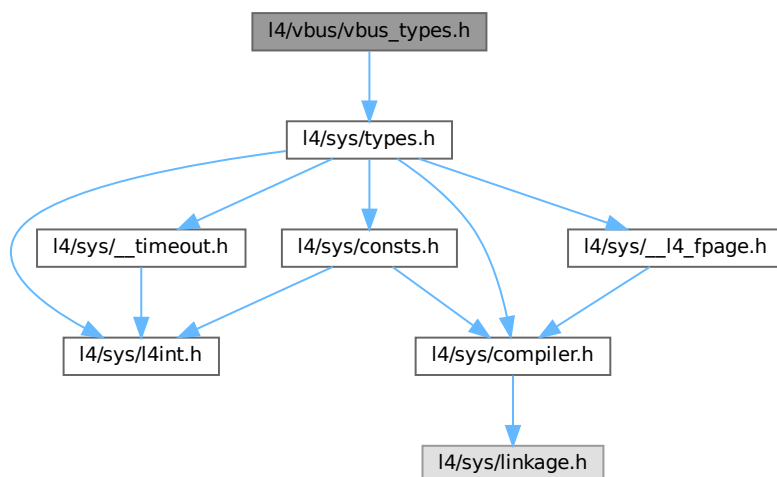
```

16.630 l4/vbus/vbus_types.h File Reference

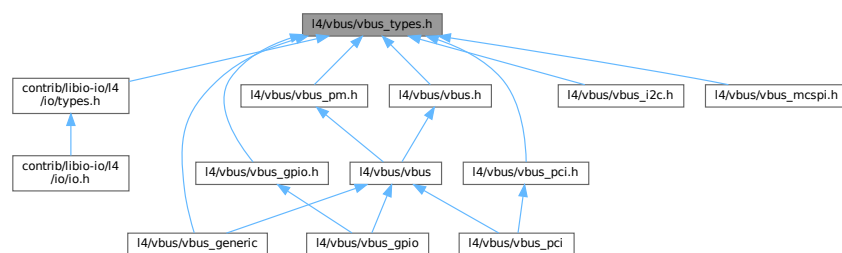
This header file contains descriptions of vbus related data types and constants.

```
#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](#).

16.630.1 Detailed Description

This header file contains descriptions of vbus related data types and constants.

Definition in file [vbus_types.h](#).

16.630.2 Enumeration Type Documentation

16.630.2.1 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](#).

16.630.2.2 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](#).

16.630.2.3 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](#).

16.631 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 };

```

16.632 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

```



```

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 }

```

16.635 l4/sys/vcpu.h File Reference

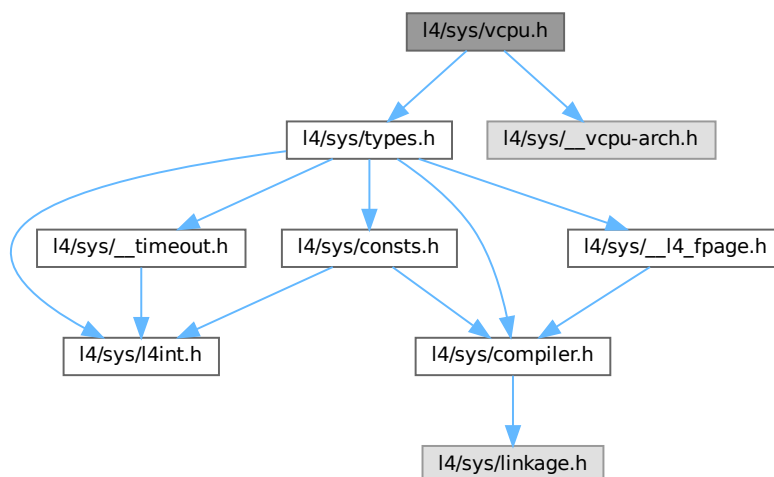
vCPU API

```

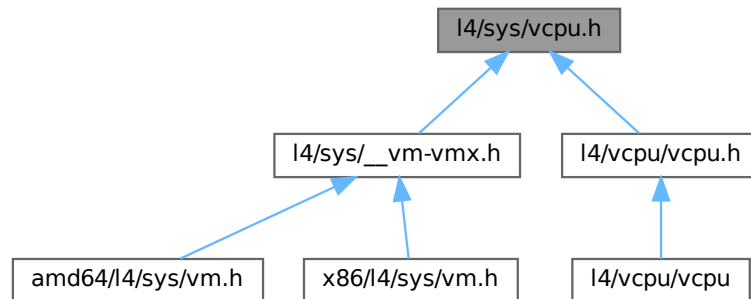
#include <l4/sys/types.h>
#include <l4/sys/__vcpu-arch.h>

```

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 }
 - enum [L4_vcpu_state_offset](#) { [L4_VCPU_OFFSET_EXT_STATE](#) = 0x400 , [L4_VCPU_OFFSET_EXT_INFOS](#) = 0x200 }
- Offsets for vCPU state layouts.*

Functions

- int [l4_vcpu_check_version](#) ([l4_vcpu_state_t](#) const *vcpu) [L4_NOTHROW](#)
Check if a vCPU state has the right version.

16.635.1 Detailed Description

vCPU API

Definition in file [vcpu.h](#).

16.635.2 Function Documentation

16.635.2.1 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

<i>vcpu</i>	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 212 of file [vcpu.h](#).

References [L4_VCPU_STATE_VERSION](#), and [l4_vcpu_state_t::version](#).

16.636 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
00189  enum L4_vcpu_state_offset
00190  {
00191      L4_VCPU_OFFSET_EXT_STATE = 0x400,
00192      L4_VCPU_OFFSET_EXT_INFOS = 0x200,
00193  };
00194
00206  L4_INLINE int
00207  l4_vcpu_check_version(l4_vcpu_state_t const *vcpu) L4_NOTHROW;
00208
00209  /* IMPLEMENTATION: ----- */
00210
00211  L4_INLINE int
00212  l4_vcpu_check_version(l4_vcpu_state_t const *vcpu) L4_NOTHROW
00213  {
00214      return vcpu->version == L4_VCPU_STATE_VERSION;
00215  }

```

16.637 l4/vcpu/vcpu.h File Reference

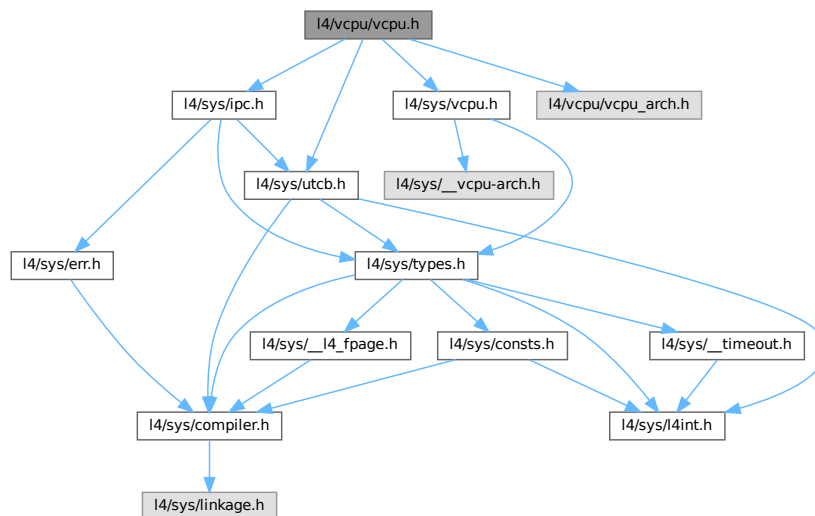
vCPU support library (C interface).

```

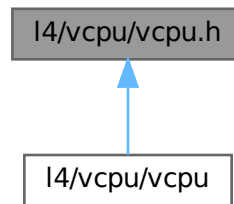
#include <l4/sys/vcpu.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
#include <l4/vcpu/vcpu_arch.h>

```

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.

16.637.1 Detailed Description

vCPU support library (C interface).

Definition in file [vcpu.h](#).

16.638 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
00253     while (1)
00254     {
00255         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

```


16.639 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

```

16.640 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\t\t\t\t\t\n\t\t\t\t\t"
    "thread:\n\t\t\t\t\t\n\t\t\t\t\t"
    #ifdef ARCH_x86

```

```

" pusha      \n\t"
" push %esp  \n\t"
" call thread_func  \n\t"
#endif
#ifdef ARCH_arm
"      push {r0-r7}          \n\t"
"      mov r0, sp           \n\t"
"      bl thread_func       \n\t"
#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
/* 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 defined(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 defined(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");
}

#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 (l4re_util_kumem_alloc(&kumem, 0, L4RE_THIS_TASK_CAP, l4re_env()->rm)
        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 vlogputc(const char c)
{
    vlogprintn(&c, 1);
}

static void thread1(void)
{
    l4_msgregs_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_msgregs_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++)
            vlogputc((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<L4Re::Util::Br_manager_hooks> 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 <l4re/c/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_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;
}

```

17.15 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 <l4/re/util/cap_alloc> // L4::Cap
#include <l4/re/dataspace>     // L4Re::Dataspace
#include <l4/re/rm>            // L4::Rm
#include <l4/re/env>           // L4::Env

```

```

#include <l4/sys/cache.h>

#include <cstring>
#include <stdio>
#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 * 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/namespace>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/re/dataspace>
#include <l4/cxx/ipc_server>

#include <l4/sys/typeinfo_svr>

#include <cstring>
#include <cstdio>
#include <unistd.h>

#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;
}

static L4Re::Util::Registry_server<> server;

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;
}

int main()
{
    L4::Cap<L4Re::Dataspace> ds;
    char *addr;

    if (!(addr = get_ds(&ds)))

```



```

    return 2;

// 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();
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
        snprintf(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(), 1);
    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)

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 >, [1168](#)

L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, [1172](#)

__iface_list

L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, [1168](#)

L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, [1172](#)

__L4UTIL_THREAD_FUNC

thread.h, [2856](#)

__check_protocols__

L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, [1168](#)

L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, [1172](#)

__kdebug_3_text

kdebug.h, [2676](#)

__kdebug_op

kdebug.h, [2677](#)

__kdebug_op_1

kdebug.h, [2678](#)

__kdebug_text

kdebug.h, [2679](#)

__vcpu-arch.h

L4_vcpu_e_field_ids, [2002](#)

L4_VCPU_STATE_VERSION, [2000](#), [2002](#), [2005](#)

~Be_file_system

L4Re::Vfs::Be_file_system, [1596](#)

~H_list_item_t

cxx::H_list_item_t< ELEM_TYPE >, [842](#)

~Unique_region

L4Re::Rm::Unique_region< T >, [1518](#)

a

L4Re::Video::Pixel_info, [1641](#), [1642](#)

access_once

cxx, [653](#)

Access_width

L4Re::Mmio_space, [1478](#)

acquire

L4Re::Inhibitor, [1464](#)

add

L4::lpc_svr::Timeout_queue, [1120](#)

L4::Thread::Modify_senders, [1282](#)

L4vcpu::State, [1742](#)

L4virtio::Svr::Driver_mem_list_t< DATA >, [1850](#)

add_block

L4virtio::Driver::Block_device, [1766](#)

add_ku_mem

L4::Task, [1255](#)

add_timeout

L4::lpc_svr::Server_iface, [1112](#)

L4::lpc_svr::Timeout_queue_hooks< BR_MAN >, [1128](#) HOOKS,

add_trusted_dataspaces

L4virtio::Svr::Device_t< DATA >, [1843](#)

AHCI driver, [74](#)

alien

L4::Thread::Attr, [1277](#)

align_to

L4::lpc::Msg, [675](#), [677](#)

all

L4::Kip::Mem_desc, [1156](#)

alloc

cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, [750](#)

cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, [757](#)

cxx::List_alloc, [850](#)

cxx::Slab< Type, Slab_size, Max_free, Alloc >, [876](#)

cxx::Slab_static< Type, Slab_size, Max_free, Alloc >, [880](#)

L4Re::Cap_alloc, [1408](#)

L4Re::Mem_alloc, [1475](#)

L4Re::Util::Counting_cap_alloc< COUNTERTYPE >, [1539](#)

alloc_buffer_demand

L4::lpc_svr::Br_manager_no_buffers, [1095](#)

L4::lpc_svr::Server_iface, [1112](#)

L4Re::Util::Br_manager, [1526](#)

alloc_descriptor

L4virtio::Driver::Virtqueue, [1799](#)

alloc_dynamic_entry

L4Re::Util::Names::Name_space, [1567](#)

alloc_fd

L4Re::Vfs::Fs, [1609](#)

alloc_max

cxx::List_alloc, [850](#)

allocate

L4Re::Dataspace, [1417](#)

L4Re::Util::Dataspace_svr, [1544](#)

amd64 Virtual Registers (UTCB), [389](#)

amd64/i4/sys/__kip-arch.h, [1997](#)

amd64/i4/sys/__vcpu-arch.h, [1998](#), [2000](#)

amd64/i4/sys/cache.h, [2539](#), [2540](#)

- amd64/l4/sys/consts.h, 2555
- amd64/l4/sys/ktrace_events.h, 2006
- amd64/l4/sys/l4int.h, 2698, 2699
- amd64/l4/sys/linkage.h, 2016
- amd64/l4/sys/segment.h, 1974, 1979
- amd64/l4/sys/utcb.h, 2750, 2752
- amd64/l4/sys/vm.h, 2021
- amd64/l4/util/bitops_arch.h, 2023
- amd64/l4/util/cpu.h, 2030, 2032
- amd64/l4/util/idt.h, 1940, 1942
- amd64/l4/util/irq.h, 2173, 2175
- amd64/l4/util/l4_macros.h, 2035, 2036
- amd64/l4/util/mbi_argv.h, 2038, 2039
- amd64/l4/util/perform.h, 1944, 1945
- amd64/l4/util/port_io.h, 1988, 1989
- amd64/l4/util/rdtsc.h, 1956, 1958
- amd64/l4/util/spin.h, 1965
- amd64/l4/util/util.h, 1967, 1969
- amd64/l4f/l4/sys/ipc.h, 2655
- amd64/l4f/l4/sys/segment.h, 1980, 1983
- amd64/l4f/l4/util/port_io.h, 1989, 1990
- Application and Server Building Blocks, 23
- Arch
 - L4::Kip::Mem_desc, 1155
- Arch_acpi_nvs
 - L4::Kip::Mem_desc, 1155
- Arch_acpi_tables
 - L4::Kip::Mem_desc, 1155
- Arch_sub_type_common
 - L4::Kip::Mem_desc, 1154
- Area
 - L4Re::Rm, 1503
- ARM Virtual Registers (UTCB), 382
- arm/l4/sys/__kip-arch.h, 1998
- arm/l4/sys/__vcpu-arch.h, 2001, 2003
- arm/l4/sys/atomic.h, 2781
- arm/l4/sys/cache.h, 2541, 2542
- arm/l4/sys/consts.h, 2556
- arm/l4/sys/ktrace_events.h, 2009
- arm/l4/sys/l4int.h, 2699, 2700
- arm/l4/sys/linkage.h, 2017
- arm/l4/sys/mem_op.h, 2018, 2020
- arm/l4/sys/task.h, 2738
- arm/l4/sys/thread.h, 2845
- arm/l4/sys/utcb.h, 2753, 2754
- arm/l4/sys/vm, 2773
- arm/l4/sys/vm.h, 2021, 2022
- arm/l4/util/bitops_arch.h, 2026, 2027
- arm/l4/util/cpu.h, 2033
- arm/l4/util/irq.h, 2175, 2176
- arm/l4/util/l4_macros.h, 2036
- arm/l4/util/mbi_argv.h, 2040, 2041
- arm/l4f/l4/sys/ipc.h, 2655, 2656
- arm/l4f/l4/sys/syscall_defs.h, 2043
- arm_smccc.h
 - l4_arm_smccc_call, 2537
- assert.h
 - l4_assert, 2777
- assign_dma_domain
 - L4vbus::Vbus, 1735, 1736
- associate
 - L4Re::Dma_space, 1432
- AT_BASE
 - ELF binary format, 607
- AT_EGID
 - ELF binary format, 607
- AT_ENTRY
 - ELF binary format, 607
- AT_EUID
 - ELF binary format, 607
- AT_EXECFD
 - ELF binary format, 607
- AT_FLAGS
 - ELF binary format, 607
- AT_GID
 - ELF binary format, 607
- AT_IGNORE
 - ELF binary format, 607
- AT_L4_AUX
 - ELF binary format, 607
- AT_L4_ENV
 - ELF binary format, 607
- AT_NOTELF
 - ELF binary format, 607
- AT_NULL
 - ELF binary format, 607
- AT_PAGESZ
 - ELF binary format, 607
- AT_PHDR
 - ELF binary format, 607
- AT_PHEMT
 - ELF binary format, 607
- AT_PHNUM
 - ELF binary format, 607
- AT_UID
 - ELF binary format, 607
- Atomic Instructions, 569
 - l4util_add16, 571
 - l4util_add16_res, 571
 - l4util_add32, 571
 - l4util_add32_res, 571
 - l4util_add8, 572
 - l4util_add8_res, 572
 - l4util_and16, 572
 - l4util_and16_res, 573
 - l4util_and32, 573
 - l4util_and32_res, 573
 - l4util_and8, 574
 - l4util_and8_res, 574
 - l4util_atomic_add, 574
 - l4util_atomic_inc, 575
 - l4util_cmpxchg, 575
 - l4util_cmpxchg16, 576
 - l4util_cmpxchg32, 576
 - l4util_cmpxchg64, 577
 - l4util_cmpxchg8, 577

- l4util_dec16, [578](#)
- l4util_dec16_res, [578](#)
- l4util_dec32, [578](#)
- l4util_dec32_res, [578](#)
- l4util_dec8, [579](#)
- l4util_dec8_res, [579](#)
- l4util_inc16, [579](#)
- l4util_inc16_res, [580](#)
- l4util_inc32, [580](#)
- l4util_inc32_res, [580](#)
- l4util_inc8, [580](#)
- l4util_inc8_res, [581](#)
- l4util_or16, [581](#)
- l4util_or16_res, [581](#)
- l4util_or32, [582](#)
- l4util_or32_res, [582](#)
- l4util_or8, [582](#)
- l4util_or8_res, [582](#)
- l4util_sub16, [583](#)
- l4util_sub16_res, [583](#)
- l4util_sub32, [583](#)
- l4util_sub32_res, [584](#)
- l4util_sub8, [584](#)
- l4util_sub8_res, [584](#)
- l4util_xchg, [585](#)
- l4util_xchg16, [585](#)
- l4util_xchg32, [585](#)
- l4util_xchg8, [586](#)
- attach
 - L4Re::Rm, [1504](#), [1505](#)
 - L4Re::Util::Event_buffer_t< PAYLOAD >, [1555](#)
- Attach_flags
 - L4Re::Rm::F, [1513](#)
- Attach_mask
 - L4Re::Rm::F, [1513](#)
- Attr
 - L4::Thread::Attr, [1277](#)
- Attribute
 - L4Re::Dma_space, [1431](#)
- Attributes
 - L4Re::Dma_space, [1431](#)
- atype
 - Elf32_Auxv, [894](#)
 - Elf64_Auxv, [904](#)
- Auxiliary data, [508](#)
- avail
 - cxx::List_alloc, [851](#)
- avail_align
 - L4virtio::Virtqueue, [1882](#)
- avail_size
 - L4virtio::Virtqueue, [1882](#)
- Avl_map
 - cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >, [732](#)
- ax
 - l4_vcpu_regs_t, [1378](#)
- b
 - L4Re::Video::Pixel_info, [1642](#)
- backtrace.h
 - l4util_backtrace, [2791](#)
- Bad_address
 - L4virtio::Svr::Bad_descriptor, [1810](#)
- Bad_descriptor
 - L4virtio::Svr::Bad_descriptor, [1810](#)
- Bad_flags
 - L4virtio::Svr::Bad_descriptor, [1810](#)
- Bad_next
 - L4virtio::Svr::Bad_descriptor, [1810](#)
- Bad_rights
 - L4virtio::Svr::Bad_descriptor, [1810](#)
- Bad_size
 - L4virtio::Svr::Bad_descriptor, [1810](#)
- Base API, [129](#)
- Base_avl_set
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [794](#)
- Basic Macros, [132](#)
 - l4_align_stack_for_direct_fncall, [135](#)
 - L4_EXPORT, [134](#)
 - L4_HIDDEN, [134](#)
 - l4_infinite_loop, [135](#)
 - L4_NOTHROW, [134](#)
- Be_file_system
 - L4Re::Vfs::Be_file_system, [1596](#)
- begin
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [795](#)
 - cxx::Bits::Bst< Node, Get_key, Compare >, [812](#), [813](#)
- Bidirectional
 - L4Re::Dma_space, [1432](#)
- bind
 - L4::lcu, [990](#)
 - L4::lommu, [1004](#)
 - L4::Thread::Attr, [1278](#)
- bind_notification_irq
 - L4virtio::Driver::Device, [1775](#)
- bind_thread
 - L4::Rcv_endpoint, [1206](#)
- bit
 - cxx::Bitmap_base, [783](#)
- Bit Manipulation, [586](#)
 - l4util_bsf, [587](#)
 - l4util_bsr, [588](#)
 - l4util_btc, [588](#)
 - l4util_btr, [588](#)
 - l4util_bts, [589](#)
 - l4util_clear_bit, [590](#)
 - l4util_complement_bit, [590](#)
 - l4util_find_first_set_bit, [591](#)
 - l4util_find_first_zero_bit, [591](#)
 - l4util_next_power2, [592](#)
 - l4util_set_bit, [592](#)
 - l4util_test_bit, [592](#)
- bit_index
 - cxx::Bitmap_base, [783](#)

- Bitmap graphics and fonts, [593](#)
- bitmap.h
 - gfxbitmap_bmap, [2299](#)
 - gfxbitmap_color_pix_t, [2299](#)
 - gfxbitmap_color_t, [2299](#)
 - gfxbitmap_convert_color, [2300](#)
 - gfxbitmap_copy, [2300](#)
 - gfxbitmap_fill, [2300](#)
 - gfxbitmap_set, [2301](#)
 - pSLIM_BMAP_START_LSB, [2298](#)
- Bits
 - cxx::Bitfield< T, LSB, MSB >, [762](#)
- bits_per_pixel
 - L4Re::Video::Pixel_info, [1642](#)
- Bits_type
 - cxx::Bitfield< T, LSB, MSB >, [761](#)
- Block_dev_base
 - L4virtio::Svr::Block_dev_base< Ds_data >, [1815](#)
- Block_device::Device_discard_feature, [709](#)
- Block_device::Device_mgr< DEV, FACTORY >, [710](#)
- Block_device::Dma_region_info, [711](#)
- Block_device::Errand::Errand, [711](#)
 - expired, [714](#)
- Block_device::Errand::Poll_errand, [715](#)
 - expired, [717](#)
- Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, bool >, [718](#)
- Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, true >, [719](#)
- Block_device::Inout_block, [720](#)
- Block_device::Inout_memory< DEV >, [721](#)
- Block_device::Mem_region_info, [722](#)
- Block_device::Partition_info, [722](#)
- Block_device::Partition_reader< DEV >, [723](#)
- Block_device::Partitioned_device< BASE_DEV >, [724](#)
- Block_device::Pending_request, [725](#)
 - fail_request, [726](#)
 - handle_request, [726](#)
 - is_owner, [727](#)
- Block_device::Pending_request::Owner, [727](#)
- Block_device::Simple_request_queue, [728](#)
- Bootloader
 - L4::Kip::Mem_desc, [1155](#)
- Bootstrap, the %L4 kernel bootstrapper, [77](#)
- bp
 - l4_vcpu_regs_t, [1378](#)
- Br_bytes
 - L4::lpc::Msg, [675](#)
- buf
 - L4Re::Util::Event_buffer_t< PAYLOAD >, [1555](#)
- buf_cp_in
 - L4::lpc, [668](#)
- buf_cp_out
 - L4::lpc, [669](#)
- buf_in
 - L4::lpc, [669](#)
- buffer
 - L4Re::Util::Event_t< PAYLOAD >, [1559](#)
- Buffer Registers (BRs), [383](#)
 - L4_BDR_IO_SHIFT, [384](#)
 - L4_BDR_MEM_SHIFT, [384](#)
 - L4_BDR_OBJ_SHIFT, [384](#)
 - l4_buffer_desc_consts_t, [383](#)
- Bufferable
 - L4Re::Dataspace::F, [1425](#)
- bus_cap
 - L4vbus::Device, [1687](#)
- bx
 - l4_vcpu_regs_t, [1378](#)
- bytes_per_pixel
 - L4Re::Video::Pixel_info, [1643](#)
- c
 - L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, [1168](#)
 - L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, [1172](#)
- C++ Exceptions, [509](#)
- C++ IPC Interface Definition., [136](#)
- C_bits
 - cxx::Bitmap_base, [782](#)
- Cache Consistency, [137](#)
 - l4_cache_clean_data, [137](#)
 - l4_cache_coherent, [138](#)
 - l4_cache_dma_coherent, [138](#)
 - l4_cache_flush_data, [139](#)
 - l4_cache_inv_data, [139](#)
- Cache_buffered
 - L4Re::Rm::F, [1514](#)
- Cache_normal
 - L4Re::Rm::F, [1514](#)
- Cache_uncached
 - L4Re::Rm::F, [1514](#)
- Cacheable
 - L4Re::Dataspace::F, [1425](#)
- Caching_mask
 - L4Re::Dataspace::F, [1425](#)
 - L4Re::Rm::F, [1514](#)
- Caching_shift
 - L4Re::Dataspace::F, [1424](#)
 - L4Re::Rm, [1504](#)
- call
 - L4::Arm_smccc, [916](#)
 - L4::lpc::loststream, [1027](#)
- Cap
 - L4::Cap< T >, [928](#)
 - L4::lpc::Cap< T >, [1017](#)
- cap
 - L4::Cap_base, [933](#)
 - L4::Invalid_capability, [999](#)
 - L4::Kobject, [1164](#)
- cap_alloc
 - L4Re Capability API, [519](#)
- Cap_base
 - L4::Cap_base, [932](#)
- cap_cast
 - L4, [660](#)

- cap_dynamic_cast
 - L4, [661](#)
- cap_equal
 - L4::Task, [1256](#)
- Cap_mask
 - L4::lpc::Cap< T >, [1017](#)
- cap_received
 - L4::lpc::Gen_fpage< T >, [1020](#)
- cap_reinterpret_cast
 - L4, [662](#)
- Cap_type
 - L4::Cap_base, [932](#)
- cap_valid
 - L4::Task, [1256](#)
- Capabilities, [140](#)
 - L4_BASE_ARM_SMCCC_CAP, [142](#)
 - L4_BASE_CAPS_LAST, [142](#)
 - L4_BASE_DEBUGGER_CAP, [142](#)
 - L4_BASE_FACTORY_CAP, [142](#)
 - L4_BASE_ICU_CAP, [142](#)
 - L4_BASE_IOMMU_CAP, [142](#)
 - L4_BASE_LOG_CAP, [142](#)
 - L4_BASE_PAGER_CAP, [142](#)
 - L4_BASE_SCHEDULER_CAP, [142](#)
 - L4_BASE_TASK_CAP, [142](#)
 - L4_BASE_THREAD_CAP, [142](#)
 - l4_cap_consts_t, [141](#)
 - l4_cap_idx_t, [141](#)
 - L4_CAP_MASK, [141](#)
 - L4_CAP_OFFSET, [141](#)
 - L4_CAP_SHIFT, [141](#)
 - L4_CAP_SIZE, [141](#)
 - l4_capability_equal, [142](#)
 - l4_default_caps_t, [142](#)
 - L4_INVALID_CAP, [141](#)
 - l4_is_invalid_cap, [143](#)
 - l4_is_valid_cap, [143](#)
- Capabilities and Naming, [18](#)
- capability
 - L4_DISABLE_COPY, [2549](#)
- Capability allocator, [456](#)
 - l4re_util_cap_last, [457](#)
- Caps
 - L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, [1294](#)
- caps
 - l4re_env_t, [1660](#)
- cast
 - L4vcpu::Vcpu, [1746](#)
- cfg_read
 - L4vbus::Pci_dev, [1717](#)
 - L4vbus::Pci_host_bridge, [1723](#)
- cfg_write
 - L4vbus::Pci_dev, [1717](#)
 - L4vbus::Pci_host_bridge, [1723](#)
- chain
 - L4::Irq_mux, [1147](#)
- change_mode
 - L4::Uart, [1331](#)
- change_queue_config
 - L4virtio::Svr::Dev_config, [1830](#)
- char_avail
 - L4::Uart, [1331](#)
- check_size
 - L4::lpc::Msg, [678](#)
- chkcap
 - L4Re, [688](#)
- chkipc
 - L4Re, [689](#)
- chksys
 - L4Re, [690](#), [691](#)
- Chunks, [532](#)
 - l4shmc_add_chunk, [533](#)
 - l4shmc_chunk_capacity, [533](#)
 - l4shmc_chunk_ptr, [534](#)
 - l4shmc_chunk_signal, [534](#)
 - l4shmc_get_chunk, [535](#)
 - l4shmc_get_chunk_to, [535](#)
 - l4shmc_iterate_chunk, [536](#)
- clamp
 - Small C++ Template Library, [559](#)
- Class
 - L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, [1168](#)
 - L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, [1172](#)
- clear
 - cxx::Bits::Basic_list< POLICY >, [808](#)
 - L4::Types::Flags< BITS_ENUM, UNDERLYING >, [1319](#)
 - L4drivers::Register_tmpl< BITS, BLOCK >, [1401](#)
 - L4Re::Dataspace, [1417](#)
 - L4Re::Util::Dataspace_svr, [1544](#)
 - L4vcpu::State, [1742](#)
- clear_bit
 - cxx::Bitmap_base, [784](#)
- Coherent
 - L4Re::Dma_space, [1432](#)
- Color_component
 - L4Re::Video::Color_component, [1628](#)
- Com_error
 - L4::Com_error, [943](#)
- Comfortable Command Line Parsing, [597](#)
 - parse_cmdline, [597](#)
- config_get
 - L4vbus::Gpio_pin, [1704](#)
- config_pad
 - L4vbus::Gpio_module, [1696](#)
 - L4vbus::Gpio_pin, [1704](#)
- config_pull
 - L4vbus::Gpio_pin, [1705](#)
- config_queue
 - L4virtio::Device, [1758](#)
 - L4virtio::Driver::Device, [1776](#)
- Cons, the Console Multiplexer, [73](#)
- Console API, [510](#)

- consumed
 - L4virtio::Svr::Virtqueue, [1873](#)
- Consumer, [536](#), [547](#)
 - l4shmc_chunk_consumed, [537](#)
 - l4shmc_chunk_size, [537](#)
 - l4shmc_chunk_try_to_take_for_reading, [538](#)
 - l4shmc_enable_chunk, [538](#)
 - l4shmc_enable_signal, [548](#)
 - l4shmc_is_chunk_ready, [538](#)
 - l4shmc_wait_any, [549](#)
 - l4shmc_wait_any_to, [549](#)
 - l4shmc_wait_any_try, [550](#)
 - l4shmc_wait_chunk, [539](#)
 - l4shmc_wait_chunk_to, [539](#)
 - l4shmc_wait_chunk_try, [540](#)
 - l4shmc_wait_signal, [550](#)
 - l4shmc_wait_signal_to, [550](#)
 - l4shmc_wait_signal_try, [551](#)
- contains
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1859](#)
- Continuous
 - L4Re::Mem_alloc, [1475](#)
- contrib/libio-io/l4/io/io.h, [2044](#), [2046](#)
- contrib/libio-io/l4/io/types.h, [2280](#)
- control
 - L4::Thread, [1266](#)
- Conventional
 - L4::Kip::Mem_desc, [1155](#)
- copy
 - L4::Cap< T >, [929](#)
 - L4::Cap_base, [934](#)
 - L4Re::Util::Dataspace_svr, [1545](#)
- copy_in
 - L4Re::Dataspace, [1418](#)
- copy_receive_cap
 - L4Re::Util::Names::Name_space, [1567](#)
- copy_to
 - L4virtio::Svr::Data_buffer, [1824](#)
- count
 - L4::Kip::Mem_desc, [1157](#), [1158](#)
- Counting_cap_alloc
 - L4Re::Util::Counting_cap_alloc< COUNTERTYPE >, [1539](#)
- CPU related functions, [594](#)
 - l4util_cpu_capabilities, [595](#)
 - l4util_cpu_capabilities_nocheck, [595](#)
 - l4util_cpu_has_cpuid, [596](#)
- cpu_allow_shutdown
 - L4::Platform_control, [1194](#)
- cpu_disable
 - L4::Platform_control, [1195](#)
- cpu_enable
 - L4::Platform_control, [1195](#)
- create
 - L4::Factory, [975](#), [976](#)
- create_buffer
 - L4Re::Video::Goos, [1634](#)
- create_factory
 - L4::Factory, [977](#)
- create_gate
 - L4::Factory, [978](#)
- create_task
 - L4::Factory, [979](#)
- create_view
 - L4Re::Video::Goos, [1635](#)
- current_flags
 - L4virtio::Svr::Request_processor, [1864](#)
- cx
 - l4_vcpu_regs_t, [1378](#)
- cxx, [651](#)
 - access_once, [653](#)
 - write_now, [654](#)
- cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >, [729](#)
 - Avl_map, [732](#)
 - insert, [733](#)
 - operator[], [734](#)
- cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >, [735](#)
- cxx::Avl_tree< Node, Get_key, Compare >, [739](#)
 - insert, [744](#)
 - Iterator, [744](#)
 - remove, [745](#)
- cxx::Avl_tree_node, [746](#)
- cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, [748](#)
 - alloc, [750](#)
 - free, [751](#)
 - free_objects, [752](#)
 - max_free_slabs, [750](#)
 - object_size, [750](#)
 - objects_per_slab, [750](#)
 - slab_size, [750](#)
 - total_objects, [753](#)
- cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i, [754](#)
- cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, [754](#)
 - alloc, [757](#)
 - free, [757](#)
 - free_objects, [758](#)
 - max_free_slabs, [756](#)
 - object_size, [756](#)
 - objects_per_slab, [756](#)
 - slab_size, [756](#)
 - total_objects, [758](#)
- cxx::Bitfield< T, LSB, MSB >, [759](#)
 - Bits, [762](#)
 - Bits_type, [761](#)
 - get, [762](#)
 - get_unshifted, [763](#)
 - Low_mask, [762](#)
 - Lsb, [762](#)
 - Mask, [762](#)
 - Masks, [762](#)
 - Msb, [762](#)

- set, [764](#)
- set_dirty, [764](#)
- set_unshifted, [765](#)
- set_unshifted_dirty, [766](#)
- Shift_type, [761](#)
- val, [767](#)
- val_dirty, [768](#)
- val_unshifted, [770](#)
- cxx::Bitfield< T, LSB, MSB >::Value< TT >, [771](#)
- cxx::Bitfield< T, LSB, MSB >::Value_base< TT >, [773](#)
- cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >, [774](#)
- cxx::Bitmap< BITS >, [775](#)
 - scan_zero, [779](#)
- cxx::Bitmap_base, [779](#)
 - bit, [783](#)
 - bit_index, [783](#)
 - C_bits, [782](#)
 - clear_bit, [784](#)
 - operator[], [784](#), [785](#)
 - scan_zero, [785](#)
 - set_bit, [786](#)
 - W_bits, [782](#)
 - word_index, [786](#)
- cxx::Bitmap_base::Bit, [787](#)
- cxx::Bitmap_base::Char< BITS >, [788](#)
- cxx::Bitmap_base::Word< BITS >, [788](#)
- cxx::Bits, [655](#)
- cxx::Bits::Avl_map_get_key< KEY_TYPE >, [790](#)
- cxx::Bits::Avl_set_get_key< KEY_TYPE >, [790](#)
- cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [791](#)
 - Base_avl_set, [794](#)
 - begin, [795](#)
 - E_exist, [794](#)
 - E_inval, [794](#)
 - E_noent, [794](#)
 - E_nomem, [794](#)
 - end, [796](#), [797](#)
 - erase, [798](#)
 - find_node, [798](#)
 - insert, [799](#)
 - lower_bound_node, [800](#)
 - rbegin, [801](#)
 - remove, [802](#)
 - rend, [803](#)
- cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node, [804](#)
 - operator->, [806](#)
 - operator*, [806](#)
 - valid, [806](#)
- cxx::Bits::Basic_list< POLICY >, [807](#)
 - clear, [808](#)
 - iter, [808](#)
- cxx::Bits::Bst< Node, Get_key, Compare >, [809](#)
 - begin, [812](#), [813](#)
 - dir, [813](#), [814](#)
 - end, [815](#)
- find, [816](#)
- find_node, [816](#)
- lower_bound_node, [817](#)
- rbegin, [818](#), [819](#)
- remove_all, [820](#)
- remove_tree, [821](#)
- rend, [822](#)
- cxx::Bits::Bst_node, [823](#)
- cxx::Bits::Direction, [825](#)
 - Direction_e, [826](#)
 - L, [826](#)
 - N, [826](#)
 - operator!, [826](#)
 - R, [826](#)
- cxx::Bits::Smart_ptr_list< ITEM >, [827](#)
 - pop_front, [830](#)
- cxx::Bits::Smart_ptr_list_item< T, STORE_T >, [830](#)
- cxx::H_list< T, POLICY >, [832](#)
 - erase, [835](#)
 - insert, [835](#)
 - insert_after, [836](#)
 - insert_before, [837](#)
 - iter, [837](#)
 - pop_front, [838](#)
 - remove, [839](#)
 - replace, [839](#)
- cxx::H_list_item_t< ELEM_TYPE >, [840](#)
 - ~H_list_item_t, [842](#)
 - H_list_item_t, [842](#)
- cxx::H_list_t< T >, [843](#)
- cxx::List< D, Alloc >, [846](#)
 - operator[], [847](#)
- cxx::List< D, Alloc >::Iter, [848](#)
- cxx::List_alloc, [849](#)
 - alloc, [850](#)
 - alloc_max, [850](#)
 - avail, [851](#)
 - free, [851](#)
 - List_alloc, [849](#)
- cxx::List_item, [852](#)
 - push_back, [853](#)
 - push_front, [854](#)
 - remove, [855](#)
- cxx::List_item::Iter, [856](#)
- cxx::List_item::T_iter< T, Poly >, [857](#)
- cxx::Lt_functor< Obj >, [859](#)
- cxx::New_allocator< _Type >, [859](#)
- cxx::Nothrow, [860](#)
- cxx::Pair< First, Second >, [861](#)
 - Pair, [862](#)
- cxx::Pair_first_compare< Cmp, Typ >, [863](#)
 - operator(), [864](#)
 - Pair_first_compare, [863](#)
- cxx::Ref_obj_list_item< T >, [864](#)
- cxx::Ref_ptr< T, CNT >, [866](#)
 - get, [869](#)
 - ptr, [869](#)
 - Ref_ptr, [868](#)

- release, [869](#)
- cxx::S_list< T, POLICY >, [870](#)
 - pop_front, [872](#)
- cxx::Slab< Type, Slab_size, Max_free, Alloc >, [873](#)
 - alloc, [876](#)
 - free, [876](#)
- cxx::Slab_static< Type, Slab_size, Max_free, Alloc >, [877](#)
 - alloc, [880](#)
- cxx::static_vector< T, IDX >, [881](#)
- cxx::String, [882](#)
 - find, [885](#)
 - from_dec, [886](#)
 - from_hex, [886](#)
 - starts_with, [887](#)
 - String, [885](#)
- cxx::Weak_ref< T >, [888](#)
- cxx::Weak_ref_base, [891](#)
- d_tag
 - Elf32_Dyn, [895](#)
 - Elf64_Dyn, [905](#)
- data
 - L4::lpc::Varg, [1078](#)
- Data_buffer
 - L4virtio::Svr::Data_buffer, [1823](#)
- data_size
 - L4virtio::Svr::Block_request< Ds_data >, [1820](#)
- data_space
 - L4Re::Vfs::Be_file, [1593](#)
 - L4Re::Vfs::Regular_file, [1622](#)
- Dataspace interface, [461](#)
 - l4re_ds_allocate, [462](#)
 - l4re_ds_clear, [463](#)
 - l4re_ds_copy_in, [463](#)
 - L4RE_DS_F_BUFFERABLE, [462](#)
 - L4RE_DS_F_CACHEABLE, [462](#)
 - L4RE_DS_F_CACHING_MASK, [462](#)
 - L4RE_DS_F_CACHING_SHIFT, [462](#)
 - L4RE_DS_F_NORMAL, [462](#)
 - L4RE_DS_F_UNCACHEABLE, [462](#)
 - l4re_ds_flags, [464](#)
 - l4re_ds_info, [464](#)
 - l4re_ds_map_flags, [462](#)
 - l4re_ds_size, [465](#)
- debug
 - L4Re::Debug_obj, [1428](#)
- Debug interface, [465](#)
 - l4re_debug_obj_debug, [466](#)
- Debugging API, [510](#)
- dec_refcnt
 - L4::Kobject, [1165](#)
- Dedicated
 - L4::Kip::Mem_desc, [1155](#)
- delete_buffer
 - L4Re::Video::Goos, [1635](#)
- delete_obj
 - L4::Task, [1257](#)
- delete_view
 - L4Re::Video::Goos, [1635](#)
- Demand
 - L4::Kobject_typeid< T >, [1176](#)
 - L4::Type_info::Demand, [1291](#)
- demand
 - L4::Kobject_typeid< T >, [1177](#)
- Deprecated List, [81](#)
- desc
 - L4virtio::Driver::Virtqueue, [1799](#)
 - L4virtio::Svr::Virtqueue, [1874](#)
 - L4virtio::Svr::Virtqueue::Head_desc, [1877](#)
- desc_align
 - L4virtio::Virtqueue, [1883](#)
- desc_avail
 - L4virtio::Svr::Virtqueue, [1874](#)
- desc_size
 - L4virtio::Virtqueue, [1883](#)
- detach
 - L4::Irq, [1134](#)
 - L4Re::Rm, [1506](#), [1507](#)
 - L4Re::Util::Event_buffer_t< PAYLOAD >, [1556](#)
- Detach_again
 - L4Re::Rm, [1504](#)
- Detach_exact
 - L4Re::Rm, [1504](#)
- Detach_flags
 - L4Re::Rm, [1503](#)
- Detach_free
 - L4Re::Rm::F, [1514](#)
- Detach_keep
 - L4Re::Rm, [1504](#)
- Detach_overlap
 - L4Re::Rm, [1504](#)
- Detach_result
 - L4Re::Rm, [1504](#)
- Detached_ds
 - L4Re::Rm, [1504](#)
- Dev_config
 - L4virtio::Svr::Dev_config, [1828](#), [1829](#)
- dev_handle
 - L4vbus::Device, [1687](#)
- Device
 - L4vbus::Device, [1686](#)
- device
 - L4vbus::Device, [1688](#)
- device_by_hid
 - L4vbus::Device, [1688](#)
- device_config
 - L4virtio::Device, [1759](#)
- device_error
 - L4virtio::Svr::Device_t< DATA >, [1844](#)
- device_notification_irq
 - L4virtio::Device, [1759](#)
- device_notify_irq
 - L4virtio::Svr::Device_t< DATA >, [1844](#)
- DF_1_CONFALT
 - ELF binary format, [608](#)
- DF_1_DIRECT

- ELF binary format, [608](#)
- DF_1_DISPRELDNE
 - ELF binary format, [608](#)
- DF_1_DISPRELPND
 - ELF binary format, [608](#)
- DF_1_ENDFILTEE
 - ELF binary format, [608](#)
- DF_1_GLOBAL
 - ELF binary format, [608](#)
- DF_1_GROUP
 - ELF binary format, [608](#)
- DF_1_INITFIRST
 - ELF binary format, [608](#)
- DF_1_INTERPOSE
 - ELF binary format, [608](#)
- DF_1_LOADFLTR
 - ELF binary format, [608](#)
- DF_1_NODEFLIB
 - ELF binary format, [608](#)
- DF_1_NODELETE
 - ELF binary format, [608](#)
- DF_1_NODUMP
 - ELF binary format, [608](#)
- DF_1_NOOPEN
 - ELF binary format, [608](#)
- DF_1_NOW
 - ELF binary format, [608](#)
- DF_1_ORIGIN
 - ELF binary format, [608](#)
- DF_BIND_NOW
 - ELF binary format, [610](#)
- DF_ORIGIN
 - ELF binary format, [610](#)
- DF_P1_GROUPPERM
 - ELF binary format, [610](#)
- DF_P1_LAZYLOAD
 - ELF binary format, [610](#)
- DF_STATIC_TLS
 - ELF binary format, [610](#)
- DF_SYMBOLIC
 - ELF binary format, [610](#)
- DF_TEXTREL
 - ELF binary format, [610](#)
- di
 - l4_vcpu_regs_t, [1378](#)
- dir
 - cxx::Bits::Bst< Node, Get_key, Compare >, [813](#), [814](#)
- Direction
 - L4Re::Dma_space, [1432](#)
- Direction_e
 - cxx::Bits::Direction, [826](#)
- disable
 - L4virtio::Virtqueue, [1884](#)
- disable_notify
 - L4virtio::Svr::Virtqueue, [1875](#)
- disassociate
 - L4Re::Dma_space, [1433](#)
- dispatch
 - L4::Basic_registry, [920](#)
 - L4::Epiface, [960](#)
 - L4::Epiface_t< Derived, IFACE, BASE, bool >, [966](#)
 - L4::lrqp_t< Derived, BASE, bool >, [1152](#)
 - L4::Server_object, [1235](#), [1236](#)
- dispatch_meta_request
 - L4::Server_object_t< IFACE, BASE >, [1241](#)
- DMA space, [191](#)
- DMA Space Interface, [457](#)
 - l4re_dma_space_associate, [458](#)
 - l4re_dma_space_disassociate, [459](#)
 - l4re_dma_space_map, [459](#)
 - l4re_dma_space_t, [458](#)
 - l4re_dma_space_unmap, [460](#)
- dma_space.h
 - L4RE_DMA_SPACE_BIDIRECTIONAL, [2313](#)
 - L4RE_DMA_SPACE_COHERENT, [2314](#)
 - l4re_dma_space_direction, [2313](#)
 - L4RE_DMA_SPACE_FROM_DEVICE, [2313](#)
 - L4RE_DMA_SPACE_NONE, [2313](#)
 - L4RE_DMA_SPACE_PHYS_SPACE, [2314](#)
 - l4re_dma_space_space_attribs, [2313](#)
 - L4RE_DMA_SPACE_TO_DEVICE, [2313](#)
- done
 - L4virtio::Svr::Data_buffer, [1824](#)
- down
 - L4::Semaphore, [1226](#)
- drive_cylinders
 - l4util_mb_drive_t, [1676](#)
- drive_mode
 - l4util_mb_drive_t, [1676](#)
- drive_number
 - l4util_mb_drive_t, [1676](#)
- driver_acknowledge
 - L4virtio::Driver::Device, [1777](#)
- driver_connect
 - L4virtio::Driver::Device, [1778](#)
- Driver_mem_region_t
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1858](#)
- drv_base
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1860](#)
- ds
 - L4virtio::Svr::Dev_config, [1830](#)
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1860](#)
- Ds_map_mask
 - L4Re::Rm::F, [1514](#)
- ds_offset
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1860](#)
- DT_BIND_NOW
 - ELF binary format, [611](#)
- DT_DEBUG
 - ELF binary format, [611](#)
- DT_ENCODING

- ELF binary format, [611](#)
- DT_FINI
 - ELF binary format, [611](#)
- DT_FINI_ARRAY
 - ELF binary format, [611](#)
- DT_FINI_ARRAYSZ
 - ELF binary format, [611](#)
- DT_FLAGS
 - ELF binary format, [611](#)
- DT_HASH
 - ELF binary format, [610](#)
- DT_HIOS
 - ELF binary format, [611](#)
- DT_HIPROC
 - ELF binary format, [611](#)
- DT_INIT
 - ELF binary format, [611](#)
- DT_INIT_ARRAY
 - ELF binary format, [611](#)
- DT_INIT_ARRAYSZ
 - ELF binary format, [611](#)
- DT_JMPREL
 - ELF binary format, [611](#)
- DT_LOOS
 - ELF binary format, [611](#)
- DT_LOPROC
 - ELF binary format, [611](#)
- DT_NEEDED
 - ELF binary format, [610](#)
- DT_NULL
 - ELF binary format, [610](#)
- DT_NUM
 - ELF binary format, [611](#)
- DT_PLTGOT
 - ELF binary format, [610](#)
- DT_PLTRELSZ
 - ELF binary format, [610](#)
- DT_PREINIT_ARRAY
 - ELF binary format, [611](#)
- DT_PREINIT_ARRAYSZ
 - ELF binary format, [611](#)
- DT_PTRREL
 - ELF binary format, [611](#)
- DT_REL
 - ELF binary format, [611](#)
- DT_RELA
 - ELF binary format, [610](#)
- DT_RELAENT
 - ELF binary format, [610](#)
- DT_RELASZ
 - ELF binary format, [610](#)
- DT_RELENT
 - ELF binary format, [611](#)
- DT_RELSZ
 - ELF binary format, [611](#)
- DT_RPATH
 - ELF binary format, [611](#)
- DT_RUNPATH
 - ELF binary format, [611](#)
- ELF binary format, [611](#)
- DT_SONAME
 - ELF binary format, [611](#)
- DT_STRSZ
 - ELF binary format, [610](#)
- DT_STRTAB
 - ELF binary format, [610](#)
- DT_SYMBOLIC
 - ELF binary format, [611](#)
- DT_SYMENT
 - ELF binary format, [611](#)
- DT_SYMTAB
 - ELF binary format, [610](#)
- DT_TEXTREL
 - ELF binary format, [611](#)
- dump
 - L4Re::Video::Color_component, [1628](#)
 - L4Re::Video::Pixel_info, [1644](#)
 - L4virtio::Virtqueue, [1885](#)
- dx
 - l4_vcpu_regs_t, [1379](#)
- E_exist
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COM-PARE, ALLOC, GET_KEY >, [794](#)
- e_flags
 - Elf32_Ehdr, [897](#)
 - Elf64_Ehdr, [907](#)
- E_inval
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COM-PARE, ALLOC, GET_KEY >, [794](#)
- e_machine
 - Elf32_Ehdr, [897](#)
 - Elf64_Ehdr, [907](#)
- E_noent
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COM-PARE, ALLOC, GET_KEY >, [794](#)
- E_nomem
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COM-PARE, ALLOC, GET_KEY >, [794](#)
- e_type
 - Elf32_Ehdr, [897](#)
 - Elf64_Ehdr, [907](#)
- e_version
 - Elf32_Ehdr, [898](#)
 - Elf64_Ehdr, [908](#)
- Eager_map
 - L4Re::Rm::F, [1513](#)
- EDID parsing functionality, [390](#)
 - libedid_block_size, [391](#)
 - libedid_check_header, [391](#)
 - libedid_checksum, [392](#)
 - libedid_consts, [391](#)
 - libedid_dump, [392](#)
 - libedid_dump_standard_timings, [392](#)
 - libedid_num_ext_blocks, [393](#)
 - libedid_pnp_id, [393](#)
 - libedid_preferred_resolution, [393](#)
 - libedid_revision, [393](#)

- libedid_version, [394](#)
- EF_ARM_ALIGN8
 - ELF binary format, [611](#)
- EI_ABIVERSION
 - ELF binary format, [612](#)
- EI_CLASS
 - ELF binary format, [612](#)
- EI_DATA
 - ELF binary format, [612](#)
- EI_MAG0
 - ELF binary format, [612](#)
- EI_MAG1
 - ELF binary format, [612](#)
- EI_MAG2
 - ELF binary format, [612](#)
- EI_MAG3
 - ELF binary format, [612](#)
- EI_NIDENT
 - ELF binary format, [606](#)
- EI_OSABI
 - ELF binary format, [612](#)
- EI_PAD
 - ELF binary format, [612](#)
- EI_VERSION
 - ELF binary format, [612](#)
- ELF binary format, [599](#)
 - AT_BASE, [607](#)
 - AT_EGID, [607](#)
 - AT_ENTRY, [607](#)
 - AT_EUID, [607](#)
 - AT_EXECFD, [607](#)
 - AT_FLAGS, [607](#)
 - AT_GID, [607](#)
 - AT_IGNORE, [607](#)
 - AT_L4_AUX, [607](#)
 - AT_L4_ENV, [607](#)
 - AT_NOTELF, [607](#)
 - AT_NULL, [607](#)
 - AT_PAGESZ, [607](#)
 - AT_PHDR, [607](#)
 - AT_PHENT, [607](#)
 - AT_PHNUM, [607](#)
 - AT_UID, [607](#)
 - DF_1_CONFALT, [608](#)
 - DF_1_DIRECT, [608](#)
 - DF_1_DISPRELDNE, [608](#)
 - DF_1_DISPRELPND, [608](#)
 - DF_1_ENDFILTEE, [608](#)
 - DF_1_GLOBAL, [608](#)
 - DF_1_GROUP, [608](#)
 - DF_1_INITFIRST, [608](#)
 - DF_1_INTERPOSE, [608](#)
 - DF_1_LOADFLTR, [608](#)
 - DF_1_NODEFLIB, [608](#)
 - DF_1_NODELETE, [608](#)
 - DF_1_NODUMP, [608](#)
 - DF_1_NOOPEN, [608](#)
 - DF_1_NOW, [608](#)
 - DF_1_ORIGIN, [608](#)
 - DF_BIND_NOW, [610](#)
 - DF_ORIGIN, [610](#)
 - DF_P1_GROUPPERM, [610](#)
 - DF_P1_LAZYLOAD, [610](#)
 - DF_STATIC_TLS, [610](#)
 - DF_SYMBOLIC, [610](#)
 - DF_TEXTREL, [610](#)
 - DT_BIND_NOW, [611](#)
 - DT_DEBUG, [611](#)
 - DT_ENCODING, [611](#)
 - DT_FINI, [611](#)
 - DT_FINI_ARRAY, [611](#)
 - DT_FINI_ARRAYSZ, [611](#)
 - DT_FLAGS, [611](#)
 - DT_HASH, [610](#)
 - DT_HIOS, [611](#)
 - DT_HIPROC, [611](#)
 - DT_INIT, [611](#)
 - DT_INIT_ARRAY, [611](#)
 - DT_INIT_ARRAYSZ, [611](#)
 - DT_JMPREL, [611](#)
 - DT_LOOS, [611](#)
 - DT_LOPROC, [611](#)
 - DT_NEEDED, [610](#)
 - DT_NULL, [610](#)
 - DT_NUM, [611](#)
 - DT_PLTGOT, [610](#)
 - DT_PLTRELSZ, [610](#)
 - DT_PREINIT_ARRAY, [611](#)
 - DT_PREINIT_ARRAYSZ, [611](#)
 - DT_PTRREL, [611](#)
 - DT_REL, [611](#)
 - DT_RELA, [610](#)
 - DT_RELAENT, [610](#)
 - DT_RELASZ, [610](#)
 - DT_RELENT, [611](#)
 - DT_RELSZ, [611](#)
 - DT_RPATH, [611](#)
 - DT_RUNPATH, [611](#)
 - DT_SONAME, [611](#)
 - DT_STRSZ, [610](#)
 - DT_STRTAB, [610](#)
 - DT_SYMBOLIC, [611](#)
 - DT_SYMENT, [611](#)
 - DT_SYMTAB, [610](#)
 - DT_TEXTREL, [611](#)
- EF_ARM_ALIGN8, [611](#)
- EI_ABIVERSION, [612](#)
- EI_CLASS, [612](#)
- EI_DATA, [612](#)
- EI_MAG0, [612](#)
- EI_MAG1, [612](#)
- EI_MAG2, [612](#)
- EI_MAG3, [612](#)
- EI_NIDENT, [606](#)
- EI_OSABI, [612](#)
- EI_PAD, [612](#)

ELF_VERSION, 612
ELF32_R_TYPE, 605
ELF32_ST_BIND, 605
ELF32_ST_TYPE, 605
ELF64_R_TYPE, 605
ELF64_ST_BIND, 605
ELF64_ST_TYPE, 606
Elf_ARM_SBs, 606
Elf_ATs, 606
Elf_Classes, 607
Elf_DATAs, 607
Elf_DF_1s, 608
Elf_DF_P1s, 608
Elf_DFs, 610
Elf_DTs, 610
Elf_EF_ARM_s, 611
Elf_EIs, 611
Elf_EMs, 612
Elf_ETs, 614
Elf_EVs, 614
Elf_MAGs, 615
Elf_NT_core, 615
Elf_NT_obj, 615
Elf_OSABIs, 616
ELF_PFs, 616
Elf_PTs, 617
Elf_R_386_s, 617
Elf_R_AARCH64_s, 618
Elf_R_ARM_s, 618
Elf_R_X86_64_s, 619
Elf_SHF_s_ARM, 620
Elf_SHFs, 620
Elf_SHNs, 621
Elf_SHTs, 621
Elf_STBs, 622
Elf_STTs, 622
ELFCLASS32, 607
ELFCLASS64, 607
ELFCLASSNONE, 607
ELFCLASSNUM, 607
ELFDATA2LSB, 608
ELFDATA2MSB, 608
ELFDATANONE, 608
ELFDATANUM, 608
ELFMAG0, 615
ELFMAG1, 615
ELFMAG2, 615
ELFMAG3, 615
ELFOSABI_AIX, 616
ELFOSABI_ARM, 616
ELFOSABI_FREEBSD, 616
ELFOSABI_HPUX, 616
ELFOSABI_IRIX, 616
ELFOSABI_LINUX, 616
ELFOSABI_MODESTO, 616
ELFOSABI_NETBSD, 616
ELFOSABI_NONE, 616
ELFOSABI_OPENBSD, 616
ELFOSABI_SOLARIS, 616
ELFOSABI_STANDALONE, 616
ELFOSABI_SYSV, 616
ELFOSABI_TRU64, 616
EM_386, 612
EM_68HC05, 613
EM_68HC08, 613
EM_68HC11, 613
EM_68HC12, 613
EM_68HC16, 613
EM_68K, 612
EM_860, 612
EM_88K, 612
EM_960, 613
EM_AARCH64, 614
EM_ALPHA, 613
EM_ALTERA_NIOS2, 614
EM_ARC, 613
EM_ARC_A5, 614
EM_ARM, 613
EM_AVR, 613
EM_COLDFIRE, 613
EM_CRIS, 613
EM_D10V, 613
EM_D30V, 613
EM_FIREPATH, 613
EM_FR20, 613
EM_FR30, 613
EM_FX66, 613
EM_H8_300, 613
EM_H8_300H, 613
EM_H8_500, 613
EM_H8S, 613
EM_HUANY, 613
EM_IA_64, 613
EM_JAVELIN, 613
EM_M32, 612
EM_M32R, 613
EM_MICROBLAZE, 614
EM_MIPS, 612
EM_MIPS_RS4_BE, 612
EM_MIPS_X, 613
EM_MMIX, 613
EM_MN10200, 613
EM_MN10300, 613
EM_NONE, 612
EM_OPENRISC, 614
EM_PARISC, 612
EM_PDSP, 613
EM_PJ, 613
EM_PPC, 613
EM_PRISM, 613
EM_RCE, 613
EM_RH32, 613
EM_SH, 613
EM_SPARC, 612
EM_SPARC32PLUS, 612
EM_SPARC64, 612

EM_SPARCV9, [613](#)
EM_ST19, [613](#)
EM_ST7, [613](#)
EM_ST9PLUS, [613](#)
EM_SVX, [613](#)
EM_TILEGX, [614](#)
EM_TILEPRO, [614](#)
EM_TRICORE, [613](#)
EM_V800, [613](#)
EM_V850, [613](#)
EM_VAX, [613](#)
EM_VPP500, [612](#)
EM_X86_64, [613](#)
EM_XTENSA, [614](#)
EM_ZSP, [613](#)
ET_CORE, [614](#)
ET_DYN, [614](#)
ET_EXEC, [614](#)
ET_HIPROC, [614](#)
ET_LOPROC, [614](#)
ET_NONE, [614](#)
ET_REL, [614](#)
EV_CURRENT, [614](#)
EV_NONE, [614](#)
NT_ASRS, [615](#)
NT_AUXV, [615](#)
NT_FPREGSET, [615](#)
NT_GWINDOWS, [615](#)
NT_LWPSINFO, [615](#)
NT_LWPSTATUS, [615](#)
NT_PLATFORM, [615](#)
NT_PRCRED, [615](#)
NT_PRFPXREG, [615](#)
NT_PRPSINFO, [615](#)
NT_PRSTATUS, [615](#)
NT_PRXREG, [615](#)
NT_PSINFO, [615](#)
NT_PSTATUS, [615](#)
NT_TASKSTRUCT, [615](#)
NT_UTSNAME, [615](#)
NT_VERSION, [616](#)
PF_ARM_SB, [606](#)
PF_MASKOS, [617](#)
PF_MASKPROC, [617](#)
PF_R, [617](#)
PF_W, [617](#)
PF_X, [617](#)
PT_DYNAMIC, [617](#)
PT_GNU_EH_FRAME, [617](#)
PT_GNU_RELRO, [617](#)
PT_GNU_STACK, [617](#)
PT_HIOS, [617](#)
PT_HIPROC, [617](#)
PT_INTERP, [617](#)
PT_L4_AUX, [617](#)
PT_L4_KIP, [617](#)
PT_L4_STACK, [617](#)
PT_LOAD, [617](#)
PT_LOOS, [617](#)
PT_LOPROC, [617](#)
PT_NOTE, [617](#)
PT_NULL, [617](#)
PT_NUM, [617](#)
PT_PHDR, [617](#)
PT_SHLIB, [617](#)
PT_TLS, [617](#)
R_386_32, [618](#)
R_386_COPY, [618](#)
R_386_GLOB_DAT, [618](#)
R_386_GOT32, [618](#)
R_386_GOTOFF, [618](#)
R_386_GOTPC, [618](#)
R_386_JMP_SLOT, [618](#)
R_386_NONE, [618](#)
R_386_NUM, [618](#)
R_386_PC32, [618](#)
R_386_PLT32, [618](#)
R_386_RELATIVE, [618](#)
R_386_TLS_DTPMOD32, [618](#)
R_386_TLS_DTPOFF32, [618](#)
R_386_TLS_GD, [618](#)
R_386_TLS_GD_32, [618](#)
R_386_TLS_GD_CALL, [618](#)
R_386_TLS_GD_POP, [618](#)
R_386_TLS_GD_PUSH, [618](#)
R_386_TLS_GOTIE, [618](#)
R_386_TLS_IE, [618](#)
R_386_TLS_IE_32, [618](#)
R_386_TLS_LDM, [618](#)
R_386_TLS_LDM_32, [618](#)
R_386_TLS_LDM_CALL, [618](#)
R_386_TLS_LDM_POP, [618](#)
R_386_TLS_LDM_PUSH, [618](#)
R_386_TLS_LDO_32, [618](#)
R_386_TLS_LE, [618](#)
R_386_TLS_LE_32, [618](#)
R_386_TLS_TPOFF, [618](#)
R_386_TLS_TPOFF32, [618](#)
R_AARCH64_NONE, [618](#)
R_ARM_ABS12, [619](#)
R_ARM_ABS16, [619](#)
R_ARM_ABS32, [619](#)
R_ARM_ABS8, [619](#)
R_ARM_COPY, [619](#)
R_ARM_GLOB_DAT, [619](#)
R_ARM_GOT32, [619](#)
R_ARM_GOTOFF, [619](#)
R_ARM_GOTPC, [619](#)
R_ARM_JUMP_SLOT, [619](#)
R_ARM_NONE, [619](#)
R_ARM_NUM, [619](#)
R_ARM_PC24, [619](#)
R_ARM_PLT32, [619](#)
R_ARM_REL32, [619](#)
R_ARM_RELATIVE, [619](#)
R_ARM_THM_PC11, [619](#)

[R_ARM_THM_PC9](#), 619
[R_X86_64_16](#), 620
[R_X86_64_32](#), 619
[R_X86_64_32S](#), 620
[R_X86_64_64](#), 619
[R_X86_64_8](#), 620
[R_X86_64_COPY](#), 619
[R_X86_64_DTPMOD64](#), 620
[R_X86_64_DTPOFF32](#), 620
[R_X86_64_DTPOFF64](#), 620
[R_X86_64_GLOB_DAT](#), 619
[R_X86_64_GOT32](#), 619
[R_X86_64_GOTPCREL](#), 619
[R_X86_64_GOTTPOFF](#), 620
[R_X86_64_JUMP_SLOT](#), 619
[R_X86_64_NONE](#), 619
[R_X86_64_PC16](#), 620
[R_X86_64_PC32](#), 619
[R_X86_64_PC8](#), 620
[R_X86_64_PLT32](#), 619
[R_X86_64_RELATIVE](#), 619
[R_X86_64_TLSD](#), 620
[R_X86_64_TLSD](#), 620
[R_X86_64_TPOFF32](#), 620
[R_X86_64_TPOFF64](#), 620
[SHF_ALLOC](#), 620
[SHF_ARM_COMDEF](#), 620
[SHF_ARM_ENTRYSECT](#), 620
[SHF_EXECINSTR](#), 620
[SHF_GROUP](#), 621
[SHF_INFO_LINK](#), 620
[SHF_LINK_ORDER](#), 620
[SHF_MASKOS](#), 621
[SHF_MASKPROC](#), 621
[SHF_MERGE](#), 620
[SHF_OS_NONCONFORMING](#), 620
[SHF_STRINGS](#), 620
[SHF_TLS](#), 621
[SHF_WRITE](#), 620
[SHN_ABS](#), 621
[SHN_COMMON](#), 621
[SHN_HIPROC](#), 621
[SHN_HIRESERVE](#), 621
[SHN_LOPROC](#), 621
[SHN_LORESERVE](#), 621
[SHN_UNDEF](#), 621
[SHT_DYNAMIC](#), 621
[SHT_DYNSYM](#), 622
[SHT_FINI_ARRAY](#), 622
[SHT_GROUP](#), 622
[SHT_HASH](#), 621
[SHT_HIOS](#), 622
[SHT_HIPROC](#), 622
[SHT_HIUSER](#), 622
[SHT_INIT_ARRAY](#), 622
[SHT_LOOS](#), 622
[SHT_LOPROC](#), 622
[SHT_LOUSER](#), 622
[SHT_NOBITS](#), 621
[SHT_NOTE](#), 621
[SHT_NULL](#), 621
[SHT_NUM](#), 622
[SHT_PREINIT_ARRAY](#), 622
[SHT_PROGBITS](#), 621
[SHT_REL](#), 621
[SHT_RELA](#), 621
[SHT_SHLIB](#), 622
[SHT_STRTAB](#), 621
[SHT_SYMTAB](#), 621
[SHT_SYMTAB_SHNDX](#), 622
[STB_GLOBAL](#), 622
[STB_HIOS](#), 622
[STB_HIPROC](#), 622
[STB_LOCAL](#), 622
[STB_LOOS](#), 622
[STB_LOPROC](#), 622
[STB_WEAK](#), 622
[STT_FILE](#), 623
[STT_FUNC](#), 623
[STT_HIOS](#), 623
[STT_HIPROC](#), 623
[STT_LOOS](#), 623
[STT_LOPROC](#), 623
[STT_NOTYPE](#), 623
[STT_OBJECT](#), 623
[STT_SECTION](#), 623
[Elf32_Auxv](#), 894
 [atype](#), 894
[Elf32_Dyn](#), 895
 [d_tag](#), 895
[Elf32_Ehdr](#), 896
 [e_flags](#), 897
 [e_machine](#), 897
 [e_type](#), 897
 [e_version](#), 898
[Elf32_Phdr](#), 898
 [p_flags](#), 899
 [p_type](#), 899
[ELF32_R_TYPE](#)
 [ELF binary format](#), 605
[Elf32_Rel](#), 900
[Elf32_Rela](#), 900
[Elf32_Shdr](#), 901
 [sh_flags](#), 902
 [sh_type](#), 902
[ELF32_ST_BIND](#)
 [ELF binary format](#), 605
[ELF32_ST_TYPE](#)
 [ELF binary format](#), 605
[Elf32_Sym](#), 903
[Elf64_Auxv](#), 904
 [atype](#), 904
[Elf64_Dyn](#), 905
 [d_tag](#), 905
[Elf64_Ehdr](#), 906
 [e_flags](#), 907

- e_machine, [907](#)
- e_type, [907](#)
- e_version, [908](#)
- Elf64_Phdr, [908](#)
 - p_flags, [909](#)
 - p_type, [909](#)
- Elf64_R_TYPE
 - ELF binary format, [605](#)
- Elf64_Rel, [910](#)
- Elf64_Rela, [910](#)
- Elf64_Shdr, [911](#)
 - sh_flags, [912](#)
 - sh_type, [912](#)
- Elf64_ST_BIND
 - ELF binary format, [605](#)
- Elf64_ST_TYPE
 - ELF binary format, [606](#)
- Elf64_Sym, [913](#)
- Elf_ARM_SBs
 - ELF binary format, [606](#)
- Elf_ATs
 - ELF binary format, [606](#)
- Elf_CIASSs
 - ELF binary format, [607](#)
- Elf_DATAs
 - ELF binary format, [607](#)
- Elf_DF_1s
 - ELF binary format, [608](#)
- Elf_DF_P1s
 - ELF binary format, [608](#)
- Elf_DFs
 - ELF binary format, [610](#)
- Elf_DTs
 - ELF binary format, [610](#)
- Elf_EF_ARM_s
 - ELF binary format, [611](#)
- Elf_EIs
 - ELF binary format, [611](#)
- Elf_EMs
 - ELF binary format, [612](#)
- Elf_ETs
 - ELF binary format, [614](#)
- Elf_EVs
 - ELF binary format, [614](#)
- Elf_MAGs
 - ELF binary format, [615](#)
- Elf_NT_s_core
 - ELF binary format, [615](#)
- Elf_NT_s_obj
 - ELF binary format, [615](#)
- Elf_OSABIs
 - ELF binary format, [616](#)
- Elf_PFs
 - ELF binary format, [616](#)
- Elf_PT_s
 - ELF binary format, [617](#)
- Elf_R_386_s
 - ELF binary format, [617](#)
- Elf_R_AARCH64_s
 - ELF binary format, [618](#)
- Elf_R_ARM_s
 - ELF binary format, [618](#)
- Elf_R_X86_64_s
 - ELF binary format, [619](#)
- Elf_SHF_s_ARM
 - ELF binary format, [620](#)
- Elf_SHFs
 - ELF binary format, [620](#)
- Elf_SHNs
 - ELF binary format, [621](#)
- Elf_SHTs
 - ELF binary format, [621](#)
- Elf_STBs
 - ELF binary format, [622](#)
- Elf_STTs
 - ELF binary format, [622](#)
- ELFCLASS32
 - ELF binary format, [607](#)
- ELFCLASS64
 - ELF binary format, [607](#)
- ELFCLASSNONE
 - ELF binary format, [607](#)
- ELFCLASSNUM
 - ELF binary format, [607](#)
- ELFDATA2LSB
 - ELF binary format, [608](#)
- ELFDATA2MSB
 - ELF binary format, [608](#)
- ELFDATANONE
 - ELF binary format, [608](#)
- ELFDATANUM
 - ELF binary format, [608](#)
- ELFMAG0
 - ELF binary format, [615](#)
- ELFMAG1
 - ELF binary format, [615](#)
- ELFMAG2
 - ELF binary format, [615](#)
- ELFMAG3
 - ELF binary format, [615](#)
- ELFOSABI_AIX
 - ELF binary format, [616](#)
- ELFOSABI_ARM
 - ELF binary format, [616](#)
- ELFOSABI_FREEBSD
 - ELF binary format, [616](#)
- ELFOSABI_HPUX
 - ELF binary format, [616](#)
- ELFOSABI_IRIX
 - ELF binary format, [616](#)
- ELFOSABI_LINUX
 - ELF binary format, [616](#)
- ELFOSABI_MODESTO
 - ELF binary format, [616](#)
- ELFOSABI_NETBSD
 - ELF binary format, [616](#)

ELFOSABI_NONE
 ELF binary format, [616](#)
ELFOSABI_OPENBSD
 ELF binary format, [616](#)
ELFOSABI_SOLARIS
 ELF binary format, [616](#)
ELFOSABI_STANDALONE
 ELF binary format, [616](#)
ELFOSABI_SYSV
 ELF binary format, [616](#)
ELFOSABI_TRU64
 ELF binary format, [616](#)
EM_386
 ELF binary format, [612](#)
EM_68HC05
 ELF binary format, [613](#)
EM_68HC08
 ELF binary format, [613](#)
EM_68HC11
 ELF binary format, [613](#)
EM_68HC12
 ELF binary format, [613](#)
EM_68HC16
 ELF binary format, [613](#)
EM_68K
 ELF binary format, [612](#)
EM_860
 ELF binary format, [612](#)
EM_88K
 ELF binary format, [612](#)
EM_960
 ELF binary format, [613](#)
EM_AARCH64
 ELF binary format, [614](#)
EM_ALPHA
 ELF binary format, [613](#)
EM_ALTERA_NIOS2
 ELF binary format, [614](#)
EM_ARC
 ELF binary format, [613](#)
EM_ARC_A5
 ELF binary format, [614](#)
EM_ARM
 ELF binary format, [613](#)
EM_AVR
 ELF binary format, [613](#)
EM_COLDFIRE
 ELF binary format, [613](#)
EM_CRIS
 ELF binary format, [613](#)
EM_D10V
 ELF binary format, [613](#)
EM_D30V
 ELF binary format, [613](#)
EM_FIREPATH
 ELF binary format, [613](#)
EM_FR20
 ELF binary format, [613](#)
EM_FR30
 ELF binary format, [613](#)
EM_FX66
 ELF binary format, [613](#)
EM_H8_300
 ELF binary format, [613](#)
EM_H8_300H
 ELF binary format, [613](#)
EM_H8_500
 ELF binary format, [613](#)
EM_H8S
 ELF binary format, [613](#)
EM_HUANY
 ELF binary format, [613](#)
EM_IA_64
 ELF binary format, [613](#)
EM_JAVELIN
 ELF binary format, [613](#)
EM_M32
 ELF binary format, [612](#)
EM_M32R
 ELF binary format, [613](#)
EM_MICROBLAZE
 ELF binary format, [614](#)
EM_MIPS
 ELF binary format, [612](#)
EM_MIPS_RS4_BE
 ELF binary format, [612](#)
EM_MIPS_X
 ELF binary format, [613](#)
EM_MMIX
 ELF binary format, [613](#)
EM_MN10200
 ELF binary format, [613](#)
EM_MN10300
 ELF binary format, [613](#)
EM_NONE
 ELF binary format, [612](#)
EM_OPENRISC
 ELF binary format, [614](#)
EM_PARISC
 ELF binary format, [612](#)
EM_PDSP
 ELF binary format, [613](#)
EM_PJ
 ELF binary format, [613](#)
EM_PPC
 ELF binary format, [613](#)
EM_PRISM
 ELF binary format, [613](#)
EM_RCE
 ELF binary format, [613](#)
EM_RH32
 ELF binary format, [613](#)
EM_SH
 ELF binary format, [613](#)
EM_SPARC
 ELF binary format, [612](#)

- EM_SPARC32PLUS
 - ELF binary format, [612](#)
- EM_SPARC64
 - ELF binary format, [612](#)
- EM_SPARCV9
 - ELF binary format, [613](#)
- EM_ST19
 - ELF binary format, [613](#)
- EM_ST7
 - ELF binary format, [613](#)
- EM_ST9PLUS
 - ELF binary format, [613](#)
- EM_SVX
 - ELF binary format, [613](#)
- EM_TILEGX
 - ELF binary format, [614](#)
- EM_TILEPRO
 - ELF binary format, [614](#)
- EM_TRICORE
 - ELF binary format, [613](#)
- EM_V800
 - ELF binary format, [613](#)
- EM_V850
 - ELF binary format, [613](#)
- EM_VAX
 - ELF binary format, [613](#)
- EM_VPP500
 - ELF binary format, [612](#)
- EM_X86_64
 - ELF binary format, [613](#)
- EM_XTENSA
 - ELF binary format, [614](#)
- EM_ZSP
 - ELF binary format, [613](#)
- empty
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1861](#)
- enable_notify
 - L4virtio::Svr::Virtqueue, [1875](#)
- enable_rx_irq
 - L4::Uart, [1332](#)
- end
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [796](#), [797](#)
 - cxx::Bits::Bst< Node, Get_key, Compare >, [815](#)
 - L4::Kip::Mem_desc, [1158](#)
- enqueue_descriptor
 - L4virtio::Driver::Virtqueue, [1800](#)
- enter_kdebug
 - kdebug.h, [2681](#)
- entry_ip
 - L4vcpu::Vcpu, [1746](#)
- entry_sp
 - L4vcpu::Vcpu, [1747](#)
- env
 - L4Re::Env, [1439](#)
- env.h
 - l4re_env_t, [2359](#)
- erase
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [798](#)
 - cxx::H_list< T, POLICY >, [835](#)
- err_no
 - L4::Runtime_error, [1215](#)
- Error
 - L4virtio::Svr::Bad_descriptor, [1810](#)
- Error codes, [144](#)
 - L4_E2BIG, [145](#)
 - L4_EACCESS, [145](#)
 - L4_EADDRNOTAVAIL, [145](#)
 - L4_EAGAIN, [145](#)
 - L4_EBADPROTO, [145](#)
 - L4_EBUSY, [145](#)
 - L4_EEXIST, [145](#)
 - L4_EFAULT, [145](#)
 - L4_EINVAL, [145](#)
 - L4_EIO, [145](#)
 - L4_EIPC_HI, [145](#)
 - L4_EIPC_LO, [145](#)
 - L4_EMMSGMISSARG, [145](#)
 - L4_EMMSGTOOLONG, [145](#)
 - L4_EMMSGTOOSHORT, [145](#)
 - L4_ENAMETOOLONG, [145](#)
 - L4_ENODEV, [145](#)
 - L4_ENOENT, [145](#)
 - L4_ENOMEM, [145](#)
 - L4_ENOREPLY, [145](#)
 - L4_ENOSPC, [145](#)
 - L4_ENOSYS, [145](#)
 - L4_ENXIO, [145](#)
 - L4_EOK, [145](#)
 - L4_EPERM, [145](#)
 - L4_ERANGE, [145](#)
 - L4_ERRNOMAX, [145](#)
 - l4_error_code_t, [145](#)
- Error Handling, [347](#)
 - l4_error, [349](#)
 - L4_IPC_ENOT_EXISTENT, [348](#)
 - l4_ipc_error, [350](#)
 - l4_ipc_error_code, [351](#)
 - L4_IPC_ERROR_MASK, [348](#)
 - l4_ipc_is_rcv_error, [351](#)
 - l4_ipc_is_snd_error, [352](#)
 - L4_IPC_REABORTED, [348](#)
 - L4_IPC_RECANCELED, [348](#)
 - L4_IPC_REMAPFAILED, [348](#)
 - L4_IPC_REMSGCUT, [348](#)
 - L4_IPC_RERCVPFTO, [348](#)
 - L4_IPC_RESNDPFTO, [348](#)
 - L4_IPC_RETIMEOUT, [348](#)
 - L4_IPC_SEABORTED, [348](#)
 - L4_IPC_SECANCELED, [348](#)
 - L4_IPC_SEMAPFAILED, [348](#)
 - L4_IPC_SEMSGCUT, [348](#)
 - L4_IPC_SERCVPFTO, [348](#)
 - L4_IPC_SESNDPFTO, [348](#)

- L4_IPC_SETIMEOUT, [348](#)
 - L4_IPC_SND_ERR_MASK, [348](#)
 - l4_ipc_tcr_error_t, [348](#)
- ET_CORE
 - ELF binary format, [614](#)
- ET_DYN
 - ELF binary format, [614](#)
- ET_EXEC
 - ELF binary format, [614](#)
- ET_HIPROC
 - ELF binary format, [614](#)
- ET_LOPROC
 - ELF binary format, [614](#)
- ET_NONE
 - ELF binary format, [614](#)
- ET_REL
 - ELF binary format, [614](#)
- EV_CURRENT
 - ELF binary format, [614](#)
- EV_NONE
 - ELF binary format, [614](#)
- Event API, [511](#)
- Event interface, [466](#)
 - l4re_event_get_axis_info, [467](#)
 - l4re_event_get_buffer, [467](#)
 - l4re_event_get_num_streams, [468](#)
 - l4re_event_get_stream_info, [468](#)
 - l4re_event_get_stream_info_for_id, [468](#)
- Event_buffer_t
 - L4Re::Event_buffer_t< PAYLOAD >, [1457](#)
- ex_regs
 - L4::Thread, [1267](#), [1268](#)
- exc_handler
 - L4::Thread::Attr, [1278](#), [1279](#)
- exception
 - L4::Exception, [969](#)
- Exception registers, [384](#)
 - l4_utcb_exc, [385](#)
 - l4_utcb_exc_is_ex_regs_exception, [385](#)
 - l4_utcb_exc_is_pf, [387](#)
 - l4_utcb_exc_pc, [387](#)
 - l4_utcb_exc_pc_set, [388](#)
- execute
 - L4Re::Ned::Cmd_control, [1488](#), [1489](#)
- expired
 - Block_device::Errand::Errand, [714](#)
 - Block_device::Errand::Poll_errand, [717](#)
 - L4::lpc_svr::Timeout, [1118](#)
- ext_alloc
 - L4vcpu::Vcpu, [1747](#)
- Extended vCPU support, [649](#)
 - l4vcpu_ext_alloc, [649](#)
- extra_str
 - L4::Runtime_error, [1215](#)
- F_above
 - L4Re::Video::View, [1649](#)
- F_auto_refresh
 - L4Re::Video::Goos, [1634](#)
- F_dyn_allocated
 - L4Re::Video::View, [1648](#)
- F_dynamic_buffers
 - L4Re::Video::Goos, [1634](#)
- F_dynamic_views
 - L4Re::Video::Goos, [1634](#)
- F_flags_mask
 - L4Re::Video::View, [1649](#)
- F_fully_dynamic
 - L4Re::Video::View, [1648](#)
- F_l4re_video_goos_auto_refresh
 - Video API, [501](#)
- F_l4re_video_goos_dynamic_buffers
 - Video API, [501](#)
- F_l4re_video_goos_dynamic_views
 - Video API, [501](#)
- F_l4re_video_goos_pointer
 - Video API, [501](#)
- F_l4re_video_view_above
 - Video API, [501](#)
- F_l4re_video_view_dyn_allocated
 - Video API, [501](#)
- F_l4re_video_view_flags_mask
 - Video API, [501](#)
- F_l4re_video_view_none
 - Video API, [501](#)
- F_l4re_video_view_set_background
 - Video API, [501](#)
- F_l4re_video_view_set_buffer
 - Video API, [501](#)
- F_l4re_video_view_set_buffer_offset
 - Video API, [501](#)
- F_l4re_video_view_set_bytes_per_line
 - Video API, [501](#)
- F_l4re_video_view_set_flags
 - Video API, [501](#)
- F_l4re_video_view_set_pixel
 - Video API, [501](#)
- F_l4re_video_view_set_position
 - Video API, [501](#)
- F_none
 - L4Re::Video::View, [1648](#)
- F_pointer
 - L4Re::Video::Goos, [1634](#)
- F_set_background
 - L4Re::Video::View, [1648](#)
- F_set_buffer
 - L4Re::Video::View, [1648](#)
- F_set_buffer_offset
 - L4Re::Video::View, [1648](#)
- F_set_bytes_per_line
 - L4Re::Video::View, [1648](#)
- F_set_flags
 - L4Re::Video::View, [1648](#)
- F_set_pixel
 - L4Re::Video::View, [1648](#)
- F_set_position
 - L4Re::Video::View, [1648](#)

- faccessat
 - L4Re::Vfs::Directory, [1599](#)
- Factory, [192](#)
 - I4_factory_create, [193](#)
 - I4_factory_create_factory, [194](#)
 - I4_factory_create_gate, [195](#)
 - I4_factory_create_irq, [196](#)
 - I4_factory_create_task, [197](#)
 - I4_factory_create_thread, [198](#)
 - I4_factory_create_vm, [199](#)
- factory
 - L4Re::Env, [1439](#)
- fail_request
 - Block_device::Pending_request, [726](#)
- fchmod
 - L4Re::Vfs::Generic_file, [1612](#)
- fdatasync
 - L4Re::Vfs::Regular_file, [1622](#)
- feature_negotiated
 - L4virtio::Driver::Device, [1779](#)
- features
 - I4_icu_info_t, [1357](#)
- Fiasco extensions, [146](#)
 - fiasco_gdt_get_entry_offset, [147](#)
 - fiasco_gdt_set, [147](#)
 - fiasco_ldt_set, [148](#)
- Fiasco real time scheduling extensions, [149](#)
- Fiasco-UX Virtual devices, [184](#)
 - L4_TYPE_VHW_FRAMEBUFFER, [185](#)
 - L4_TYPE_VHW_INPUT, [185](#)
 - L4_TYPE_VHW_NET, [185](#)
 - L4_TYPE_VHW_NONE, [185](#)
 - I4_vhw_entry_type, [184](#)
- fiasco_amd64_segment_info
 - segment.h, [1976](#)
- fiasco_amd64_set_fs
 - segment.h, [1977](#), [1981](#)
- fiasco_amd64_set_segment_base
 - segment.h, [1978](#), [1982](#)
- fiasco_gdt_get_entry_offset
 - Fiasco extensions, [147](#)
- fiasco_gdt_set
 - Fiasco extensions, [147](#)
- fiasco_ldt_set
 - Fiasco extensions, [148](#)
- fiasco_tbuf_log
 - Kernel Tracing, [157](#)
- fiasco_tbuf_log_3val
 - Kernel Tracing, [157](#)
- fiasco_tbuf_log_binary
 - Kernel Tracing, [158](#)
- finalize_request
 - L4virtio::Svr::Block_dev_base< Ds_data >, [1816](#)
- find
 - cxx::Bits::Bst< Node, Get_key, Compare >, [816](#)
 - cxx::String, [885](#)
 - L4::Basic_registry, [921](#)
 - L4Re::Rm, [1508](#)
 - L4virtio::Svr::Driver_mem_list_t< DATA >, [1850](#)
- find_next_used
 - L4virtio::Driver::Virtqueue, [1801](#)
- find_node
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [798](#)
 - cxx::Bits::Bst< Node, Get_key, Compare >, [816](#)
- finish_rx
 - L4virtio::Driver::Virtio_net_device, [1789](#)
- first
 - L4::Kip::Mem_desc, [1159](#)
- first_free_cap
 - L4Re::Env, [1440](#)
- first_free_utcb
 - L4Re::Env, [1440](#)
- Flags
 - L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, [1294](#)
 - L4::Types::Flags< BITS_ENUM, UNDERLYING >, [1318](#)
 - L4Re::Dataspace::F, [1424](#)
 - L4Re::Video::Goos, [1634](#)
 - L4Re::Video::View, [1648](#)
- flags
 - I4_exc_regs_t, [1354](#)
 - I4_msgtag_t, [1363](#)
 - L4Re::Dataspace, [1419](#)
 - I4re_env_cap_entry_t, [1658](#)
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1861](#)
- Flex pages, [159](#)
 - L4_CAP_FPAGE_D, [163](#)
 - L4_CAP_FPAGE_R, [163](#)
 - L4_cap_fpage_rights, [162](#)
 - L4_CAP_FPAGE_RO, [163](#)
 - L4_CAP_FPAGE_RS, [163](#)
 - L4_CAP_FPAGE_RSD, [163](#)
 - L4_CAP_FPAGE_RW, [163](#)
 - L4_CAP_FPAGE_RWD, [163](#)
 - L4_CAP_FPAGE_RWS, [163](#)
 - L4_CAP_FPAGE_RWSD, [163](#)
 - L4_CAP_FPAGE_S, [162](#)
 - L4_CAP_FPAGE_W, [162](#)
 - I4_fpage, [166](#)
 - L4_FPAGE_ADDR_BITS, [164](#)
 - L4_FPAGE_ADDR_SHIFT, [164](#)
 - I4_fpage_all, [166](#)
 - L4_FPAGE_BUFFERABLE, [164](#)
 - L4_FPAGE_C_IPCGATE_SVR, [166](#)
 - L4_FPAGE_C_NO_REF_CNT, [166](#)
 - L4_FPAGE_C_OBJ_RIGHT1, [166](#)
 - L4_FPAGE_C_OBJ_RIGHT2, [166](#)
 - L4_FPAGE_C_OBJ_RIGHT3, [166](#)
 - L4_FPAGE_C_OBJ_RIGHTS, [166](#)
 - L4_FPAGE_C_REF_CNT, [165](#)
 - L4_FPAGE_CACHE_OPT, [164](#)
 - I4_fpage_cacheability_opt_t, [163](#)
 - L4_FPAGE_CACHEABLE, [164](#)

- L4_fpage_consts, [164](#)
- L4_fpage_contains, [167](#)
- L4_fpage_control, [164](#)
- L4_FPAGE_CONTROL_MASK, [164](#)
- L4_FPAGE_CONTROL_OFFSET_SHIFT, [164](#)
- L4_fpage_invalid, [167](#)
- L4_FPAGE_IO, [165](#)
- L4_fpage_ioport, [168](#)
- L4_fpage_max_order, [168](#)
- L4_fpage_memaddr, [169](#)
- L4_FPAGE_MEMORY, [165](#)
- L4_FPAGE_OBJ, [165](#)
- L4_fpage_obj, [170](#)
- L4_fpage_page, [170](#)
- L4_fpage_rights, [164](#)
- L4_fpage_rights, [171](#)
- L4_FPAGE_RIGHTS_ALL, [164](#)
- L4_FPAGE_RIGHTS_BITS, [164](#)
- L4_FPAGE_RIGHTS_MASK, [164](#)
- L4_FPAGE_RIGHTS_SHIFT, [164](#)
- L4_FPAGE_RO, [165](#)
- L4_FPAGE_RW, [165](#)
- L4_FPAGE_RWX, [165](#)
- L4_FPAGE_RX, [165](#)
- L4_fpage_set_rights, [171](#)
- L4_fpage_size, [172](#)
- L4_FPAGE_SIZE_BITS, [164](#)
- L4_FPAGE_SIZE_SHIFT, [164](#)
- L4_FPAGE_SPECIAL, [165](#)
- L4_fpage_type, [165](#)
- L4_fpage_type, [173](#)
- L4_FPAGE_TYPE_BITS, [164](#)
- L4_FPAGE_TYPE_SHIFT, [164](#)
- L4_FPAGE_UNCACHEABLE, [164](#)
- L4_FPAGE_W, [165](#)
- L4_FPAGE_X, [165](#)
- L4_iofpage, [173](#)
- L4_IOPORT_MAX, [162](#)
- L4_is_fpage_writable, [174](#)
- L4_obj_fpage, [174](#)
- L4_obj_fpage_ctl, [165](#)
- L4_WHOLE_ADDRESS_SPACE, [162](#)
- L4_WHOLE_IOADDRESS_SPACE, [162](#)
- font.h
 - gfxbitmap_font_data, [2304](#)
 - gfxbitmap_font_get, [2304](#)
 - gfxbitmap_font_height, [2305](#)
 - gfxbitmap_font_init, [2305](#)
 - gfxbitmap_font_text, [2305](#)
 - gfxbitmap_font_text_scale, [2306](#)
 - gfxbitmap_font_width, [2307](#)
- foreach_available_event
 - L4Re::Util::Event_buffer_consumer_t< PAYLOAD
>, [1550](#)
- fpage
 - L4::Cap_base, [935](#)
- free
 - cxx::Base_slab< Obj_size, Slab_size, Max_free,
Alloc >, [751](#)
 - cxx::Base_slab_static< Obj_size, Slab_size,
Max_free, Alloc >, [757](#)
 - cxx::List_alloc, [851](#)
 - cxx::Slab< Type, Slab_size, Max_free, Alloc >,
[876](#)
 - L4Re::Cap_alloc, [1409](#)
 - L4Re::Util::Counting_cap_alloc< COUNTERTYPE
>, [1540](#)
 - free_area
 - L4Re::Rm, [1509](#)
 - free_capability
 - L4Re::Util::Names::Name_space, [1567](#)
 - free_descriptor
 - L4virtio::Driver::Virtqueue, [1801](#)
 - free_dynamic_entry
 - L4Re::Util::Names::Name_space, [1568](#)
 - free_epiface
 - L4Re::Util::Names::Name_space, [1568](#)
 - free_fd
 - L4Re::Vfs::Fs, [1609](#)
 - free_objects
 - cxx::Base_slab< Obj_size, Slab_size, Max_free,
Alloc >, [752](#)
 - cxx::Base_slab_static< Obj_size, Slab_size,
Max_free, Alloc >, [758](#)
 - from_ci
 - L4::lpc::Cap< T >, [1018](#)
 - from_dec
 - cxx::String, [886](#)
 - From_device
 - L4Re::Dma_space, [1432](#)
 - from_hex
 - cxx::String, [886](#)
 - from_raw
 - L4::Types::Flags< BITS_ENUM, UNDERLYING >,
[1319](#)
 - fstat64
 - L4Re::Vfs::Be_file, [1593](#)
 - L4Re::Vfs::Generic_file, [1612](#)
 - fsync
 - L4Re::Vfs::Regular_file, [1622](#)
 - ftruncate64
 - L4Re::Vfs::Regular_file, [1622](#)
 - full
 - L4virtio::Svr::Driver_mem_list_t< DATA >, [1851](#)
 - Functions for rendering bitmap data in frame buffers, [594](#)
 - Functions for rendering bitmap fonts to frame buffers,
[594](#)
 - Functions to manipulate the local IDT, [623](#)
- g
 - L4Re::Video::Pixel_info, [1644](#)
- generic_write
 - L4::Uart, [1332](#)
- get
 - cxx::Bitfield< T, LSB, MSB >, [762](#)
 - cxx::Ref_ptr< T, CNT >, [869](#)

- L4::lpc::loststream, [1027](#), [1028](#)
- L4::lpc::lstream, [1036](#), [1037](#)
- L4Re::Env, [1441](#)
- L4Re::Rm::Unique_region< T >, [1518](#)
- L4Re::Video::Color_component, [1628](#)
- L4vbus::Gpio_module, [1696](#)
- L4vbus::Gpio_pin, [1706](#)
- L4virtio::Ptr< T >, [1807](#)
- get_areas
 - L4Re::Rm, [1509](#)
- get_attr
 - L4::Vcon, [1341](#)
- get_avail_idx
 - L4virtio::Virtqueue, [1885](#)
- get_axis_info
 - L4Re::Event, [1452](#)
- get_buffer
 - L4Re::Event, [1453](#)
- get_buffer_demand
 - L4::Epiface, [961](#)
 - L4::Server_object_t< IFACE, BASE >, [1241](#)
- get_cap
 - L4Re::Env, [1441](#), [1443](#)
- get_cap_alloc
 - L4Re::Cap_alloc, [1409](#)
- get_char
 - L4::Uart, [1333](#)
- get_cmd
 - L4virtio::Svr::Dev_config, [1830](#)
- get_epiface
 - L4Re::Util::Names::Name_space, [1568](#)
- get_fb
 - L4Re::Util::Video::Goos_svr, [1588](#)
- get_file
 - L4Re::Vfs::Fs, [1609](#)
- get_infos
 - L4::lpc_gate, [1092](#)
- get_lock
 - L4Re::Vfs::Regular_file, [1623](#)
- get_num_streams
 - L4Re::Event, [1453](#)
- get_object_name
 - L4::Debugger, [947](#)
- get_random
 - L4Re::Random, [1497](#)
- get_rcv_cap
 - L4::lpc_svr::Server_iface, [1113](#)
 - L4Re::Util::Br_manager, [1527](#)
- get_regions
 - L4Re::Rm, [1510](#)
- get_resource
 - L4vbus::Device, [1689](#)
- get_static_buffer
 - L4Re::Video::Goos, [1636](#)
- get_status_flags
 - L4Re::Vfs::Generic_file, [1613](#)
- get_stream_info
 - L4Re::Event, [1453](#)
- get_stream_info_for_id
 - L4Re::Event, [1454](#)
- get_stream_state_for_id
 - L4Re::Event, [1454](#)
- get_tail_avail_idx
 - L4virtio::Virtqueue, [1886](#)
- get_unshifted
 - cxx::Bitfield< T, LSB, MSB >, [763](#)
- get_value
 - L4::lpc::Varg, [1078](#)
- get_writeback
 - L4virtio::Svr::Block_dev_base< Ds_data >, [1817](#)
- gfxbitmap_bmap
 - bitmap.h, [2299](#)
- gfxbitmap_color_pix_t
 - bitmap.h, [2299](#)
- gfxbitmap_color_t
 - bitmap.h, [2299](#)
- gfxbitmap_convert_color
 - bitmap.h, [2300](#)
- gfxbitmap_copy
 - bitmap.h, [2300](#)
- gfxbitmap_fill
 - bitmap.h, [2300](#)
- gfxbitmap_font_data
 - font.h, [2304](#)
- gfxbitmap_font_get
 - font.h, [2304](#)
- gfxbitmap_font_height
 - font.h, [2305](#)
- gfxbitmap_font_init
 - font.h, [2305](#)
- gfxbitmap_font_text
 - font.h, [2305](#)
- gfxbitmap_font_text_scale
 - font.h, [2306](#)
- gfxbitmap_font_width
 - font.h, [2307](#)
- gfxbitmap_offset, [914](#)
- gfxbitmap_set
 - bitmap.h, [2301](#)
- global_id
 - L4::Debugger, [948](#)
- gran_offset
 - l4_sched_cpu_set_t, [1367](#)
- granularity
 - l4_sched_cpu_set_t, [1365](#)
- guest_features
 - L4virtio::Svr::Dev_config, [1831](#)
- H_list_item_t
 - cxx::H_list_item_t< ELEM_TYPE >, [842](#)
- handle_expired_timeouts
 - L4::lpc_svr::Timeout_queue, [1120](#)
- handle_mem_cmd_write
 - L4virtio::Svr::Device_t< DATA >, [1844](#)
- handle_request
 - Block_device::Pending_request, [726](#)
- has_alpha

- L4Re::Video::Pixel_info, 1645
- has_more
 - L4virtio::Svr::Request_processor, 1864
- hdr
 - L4virtio::Svr::Dev_config, 1832
- i
 - L4vcpu::Vcpu, 1748
- IA32 Port I/O API, 624
 - I4util_in16, 624
 - I4util_in32, 625
 - I4util_in8, 625
 - I4util_ins16, 626
 - I4util_ins32, 626
 - I4util_ins8, 626
 - I4util_out16, 627
 - I4util_out32, 627
 - I4util_out8, 627
 - I4util_outs16, 629
 - I4util_outs32, 629
 - I4util_outs8, 630
- id
 - L4::Kobject_typeid< T >, 1177
- id_received
 - L4::lpc::Gen_fpage< T >, 1020
- idle_time
 - L4::Scheduler, 1219
- In_area
 - L4Re::Rm::F, 1513
- include.mk - Header File Role, 32
- indirect_bfm_t
 - L4virtio::Virtqueue::Desc::Flags, 1899
- Info
 - L4::Kip::Mem_desc, 1155
- info
 - L4::lcu, 991
 - L4::Scheduler, 1220
 - L4Re::Dataspace, 1419
 - L4Re::Video::Goos, 1636
 - L4Re::Video::View, 1649
- Info_acpi_rsdp
 - L4::Kip::Mem_desc, 1155
- Info_sub_type
 - L4::Kip::Mem_desc, 1155
- inhibitor.h
 - I4re_inhibitor_acquire, 2320
 - I4re_inhibitor_next_lock_info, 2321
 - I4re_inhibitor_release, 2321
- init
 - L4Re::Util::Event_t< PAYLOAD >, 1560
 - L4virtio::Svr::Driver_mem_list_t< DATA >, 1852
- init_infos
 - L4Re::Util::Video::Goos_svr, 1588
- init_mem_info
 - L4virtio::Svr::Device_t< DATA >, 1845
- init_poll
 - L4Re::Util::Event_t< PAYLOAD >, 1561
- init_queue
 - L4virtio::Driver::Virtqueue, 1802, 1803
- Initial Environment, 469
 - I4re_env, 470
 - I4re_env_get_cap, 470
 - I4re_env_get_cap_e, 471
 - I4re_env_get_cap_l, 472
 - I4re_kip, 473
- Initial Environment and Application Bootstrapping, 20
- Initial Memory Allocator and Factory, 23
- initial_caps
 - L4Re::Env, 1443
- initialize_rings
 - L4virtio::Driver::Virtqueue, 1804
- insert
 - cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >, 733
 - cxx::Avl_tree< Node, Get_key, Compare >, 744
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 799
 - cxx::H_list< T, POLICY >, 835
- insert_after
 - cxx::H_list< T, POLICY >, 836
- insert_before
 - cxx::H_list< T, POLICY >, 837
- Integer Types, 176
- interface
 - L4::Meta, 1183
- Interface Definition Language, 25
- Interface for asynchronous ISR handlers with a given IRQ capability., 403
 - I4irq_request_cap, 403
- Interface for asynchronous ISR handlers., 401
 - I4irq_release, 402
 - I4irq_request, 402
- Interface using direct functionality., 404, 408
 - I4irq_attach, 405
 - I4irq_attach_cap, 409
 - I4irq_attach_cap_ft, 409
 - I4irq_attach_ft, 405
 - I4irq_attach_thread, 405
 - I4irq_attach_thread_cap, 409
 - I4irq_attach_thread_cap_ft, 410
 - I4irq_attach_thread_ft, 406
 - I4irq_detach, 406
 - I4irq_unmask, 407
 - I4irq_unmask_and_wait_any, 407
 - I4irq_wait, 407
 - I4irq_wait_any, 408
- Internal, 524
- Internal constants, 557
- Internal functions, 630
- Internal Helpers, 136
- internal_loop
 - L4::Server< LOOP_HOOKS >, 1231
- Interrupt controller, 217
 - I4_icu_bind, 219
 - I4_icu_bind_u, 220
 - L4_ICU_FLAG_MSI, 219
 - L4_icu_flags, 219

- l4_icu_info, [222](#)
- l4_icu_info_t, [219](#)
- l4_icu_info_u, [222](#)
- l4_icu_mask, [223](#)
- l4_icu_mask_u, [224](#)
- l4_icu_msi_info, [225](#)
- l4_icu_msi_info_u, [226](#)
- l4_icu_set_mode, [227](#)
- l4_icu_set_mode_u, [228](#)
- l4_icu_unbind, [229](#)
- l4_icu_unbind_u, [230](#)
- l4_icu_unmask, [231](#)
- l4_icu_unmask_u, [232](#)
- Introduction, [3](#)
- Invalid
 - L4::Cap_base, [932](#)
 - L4virtio::Ptr< T >, [1807](#)
- Invalid_capability
 - L4::Invalid_capability, [999](#)
- Invalid_type
 - L4virtio::Ptr< T >, [1807](#)
- IO interface, [394](#)
 - L4IO_DEVICE_ANY, [396](#)
 - L4IO_DEVICE_INVALID, [396](#)
 - L4IO_DEVICE_OTHER, [396](#)
 - L4IO_DEVICE_PCI, [396](#)
 - l4io_device_types_t, [395](#)
 - L4IO_DEVICE_USB, [396](#)
 - l4io_has_resource, [396](#)
 - l4io_iomem_flags_t, [396](#)
 - l4io_lookup_device, [397](#)
 - l4io_lookup_resource, [397](#)
 - L4IO_MEM_CACHED, [396](#)
 - L4IO_MEM_EAGER_MAP, [396](#)
 - L4IO_MEM_NONCACHED, [396](#)
 - L4IO_MEM_USE_MTRR, [396](#)
 - L4IO_MEM_USE_RESERVED_AREA, [396](#)
 - l4io_release_iomem, [398](#)
 - l4io_release_ioport, [398](#)
 - l4io_request_iomem, [398](#)
 - l4io_request_iomem_region, [399](#)
 - l4io_request_ioport, [400](#)
 - l4io_request_resource_iomem, [400](#)
 - L4IO_RESOURCE_ANY, [396](#)
 - L4IO_RESOURCE_INVALID, [396](#)
 - L4IO_RESOURCE_IRQ, [396](#)
 - L4IO_RESOURCE_MEM, [396](#)
 - L4IO_RESOURCE_PORT, [396](#)
 - l4io_resource_t, [395](#)
 - l4io_resource_types_t, [396](#)
- Io, the Io Server, [53](#)
- io.h
 - l4io_get_root_device, [2045](#)
 - l4io_iterate_devices, [2045](#)
 - l4io_request_all_ioports, [2045](#)
 - l4io_request_icu, [2046](#)
- io_page_fault
 - L4::Io_pager, [1001](#)
- ioctl
 - L4Re::Vfs::Special_file, [1626](#)
- loststream
 - L4::lpc::loststream, [1026](#)
- IPC-Gate API, [200](#)
 - l4_ipc_gate_get_infos, [201](#)
 - l4_rcv_ep_bind_thread, [202](#)
- ipc.h
 - l4_ipc_to_errno, [2659](#)
- ipc_client
 - L4_RPC_DEF, [2577](#)
- ipc_iface
 - L4_INLINE_RPC, [2584](#)
 - L4_INLINE_RPC_NF, [2585](#)
 - L4_INLINE_RPC_NF_OP, [2585](#)
 - L4_INLINE_RPC_OP, [2586](#)
 - L4_RPC, [2586](#)
 - L4_RPC_NF, [2587](#)
 - L4_RPC_NF_OP, [2587](#)
 - L4_RPC_OP, [2587](#)
- ipc_stream
 - operator<<, [2107–2109](#)
 - operator>>, [2109, 2111–2114](#)
- irq
 - L4Re::Util::Event_t< PAYLOAD >, [1561](#)
- IRQ handling library, [401](#)
- irq.h
 - l4util_irq_acknowledge, [2174, 2184](#)
- irq_disable_save
 - L4vcpu::Vcpu, [1748](#)
- irq_enable
 - L4vbus::Pci_dev, [1718](#)
 - L4vbus::Pci_host_bridge, [1724](#)
 - L4vcpu::Vcpu, [1748](#)
- irq_restore
 - L4vcpu::Vcpu, [1749](#)
- IRQs, [203](#)
 - l4_irq_detach, [205](#)
 - l4_irq_detach_u, [206](#)
 - L4_IRQ_F_BOTH, [205](#)
 - L4_IRQ_F_BOTH_EDGE, [205](#)
 - L4_IRQ_F_CLEAR_WAKEUP, [205](#)
 - L4_IRQ_F_EDGE, [205](#)
 - L4_IRQ_F_LEVEL, [205](#)
 - L4_IRQ_F_LEVEL_HIGH, [205](#)
 - L4_IRQ_F_LEVEL_LOW, [205](#)
 - L4_IRQ_F_MASK, [205](#)
 - L4_IRQ_F_NEG, [205](#)
 - L4_IRQ_F_NEG_EDGE, [205](#)
 - L4_IRQ_F_NONE, [205](#)
 - L4_IRQ_F_POS, [205](#)
 - L4_IRQ_F_POS_EDGE, [205](#)
 - L4_IRQ_F_SET_WAKEUP, [205](#)
 - l4_irq_mode, [204](#)
 - l4_irq_mux_chain, [207](#)
 - l4_irq_mux_chain_u, [208](#)
 - l4_irq_receive, [209](#)
 - l4_irq_receive_u, [210](#)

- l4_irq_trigger, 211
- l4_irq_trigger_u, 212
- l4_irq_unmask, 213
- l4_irq_unmask_u, 214
- l4_irq_wait, 215
- l4_irq_wait_u, 216
- is_compatible
 - L4vbus::Device, 1690
- is_compound
 - L4::lpc::Gen_fpage< T >, 1020
- is_irq_entry
 - L4vcpu::Vcpu, 1750
- is_nil
 - L4::lpc::Varg, 1079
- is_of
 - L4::lpc::Varg, 1079
- is_of_int
 - L4::lpc::Varg, 1080
- is_online
 - L4::Scheduler, 1221
- is_owner
 - Block_device::Pending_request, 727
- is_page_fault_entry
 - L4vcpu::Vcpu, 1750
- is_static
 - L4Re::Util::Dataspace_svr, 1545
- is_valid
 - L4::Cap_base, 936
 - L4Re::Rm::Unique_region< T >, 1519
 - L4virtio::Ptr< T >, 1808
- is_virtual
 - L4::Kip::Mem_desc, 1160
- is_writable
 - L4virtio::Svr::Driver_mem_region_t< DATA >, 1861
- Istream
 - L4::lpc::Istream, 1035
- Item_bytes
 - L4::lpc::Msg, 675
- Item_words
 - L4::lpc::Msg, 675
- iter
 - cxx::Bits::Basic_list< POLICY >, 808
 - cxx::H_list< T, POLICY >, 837
- Iterator
 - cxx::Avl_tree< Node, Get_key, Compare >, 744
- kdebug.h
 - __kdebug_3_text, 2676
 - __kdebug_op, 2677
 - __kdebug_op_1, 2678
 - __kdebug_text, 2679
 - enter_kdebug, 2681
 - l4_kdebug_ops_t, 2675
 - outchar, 2681
 - outdec, 2682
 - outhex12, 2683
 - outhex16, 2683
 - outhex20, 2684
 - outhex32, 2684
 - outhex64, 2685
 - outhex8, 2686
 - outnstring, 2686
 - outstring, 2687
 - outumword, 2688
- Kept_ds
 - L4Re::Rm, 1504
- Kernel Debugger, 150
 - l4_debugger_get_object_name, 151
 - l4_debugger_global_id, 151
 - l4_debugger_kobj_to_id, 152
 - l4_debugger_query_log_name, 153
 - l4_debugger_query_log_typeid, 154
 - l4_debugger_set_object_name, 154
 - l4_debugger_switch_log, 155
- Kernel Factory, 40
- Kernel Interface Page, 177
 - l4_kernel_info_version_offset, 179
 - l4_kip, 180
 - l4_kip_clock, 180
 - l4_kip_clock_lw, 181
 - l4_kip_clock_ns, 182
 - L4_KIP_OFFS_READ_NS, 179
 - L4_KIP_OFFS_READ_US, 179
 - l4_kip_version, 182
 - l4_kip_version_string, 183
- Kernel Interface Page API, 631
 - l4util_kip_for_each_feature, 631
 - l4util_kip_kernel_abi_version, 632
 - l4util_kip_kernel_has_feature, 632
 - l4util_kip_kernel_is_ux, 632
- Kernel Objects, 190
- Kernel Tracing, 156
 - fiasco_tbuf_log, 157
 - fiasco_tbuf_log_3val, 157
 - fiasco_tbuf_log_binary, 158
- Kernel-provided semaphore, 233
 - l4_semaphore_down, 233
 - l4_semaphore_up, 234
- kip.h
 - l4_kip_for_each_feature, 2820
 - l4_kip_kernel_has_feature, 2820
- kobj_to_id
 - L4::Debugger, 948
- kobject_typeid
 - L4 kernel object type information, 236
- Kumem allocator utility, 474
- Kumem utilities, 516
 - kumem_alloc, 516
- kumem_alloc
 - Kumem utilities, 516
- kumem_alloc.h
 - l4re_util_kumem_alloc, 2335
- L
 - cxx::Bits::Direction, 826
- L4, 655
 - cap_cast, 660

- cap_dynamic_cast, [661](#)
- cap_reinterpret_cast, [662](#)
- PROTO_ANY, [659](#)
- PROTO_EMPTY, [659](#)
- round_order, [663](#)
- throw_ipc_exception, [664](#), [665](#)
- trunc_order, [665](#)
- L4 IPC Opcodes, [410](#)
 - L4_ICU_OP_BIND, [411](#)
 - L4_ICU_OP_INFO, [412](#)
 - L4_ICU_OP_MASK, [412](#)
 - L4_ICU_OP_MSI_INFO, [412](#)
 - L4_ICU_OP_SET_MODE, [412](#)
 - L4_ICU_OP_UNBIND, [411](#)
 - L4_ICU_OP_UNMASK, [412](#)
 - L4_icu_opcode, [411](#)
 - L4_IPC_GATE_BIND_OP, [412](#)
 - L4_IPC_GATE_GET_INFO_OP, [412](#)
 - L4_ipc_gate_ops, [412](#)
 - L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP, [413](#)
 - L4_PLATFORM_CTL_CPU_DISABLE_OP, [413](#)
 - L4_PLATFORM_CTL_CPU_ENABLE_OP, [413](#)
 - L4_platform_ctl_ops, [412](#)
 - L4_PLATFORM_CTL_SYS_SHUTDOWN_OP, [413](#)
 - L4_PLATFORM_CTL_SYS_SUSPEND_OP, [413](#)
 - L4_TASK_ADD_KU_MEM_OP, [413](#)
 - L4_TASK_CAP_INFO_OP, [413](#)
 - L4_TASK_LDT_SET_X86_OP, [413](#)
 - L4_TASK_MAP_OP, [413](#)
 - L4_TASK_MAP_VGICC_ARM_OP, [413](#)
 - L4_task_ops, [413](#)
 - L4_TASK_UNMAP_OP, [413](#)
 - L4_THREAD_AMD64_GET_SEGMENT_INFO_OP, [413](#)
 - L4_THREAD_AMD64_SET_SEGMENT_BASE_OP, [413](#)
 - L4_THREAD_ARM_TPIDRURO_OP, [413](#)
 - L4_THREAD_CONTROL_OP, [413](#)
 - L4_THREAD_EX_REGS_OP, [413](#)
 - L4_THREAD_MODIFY_SENDER_OP, [413](#)
 - L4_THREAD_OPCODE_MASK, [413](#)
 - L4_thread_ops, [413](#)
 - L4_THREAD_REGISTER_DELETE_IRQ_OP, [413](#)
 - L4_THREAD_STATS_OP, [413](#)
 - L4_THREAD_SWITCH_OP, [413](#)
 - L4_THREAD_VCPU_CONTROL_OP, [413](#)
 - L4_THREAD_VCPU_RESUME_OP, [413](#)
 - L4_THREAD_X86_GDT_OP, [413](#)
 - L4_VCON_GET_ATTR_OP, [414](#)
 - L4_vcon_ops, [414](#)
 - L4_VCON_READ_OP, [414](#)
 - L4_VCON_SET_ATTR_OP, [414](#)
 - L4_VCON_WRITE_OP, [414](#)
- L4 kernel object type information, [235](#)
 - kobject_typeid, [236](#)
- L4 Vbus functions, [426](#)
 - l4vbus_assign_dma_domain, [428](#)
 - L4vbus_dma_domain_assign_flags, [427](#)
 - L4VBUS_DMAD_BIND, [428](#)
 - L4VBUS_DMAD_KERNEL_DMA_SPACE, [428](#)
 - L4VBUS_DMAD_L4RE_DMA_SPACE, [428](#)
 - L4VBUS_DMAD_UNBIND, [428](#)
 - l4vbus_get_adr, [429](#)
 - l4vbus_get_device, [429](#)
 - l4vbus_get_device_by_hid, [430](#)
 - l4vbus_get_hid, [431](#)
 - l4vbus_get_next_device, [431](#)
 - l4vbus_get_resource, [433](#)
 - l4vbus_is_compatible, [434](#)
 - l4vbus_release_ioport, [434](#)
 - l4vbus_request_ioport, [435](#)
 - l4vbus_vicu_get_cap, [436](#)
- L4 VIRTIO Block Device, [415](#)
 - L4virtio_block_operations, [415](#)
 - L4VIRTIO_BLOCK_S_IOERR, [416](#)
 - L4VIRTIO_BLOCK_S_OK, [416](#)
 - L4VIRTIO_BLOCK_S_UNSUPP, [416](#)
 - L4virtio_block_status, [416](#)
 - L4VIRTIO_BLOCK_T_DISCARD, [416](#)
 - L4VIRTIO_BLOCK_T_FLUSH, [416](#)
 - L4VIRTIO_BLOCK_T_GET_ID, [416](#)
 - L4VIRTIO_BLOCK_T_IN, [416](#)
 - L4VIRTIO_BLOCK_T_OUT, [416](#)
 - L4VIRTIO_BLOCK_T_WRITE_ZEROES, [416](#)
- L4 VIRTIO Input Device, [416](#)
- L4 VIRTIO Interface, [414](#)
- L4 VIRTIO Network Device, [417](#)
- L4 VIRTIO Transport Layer, [418](#)
 - L4_virtio_cmd, [420](#)
 - L4_virtio_irq_status, [420](#)
 - L4_virtio_opcodes, [421](#)
 - L4VIRTIO_CMD_CFG_QUEUE, [420](#)
 - L4VIRTIO_CMD_MASK, [420](#)
 - L4VIRTIO_CMD_NONE, [420](#)
 - L4VIRTIO_CMD_SET_STATUS, [420](#)
 - l4virtio_config_queue, [422](#)
 - l4virtio_config_queue_t, [420](#)
 - l4virtio_config_queues, [423](#)
 - l4virtio_device_config, [423](#)
 - l4virtio_device_config_ds, [423](#)
 - L4virtio_device_ids, [421](#)
 - l4virtio_device_notification_irq, [424](#)
 - L4virtio_device_status, [422](#)
 - L4virtio_feature_bits, [422](#)
 - L4VIRTIO_FEATURE_CMD_CONFIG, [422](#)
 - L4VIRTIO_FEATURE_VERSION_1, [422](#)
 - L4VIRTIO_ID_9P, [421](#)
 - L4VIRTIO_ID_BALLOON, [421](#)
 - L4VIRTIO_ID_BLOCK, [421](#)
 - L4VIRTIO_ID_CAIF, [421](#)
 - L4VIRTIO_ID_CONSOLE, [421](#)
 - L4VIRTIO_ID_CRYPT, [421](#)
 - L4VIRTIO_ID_GPU, [421](#)
 - L4VIRTIO_ID_INPUT, [421](#)

- L4VIRTIO_ID_NET, [421](#)
- L4VIRTIO_ID_RNG, [421](#)
- L4VIRTIO_ID_RPMMSG, [421](#)
- L4VIRTIO_ID_RPROC_SERIAL, [421](#)
- L4VIRTIO_ID_SCSI, [421](#)
- L4VIRTIO_ID_SOCKET, [421](#)
- L4VIRTIO_ID_VSOCKET, [421](#)
- L4VIRTIO_IRQ_STATUS_CONFIG, [421](#)
- L4VIRTIO_IRQ_STATUS_VRING, [421](#)
- L4VIRTIO_OP_CONFIG_QUEUE, [421](#)
- L4VIRTIO_OP_DEVICE_CONFIG, [421](#)
- L4VIRTIO_OP_GET_DEVICE_IRQ, [421](#)
- L4VIRTIO_OP_REGISTER_DS, [421](#)
- L4VIRTIO_OP_SET_STATUS, [421](#)
- l4virtio_register_ds, [424](#)
- l4virtio_set_status, [425](#)
- L4VIRTIO_STATUS_ACKNOWLEDGE, [422](#)
- L4VIRTIO_STATUS_DEVICE_NEEDS_RESET, [422](#)
- L4VIRTIO_STATUS_DRIVER, [422](#)
- L4VIRTIO_STATUS_DRIVER_OK, [422](#)
- L4VIRTIO_STATUS_FAILED, [422](#)
- L4VIRTIO_STATUS_FEATURES_OK, [422](#)
- l4/cxx/alloc.h, [2047](#), [2048](#)
- l4/cxx/arith, [2048](#)
- l4/cxx/atomic.h, [2782](#), [2783](#)
- l4/cxx/avl_map, [2049](#), [2050](#)
- l4/cxx/avl_set, [2052](#), [2053](#)
- l4/cxx/avl_tree, [2056](#), [2058](#)
- l4/cxx/basic_ostream, [2062](#), [2063](#)
- l4/cxx/basic_vector.h, [2066](#), [2067](#)
- l4/cxx/bitfield, [2068](#)
- l4/cxx/bitmap, [2070](#)
- l4/cxx/bits/bst.h, [2072](#), [2075](#)
- l4/cxx/bits/bst_base.h, [2077](#), [2079](#)
- l4/cxx/bits/bst_iter.h, [2080](#), [2082](#)
- l4/cxx/bits/list_basics.h, [2083](#)
- l4/cxx/bits/smart_ptr_list.h, [2085](#), [2087](#)
- l4/cxx/bits/type_traits.h, [2088](#)
- l4/cxx/dlist, [2091](#)
- l4/cxx/exceptions, [2094](#), [2096](#)
- l4/cxx/hlist, [2098](#)
- l4/cxx/iostream, [2101](#), [2102](#)
- l4/cxx/ipc_helper, [2102](#), [2103](#)
- l4/cxx/ipc_server, [2595](#), [2597](#)
- l4/cxx/ipc_stream, [2104](#), [2115](#)
- l4/cxx/ipc_timeout_queue, [2123](#)
- l4/cxx/l4iostream, [2125](#), [2126](#)
- l4/cxx/l4types.h, [2126](#), [2127](#)
- l4/cxx/list, [2128](#)
- l4/cxx/list_alloc, [2132](#)
- l4/cxx/main_thread, [2137](#), [2138](#)
- l4/cxx/minmax, [2139](#)
- l4/cxx/observer, [2139](#)
- l4/cxx/pair, [2140](#), [2141](#)
- l4/cxx/ref_ptr, [2142](#)
- l4/cxx/ref_ptr_list, [2145](#), [2147](#)
- l4/cxx/slab_alloc, [2147](#)
- l4/cxx/slist, [2151](#)
- l4/cxx/static_container, [2153](#)
- l4/cxx/static_vector, [2154](#)
- l4/cxx/std_alloc, [2155](#)
- l4/cxx/std_exc_io, [2156](#)
- l4/cxx/std_ops, [2157](#)
- l4/cxx/string, [2157](#)
- l4/cxx/string.h, [2160](#), [2163](#)
- l4/cxx/thread, [2742](#), [2744](#)
- l4/cxx/type_list, [2163](#)
- l4/cxx/type_traits, [2164](#)
- l4/cxx/unique_ptr, [2168](#)
- l4/cxx/unique_ptr_list, [2170](#), [2171](#)
- l4/cxx/utils, [2171](#)
- l4/cxx/weak_ref, [2172](#)
- l4/irq/irq.h, [2177](#), [2178](#)
- l4/l4re_vfs/backend, [2186](#)
- l4/l4re_vfs/impl/default_ops_impl.h, [2189](#)
- l4/l4re_vfs/impl/fd_store.h, [2190](#)
- l4/l4re_vfs/impl/fd_store_impl.h, [2191](#)
- l4/l4re_vfs/impl/ns_fs.h, [2191](#)
- l4/l4re_vfs/impl/ns_fs_impl.h, [2192](#)
- l4/l4re_vfs/impl/ro_file.h, [2197](#)
- l4/l4re_vfs/impl/ro_file_impl.h, [2198](#)
- l4/l4re_vfs/impl/vcon_stream.h, [2199](#)
- l4/l4re_vfs/impl/vcon_stream_impl.h, [2200](#)
- l4/l4re_vfs/impl/vfs_impl.h, [2202](#)
- l4/l4re_vfs/vfs.h, [2214](#)
- l4/l4virtio/client/l4virtio, [2224](#)
- l4/l4virtio/client/virtio-block, [2243](#)
- l4/l4virtio/client/virtio-net, [2221](#)
- l4/l4virtio/l4virtio, [2227](#)
- l4/l4virtio/server/l4virtio, [2228](#)
- l4/l4virtio/server/virtio, [2239](#)
- l4/l4virtio/server/virtio-block, [2246](#)
- l4/l4virtio/virtio.h, [2252](#)
- l4/l4virtio/virtio_block.h, [2255](#)
- l4/l4virtio/virtio_input.h, [2256](#)
- l4/l4virtio/virtio_net.h, [2257](#)
- l4/l4virtio/virtqueue, [2257](#)
- l4/libblock-device/block_device_mgr.h, [2262](#)
- l4/libblock-device/debug.h, [2267](#)
- l4/libblock-device/device.h, [2269](#)
- l4/libblock-device/errand.h, [2270](#)
- l4/libblock-device/gpt.h, [2272](#)
- l4/libblock-device/inout_memory.h, [2272](#)
- l4/libblock-device/part_device.h, [2274](#)
- l4/libblock-device/partition.h, [2276](#)
- l4/libblock-device/request_queue.h, [2279](#)
- l4/libblock-device/types.h, [2281](#)
- l4/libblock-device/virtio_client.h, [2287](#)
- l4/libedid/edid.h, [2295](#), [2296](#)
- l4/libgfxbitmap/bitmap.h, [2297](#), [2301](#)
- l4/libgfxbitmap/font.h, [2302](#), [2307](#)
- l4/libgfxbitmap/support, [2308](#), [2309](#)
- l4/re/c/dataspace.h, [2309](#), [2311](#)
- l4/re/c/debug.h, [2267](#), [2268](#)
- l4/re/c/dma_space.h, [2312](#), [2314](#)

l4/re/c/event.h, 2315, 2316
l4/re/c/event_buffer.h, 2319
l4/re/c/inhibitor.h, 2319, 2322
l4/re/c/log.h, 2322, 2323
l4/re/c/mem_alloc.h, 2324, 2326
l4/re/c/namespace.h, 2326, 2328
l4/re/c/rm.h, 2328, 2330
l4/re/c/util/cap_alloc.h, 2332, 2333
l4/re/c/util/kumem_alloc.h, 2334, 2335
l4/re/c/util/video/goos_fb.h, 2336, 2337
l4/re/c/video/colors.h, 2338, 2339
l4/re/c/video/goos.h, 2340, 2342
l4/re/c/video/view.h, 2343, 2345
l4/re/cap_alloc, 2412, 2414
l4/re/console, 2346
l4/re/consts, 2566, 2567
l4/re/consts.h, 2557, 2558
l4/re/dataspace, 2346, 2348
l4/re/dataspace-sys.h, 2349, 2350
l4/re/debug, 2427, 2429
l4/re/dma_space, 2350, 2352
l4/re/elf_aux.h, 2353, 2354
l4/re/env, 2355, 2356
l4/re/env.h, 2358, 2360
l4/re/error_helper, 2361, 2363
l4/re/event, 2432
l4/re/event-sys.h, 2364
l4/re/event.h, 2317, 2318
l4/re/event_enums.h, 2365
l4/re/impl/dataspace_impl.h, 2371, 2372
l4/re/impl/mem_alloc_impl.h, 2374, 2375
l4/re/impl/namespace_impl.h, 2375, 2376
l4/re/impl/rm_impl.h, 2378, 2379
l4/re/inhibitor, 2380
l4/re/inhibitor-sys.h, 2381
l4/re/l4aux.h, 2381, 2382
l4/re/log, 2382, 2384
l4/re/log-sys.h, 2384, 2385
l4/re/mem_alloc, 2385, 2387
l4/re/mem_alloc-sys.h, 2387, 2388
l4/re/mmio_space, 2389, 2390
l4/re/namespace, 2390, 2392
l4/re/namespace-sys.h, 2393, 2394
l4/re/parent, 2394, 2396
l4/re/parent-sys.h, 2396, 2397
l4/re/protocols.h, 2397, 2398
l4/re/random, 2399, 2400
l4/re/rm, 2400, 2402
l4/re/rm-sys.h, 2406
l4/re/shared_cap, 2461, 2463
l4/re/unique_cap, 2467, 2469
l4/re/util/bitmap_cap_alloc, 2407, 2408
l4/re/util/br_manager, 2409
l4/re/util/cap, 2411, 2412
l4/re/util/cap_alloc, 2416, 2418
l4/re/util/cap_alloc_impl.h, 2419, 2420
l4/re/util/counting_cap_alloc, 2421, 2423
l4/re/util/dataspace_svr, 2425
l4/re/util/debug, 2429
l4/re/util/env_ns, 2431
l4/re/util/event, 2435, 2436
l4/re/util/event_buffer, 2437
l4/re/util/event_svr, 2438
l4/re/util/icu_svr, 2440
l4/re/util/item_alloc, 2442, 2444
l4/re/util/kumem_alloc, 2445, 2446
l4/re/util/meta, 2706
l4/re/util/name_space_svr, 2446
l4/re/util/object_registry, 2449
l4/re/util/poll_timeout_kipclock, 2452
l4/re/util/region_mapping, 2452, 2454
l4/re/util/region_mapping_svr_2, 2459
l4/re/util/shared_cap, 2464, 2466
l4/re/util/unique_cap, 2469, 2471
l4/re/util/vcon_svr, 2472
l4/re/util/video/goos_fb, 2473
l4/re/util/video/goos_svr, 2474
l4/re/video/colors, 2476
l4/re/video/goos, 2477
l4/re/video/goos-sys.h, 2480, 2481
l4/re/video/view, 2482
l4/shmc/ringbuf.h, 2482, 2490
l4/shmc/shmc.h, 2491, 2494
l4/sigma0/sigma0.h, 2496, 2498
l4/sys/_kernel_object_impl.h, 2500
l4/sys/_kip-32bit.h, 2501
l4/sys/_kip-64bit.h, 2502
l4/sys/_ktrace-impl.h, 2503, 2505
l4/sys/_l4_fpage.h, 2506
l4/sys/_task-arm.h, 2510
l4/sys/_timeout.h, 2510
l4/sys/_typeinfo.h, 2513, 2515
l4/sys/_vcpu-arm.h, 2525
l4/sys/_vm-arm.h, 2526, 2528
l4/sys/_vm-svm.h, 2528
l4/sys/_vm-vmx.h, 2530
l4/sys/arm_smccc, 2534, 2536
l4/sys/arm_smccc.h, 2536, 2538
l4/sys/assert.h, 2775, 2777
l4/sys/cache.h, 2543, 2545
l4/sys/capability, 2547, 2549
l4/sys/compiler.h, 2551, 2552
l4/sys/consts.h, 2559, 2561
l4/sys/cxx/capability.h, 2564
l4/sys/cxx/consts, 2567
l4/sys/cxx/ipc_array, 2568
l4/sys/cxx/ipc_basics, 2572
l4/sys/cxx/ipc_client, 2575, 2578
l4/sys/cxx/ipc_epiface, 2578
l4/sys/cxx/ipc_iface, 2582, 2588
l4/sys/cxx/ipc_legacy, 2593
l4/sys/cxx/ipc_ret_array, 2594
l4/sys/cxx/ipc_server, 2598
l4/sys/cxx/ipc_server_loop, 2601
l4/sys/cxx/ipc_string, 2604
l4/sys/cxx/ipc_types, 2606, 2608

[l4/sys/cxx/ipc_varg](#), 2614
[l4/sys/cxx/smart_capability_1x](#), 2620, 2621
[l4/sys/cxx/types](#), 2623, 2624
[l4/sys/debugger](#), 2627, 2628
[l4/sys/debugger.h](#), 2629, 2630
[l4/sys/err.h](#), 2633, 2635
[l4/sys/exception](#), 2635, 2637
[l4/sys/factory](#), 2637, 2639
[l4/sys/factory.h](#), 2641, 2642
[l4/sys/icu](#), 2647, 2648
[l4/sys/icu.h](#), 2648, 2651
[l4/sys/iommu](#), 2654
[l4/sys/ipc.h](#), 2657, 2659
[l4/sys/ipc_gate](#), 2664, 2666
[l4/sys/ipc_gate.h](#), 2666, 2669
[l4/sys/irq](#), 2670, 2671
[l4/sys/irq.h](#), 2179, 2181
[l4/sys/kdebug.h](#), 2673, 2689
[l4/sys/kernel_object.h](#), 2691, 2692
[l4/sys/kip](#), 2693, 2694
[l4/sys/kip.h](#), 2818, 2821
[l4/sys/kobject](#), 2695
[l4/sys/ktrace.h](#), 2696, 2698
[l4/sys/l4int.h](#), 2700, 2701
[l4/sys/memdesc.h](#), 2702, 2704
[l4/sys/meta](#), 2706, 2708
[l4/sys/pager](#), 2708, 2710
[l4/sys/platform_control](#), 2710, 2712
[l4/sys/platform_control.h](#), 2712, 2714
[l4/sys/rcv_endpoint](#), 2716, 2718
[l4/sys/rcv_endpoint.h](#), 2718, 2720
[l4/sys/scheduler](#), 2721, 2722
[l4/sys/scheduler.h](#), 2723, 2725
[l4/sys/semaphore](#), 2727, 2729
[l4/sys/semaphore.h](#), 2729, 2731
[l4/sys/smart_capability](#), 2732, 2733
[l4/sys/task](#), 2735, 2737
[l4/sys/task.h](#), 2738, 2739
[l4/sys/thread](#), 2744, 2746
[l4/sys/thread.h](#), 2846, 2848
[l4/sys/typeinfo_svr](#), 2748, 2750
[l4/sys/types.h](#), 2282, 2284
[l4/sys/utcb.h](#), 2755, 2757
[l4/sys/vcon](#), 2762, 2764
[l4/sys/vcon.h](#), 2764, 2767
[l4/sys/vcpu](#), 2878, 2880
[l4/sys/vhw.h](#), 2770, 2772
[l4/sys/vm](#), 2773, 2775
[l4/util/assert.h](#), 2778, 2779
[l4/util/atomic.h](#), 2783, 2786
[l4/util/backtrace.h](#), 2790, 2792
[l4/util/base64.h](#), 2792, 2793
[l4/util/bitops.h](#), 2793, 2796
[l4/util/elf.h](#), 2799, 2805
[l4/util/getopt.h](#), 2815, 2816
[l4/util/keymap.h](#), 2817, 2818
[l4/util/kip.h](#), 2822, 2823
[l4/util/kprintf.h](#), 2824, 2825
[l4/util/l4_macros.h](#), 2037
[l4/util/l4mod.h](#), 2825, 2826
[l4/util/list_alloc.h](#), 2827, 2829
[l4/util/llulc.h](#), 2829
[l4/util/lock.h](#), 2830, 2831
[l4/util/mb_info.h](#), 2832, 2836
[l4/util/parse_cmd.h](#), 2840, 2841
[l4/util/rand.h](#), 2841, 2843
[l4/util/splitlog2.h](#), 2843, 2844
[l4/util/thread.h](#), 2855, 2856
[l4/util/util.h](#), 1970
[l4/vbus/vbus](#), 2857
[l4/vbus/vbus.h](#), 2858, 2861
[l4/vbus/vbus_generic](#), 2862
[l4/vbus/vbus_gpio](#), 2862
[l4/vbus/vbus_gpio-ops.h](#), 2864
[l4/vbus/vbus_gpio.h](#), 2864
[l4/vbus/vbus_i2c.h](#), 2865
[l4/vbus/vbus_inhibitor.h](#), 2865
[l4/vbus/vbus_interfaces.h](#), 2865, 2868
[l4/vbus/vbus_mcspi.h](#), 2869
[l4/vbus/vbus_pci](#), 2869
[l4/vbus/vbus_pci-ops.h](#), 2870
[l4/vbus/vbus_pci.h](#), 2871
[l4/vbus/vbus_pm-ops.h](#), 2871
[l4/vbus/vbus_pm.h](#), 2871
[l4/vbus/vbus_types.h](#), 2872, 2874
[l4/vbus/vdevice-ops.h](#), 2875
[l4/vcpu/vcpu](#), 2876
[l4/vcpu/vcpu.h](#), 2881, 2883
[L4::Alloc_list](#), 915
[L4::Arm_smccc](#), 915
 [call](#), 916
[L4::Base_exception](#), 917
[L4::Basic_registry](#), 919
 [dispatch](#), 920
 [find](#), 921
[L4::Bounds_error](#), 922
[L4::Cap< T >](#), 924
 [Cap](#), 928
 [copy](#), 929
 [move](#), 929
[L4::Cap_base](#), 930
 [cap](#), 933
 [Cap_base](#), 932
 [Cap_type](#), 932
 [copy](#), 934
 [fpage](#), 935
 [Invalid](#), 932
 [is_valid](#), 936
 [move](#), 937
 [No_init](#), 932
 [No_init_type](#), 932
 [snd_base](#), 938
 [validate](#), 938, 939
[L4::Com_error](#), 940
 [Com_error](#), 943
[L4::Debugger](#), 944

- get_object_name, 947
- global_id, 948
- kobj_to_id, 948
- query_log_name, 949
- query_log_typeid, 950
- set_object_name, 951
- switch_log, 951
- L4::Element_already_exists, 952
- L4::Element_not_found, 955
- L4::Epiface, 958
 - dispatch, 960
 - get_buffer_demand, 961
 - obj_cap, 961
 - server_iface, 961
 - set_server, 962
- L4::Epiface_t< Derived, IFACE, BASE, bool >, 963
 - dispatch, 966
- L4::Epiface_t0< RPC_IFACE, BASE >, 966
 - obj_cap, 968
- L4::Exception, 969
 - exception, 969
- L4::Exception_tracer, 970
- L4::Factory, 972
 - create, 975, 976
 - create_factory, 977
 - create_gate, 978
 - create_task, 979
- L4::Factory::Lstr, 980
 - Lstr, 981
- L4::Factory::Nil, 982
- L4::Factory::S, 982
 - operator l4_msgtag_t, 984
 - put, 985, 986
 - S, 984
- L4::lcu, 986
 - bind, 990
 - info, 991
 - mask, 992
 - msi_info, 993
 - set_mode, 993
 - unbind, 994
- L4::lcu::Info, 995
- L4::Invalid_capability, 997
 - cap, 999
 - Invalid_capability, 999
- L4::io_pager, 1000
 - io_page_fault, 1001
- L4::lommu, 1002
 - bind, 1004
 - unbind, 1004
- L4::IOModifier, 1004
- L4::lpc, 666
 - buf_cp_in, 668
 - buf_cp_out, 669
 - buf_in, 669
 - make_cap, 670
 - make_cap_full, 670
 - make_cap_rw, 671
 - make_cap_rws, 672
 - msg_ptr, 672
 - read, 673
 - str_cp_in, 673
- L4::lpc::Array< ELEM_TYPE, LEN_TYPE >, 1005
- L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >, 1008
- L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >, 1009
- L4::lpc::As_value< T >, 1010
- L4::lpc::Buf_item, 1011
- L4::lpc::Call, 1012
- L4::lpc::Call_t< RIGHTS >, 1013
- L4::lpc::Call_zero_send_timeout, 1014
- L4::lpc::Cap< T >, 1016
 - Cap, 1017
 - Cap_mask, 1017
 - from_ci, 1018
 - Rights_mask, 1017
- L4::lpc::Gen_fpage< T >, 1018
 - cap_received, 1020
 - id_received, 1020
 - is_compound, 1020
 - local_id_received, 1020
- L4::lpc::In_out< T >, 1021
- L4::lpc::Iostream, 1022
 - call, 1027
 - get, 1027, 1028
 - lostream, 1026
 - put, 1029
 - reply_and_wait, 1029, 1030
 - reset, 1031
- L4::lpc::Istream, 1032
 - get, 1036, 1037
 - Istream, 1035
 - receive, 1038
 - reset, 1039
 - skip, 1039
 - tag, 1040
 - wait, 1041, 1042
- L4::lpc::Msg, 674
 - align_to, 675, 677
 - Br_bytes, 675
 - check_size, 678
 - Item_bytes, 675
 - Item_words, 675
 - Mr_bytes, 675
 - Mr_words, 675
 - msg_add, 679
 - msg_get, 679
 - Word_bytes, 675
- L4::lpc::Msg::CInt_val_ops< MTYPE, DIR, CLASS >, 1043
- L4::lpc::Msg::Cls_buffer, 1044
- L4::lpc::Msg::Cls_data, 1045
- L4::lpc::Msg::Cls_item, 1046
- L4::lpc::Msg::Dir_in, 1047
- L4::lpc::Msg::Dir_out, 1048
- L4::lpc::Msg::Do_in_data, 1049

- L4::lpc::Msg::Do_in_items, [1050](#)
- L4::lpc::Msg::Do_out_data, [1051](#)
- L4::lpc::Msg::Do_out_items, [1052](#)
- L4::lpc::Msg::Do_rcv_buffers, [1053](#)
- L4::lpc::Msg::Elem< Array< A, LEN > >, [1056](#)
- L4::lpc::Msg::Elem< Array< A, LEN > & >, [1055](#)
- L4::lpc::Msg::Elem< Array_ref< A, LEN > & >, [1057](#)
- L4::lpc::Msg::Is_valid_rpc_type< T >, [1058](#)
- L4::lpc::Msg::Svr_arg_pack< IPC_TYPE >, [1060](#)
- L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >, [1060](#)
- L4::lpc::Msg_ptr< T >, [1061](#)
 - Msg_ptr, [1062](#)
- L4::lpc::Opt< T >, [1062](#)
- L4::lpc::Ostream, [1064](#)
 - put, [1067](#), [1068](#)
 - send, [1068](#)
 - tag, [1069](#)
- L4::lpc::Out< T >, [1070](#)
- L4::lpc::Ret_array< T >, [1071](#)
- L4::lpc::Send_only, [1072](#)
- L4::lpc::Small_buf, [1073](#)
 - Small_buf, [1073](#), [1074](#)
- L4::lpc::Snd_item, [1074](#)
- L4::lpc::Str_cp_in< T >, [1075](#)
 - Str_cp_in, [1076](#)
- L4::lpc::Varg, [1076](#)
 - data, [1078](#)
 - get_value, [1078](#)
 - is_nil, [1079](#)
 - is_of, [1079](#)
 - is_of_int, [1080](#)
 - length, [1080](#)
 - tag, [1080](#)
 - type, [1081](#)
 - value, [1081](#)
- L4::lpc::Varg_list< MAX >, [1082](#)
- L4::lpc::Varg_list_ref, [1084](#)
 - Varg_list_ref, [1086](#)
- L4::lpc::Varg_list_ref::Iterator, [1086](#)
- L4::lpc_gate, [1087](#)
 - get_infos, [1092](#)
- L4::lpc_svr, [680](#)
- L4::lpc_svr::Br_manager_no_buffers, [1092](#)
 - alloc_buffer_demand, [1095](#)
- L4::lpc_svr::Compound_reply, [1096](#)
- L4::lpc_svr::Default_loop_hooks, [1098](#)
- L4::lpc_svr::Default_setup_wait, [1101](#)
- L4::lpc_svr::Default_timeout, [1102](#)
- L4::lpc_svr::Direct_dispatch< R >, [1104](#)
- L4::lpc_svr::Direct_dispatch< R * >, [1106](#)
- L4::lpc_svr::Exc_dispatch< R, Exc >, [1107](#)
- L4::lpc_svr::Ignore_errors, [1109](#)
- L4::lpc_svr::Server_iface, [1110](#)
 - add_timeout, [1112](#)
 - alloc_buffer_demand, [1112](#)
 - get_rcv_cap, [1113](#)
 - rcv_cap, [1113](#), [1114](#)
 - realloc_rcv_cap, [1115](#)
 - remove_timeout, [1115](#)
- L4::lpc_svr::Timeout, [1116](#)
 - expired, [1118](#)
 - timeout, [1118](#)
- L4::lpc_svr::Timeout_queue, [1119](#)
 - add, [1120](#)
 - handle_expired_timeouts, [1120](#)
 - next_timeout, [1121](#)
 - remove, [1122](#)
 - timeout_expired, [1123](#)
- L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >, [1124](#)
 - add_timeout, [1128](#)
 - remove_timeout, [1129](#)
- L4::lrq, [1130](#)
 - detach, [1134](#)
 - receive, [1135](#)
 - unmask, [1136](#)
 - wait, [1137](#)
- L4::lrq_eoi, [1138](#)
 - unmask, [1139](#)
- L4::lrq_handler_object, [1140](#)
- L4::lrq_mux, [1144](#)
 - chain, [1147](#)
- L4::lrqp_t< Derived, BASE, bool >, [1148](#)
 - dispatch, [1152](#)
 - obj_cap, [1152](#)
- L4::Kip::Mem_desc, [1153](#)
 - all, [1156](#)
 - Arch, [1155](#)
 - Arch_acpi_nvs, [1155](#)
 - Arch_acpi_tables, [1155](#)
 - Arch_sub_type_common, [1154](#)
 - Bootloader, [1155](#)
 - Conventional, [1155](#)
 - count, [1157](#), [1158](#)
 - Dedicated, [1155](#)
 - end, [1158](#)
 - first, [1159](#)
 - Info, [1155](#)
 - Info_acpi_rsd, [1155](#)
 - Info_sub_type, [1155](#)
 - is_virtual, [1160](#)
 - Mem_desc, [1155](#)
 - Mem_type, [1155](#)
 - Reserved, [1155](#)
 - set, [1160](#)
 - Shared, [1155](#)
 - size, [1161](#)
 - start, [1161](#)
 - sub_type, [1162](#)
 - type, [1162](#)
 - Undefined, [1155](#)
- L4::Kobject, [1163](#)
 - cap, [1164](#)
 - dec_refcnt, [1165](#)

- L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, 1166
 - __lface, 1168
 - __lface_list, 1168
 - __check_protocols__, 1168
 - c, 1168
 - Class, 1168
- L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, 1169
 - __lface, 1172
 - __lface_list, 1172
 - __check_protocols__, 1172
 - c, 1172
 - Class, 1172
- L4::Kobject_demand< T >, 1173
- L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >, 1174
- L4::Kobject_typeid< T >, 1175
 - Demand, 1176
 - demand, 1177
 - id, 1177
 - proto_dispatch, 1177
- L4::Kobject_typeid< void >, 1178
- L4::Kobject_x< Derived, ARGS >, 1179
- L4::Meta, 1180
 - interface, 1183
 - num_interfaces, 1184
 - supports, 1184
- L4::Out_of_memory, 1185
- L4::Pager, 1188
 - page_fault, 1190
- L4::Platform_control, 1192
 - cpu_allow_shutdown, 1194
 - cpu_disable, 1195
 - cpu_enable, 1195
 - system_shutdown, 1195
 - system_suspend, 1196
- L4::Poll_timeout_counter, 1196
 - Poll_timeout_counter, 1197
 - set, 1198
 - timed_out, 1198
- L4::Poll_timeout_kipclock, 1199
 - Poll_timeout_kipclock, 1200
 - set, 1200
 - test, 1201
 - timed_out, 1202
- L4::Proto_t< P >, 1202
- L4::Rcv_endpoint, 1203
 - bind_thread, 1206
- L4::Registry_iface, 1207
 - register_irq_obj, 1209
 - register_obj, 1209, 1210
 - unregister_obj, 1211
- L4::Runtime_error, 1211
 - err_no, 1215
 - extra_str, 1215
 - Runtime_error, 1214
- L4::Scheduler, 1215
- idle_time, 1219
- info, 1220
- is_online, 1221
- run_thread, 1221
- L4::Semaphore, 1222
 - down, 1226
 - up, 1227
- L4::Server< LOOP_HOOKS >, 1228
 - internal_loop, 1231
 - loop, 1232
 - Server, 1231
- L4::Server_object, 1233
 - dispatch, 1235, 1236
- L4::Server_object_t< IFACE, BASE >, 1237
 - dispatch_meta_request, 1241
 - get_buffer_demand, 1241
 - proto_dispatch, 1241
- L4::Server_object_x< Derived, IFACE, BASE >, 1243
- L4::Smart_cap< T, SMART >, 1246
 - Smart_cap, 1250
- L4::String, 1250
- L4::Task, 1251
 - add_ku_mem, 1255
 - cap_equal, 1256
 - cap_valid, 1256
 - delete_obj, 1257
 - map, 1258
 - release_cap, 1259
 - unmap, 1260
 - unmap_batch, 1261
- L4::Thread, 1262
 - control, 1266
 - ex_regs, 1267, 1268
 - modify_senders, 1269
 - register_del_irq, 1270
 - stats_time, 1271
 - switch_to, 1271
 - vcpu_control, 1272
 - vcpu_control_ext, 1273
 - vcpu_resume_commit, 1274
 - vcpu_resume_start, 1275
- L4::Thread::Attr, 1276
 - alien, 1277
 - Attr, 1277
 - bind, 1278
 - exc_handler, 1278, 1279
 - pager, 1280
 - ux_host_syscall, 1281
- L4::Thread::Modify_senders, 1281
 - add, 1282
- L4::Triggerable, 1283
 - trigger, 1286
- L4::Type_info, 1287
- L4::Type_info::Demand, 1288
 - Demand, 1291
 - no_demand, 1291
- L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, 1292

- Caps, [1294](#)
- Flags, [1294](#)
- Mem, [1294](#)
- Ports, [1294](#)
- L4::Type_info::Demand_union_t< D1, D2 >, [1294](#)
- L4::Typeid, [681](#)
- L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y >, [1299](#)
- L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >, [1299](#)
- L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y >, [1301](#)
- L4::Typeid::Detail::_Rpc< OPCODE, O, X >, [1297](#)
- L4::Typeid::Detail::Rpc_end, [1301](#)
- L4::Typeid::P_dispatch< LIST >, [1302](#)
- L4::Typeid::Raw_ipc< CLASS >, [1303](#)
- L4::Typeid::Rpc_nocode< OPERATION >, [1304](#)
- L4::Typeid::Rpc< RPCS >, [1306](#)
- L4::Typeid::Rpc_code< OPCODE_TYPE >, [1308](#)
- L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >, [1309](#)
- L4::Typeid::Rpc_sys< ARG >, [1311](#)
- L4::Types, [682](#)
- L4::Types::Bool< V >, [1313](#)
- L4::Types::False, [1314](#)
- L4::Types::Flags< BITS_ENUM, UNDERLYING >, [1316](#)
 - clear, [1319](#)
 - Flags, [1318](#)
 - from_raw, [1319](#)
 - None, [1318](#)
 - None_type, [1318](#)
- L4::Types::Flags_ops_t< DT >, [1320](#)
- L4::Types::Flags_t< DT, T >, [1322](#)
- L4::Types::Int_for_size< SIZE, bool >, [1324](#)
- L4::Types::Int_for_type< T >, [1325](#)
- L4::Types::Same< A, B >, [1326](#)
- L4::Types::True, [1328](#)
- L4::Uart, [1330](#)
 - change_mode, [1331](#)
 - char_avail, [1331](#)
 - enable_rx_irq, [1332](#)
 - generic_write, [1332](#)
 - get_char, [1333](#)
 - mode, [1333](#)
 - rate, [1333](#)
 - shutdown, [1333](#)
 - startup, [1334](#)
 - write, [1334](#)
- L4::Unknown_error, [1335](#)
- L4::Vcon, [1337](#)
 - get_attr, [1341](#)
 - read, [1342](#)
 - read_with_flags, [1343](#)
 - send, [1343](#)
 - set_attr, [1344](#)
 - write, [1345](#)
- L4::Vm, [1346](#)
 - vgicc_map, [1350](#)
- l4_addr_consts_t
 - Memory related, [324](#)
- l4_align_stack_for_direct_fncall
 - Basic Macros, [135](#)
- L4_AMD64_SEGMENT_FS
 - segment.h, [1975](#)
- L4_AMD64_SEGMENT_GS
 - segment.h, [1975](#)
- l4_arm_smccc_call
 - arm_smccc.h, [2537](#)
- l4_assert
 - assert.h, [2777](#)
- L4_BASE_ARM_SMCCC_CAP
 - Capabilities, [142](#)
- L4_BASE_CAPS_LAST
 - Capabilities, [142](#)
- L4_BASE_DEBUGGER_CAP
 - Capabilities, [142](#)
- L4_BASE_FACTORY_CAP
 - Capabilities, [142](#)
- L4_BASE_ICU_CAP
 - Capabilities, [142](#)
- L4_BASE_IOMMU_CAP
 - Capabilities, [142](#)
- L4_BASE_LOG_CAP
 - Capabilities, [142](#)
- L4_BASE_PAGER_CAP
 - Capabilities, [142](#)
- L4_BASE_SCHEDULER_CAP
 - Capabilities, [142](#)
- L4_BASE_TASK_CAP
 - Capabilities, [142](#)
- L4_BASE_THREAD_CAP
 - Capabilities, [142](#)
- L4_BDR_IO_SHIFT
 - Buffer Registers (BRs), [384](#)
- L4_BDR_MEM_SHIFT
 - Buffer Registers (BRs), [384](#)
- L4_BDR_OBJ_SHIFT
 - Buffer Registers (BRs), [384](#)
- l4_buf_regs_t, [1351](#)
- l4_buffer_desc_consts_t
 - Buffer Registers (BRs), [383](#)
- l4_busy_wait_ns
 - Timestamp Counter, [635](#)
- l4_busy_wait_us
 - Timestamp Counter, [636](#)
- l4_bytes_to_mwords
 - Memory related, [325](#)
- l4_cache_clean_data
 - Cache Consistency, [137](#)
- l4_cache_coherent
 - Cache Consistency, [138](#)
- l4_cache_dma_coherent
 - Cache Consistency, [138](#)
- l4_cache_flush_data
 - Cache Consistency, [139](#)

- `l4_cache_inv_data`
 - Cache Consistency, [139](#)
- `l4_calibrate_tsc`
 - Timestamp Counter, [636](#)
- `l4_cap_consts_t`
 - Capabilities, [141](#)
- `L4_CAP_FPAGE_D`
 - Flex pages, [163](#)
- `L4_CAP_FPAGE_R`
 - Flex pages, [163](#)
- `L4_cap_fpage_rights`
 - Flex pages, [162](#)
- `L4_CAP_FPAGE_RO`
 - Flex pages, [163](#)
- `L4_CAP_FPAGE_RS`
 - Flex pages, [163](#)
- `L4_CAP_FPAGE_RSD`
 - Flex pages, [163](#)
- `L4_CAP_FPAGE_RW`
 - Flex pages, [163](#)
- `L4_CAP_FPAGE_RWD`
 - Flex pages, [163](#)
- `L4_CAP_FPAGE_RWS`
 - Flex pages, [163](#)
- `L4_CAP_FPAGE_RWSD`
 - Flex pages, [163](#)
- `L4_CAP_FPAGE_S`
 - Flex pages, [162](#)
- `L4_CAP_FPAGE_W`
 - Flex pages, [162](#)
- `l4_cap_idx_t`
 - Capabilities, [141](#)
- `L4_CAP_MASK`
 - Capabilities, [141](#)
- `L4_CAP_OFFSET`
 - Capabilities, [141](#)
- `L4_CAP_SHIFT`
 - Capabilities, [141](#)
- `L4_CAP_SIZE`
 - Capabilities, [141](#)
- `l4_capability_equal`
 - Capabilities, [142](#)
- `l4_capability_next`
 - types.h, [2284](#)
- `l4_debugger_get_object_name`
 - Kernel Debugger, [151](#)
- `l4_debugger_global_id`
 - Kernel Debugger, [151](#)
- `l4_debugger_kobj_to_id`
 - Kernel Debugger, [152](#)
- `l4_debugger_query_log_name`
 - Kernel Debugger, [153](#)
- `l4_debugger_query_log_typeid`
 - Kernel Debugger, [154](#)
- `l4_debugger_set_object_name`
 - Kernel Debugger, [154](#)
- `l4_debugger_switch_log`
 - Kernel Debugger, [155](#)
- `l4_default_caps_t`
 - Capabilities, [142](#)
- `L4_DISABLE_COPY`
 - capability, [2549](#)
- `L4_E2BIG`
 - Error codes, [145](#)
- `L4_EACCESS`
 - Error codes, [145](#)
- `L4_EADDRNOTAVAIL`
 - Error codes, [145](#)
- `L4_EAGAIN`
 - Error codes, [145](#)
- `L4_EBADPROTO`
 - Error codes, [145](#)
- `L4_EBUSY`
 - Error codes, [145](#)
- `L4_EEXIST`
 - Error codes, [145](#)
- `L4_EFAULT`
 - Error codes, [145](#)
- `L4_EINVAL`
 - Error codes, [145](#)
- `L4_EIO`
 - Error codes, [145](#)
- `L4_EIPC_HI`
 - Error codes, [145](#)
- `L4_EIPC_LO`
 - Error codes, [145](#)
- `L4_MSGMISSARG`
 - Error codes, [145](#)
- `L4_MSGTOOLONG`
 - Error codes, [145](#)
- `L4_MSGTOOSHORT`
 - Error codes, [145](#)
- `L4_ENAMETOOLONG`
 - Error codes, [145](#)
- `L4_ENODEV`
 - Error codes, [145](#)
- `L4_ENOENT`
 - Error codes, [145](#)
- `L4_ENOMEM`
 - Error codes, [145](#)
- `L4_ENOREPLY`
 - Error codes, [145](#)
- `L4_ENOSPC`
 - Error codes, [145](#)
- `L4_ENOSYS`
 - Error codes, [145](#)
- `L4_ENXIO`
 - Error codes, [145](#)
- `L4_EOK`
 - Error codes, [145](#)
- `L4_EPERM`
 - Error codes, [145](#)
- `L4_ERANGE`
 - Error codes, [145](#)
- `L4_ERRNOMAX`
 - Error codes, [145](#)

- l4_error
 - Error Handling, [349](#)
- l4_error_code_t
 - Error codes, [145](#)
- l4_exc_regs_t, [1352](#)
 - flags, [1354](#)
 - ss, [1354](#)
- L4_EXPORT
 - Basic Macros, [134](#)
- l4_factory_create
 - Factory, [193](#)
- l4_factory_create_factory
 - Factory, [194](#)
- l4_factory_create_gate
 - Factory, [195](#)
- l4_factory_create_irq
 - Factory, [196](#)
- l4_factory_create_task
 - Factory, [197](#)
- l4_factory_create_thread
 - Factory, [198](#)
- l4_factory_create_vm
 - Factory, [199](#)
- L4_FP_ALL_SPACES
 - Task, [250](#)
- L4_FP_DELETE_OBJ
 - Task, [250](#)
- L4_FP_OTHER_SPACES
 - Task, [250](#)
- l4_fpage
 - Flex pages, [166](#)
- L4_FPAGE_ADDR_BITS
 - Flex pages, [164](#)
- L4_FPAGE_ADDR_SHIFT
 - Flex pages, [164](#)
- l4_fpage_all
 - Flex pages, [166](#)
- L4_FPAGE_BUFFERABLE
 - Flex pages, [164](#)
- L4_FPAGE_C_IPCGATE_SVR
 - Flex pages, [166](#)
- L4_FPAGE_C_NO_REF_CNT
 - Flex pages, [166](#)
- L4_FPAGE_C_OBJ_RIGHT1
 - Flex pages, [166](#)
- L4_FPAGE_C_OBJ_RIGHT2
 - Flex pages, [166](#)
- L4_FPAGE_C_OBJ_RIGHT3
 - Flex pages, [166](#)
- L4_FPAGE_C_OBJ_RIGHTS
 - Flex pages, [166](#)
- L4_FPAGE_C_REF_CNT
 - Flex pages, [165](#)
- L4_FPAGE_CACHE_OPT
 - Flex pages, [164](#)
- l4_fpage_cacheability_opt_t
 - Flex pages, [163](#)
- L4_FPAGE_CACHEABLE
 - Flex pages, [164](#)
- L4_fpage_consts
 - Flex pages, [164](#)
- l4_fpage_contains
 - Flex pages, [167](#)
- L4_fpage_control
 - Flex pages, [164](#)
- L4_FPAGE_CONTROL_MASK
 - Flex pages, [164](#)
- L4_FPAGE_CONTROL_OFFSET_SHIFT
 - Flex pages, [164](#)
- l4_fpage_invalid
 - Flex pages, [167](#)
- L4_FPAGE_IO
 - Flex pages, [165](#)
- l4_fpage_ioport
 - Flex pages, [168](#)
- l4_fpage_max_order
 - Flex pages, [168](#)
- l4_fpage_memaddr
 - Flex pages, [169](#)
- L4_FPAGE_MEMORY
 - Flex pages, [165](#)
- L4_FPAGE_OBJ
 - Flex pages, [165](#)
- l4_fpage_obj
 - Flex pages, [170](#)
- l4_fpage_page
 - Flex pages, [170](#)
- L4_fpage_rights
 - Flex pages, [164](#)
- l4_fpage_rights
 - Flex pages, [171](#)
- L4_FPAGE_RIGHTS_ALL
 - Flex pages, [164](#)
- L4_FPAGE_RIGHTS_BITS
 - Flex pages, [164](#)
- L4_FPAGE_RIGHTS_MASK
 - Flex pages, [164](#)
- L4_FPAGE_RIGHTS_SHIFT
 - Flex pages, [164](#)
- L4_FPAGE_RO
 - Flex pages, [165](#)
- L4_FPAGE_RW
 - Flex pages, [165](#)
- L4_FPAGE_RWX
 - Flex pages, [165](#)
- L4_FPAGE_RX
 - Flex pages, [165](#)
- l4_fpage_set_rights
 - Flex pages, [171](#)
- l4_fpage_size
 - Flex pages, [172](#)
- L4_FPAGE_SIZE_BITS
 - Flex pages, [164](#)
- L4_FPAGE_SIZE_SHIFT
 - Flex pages, [164](#)
- L4_FPAGE_SPECIAL

- Flex pages, [165](#)
- `l4_fpage_t`, [1355](#)
- `L4_fpage_type`
 - Flex pages, [165](#)
- `l4_fpage_type`
 - Flex pages, [173](#)
- `L4_FPAGE_TYPE_BITS`
 - Flex pages, [164](#)
- `L4_FPAGE_TYPE_SHIFT`
 - Flex pages, [164](#)
- `L4_FPAGE_UNCACHEABLE`
 - Flex pages, [164](#)
- `L4_FPAGE_W`
 - Flex pages, [165](#)
- `L4_FPAGE_X`
 - Flex pages, [165](#)
- `l4_get_hz`
 - Timestamp Counter, [637](#)
- `L4_HIDDEN`
 - Basic Macros, [134](#)
- `l4_icu_bind`
 - Interrupt controller, [219](#)
- `l4_icu_bind_u`
 - Interrupt controller, [220](#)
- `L4_ICU_FLAG_MSI`
 - Interrupt controller, [219](#)
- `L4_icu_flags`
 - Interrupt controller, [219](#)
- `l4_icu_info`
 - Interrupt controller, [222](#)
- `l4_icu_info_t`, [1356](#)
 - features, [1357](#)
 - Interrupt controller, [219](#)
- `l4_icu_info_u`
 - Interrupt controller, [222](#)
- `l4_icu_mask`
 - Interrupt controller, [223](#)
- `l4_icu_mask_u`
 - Interrupt controller, [224](#)
- `l4_icu_msi_info`
 - Interrupt controller, [225](#)
- `l4_icu_msi_info_t`, [1357](#)
- `l4_icu_msi_info_u`
 - Interrupt controller, [226](#)
- `L4_ICU_OP_BIND`
 - L4 IPC Opcodes, [411](#)
- `L4_ICU_OP_INFO`
 - L4 IPC Opcodes, [412](#)
- `L4_ICU_OP_MASK`
 - L4 IPC Opcodes, [412](#)
- `L4_ICU_OP_MSI_INFO`
 - L4 IPC Opcodes, [412](#)
- `L4_ICU_OP_SET_MODE`
 - L4 IPC Opcodes, [412](#)
- `L4_ICU_OP_UNBIND`
 - L4 IPC Opcodes, [411](#)
- `L4_ICU_OP_UNMASK`
 - L4 IPC Opcodes, [412](#)
- `L4_icu_opcode`
 - L4 IPC Opcodes, [411](#)
- `l4_icu_set_mode`
 - Interrupt controller, [227](#)
- `l4_icu_set_mode_u`
 - Interrupt controller, [228](#)
- `l4_icu_unbind`
 - Interrupt controller, [229](#)
- `l4_icu_unbind_u`
 - Interrupt controller, [230](#)
- `l4_icu_unmask`
 - Interrupt controller, [231](#)
- `l4_icu_unmask_u`
 - Interrupt controller, [232](#)
- `l4_infinite_loop`
 - Basic Macros, [135](#)
- `L4_INLINE_RPC`
 - `ipc_iface`, [2584](#)
- `L4_INLINE_RPC_NF`
 - `ipc_iface`, [2585](#)
- `L4_INLINE_RPC_NF_OP`
 - `ipc_iface`, [2585](#)
- `L4_INLINE_RPC_OP`
 - `ipc_iface`, [2586](#)
- `L4_INVALID_ADDR`
 - Memory related, [325](#)
- `L4_INVALID_CAP`
 - Capabilities, [141](#)
- `l4_iofpage`
 - Flex pages, [173](#)
- `L4_IOPORT_MAX`
 - Flex pages, [162](#)
- `l4_ipc`
 - Object Invocation, [332](#)
- `l4_ipc_call`
 - Object Invocation, [333](#)
- `L4_IPC_ENOT_EXISTENT`
 - Error Handling, [348](#)
- `l4_ipc_error`
 - Error Handling, [350](#)
- `l4_ipc_error_code`
 - Error Handling, [351](#)
- `L4_IPC_ERROR_MASK`
 - Error Handling, [348](#)
- `L4_IPC_GATE_BIND_OP`
 - L4 IPC Opcodes, [412](#)
- `L4_IPC_GATE_GET_INFO_OP`
 - L4 IPC Opcodes, [412](#)
- `l4_ipc_gate_get_infos`
 - IPC-Gate API, [201](#)
- `l4_ipc_gate_ops`
 - L4 IPC Opcodes, [412](#)
- `l4_ipc_is_rcv_error`
 - Error Handling, [351](#)
- `l4_ipc_is_snd_error`
 - Error Handling, [352](#)
- `L4_IPC_REABORTED`
 - Error Handling, [348](#)

- L4_IPC_RECANCELED
 - Error Handling, [348](#)
- l4_ipc_receive
 - Object Invocation, [335](#)
- L4_IPC_REMAPFAILED
 - Error Handling, [348](#)
- L4_IPC_REMSGCUT
 - Error Handling, [348](#)
- l4_ipc_reply_and_wait
 - Object Invocation, [338](#)
- L4_IPC_RERCVPFTO
 - Error Handling, [348](#)
- L4_IPC_RESNDPFTO
 - Error Handling, [348](#)
- L4_IPC_RETIMEOUT
 - Error Handling, [348](#)
- L4_IPC_SEABORTED
 - Error Handling, [348](#)
- L4_IPC_SECANCELED
 - Error Handling, [348](#)
- L4_IPC_SEMAPFAILED
 - Error Handling, [348](#)
- L4_IPC_SEMSGCUT
 - Error Handling, [348](#)
- l4_ipc_send
 - Object Invocation, [339](#)
- l4_ipc_send_and_wait
 - Object Invocation, [341](#)
- L4_IPC_SERCVPFTO
 - Error Handling, [348](#)
- L4_IPC_SESNDFPFTO
 - Error Handling, [348](#)
- L4_IPC_SETIMEOUT
 - Error Handling, [348](#)
- l4_ipc_sleep
 - Object Invocation, [342](#)
- l4_ipc_sleep_ms
 - Object Invocation, [343](#)
- l4_ipc_sleep_us
 - Object Invocation, [344](#)
- L4_IPC_SND_ERR_MASK
 - Error Handling, [348](#)
- l4_ipc_tcr_error_t
 - Error Handling, [348](#)
- l4_ipc_timeout
 - Timeouts, [372](#)
- L4_IPC_TIMEOUT_0
 - Timeouts, [372](#)
- l4_ipc_to_errno
 - ipc.h, [2659](#)
- l4_ipc_wait
 - Object Invocation, [345](#)
- l4_irq_detach
 - IRQs, [205](#)
- l4_irq_detach_u
 - IRQs, [206](#)
- L4_IRQ_F_BOTH
 - IRQs, [205](#)
- L4_IRQ_F_BOTH_EDGE
 - IRQs, [205](#)
- L4_IRQ_F_CLEAR_WAKEUP
 - IRQs, [205](#)
- L4_IRQ_F_EDGE
 - IRQs, [205](#)
- L4_IRQ_F_LEVEL
 - IRQs, [205](#)
- L4_IRQ_F_LEVEL_HIGH
 - IRQs, [205](#)
- L4_IRQ_F_LEVEL_LOW
 - IRQs, [205](#)
- L4_IRQ_F_MASK
 - IRQs, [205](#)
- L4_IRQ_F_NEG
 - IRQs, [205](#)
- L4_IRQ_F_NEG_EDGE
 - IRQs, [205](#)
- L4_IRQ_F_NONE
 - IRQs, [205](#)
- L4_IRQ_F_POS
 - IRQs, [205](#)
- L4_IRQ_F_POS_EDGE
 - IRQs, [205](#)
- L4_IRQ_F_SET_WAKEUP
 - IRQs, [205](#)
- L4_irq_mode
 - IRQs, [204](#)
- l4_irq_mux_chain
 - IRQs, [207](#)
- l4_irq_mux_chain_u
 - IRQs, [208](#)
- l4_irq_receive
 - IRQs, [209](#)
- l4_irq_receive_u
 - IRQs, [210](#)
- l4_irq_trigger
 - IRQs, [211](#)
- l4_irq_trigger_u
 - IRQs, [212](#)
- l4_irq_unmask
 - IRQs, [213](#)
- l4_irq_unmask_u
 - IRQs, [214](#)
- l4_irq_wait
 - IRQs, [215](#)
- l4_irq_wait_u
 - IRQs, [216](#)
- l4_is_fpage_writable
 - Flex pages, [174](#)
- l4_is_invalid_cap
 - Capabilities, [143](#)
- l4_is_valid_cap
 - Capabilities, [143](#)
- L4_ITEM_CONT
 - Message Items, [354](#)
- L4_ITEM_MAP
 - Message Items, [353](#)

- `l4_kdebug_ops_t`
 - `kdebug.h`, 2675
- `l4_kernel_info_get_mem_desc_end`
 - Memory descriptors (C version), 187
- `l4_kernel_info_get_mem_desc_is_virtual`
 - Memory descriptors (C version), 188
- `l4_kernel_info_get_mem_desc_start`
 - Memory descriptors (C version), 188
- `l4_kernel_info_get_mem_desc_subtype`
 - Memory descriptors (C version), 188
- `l4_kernel_info_get_mem_desc_type`
 - Memory descriptors (C version), 188
- `l4_kernel_info_get_num_mem_descs`
 - Memory descriptors (C version), 189
- `l4_kernel_info_mem_desc_t`, 1358
 - Memory descriptors (C version), 186
- `l4_kernel_info_set_mem_desc`
 - Memory descriptors (C version), 189
- `l4_kernel_info_t`, 1359
- `l4_kernel_info_version_offset`
 - Kernel Interface Page, 179
- `l4_kip`
 - Kernel Interface Page, 180
- `l4_kip_clock`
 - Kernel Interface Page, 180
- `l4_kip_clock_lw`
 - Kernel Interface Page, 181
- `l4_kip_clock_ns`
 - Kernel Interface Page, 182
- `l4_kip_for_each_feature`
 - `kip.h`, 2820
- `l4_kip_kernel_has_feature`
 - `kip.h`, 2820
- `L4_KIP_OFFS_READ_NS`
 - Kernel Interface Page, 179
- `L4_KIP_OFFS_READ_US`
 - Kernel Interface Page, 179
- `l4_kip_version`
 - Kernel Interface Page, 182
- `l4_kip_version_string`
 - Kernel Interface Page, 183
- `L4_LOG2_PAGESIZE`
 - Memory related, 322
- `L4_LOG2_SUPERPAGESIZE`
 - Memory related, 322
- `l4_map_control`
 - Message Items, 354
- `L4_MAP_ITEM_GRANT`
 - Message Items, 354
- `L4_MAP_ITEM_MAP`
 - Message Items, 354
- `l4_map_obj_control`
 - Message Items, 355
- `l4_mem_archspecific_acpi_nvs`
 - Memory descriptors (C version), 187
- `l4_mem_archspecific_acpi_tables`
 - Memory descriptors (C version), 187
- `l4_mem_archspecific_sub_type_common_t`
 - Memory descriptors (C version), 186
- `l4_mem_info_acpi_rsdp`
 - Memory descriptors (C version), 187
- `l4_mem_info_sub_type_t`
 - Memory descriptors (C version), 187
- `L4_mem_op_widths`
 - Memory operations., 319
- `l4_mem_read`
 - Memory operations., 320
- `l4_mem_type_archspecific`
 - Memory descriptors (C version), 187
- `l4_mem_type_bootloader`
 - Memory descriptors (C version), 187
- `l4_mem_type_conventional`
 - Memory descriptors (C version), 187
- `l4_mem_type_dedicated`
 - Memory descriptors (C version), 187
- `l4_mem_type_info`
 - Memory descriptors (C version), 187
- `l4_mem_type_reserved`
 - Memory descriptors (C version), 187
- `l4_mem_type_shared`
 - Memory descriptors (C version), 187
- `l4_mem_type_t`
 - Memory descriptors (C version), 187
- `l4_mem_type_undefined`
 - Memory descriptors (C version), 187
- `L4_MEM_WIDTH_1BYTE`
 - Memory operations., 319
- `L4_MEM_WIDTH_2BYTE`
 - Memory operations., 319
- `L4_MEM_WIDTH_4BYTE`
 - Memory operations., 319
- `l4_mem_write`
 - Memory operations., 320
- `l4_msg_item_consts_t`
 - Message Items, 353
- `l4_msg_regs_t`, 1361
- `l4_msgtag`
 - Message Tag, 360
- `L4_MSGTAG_ERROR`
 - Message Tag, 358
- `L4_MSGTAG_FLAGS`
 - Message Tag, 358
- `L4_msgtag_flags`
 - Message Tag, 358
- `l4_msgtag_flags`
 - Message Tag, 361
- `l4_msgtag_has_error`
 - Message Tag, 362
- `l4_msgtag_is_exception`
 - Message Tag, 362
- `l4_msgtag_is_io_page_fault`
 - Message Tag, 364
- `l4_msgtag_is_page_fault`
 - Message Tag, 365
- `l4_msgtag_is_preemption`
 - Message Tag, 365

- l4_msgtag_is_sigma0
 - Message Tag, [366](#)
- l4_msgtag_is_sys_exception
 - Message Tag, [367](#)
- l4_msgtag_items
 - Message Tag, [367](#)
- l4_msgtag_label
 - Message Tag, [368](#)
- L4_msgtag_protocol
 - Message Tag, [358](#)
- L4_MSGTAG_SCHEDULE
 - Message Tag, [358](#)
- l4_msgtag_t, [1362](#)
 - flags, [1363](#)
 - Message Tag, [358](#)
- L4_MSGTAG_TRANSFER_FPU
 - Message Tag, [358](#)
- l4_msgtag_words
 - Message Tag, [369](#)
- L4_NOTHROW
 - Basic Macros, [134](#)
- l4_ns_to_tsc
 - Timestamp Counter, [637](#)
- l4_obj_fpage
 - Flex pages, [174](#)
- L4_obj_fpage_ctl
 - Flex pages, [165](#)
- L4_PAGEMASK
 - Memory related, [323](#)
- L4_PAGESHIFT
 - Memory related, [323](#)
- l4_platform_ctl_cpu_allow_shutdown
 - Platform Control C API, [237](#)
- L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP
 - L4 IPC Opcodes, [413](#)
- l4_platform_ctl_cpu_disable
 - Platform Control C API, [238](#)
- L4_PLATFORM_CTL_CPU_DISABLE_OP
 - L4 IPC Opcodes, [413](#)
- l4_platform_ctl_cpu_enable
 - Platform Control C API, [238](#)
- L4_PLATFORM_CTL_CPU_ENABLE_OP
 - L4 IPC Opcodes, [413](#)
- L4_platform_ctl_ops
 - L4 IPC Opcodes, [412](#)
- L4_platform_ctl_proto
 - Message Tag, [359](#)
- L4_PLATFORM_CTL_SYS_SHUTDOWN_OP
 - L4 IPC Opcodes, [413](#)
- L4_PLATFORM_CTL_SYS_SUSPEND_OP
 - L4 IPC Opcodes, [413](#)
- l4_platform_ctl_system_shutdown
 - Platform Control C API, [239](#)
- l4_platform_ctl_system_suspend
 - Platform Control C API, [240](#)
- L4_PROTO_ALLOW_SYSCALL
 - Message Tag, [359](#)
- L4_PROTO_DEBUGGER
 - Message Tag, [359](#)
- L4_PROTO_DMA_SPACE
 - Message Tag, [359](#)
- L4_PROTO_EXCEPTION
 - Message Tag, [359](#)
- L4_PROTO_FACTORY
 - Message Tag, [359](#)
- L4_PROTO_IO_PAGE_FAULT
 - Message Tag, [359](#)
- L4_PROTO_IOMMU
 - Message Tag, [359](#)
- L4_PROTO_IRQ
 - Message Tag, [359](#)
- L4_PROTO_IRQ_MUX
 - Message Tag, [359](#)
- L4_PROTO_IRQ_SENDER
 - Message Tag, [359](#)
- L4_PROTO_KOBJECT
 - Message Tag, [359](#)
- L4_PROTO_LOG
 - Message Tag, [359](#)
- L4_PROTO_META
 - Message Tag, [359](#)
- L4_PROTO_NONE
 - Message Tag, [359](#)
- L4_PROTO_PAGE_FAULT
 - Message Tag, [359](#)
- L4_PROTO_PF_EXCEPTION
 - Message Tag, [359](#)
- L4_PROTO_PLATFORM_CTL
 - Message Tag, [359](#)
- L4_PROTO_PREEMPTION
 - Message Tag, [359](#)
- L4_PROTO_SCHEDULER
 - Message Tag, [359](#)
- L4_PROTO_SEMAPHORE
 - Message Tag, [359](#)
- L4_PROTO_SIGMA0
 - Message Tag, [359](#)
- L4_PROTO_SMCCC
 - Message Tag, [359](#)
- L4_PROTO_SYS_EXCEPTION
 - Message Tag, [359](#)
- L4_PROTO_TASK
 - Message Tag, [359](#)
- L4_PROTO_THREAD
 - Message Tag, [359](#)
- L4_PROTO_VM
 - Message Tag, [359](#)
- L4_RCV_EP_BIND_OP
 - rcv_endpoint.h, [2720](#)
- l4_rcv_ep_bind_thread
 - IPC-Gate API, [202](#)
- L4_rcv_ep_ops
 - rcv_endpoint.h, [2720](#)
- L4_RCV_ITEM_LOCAL_ID
 - Message Items, [354](#)
- L4_RCV_ITEM_SINGLE_CAP

- Message Items, [354](#)
- l4_rcv_timeout
 - Timeouts, [373](#)
- l4_rdpmc
 - Timestamp Counter, [638](#)
- l4_rdpmc_32
 - Timestamp Counter, [638](#)
- l4_rdtsc
 - Timestamp Counter, [638](#)
- l4_rdtsc_32
 - Timestamp Counter, [639](#)
- l4_round_page
 - Memory related, [325](#)
- l4_round_size
 - Memory related, [326](#)
- L4_RPC
 - ipc_iface, [2586](#)
- L4_RPC_DEF
 - ipc_client, [2577](#)
- L4_RPC_NF
 - ipc_iface, [2587](#)
- L4_RPC_NF_OP
 - ipc_iface, [2587](#)
- L4_RPC_OP
 - ipc_iface, [2587](#)
- l4_sched_cpu_set
 - Scheduler, [243](#)
- l4_sched_cpu_set_t, [1364](#)
 - gran_offset, [1367](#)
 - granularity, [1365](#)
 - offset, [1365](#)
 - set, [1365](#)
- l4_sched_param
 - Scheduler, [244](#)
- l4_sched_param_t, [1368](#)
- L4_SCHEDULER_CLASS_FIXED_PRIO
 - Scheduler, [243](#)
- L4_SCHEDULER_CLASS_WFQ
 - Scheduler, [243](#)
- L4_scheduler_classes
 - Scheduler, [242](#)
- l4_scheduler_idle_time
 - Scheduler, [244](#)
- L4_SCHEDULER_IDLE_TIME_OP
 - Scheduler, [243](#)
- l4_scheduler_info
 - Scheduler, [245](#)
- L4_SCHEDULER_INFO_OP
 - Scheduler, [243](#)
- l4_scheduler_info_with_classes
 - Scheduler, [246](#)
- l4_scheduler_is_online
 - Scheduler, [247](#)
- L4_scheduler_ops
 - Scheduler, [243](#)
- l4_scheduler_run_thread
 - Scheduler, [247](#)
- L4_SCHEDULER_RUN_THREAD_OP
 - Scheduler, [243](#)
- l4_semaphore_down
 - Kernel-provided semaphore, [233](#)
- l4_semaphore_up
 - Kernel-provided semaphore, [234](#)
- l4_sleep
 - util.h, [1968](#), [1972](#)
 - Utility Functions, [564](#)
- l4_snd_fpage_t, [1369](#)
- l4_snd_timeout
 - Timeouts, [373](#)
- l4_sndfpage_add
 - Object Invocation, [346](#)
- L4_SUPERPAGEMASK
 - Memory related, [323](#)
- L4_SUPERPAGESHIFT
 - Memory related, [324](#)
- L4_SUPERPAGESIZE
 - Memory related, [324](#)
- L4_sys_segment
 - segment.h, [1975](#)
- l4_syscall_flags_t
 - Object Invocation, [331](#)
- L4_SYSF_CALL
 - Object Invocation, [331](#)
- L4_SYSF_NONE
 - Object Invocation, [331](#)
- L4_SYSF_OPEN_WAIT
 - Object Invocation, [331](#)
- L4_SYSF_RECV
 - Object Invocation, [331](#)
- L4_SYSF_REPLY
 - Object Invocation, [331](#)
- L4_SYSF_REPLY_AND_WAIT
 - Object Invocation, [331](#)
- L4_SYSF_SEND
 - Object Invocation, [331](#)
- L4_SYSF_SEND_AND_WAIT
 - Object Invocation, [331](#)
- L4_SYSF_WAIT
 - Object Invocation, [331](#)
- l4_task_add_ku_mem
 - Task, [250](#)
- L4_TASK_ADD_KU_MEM_OP
 - L4 IPC Opcodes, [413](#)
- l4_task_cap_equal
 - Task, [251](#)
- L4_TASK_CAP_INFO_OP
 - L4 IPC Opcodes, [413](#)
- l4_task_cap_valid
 - Task, [252](#)
- l4_task_delete_obj
 - Task, [253](#)
- L4_TASK_LDT_SET_X86_OP
 - L4 IPC Opcodes, [413](#)
- l4_task_ldt_x86_consts
 - segment.h, [1975](#), [1985](#)
- L4_TASK_LDT_X86_ENTRY_SIZE

- segment.h, [1976](#), [1986](#)
- L4_TASK_LDT_X86_MAX_ENTRIES
 - segment.h, [1976](#), [1986](#)
- l4_task_map
 - Task, [253](#)
- L4_TASK_MAP_OP
 - L4 IPC Opcodes, [413](#)
- L4_TASK_MAP_VGICC_ARM_OP
 - L4 IPC Opcodes, [413](#)
- L4_task_ops
 - L4 IPC Opcodes, [413](#)
- l4_task_release_cap
 - Task, [255](#)
- l4_task_unmap
 - Task, [256](#)
- l4_task_unmap_batch
 - Task, [257](#)
- L4_TASK_UNMAP_OP
 - L4 IPC Opcodes, [413](#)
- l4_task_vgicc_map
 - Task, [258](#)
- L4_THREAD_AMD64_GET_SEGMENT_INFO_OP
 - L4 IPC Opcodes, [413](#)
- L4_THREAD_AMD64_SET_SEGMENT_BASE_OP
 - L4 IPC Opcodes, [413](#)
- l4_thread_arm_set_tpidruro
 - Thread, [262](#)
- L4_THREAD_ARM_TPIDRURO_OP
 - L4 IPC Opcodes, [413](#)
- L4_THREAD_CONTROL_ALIEN
 - Thread, [262](#)
- l4_thread_control_alien
 - Thread control, [280](#)
- l4_thread_control_bind
 - Thread control, [280](#)
- L4_THREAD_CONTROL_BIND_TASK
 - Thread, [262](#)
- l4_thread_control_commit
 - Thread control, [281](#)
- l4_thread_control_exc_handler
 - Thread control, [282](#)
- L4_thread_control_flags
 - Thread, [261](#)
- L4_THREAD_CONTROL_MR_IDX_BIND_TASK
 - Thread, [262](#)
- L4_THREAD_CONTROL_MR_IDX_BIND_UTCB
 - Thread, [262](#)
- L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER
 - Thread, [262](#)
- L4_THREAD_CONTROL_MR_IDX_FLAG_VALS
 - Thread, [262](#)
- L4_THREAD_CONTROL_MR_IDX_FLAGS
 - Thread, [262](#)
- L4_THREAD_CONTROL_MR_IDX_PAGER
 - Thread, [262](#)
- L4_thread_control_mr_indices
 - Thread, [262](#)
- L4_THREAD_CONTROL_OP
 - L4 IPC Opcodes, [413](#)
- l4_thread_control_pager
 - Thread control, [283](#)
- L4_THREAD_CONTROL_SET_EXC_HANDLER
 - Thread, [262](#)
- L4_THREAD_CONTROL_SET_PAGER
 - Thread, [262](#)
- l4_thread_control_start
 - Thread control, [283](#)
- l4_thread_control_ux_host_syscall
 - Thread control, [284](#)
- L4_THREAD_CONTROL_UX_NATIVE
 - Thread, [262](#)
- l4_thread_ex_regs
 - Thread, [263](#)
- L4_THREAD_EX_REGS_CANCEL
 - Thread, [262](#)
- L4_thread_ex_regs_flags
 - Thread, [262](#)
- L4_THREAD_EX_REGS_OP
 - L4 IPC Opcodes, [413](#)
- l4_thread_ex_regs_ret
 - Thread, [264](#)
- l4_thread_ex_regs_ret_u
 - Thread, [265](#)
- L4_THREAD_EX_REGS_TRIGGER_EXCEPTION
 - Thread, [262](#)
- l4_thread_ex_regs_u
 - Thread, [266](#)
- l4_thread_modify_sender_add
 - Thread, [267](#)
- l4_thread_modify_sender_commit
 - Thread, [268](#)
- L4_THREAD_MODIFY_SENDER_OP
 - L4 IPC Opcodes, [413](#)
- l4_thread_modify_sender_start
 - Thread, [269](#)
- L4_THREAD_OPCODE_MASK
 - L4 IPC Opcodes, [413](#)
- L4_thread_ops
 - L4 IPC Opcodes, [413](#)
- l4_thread_register_del_irq
 - Thread, [270](#)
- L4_THREAD_REGISTER_DELETE_IRQ_OP
 - L4 IPC Opcodes, [413](#)
- l4_thread_regs_t, [1370](#)
- L4_THREAD_STATS_OP
 - L4 IPC Opcodes, [413](#)
- l4_thread_stats_time
 - Thread, [271](#)
- l4_thread_switch
 - Thread, [271](#)
- L4_THREAD_SWITCH_OP
 - L4 IPC Opcodes, [413](#)
- l4_thread_vcpu_control
 - Thread, [272](#)
- l4_thread_vcpu_control_ext
 - Thread, [273](#)

- `l4_thread_vcpu_control_ext_u`
 - Thread, [274](#)
- `L4_THREAD_VCPU_CONTROL_OP`
 - L4 IPC Opcodes, [413](#)
- `l4_thread_vcpu_control_u`
 - Thread, [275](#)
- `l4_thread_vcpu_resume_commit`
 - Thread, [276](#)
- `L4_THREAD_VCPU_RESUME_OP`
 - L4 IPC Opcodes, [413](#)
- `l4_thread_vcpu_resume_start`
 - Thread, [277](#)
- `L4_THREAD_X86_GDT_OP`
 - L4 IPC Opcodes, [413](#)
- `l4_thread_yield`
 - Thread, [278](#)
- `l4_timeout`
 - Timeouts, [373](#)
- `l4_timeout_abs`
 - Timeouts, [374](#)
- `l4_timeout_get`
 - Timeouts, [375](#)
- `l4_timeout_is_absolute`
 - Timeouts, [376](#)
- `l4_timeout_rel`
 - Timeouts, [376](#)
- `l4_timeout_rel_get`
 - Timeouts, [377](#)
- `l4_timeout_s`, [1371](#)
 - Timeouts, [372](#)
- `l4_timeout_t`, [1372](#)
 - Timeouts, [372](#)
- `l4_touch_ro`
 - Utility Functions, [565](#)
- `l4_touch_rw`
 - Utility Functions, [565](#)
- `l4_trunc_page`
 - Memory related, [327](#)
- `l4_trunc_size`
 - Memory related, [328](#)
- `l4_tsc_init`
 - Timestamp Counter, [639](#)
- `l4_tsc_to_ns`
 - Timestamp Counter, [640](#)
- `l4_tsc_to_s_and_ns`
 - Timestamp Counter, [640](#)
- `l4_tsc_to_us`
 - Timestamp Counter, [641](#)
- `L4_TYPE_VHW_FRAMEBUFFER`
 - Fiasco-UX Virtual devices, [185](#)
- `L4_TYPE_VHW_INPUT`
 - Fiasco-UX Virtual devices, [185](#)
- `L4_TYPE_VHW_NET`
 - Fiasco-UX Virtual devices, [185](#)
- `L4_TYPE_VHW_NONE`
 - Fiasco-UX Virtual devices, [185](#)
- `L4_TYPES_FLAGS_OPS_DEF`
 - types, [2624](#)
- `l4_unmap_flags_t`
 - Task, [250](#)
- `l4_usleep`
 - Utility Functions, [566](#)
- `l4_utcb_br`
 - Virtual Registers (UTCBs), [380](#)
- `L4_UTCB_BUF_REGS_OFFSET`
 - x86 Virtual Registers (UTCB), [390](#)
- `L4_utcb_consts_x86`
 - x86 Virtual Registers (UTCB), [390](#)
- `l4_utcb_exc`
 - Exception registers, [385](#)
- `l4_utcb_exc_is_ex_regs_exception`
 - Exception registers, [385](#)
- `l4_utcb_exc_is_pf`
 - Exception registers, [387](#)
- `l4_utcb_exc_pc`
 - Exception registers, [387](#)
- `l4_utcb_exc_pc_set`
 - Exception registers, [388](#)
- `L4_UTCB_EXCEPTION_REGS_SIZE`
 - x86 Virtual Registers (UTCB), [390](#)
- `L4_UTCB_GENERIC_BUFFERS_SIZE`
 - x86 Virtual Registers (UTCB), [390](#)
- `L4_UTCB_GENERIC_DATA_SIZE`
 - x86 Virtual Registers (UTCB), [390](#)
- `L4_UTCB_INHERIT_FPU`
 - x86 Virtual Registers (UTCB), [390](#)
- `l4_utcb_mr`
 - Virtual Registers (UTCBs), [380](#)
- `l4_utcb_mr64_idx`
 - Timeouts, [377](#)
- `L4_UTCB_MSG_REGS_OFFSET`
 - x86 Virtual Registers (UTCB), [390](#)
- `L4_UTCB_OFFSET`
 - x86 Virtual Registers (UTCB), [390](#)
- `l4_utcb_t`
 - Virtual Registers (UTCBs), [380](#)
- `l4_utcb_tcr`
 - Virtual Registers (UTCBs), [381](#)
- `L4_UTCB_THREAD_REGS_OFFSET`
 - x86 Virtual Registers (UTCB), [390](#)
- `l4_vcon_attr_t`, [1373](#)
 - set_raw, [1374](#)
 - Virtual Console, [291](#)
- `L4_VCON_ECHO`
 - Virtual Console, [291](#)
- `l4_vcon_get_attr`
 - Virtual Console, [292](#)
- `L4_VCON_GET_ATTR_OP`
 - L4 IPC Opcodes, [414](#)
- `l4_vcon_get_attr_u`
 - Virtual Console, [293](#)
- `L4_vcon_i_flags`
 - Virtual Console, [291](#)
- `L4_VCON_ICANON`
 - Virtual Console, [291](#)
- `L4_VCON_ICRNL`

- Virtual Console, [291](#)
- L4_VCON_IGNCR
 - Virtual Console, [291](#)
- L4_VCON_INLCR
 - Virtual Console, [291](#)
- L4_vcon_l_flags
 - Virtual Console, [291](#)
- L4_vcon_o_flags
 - Virtual Console, [291](#)
- L4_VCON_OCRNL
 - Virtual Console, [292](#)
- L4_VCON_ONLCR
 - Virtual Console, [292](#)
- L4_VCON_ONLRET
 - Virtual Console, [292](#)
- L4_vcon_ops
 - L4 IPC Opcodes, [414](#)
- l4_vcon_read
 - Virtual Console, [294](#)
- L4_vcon_read_flags
 - vcon.h, [2767](#)
- L4_VCON_READ_OP
 - L4 IPC Opcodes, [414](#)
- L4_VCON_READ_SIZE
 - Virtual Console, [292](#)
- L4_VCON_READ_SIZE_MASK
 - vcon.h, [2767](#)
- L4_VCON_READ_STAT_BREAK
 - vcon.h, [2767](#)
- L4_VCON_READ_STAT_DONE
 - vcon.h, [2767](#)
- l4_vcon_read_u
 - Virtual Console, [295](#)
- l4_vcon_read_with_flags
 - Virtual Console, [296](#)
- l4_vcon_send
 - Virtual Console, [297](#)
- l4_vcon_send_u
 - Virtual Console, [298](#)
- l4_vcon_set_attr
 - Virtual Console, [299](#)
- L4_VCON_SET_ATTR_OP
 - L4 IPC Opcodes, [414](#)
- l4_vcon_set_attr_raw
 - Virtual Console, [300](#)
- l4_vcon_set_attr_u
 - Virtual Console, [300](#)
- L4_vcon_size_consts
 - Virtual Console, [292](#)
- l4_vcon_write
 - Virtual Console, [301](#)
- L4_VCON_WRITE_OP
 - L4 IPC Opcodes, [414](#)
- L4_VCON_WRITE_SIZE
 - Virtual Console, [292](#)
- l4_vcon_write_u
 - Virtual Console, [302](#)
- l4_vcpu_arch_state_t, [1374](#)
- l4_vcpu_check_version
 - vcpu.h, [2880](#)
- L4_vcpu_e_field_ids
 - __vcpu-arch.h, [2002](#)
- L4_VCPU_F_EXCEPTIONS
 - vCPU API, [288](#)
- L4_VCPU_F_FPU_ENABLED
 - vCPU API, [288](#)
- L4_VCPU_F_IRQ
 - vCPU API, [287](#)
- L4_VCPU_F_PAGE_FAULTS
 - vCPU API, [287](#)
- L4_VCPU_F_USER_MODE
 - vCPU API, [288](#)
- l4_vcpu_ipc_regs_t, [1375](#)
- L4_VCPU_OFFSET_EXT_INFOS
 - vCPU API, [288](#)
- L4_VCPU_OFFSET_EXT_STATE
 - vCPU API, [288](#)
- l4_vcpu_regs_t, [1376](#)
 - ax, [1378](#)
 - bp, [1378](#)
 - bx, [1378](#)
 - cx, [1378](#)
 - di, [1378](#)
 - dx, [1379](#)
 - si, [1379](#)
- L4_VCPU_SF_IRQ_PENDING
 - vCPU API, [288](#)
- L4_vcpu_state_flags
 - vCPU API, [287](#)
- L4_vcpu_state_offset
 - vCPU API, [288](#)
- l4_vcpu_state_t, [1380](#)
 - version, [1382](#)
- L4_VCPU_STATE_VERSION
 - __vcpu-arch.h, [2000](#), [2002](#), [2005](#)
- L4_vcpu_sticky_flags
 - vCPU API, [288](#)
- l4_vhw_descriptor, [1383](#)
- l4_vhw_entry, [1384](#)
- l4_vhw_entry_type
 - Fiasco-UX Virtual devices, [184](#)
- L4_virtio_cmd
 - L4 VIRTIO Transport Layer, [420](#)
- L4_virtio_irq_status
 - L4 VIRTIO Transport Layer, [420](#)
- L4_virtio_opcodes
 - L4 VIRTIO Transport Layer, [421](#)
- l4_vm_svm_vmcb_control_area, [1385](#)
- l4_vm_svm_vmcb_state_save_area, [1386](#)
- l4_vm_svm_vmcb_state_save_area_seg, [1387](#)
- l4_vm_svm_vmcb_t, [1387](#)
- l4_vm_tz_state, [1388](#)
- L4_VM_VMX_BASIC_REG
 - VM API for VMX, [308](#)
- L4_vm_vmx_caps_regs
 - VM API for VMX, [307](#)

- `l4_vm_vmx_clear`
VM API for VMX, [308](#)
- `L4_VM_VMX_CR0_FIXED0_REG`
VM API for VMX, [308](#)
- `L4_VM_VMX_CR0_FIXED1_REG`
VM API for VMX, [308](#)
- `L4_VM_VMX_CR4_FIXED0_REG`
VM API for VMX, [308](#)
- `L4_VM_VMX_CR4_FIXED1_REG`
VM API for VMX, [308](#)
- `L4_vm_vmx_dfl1_regs`
VM API for VMX, [308](#)
- `L4_VM_VMX_ENTRY_CTL5_DFL1_REG`
VM API for VMX, [308](#)
- `L4_VM_VMX_EPT_VPID_CAP_REG`
VM API for VMX, [308](#)
- `L4_VM_VMX_EXIT_CTL5_DFL1_REG`
VM API for VMX, [308](#)
- `l4_vm_vmx_field_len`
VM API for VMX, [309](#)
- `l4_vm_vmx_field_order`
VM API for VMX, [309](#)
- `l4_vm_vmx_get_caps`
VM API for VMX, [310](#)
- `l4_vm_vmx_get_caps_default1`
VM API for VMX, [310](#)
- `l4_vm_vmx_get_cr2_index`
VM API for VMX, [311](#)
- `L4_VM_VMX_MISC_REG`
VM API for VMX, [308](#)
- `L4_VM_VMX_NUM_CAPS_REGS`
VM API for VMX, [308](#)
- `L4_VM_VMX_NUM_DFL1_REGS`
VM API for VMX, [308](#)
- `L4_VM_VMX_PINBASED_CTL5_DFL1_REG`
VM API for VMX, [308](#)
- `L4_VM_VMX_PROCBASED_CTL5_2_REG`
VM API for VMX, [308](#)
- `L4_VM_VMX_PROCBASED_CTL5_DFL1_REG`
VM API for VMX, [308](#)
- `l4_vm_vmx_ptr_load`
VM API for VMX, [311](#)
- `l4_vm_vmx_read`
VM API for VMX, [312](#)
- `l4_vm_vmx_read_16`
VM API for VMX, [313](#)
- `l4_vm_vmx_read_32`
VM API for VMX, [313](#)
- `l4_vm_vmx_read_64`
VM API for VMX, [314](#)
- `l4_vm_vmx_read_nat`
VM API for VMX, [315](#)
- `L4_VM_VMX_TRUE_ENTRY_CTL5_REG`
VM API for VMX, [308](#)
- `L4_VM_VMX_TRUE_EXIT_CTL5_REG`
VM API for VMX, [308](#)
- `L4_VM_VMX_TRUE_PINBASED_CTL5_REG`
VM API for VMX, [308](#)
- `L4_VM_VMX_TRUE_PROCBASED_CTL5_REG`
VM API for VMX, [308](#)
- `L4_VM_VMX_VMCS_CR2`
VM API for VMX, [307](#)
- `L4_VM_VMX_VMCS_ENUM_REG`
VM API for VMX, [308](#)
- `L4_VM_VMX_VMCS_MSR_CSTAR`
VM API for VMX, [307](#)
- `L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE`
VM API for VMX, [307](#)
- `L4_VM_VMX_VMCS_MSR_LSTAR`
VM API for VMX, [307](#)
- `L4_VM_VMX_VMCS_MSR_STAR`
VM API for VMX, [307](#)
- `L4_VM_VMX_VMCS_MSR_SYSCALL_MASK`
VM API for VMX, [307](#)
- `L4_VM_VMX_VMCS_MSR_TSC_AUX`
VM API for VMX, [307](#)
- `L4_VM_VMX_VMCS_XCR0`
VM API for VMX, [307](#)
- `l4_vm_vmx_write`
VM API for VMX, [315](#)
- `l4_vm_vmx_write_16`
VM API for VMX, [316](#)
- `l4_vm_vmx_write_32`
VM API for VMX, [317](#)
- `l4_vm_vmx_write_64`
VM API for VMX, [317](#)
- `l4_vm_vmx_write_nat`
VM API for VMX, [318](#)
- `L4_WHOLE_ADDRESS_SPACE`
Flex pages, [162](#)
- `L4_WHOLE_IOADDRESS_SPACE`
Flex pages, [162](#)
- `L4drivers::Mmio_register_block< MAX_BITS >`, [1389](#)
- `L4drivers::Register_block< MAX_BITS, BLOCK >`,
[1390](#)
operator[], [1392](#)
r, [1394](#)
- `L4drivers::Register_block_base< MAX_BITS >`, [1395](#)
- `L4drivers::Register_block_impl< BASE, MAX_BITS >`,
[1396](#)
- `L4drivers::Register_block_tmpl< BLOCK >`, [1397](#)
- `L4drivers::Register_tmpl< BITS, BLOCK >`, [1399](#)
clear, [1401](#)
modify, [1401](#)
operator=, [1402](#)
set, [1402](#)
write, [1402](#)
- `L4drivers::Ro_register_block< MAX_BITS, BLOCK >`,
[1403](#)
operator[], [1404](#)
r, [1404](#)
- `L4drivers::Ro_register_tmpl< BITS, BLOCK >`, [1405](#)
operator value_type, [1406](#)
read, [1406](#)
- `L4IO_DEVICE_ANY`
IO interface, [396](#)

- L4IO_DEVICE_INVALID
 - IO interface, [396](#)
- L4IO_DEVICE_OTHER
 - IO interface, [396](#)
- L4IO_DEVICE_PCI
 - IO interface, [396](#)
- l4io_device_types_t
 - IO interface, [395](#)
- L4IO_DEVICE_USB
 - IO interface, [396](#)
- l4io_get_root_device
 - io.h, [2045](#)
- l4io_has_resource
 - IO interface, [396](#)
- l4io_iomem_flags_t
 - IO interface, [396](#)
- l4io_iterate_devices
 - io.h, [2045](#)
- l4io_lookup_device
 - IO interface, [397](#)
- l4io_lookup_resource
 - IO interface, [397](#)
- L4IO_MEM_CACHED
 - IO interface, [396](#)
- L4IO_MEM_EAGER_MAP
 - IO interface, [396](#)
- L4IO_MEM_NONCACHED
 - IO interface, [396](#)
- L4IO_MEM_USE_MTRR
 - IO interface, [396](#)
- L4IO_MEM_USE_RESERVED_AREA
 - IO interface, [396](#)
- l4io_release_iomem
 - IO interface, [398](#)
- l4io_release_ioport
 - IO interface, [398](#)
- l4io_request_all_ioports
 - io.h, [2045](#)
- l4io_request_icu
 - io.h, [2046](#)
- l4io_request_iomem
 - IO interface, [398](#)
- l4io_request_iomem_region
 - IO interface, [399](#)
- l4io_request_ioport
 - IO interface, [400](#)
- l4io_request_resource_iomem
 - IO interface, [400](#)
- L4IO_RESOURCE_ANY
 - IO interface, [396](#)
- L4IO_RESOURCE_INVALID
 - IO interface, [396](#)
- L4IO_RESOURCE_IRQ
 - IO interface, [396](#)
- L4IO_RESOURCE_MEM
 - IO interface, [396](#)
- L4IO_RESOURCE_PORT
 - IO interface, [396](#)
- l4io_resource_t
 - IO interface, [395](#)
- l4io_resource_types_t
 - IO interface, [396](#)
- l4irq_attach
 - Interface using direct functionality., [405](#)
- l4irq_attach_cap
 - Interface using direct functionality., [409](#)
- l4irq_attach_cap_ft
 - Interface using direct functionality., [409](#)
- l4irq_attach_ft
 - Interface using direct functionality., [405](#)
- l4irq_attach_thread
 - Interface using direct functionality., [405](#)
- l4irq_attach_thread_cap
 - Interface using direct functionality., [409](#)
- l4irq_attach_thread_cap_ft
 - Interface using direct functionality., [410](#)
- l4irq_attach_thread_ft
 - Interface using direct functionality., [406](#)
- l4irq_detach
 - Interface using direct functionality., [406](#)
- l4irq_release
 - Interface for asynchronous ISR handlers., [402](#)
- l4irq_request
 - Interface for asynchronous ISR handlers., [402](#)
- l4irq_request_cap
 - Interface for asynchronous ISR handlers with a given IRQ capability., [403](#)
- l4irq_unmask
 - Interface using direct functionality., [407](#)
- l4irq_unmask_and_wait_any
 - Interface using direct functionality., [407](#)
- l4irq_wait
 - Interface using direct functionality., [407](#)
- l4irq_wait_any
 - Interface using direct functionality., [408](#)
- l4la_alloc
 - list_alloc.h, [2828](#)
- l4la_avail
 - list_alloc.h, [2828](#)
- l4la_dump
 - list_alloc.h, [2828](#)
- l4la_free
 - list_alloc.h, [2828](#)
- l4la_init
 - list_alloc.h, [2829](#)
- l4mod.h
 - L4util_l4mod_mod_flag_kernel, [2826](#)
 - L4util_l4mod_mod_flag_mask, [2826](#)
 - L4util_l4mod_mod_flag_roottask, [2826](#)
 - L4util_l4mod_mod_flag_sigma0, [2826](#)
 - L4util_l4mod_mod_flag_unspec, [2826](#)
 - L4util_l4mod_mod_info_flag, [2826](#)
- L4Re, [682](#)
 - chkcapp, [688](#)
 - chkipc, [689](#)
 - chksys, [690](#), [691](#)

- make_shared_cap, 693
- make_shared_del_cap, 694
- make_unique_cap, 695
- make_unique_del_cap, 695
- Shared_cap, 685
- shared_cap, 685
- Shared_del_cap, 685
- shared_del_cap, 686
- throw_error, 696
- Unique_cap, 686
- unique_cap, 687
- Unique_del_cap, 687
- unique_del_cap, 687
- L4Re Build System, 28
- L4Re C Interface, 454
- L4Re C++ Interface, 506
- L4Re Capability API, 517
 - cap_alloc, 519
 - make_ref_cap, 518
 - make_ref_del_cap, 519
- L4Re ELF Auxiliary Information, 512
 - L4RE_ELF_AUX_ELEM, 513
 - L4RE_ELF_AUX_ELEM_T, 513
 - L4RE_ELF_AUX_T_KIP_ADDR, 514
 - L4RE_ELF_AUX_T_NONE, 514
 - L4RE_ELF_AUX_T_STACK_ADDR, 514
 - L4RE_ELF_AUX_T_STACK_SIZE, 514
 - L4RE_ELF_AUX_T_VMA, 514
- L4Re Protocol identifiers, 514
- L4Re Servers, 43
- L4Re Util C Interface, 474
- L4Re Util C++ Interface, 515
- L4Re::Cap_alloc, 1407
 - alloc, 1408
 - free, 1409
 - get_cap_alloc, 1409
- L4Re::Console, 1410
- L4Re::Dataspace, 1413
 - allocate, 1417
 - clear, 1417
 - copy_in, 1418
 - flags, 1419
 - info, 1419
 - map, 1421
 - map_region, 1422
 - size, 1423
- L4Re::Dataspace::F, 1424
 - Bufferable, 1425
 - Cacheable, 1425
 - Caching_mask, 1425
 - Caching_shift, 1424
 - Flags, 1424
 - Normal, 1425
 - R, 1425
 - Rights_mask, 1425
 - Ro, 1425
 - RW, 1425
 - RWX, 1425
 - RX, 1425
 - Uncacheable, 1425
 - W, 1425
 - X, 1425
- L4Re::Dataspace::Stats, 1425
- L4Re::Debug_obj, 1426
 - debug, 1428
- L4Re::Default_event_payload, 1429
- L4Re::Dma_space, 1430
 - associate, 1432
 - Attribute, 1431
 - Attributes, 1431
 - Bidirectional, 1432
 - Coherent, 1432
 - Direction, 1432
 - disassociate, 1433
 - From_device, 1432
 - map, 1433
 - No_sync, 1432
 - None, 1432
 - Phys_space, 1432
 - Space_attr, 1432
 - To_device, 1432
 - unmap, 1435
- L4Re::Env, 1436
 - env, 1439
 - factory, 1439
 - first_free_cap, 1440
 - first_free_utcb, 1440
 - get, 1441
 - get_cap, 1441, 1443
 - initial_caps, 1443
 - log, 1444
 - main_thread, 1444
 - mem_alloc, 1445
 - parent, 1445, 1446
 - rm, 1446
 - scheduler, 1447
 - task, 1447
 - utcb_area, 1447, 1448
- L4Re::Event, 1448
 - get_axis_info, 1452
 - get_buffer, 1453
 - get_num_streams, 1453
 - get_stream_info, 1453
 - get_stream_info_for_id, 1454
 - get_stream_state_for_id, 1454
- L4Re::Event_buffer_t< PAYLOAD >, 1455
 - Event_buffer_t, 1457
 - next, 1458
 - put, 1458
- L4Re::Event_buffer_t< PAYLOAD >::Event, 1459
- L4Re::Inhibitor, 1460
 - acquire, 1464
 - Name_max, 1464
 - next_lock_info, 1464
 - release, 1465
- L4Re::Log, 1465

- print, [1470](#)
- println, [1470](#)
- L4Re::Mem_alloc, [1470](#)
 - alloc, [1475](#)
 - Continuous, [1475](#)
 - Mem_alloc_flags, [1474](#)
 - Pinned, [1475](#)
 - Super_pages, [1475](#)
- L4Re::Mmio_space, [1476](#)
 - Access_width, [1478](#)
 - mmio_read, [1479](#)
 - mmio_write, [1479](#)
 - Wd_16bit, [1479](#)
 - Wd_32bit, [1479](#)
 - Wd_64bit, [1479](#)
 - Wd_8bit, [1479](#)
- L4Re::Namespace, [1480](#)
 - Link, [1484](#)
 - Overwrite, [1484](#)
 - Partly_resolved, [1483](#)
 - query, [1484](#)
 - Query_result_flags, [1483](#)
 - Query_timeout, [1483](#)
 - Register_flags, [1483](#)
 - register_obj, [1485](#)
 - Ro, [1483](#)
 - Rs, [1484](#)
 - Rw, [1484](#)
 - Rws, [1484](#)
 - Strong, [1484](#)
 - To_default, [1483](#)
 - To_non_blocking, [1483](#)
 - Trusted, [1484](#)
 - unlink, [1486](#)
- L4Re::Ned::Cmd_control, [1487](#)
 - execute, [1488](#), [1489](#)
- L4Re::Parent, [1489](#)
 - signal, [1492](#)
- L4Re::Random, [1493](#)
 - get_random, [1497](#)
- L4Re::Rm, [1498](#)
 - Area, [1503](#)
 - attach, [1504](#), [1505](#)
 - Caching_shift, [1504](#)
 - detach, [1506](#), [1507](#)
 - Detach_again, [1504](#)
 - Detach_exact, [1504](#)
 - Detach_flags, [1503](#)
 - Detach_keep, [1504](#)
 - Detach_overlap, [1504](#)
 - Detach_result, [1504](#)
 - Detached_ds, [1504](#)
 - find, [1508](#)
 - free_area, [1509](#)
 - get_areas, [1509](#)
 - get_regions, [1510](#)
 - Kept_ds, [1504](#)
 - Region, [1503](#)
 - Region_flag_shifts, [1504](#)
 - reserve_area, [1510](#), [1511](#)
 - Split_ds, [1504](#)
- L4Re::Rm::F, [1512](#)
 - Attach_flags, [1513](#)
 - Attach_mask, [1513](#)
 - Cache_buffered, [1514](#)
 - Cache_normal, [1514](#)
 - Cache_uncached, [1514](#)
 - Caching_mask, [1514](#)
 - Detach_free, [1514](#)
 - Ds_map_mask, [1514](#)
 - Eager_map, [1513](#)
 - In_area, [1513](#)
 - Pager, [1514](#)
 - R, [1513](#)
 - Region_flags, [1513](#)
 - Region_flags_mask, [1514](#)
 - Reserved, [1514](#)
 - Rights_mask, [1513](#)
 - RW, [1513](#)
 - RWX, [1514](#)
 - RX, [1513](#)
 - Search_addr, [1513](#)
 - W, [1513](#)
 - X, [1513](#)
- L4Re::Rm::Range, [1514](#)
- L4Re::Rm::Unique_region< T >, [1515](#)
 - ~Unique_region, [1518](#)
 - get, [1518](#)
 - is_valid, [1519](#)
 - operator=, [1519](#)
 - release, [1520](#)
 - reset, [1520](#)
 - Unique_region, [1517](#), [1518](#)
- L4Re::Smart_cap_auto< Unmap_flags >, [1521](#)
- L4Re::Smart_count_cap< Unmap_flags >, [1522](#)
- L4Re::Util, [697](#)
 - make_shared_cap, [705](#)
 - make_shared_del_cap, [705](#)
 - make_unique_cap, [705](#)
 - make_unique_del_cap, [705](#)
 - Shared_cap, [700](#)
 - shared_cap, [700](#)
 - Shared_del_cap, [701](#)
 - shared_del_cap, [701](#)
 - Unique_cap, [702](#)
 - unique_cap, [703](#)
 - Unique_del_cap, [703](#)
 - unique_del_cap, [704](#)
- L4Re::Util::Br_manager, [1523](#)
 - alloc_buffer_demand, [1526](#)
 - get_rcv_cap, [1527](#)
 - realloc_rcv_cap, [1527](#)
 - set_rcv_cap_flags, [1528](#)
- L4Re::Util::Br_manager_hooks, [1529](#)
- L4Re::Util::Br_manager_timeout_hooks, [1531](#)
- L4Re::Util::Cap_alloc_base, [1535](#)

- L4Re::Util::Counter< COUNTER >, 1536
- L4Re::Util::Counter_atomic< COUNTER >, 1536
- L4Re::Util::Counting_cap_alloc< COUNTERTYPE >, 1537
 - alloc, 1539
 - Counting_cap_alloc, 1539
 - free, 1540
 - release, 1541
 - setup, 1542
 - take, 1542
- L4Re::Util::Dataspace_svr, 1543
 - allocate, 1544
 - clear, 1544
 - copy, 1545
 - is_static, 1545
 - map, 1545
 - map_hook, 1546
 - page_shift, 1546
 - release, 1547
 - take, 1547
- L4Re::Util::Event_buffer_consumer_t< PAYLOAD >, 1548
 - foreach_available_event, 1550
 - process, 1551
- L4Re::Util::Event_buffer_t< PAYLOAD >, 1552
 - attach, 1555
 - buf, 1555
 - detach, 1556
- L4Re::Util::Event_svr< SVR >, 1556
- L4Re::Util::Event_t< PAYLOAD >, 1558
 - buffer, 1559
 - init, 1560
 - init_poll, 1561
 - irq, 1561
 - Mode, 1559
 - Mode_irq, 1559
 - Mode_polling, 1559
- L4Re::Util::Item_alloc_base, 1562
- L4Re::Util::Names::Name, 1562
- L4Re::Util::Names::Name_space, 1566
 - alloc_dynamic_entry, 1567
 - copy_receive_cap, 1567
 - free_capability, 1567
 - free_dynamic_entry, 1568
 - free_epiface, 1568
 - get_epiface, 1568
- L4Re::Util::Object_registry, 1569
 - Object_registry, 1572
 - register_irq_obj, 1573
 - register_obj, 1573, 1574
 - unregister_obj, 1575
- L4Re::Util::Ref_cap< T >, 1576
- L4Re::Util::Ref_del_cap< T >, 1577
- L4Re::Util::Registry_server< LOOP_HOOKS >, 1578
 - loop, 1583
 - Registry_server, 1582, 1583
- L4Re::Util::Smart_cap_auto< Unmap_flags >, 1584
- L4Re::Util::Smart_count_cap< Unmap_flags >, 1585
- L4Re::Util::Vcon_svr< SVR >, 1586
- L4Re::Util::Video::Goos_svr, 1587
 - get_fb, 1588
 - init_infos, 1588
 - refresh, 1589
 - screen_info, 1589
 - view_info, 1589
- L4Re::Vfs, 706
- L4Re::Vfs::Be_file, 1590
 - data_space, 1593
 - fstat64, 1593
 - unlock_all_locks, 1593
- L4Re::Vfs::Be_file_system, 1594
 - ~Be_file_system, 1596
 - Be_file_system, 1596
 - type, 1596
- L4Re::Vfs::Directory, 1597
 - faccessat, 1599
 - link, 1599
 - mkdir, 1599
 - rename, 1600
 - rmdir, 1600
 - symlink, 1601
 - unlink, 1601
- L4Re::Vfs::File, 1602
- L4Re::Vfs::File_system, 1605
 - mount, 1606
 - type, 1606
- L4Re::Vfs::Fs, 1607
 - alloc_fd, 1609
 - free_fd, 1609
 - get_file, 1609
 - mount, 1610
 - set_fd, 1610
- L4Re::Vfs::Generic_file, 1611
 - fchmod, 1612
 - fstat64, 1612
 - get_status_flags, 1613
 - set_status_flags, 1613
 - unlock_all_locks, 1614
- L4Re::Vfs::Mman, 1614
- L4Re::Vfs::Ops, 1616
- L4Re::Vfs::Regular_file, 1619
 - data_space, 1622
 - fdatasync, 1622
 - fsync, 1622
 - ftruncate64, 1622
 - get_lock, 1623
 - lseek64, 1623
 - readv, 1623
 - set_lock, 1624
 - writew, 1624
- L4Re::Vfs::Special_file, 1625
 - ioctl, 1626
- L4Re::Video::Color_component, 1627
 - Color_component, 1628
 - dump, 1628
 - get, 1628

- operator==, [1629](#)
- set, [1629](#)
- shift, [1629](#)
- size, [1630](#)
- L4Re::Video::Goos, [1631](#)
 - create_buffer, [1634](#)
 - create_view, [1635](#)
 - delete_buffer, [1635](#)
 - delete_view, [1635](#)
 - F_auto_refresh, [1634](#)
 - F_dynamic_buffers, [1634](#)
 - F_dynamic_views, [1634](#)
 - F_pointer, [1634](#)
 - Flags, [1634](#)
 - get_static_buffer, [1636](#)
 - info, [1636](#)
 - view, [1637](#)
- L4Re::Video::Goos::Info, [1637](#)
- L4Re::Video::Pixel_info, [1639](#)
 - a, [1641](#), [1642](#)
 - b, [1642](#)
 - bits_per_pixel, [1642](#)
 - bytes_per_pixel, [1643](#)
 - dump, [1644](#)
 - g, [1644](#)
 - has_alpha, [1645](#)
 - operator==, [1645](#)
 - padding, [1645](#)
 - Pixel_info, [1641](#)
 - r, [1646](#)
- L4Re::Video::View, [1647](#)
 - F_above, [1649](#)
 - F_dyn_allocated, [1648](#)
 - F_flags_mask, [1649](#)
 - F_fully_dynamic, [1648](#)
 - F_none, [1648](#)
 - F_set_background, [1648](#)
 - F_set_buffer, [1648](#)
 - F_set_buffer_offset, [1648](#)
 - F_set_bytes_per_line, [1648](#)
 - F_set_flags, [1648](#)
 - F_set_pixel, [1648](#)
 - F_set_position, [1648](#)
 - Flags, [1648](#)
 - info, [1649](#)
 - refresh, [1649](#)
 - set_info, [1650](#)
 - set_viewport, [1650](#)
 - stack, [1651](#)
 - V_flags, [1649](#)
- L4Re::Video::View::Info, [1651](#)
- l4re_aux_t, [1654](#)
- l4re_debug_obj_debug
 - Debug interface, [466](#)
- l4re_dma_space_associate
 - DMA Space Interface, [458](#)
- L4RE_DMA_SPACE_BIDIRECTIONAL
 - dma_space.h, [2313](#)
- L4RE_DMA_SPACE_COHERENT
 - dma_space.h, [2314](#)
- l4re_dma_space_direction
 - dma_space.h, [2313](#)
- l4re_dma_space_disassociate
 - DMA Space Interface, [459](#)
- L4RE_DMA_SPACE_FROM_DEVICE
 - dma_space.h, [2313](#)
- l4re_dma_space_map
 - DMA Space Interface, [459](#)
- L4RE_DMA_SPACE_NONE
 - dma_space.h, [2313](#)
- L4RE_DMA_SPACE_PHYS_SPACE
 - dma_space.h, [2314](#)
- l4re_dma_space_space_attribs
 - dma_space.h, [2313](#)
- l4re_dma_space_t
 - DMA Space Interface, [458](#)
- L4RE_DMA_SPACE_TO_DEVICE
 - dma_space.h, [2313](#)
- l4re_dma_space_unmap
 - DMA Space Interface, [460](#)
- l4re_ds_allocate
 - Dataspace interface, [462](#)
- l4re_ds_clear
 - Dataspace interface, [463](#)
- l4re_ds_copy_in
 - Dataspace interface, [463](#)
- L4RE_DS_F_BUFFERABLE
 - Dataspace interface, [462](#)
- L4RE_DS_F_CACHEABLE
 - Dataspace interface, [462](#)
- L4RE_DS_F_CACHING_MASK
 - Dataspace interface, [462](#)
- L4RE_DS_F_CACHING_SHIFT
 - Dataspace interface, [462](#)
- L4RE_DS_F_NORMAL
 - Dataspace interface, [462](#)
- L4RE_DS_F_UNCACHEABLE
 - Dataspace interface, [462](#)
- l4re_ds_flags
 - Dataspace interface, [464](#)
- l4re_ds_info
 - Dataspace interface, [464](#)
- l4re_ds_map_flags
 - Dataspace interface, [462](#)
- l4re_ds_size
 - Dataspace interface, [465](#)
- l4re_ds_stats_t, [1655](#)
- L4RE_ELF_AUX_ELEM
 - L4Re ELF Auxiliary Information, [513](#)
- L4RE_ELF_AUX_ELEM_T
 - L4Re ELF Auxiliary Information, [513](#)
- l4re_elf_aux_mword_t, [1655](#)
- l4re_elf_aux_t, [1656](#)
- L4RE_ELF_AUX_T_KIP_ADDR
 - L4Re ELF Auxiliary Information, [514](#)
- L4RE_ELF_AUX_T_NONE

- L4Re ELF Auxiliary Information, [514](#)
- L4RE_ELF_AUX_T_STACK_ADDR
 - L4Re ELF Auxiliary Information, [514](#)
- L4RE_ELF_AUX_T_STACK_SIZE
 - L4Re ELF Auxiliary Information, [514](#)
- L4RE_ELF_AUX_T_VMA
 - L4Re ELF Auxiliary Information, [514](#)
- l4re_elf_aux_vma_t, [1656](#)
- l4re_env
 - Initial Environment, [470](#)
- l4re_env_cap_entry_t, [1657](#)
 - flags, [1658](#)
 - l4re_env_cap_entry_t, [1658](#)
- l4re_env_get_cap
 - Initial Environment, [470](#)
- l4re_env_get_cap_e
 - Initial Environment, [471](#)
- l4re_env_get_cap_l
 - Initial Environment, [472](#)
- l4re_env_t, [1659](#)
 - caps, [1660](#)
 - env.h, [2359](#)
- l4re_event_get_axis_info
 - Event interface, [467](#)
- l4re_event_get_buffer
 - Event interface, [467](#)
- l4re_event_get_num_streams
 - Event interface, [468](#)
- l4re_event_get_stream_info
 - Event interface, [468](#)
- l4re_event_get_stream_info_for_id
 - Event interface, [468](#)
- l4re_event_t, [1661](#)
- l4re_inhibitor_acquire
 - inhibitor.h, [2320](#)
- l4re_inhibitor_next_lock_info
 - inhibitor.h, [2321](#)
- l4re_inhibitor_release
 - inhibitor.h, [2321](#)
- l4re_kip
 - Initial Environment, [473](#)
- l4re_log_print
 - Log interface, [475](#)
- l4re_log_print_srv
 - Log interface, [475](#)
- l4re_log_printn
 - Log interface, [476](#)
- l4re_log_printn_srv
 - Log interface, [477](#)
- l4re_ma_alloc
 - Memory allocator, [478](#)
- l4re_ma_alloc_align
 - Memory allocator, [479](#)
- l4re_ma_alloc_align_srv
 - Memory allocator, [480](#)
- l4re_ma_flags
 - Memory allocator, [478](#)
- l4re_ns_query_srv
 - Namespace interface, [482](#)
- l4re_ns_query_to_srv
 - Namespace interface, [483](#)
- l4re_ns_register_flags
 - Namespace interface, [482](#)
- l4re_ns_register_obj_srv
 - Namespace interface, [484](#)
- L4RE_PROTO_DATASPACE
 - protocols.h, [2398](#)
- L4RE_PROTO_DEBUG
 - protocols.h, [2398](#)
- L4RE_PROTO_DMA_SPACE
 - protocols.h, [2398](#)
- L4RE_PROTO_EVENT
 - protocols.h, [2398](#)
- L4RE_PROTO_GOOS
 - protocols.h, [2398](#)
- L4RE_PROTO_INHIBITOR
 - protocols.h, [2398](#)
- L4RE_PROTO_MMIO_SPACE
 - protocols.h, [2398](#)
- L4RE_PROTO_NAMESPACE
 - protocols.h, [2398](#)
- L4RE_PROTO_PARENT
 - protocols.h, [2398](#)
- L4RE_PROTO_RM
 - protocols.h, [2398](#)
- L4RE_PROTO_RSVD_1
 - protocols.h, [2398](#)
- L4re_protocols
 - protocols.h, [2398](#)
- l4re_rm_attach
 - Region map interface, [487](#)
- l4re_rm_attach_srv
 - Region map interface, [488](#)
- L4RE_RM_CACHING_SHIFT
 - Region map interface, [486](#)
- l4re_rm_detach
 - Region map interface, [488](#)
- l4re_rm_detach_ds
 - Region map interface, [490](#)
- l4re_rm_detach_ds_unmap
 - Region map interface, [491](#)
- l4re_rm_detach_srv
 - Region map interface, [492](#)
- l4re_rm_detach_unmap
 - Region map interface, [492](#)
- L4RE_RM_F_ATTACH_FLAGS
 - Region map interface, [486](#)
- L4RE_RM_F_CACHE_BUFFERED
 - Region map interface, [486](#)
- L4RE_RM_F_CACHE_NORMAL
 - Region map interface, [486](#)
- L4RE_RM_F_CACHE_UNCACHED
 - Region map interface, [486](#)
- L4RE_RM_F_CACHING
 - Region map interface, [486](#)
- L4RE_RM_F_EAGER_MAP

- Region map interface, [486](#)
- L4RE_RM_F_IN_AREA
 - Region map interface, [486](#)
- L4RE_RM_F_NO_ALIAS
 - Region map interface, [486](#)
- L4RE_RM_F_PAGER
 - Region map interface, [486](#)
- L4RE_RM_F_R
 - Region map interface, [486](#)
- L4RE_RM_F_RESERVED
 - Region map interface, [486](#)
- L4RE_RM_F_SEARCH_ADDR
 - Region map interface, [486](#)
- l4re_rm_find
 - Region map interface, [493](#)
- l4re_rm_find_srv
 - Region map interface, [494](#)
- l4re_rm_flags_values
 - Region map interface, [486](#)
- l4re_rm_free_area
 - Region map interface, [495](#)
- l4re_rm_free_area_srv
 - Region map interface, [496](#)
- L4RE_RM_REGION_FLAGS
 - Region map interface, [486](#)
- l4re_rm_reserve_area
 - Region map interface, [496](#)
- l4re_rm_reserve_area_srv
 - Region map interface, [497](#)
- l4re_rm_show_lists
 - Region map interface, [498](#)
- l4re_util_cap_last
 - Capability allocator, [457](#)
- l4re_util_kumem_alloc
 - kumem_alloc.h, [2335](#)
- l4re_video_color_component_t, [1662](#)
- l4re_video_goos_create_buffer
 - Video API, [501](#)
- l4re_video_goos_create_view
 - Video API, [502](#)
- l4re_video_goos_delete_buffer
 - Video API, [502](#)
- l4re_video_goos_delete_view
 - Video API, [502](#)
- l4re_video_goos_get_static_buffer
 - Video API, [502](#)
- l4re_video_goos_get_view
 - Video API, [503](#)
- l4re_video_goos_info
 - Video API, [503](#)
- l4re_video_goos_info_flags_t
 - Video API, [500](#)
- l4re_video_goos_info_t, [1662](#)
- l4re_video_goos_refresh
 - Video API, [504](#)
- l4re_video_pixel_info_t, [1664](#)
- l4re_video_view_get_info
 - Video API, [504](#)
- l4re_video_view_info_flags_t
 - Video API, [501](#)
- l4re_video_view_info_t, [1665](#)
- l4re_video_view_refresh
 - Video API, [504](#)
- l4re_video_view_set_info
 - Video API, [505](#)
- l4re_video_view_set_viewport
 - Video API, [505](#)
- l4re_video_view_stack
 - Video API, [505](#)
- l4re_video_view_t, [1666](#)
 - Video API, [500](#)
- L4SHM-based ring buffer implementation, [524](#)
- l4shmc_add_chunk
 - Chunks, [533](#)
- l4shmc_add_signal
 - Signals, [545](#)
- l4shmc_area_overhead
 - Shared Memory Library, [528](#)
- l4shmc_area_size
 - Shared Memory Library, [528](#)
- l4shmc_area_size_free
 - Shared Memory Library, [529](#)
- l4shmc_attach
 - Shared Memory Library, [529](#)
- l4shmc_attach_signal
 - Signals, [545](#)
- l4shmc_check_magic
 - Signals, [546](#)
- l4shmc_chunk_capacity
 - Chunks, [533](#)
- l4shmc_chunk_consumed
 - Consumer, [537](#)
- l4shmc_chunk_overhead
 - Shared Memory Library, [530](#)
- l4shmc_chunk_ptr
 - Chunks, [534](#)
- l4shmc_chunk_ready
 - Producer, [541](#)
- l4shmc_chunk_ready_sig
 - Producer, [541](#)
- l4shmc_chunk_signal
 - Chunks, [534](#)
- l4shmc_chunk_size
 - Consumer, [537](#)
- l4shmc_chunk_try_to_take
 - Producer, [541](#)
- l4shmc_chunk_try_to_take_for_overwriting
 - Producer, [543](#)
- l4shmc_chunk_try_to_take_for_reading
 - Consumer, [538](#)
- l4shmc_chunk_try_to_take_for_writing
 - Producer, [543](#)
- l4shmc_connect_chunk_signal
 - Shared Memory Library, [530](#)
- l4shmc_create
 - Shared Memory Library, [530](#)

- `l4shmc_enable_chunk`
 - Consumer, [538](#)
- `l4shmc_enable_signal`
 - Consumer, [548](#)
- `l4shmc_get_chunk`
 - Chunks, [535](#)
- `l4shmc_get_chunk_to`
 - Chunks, [535](#)
- `l4shmc_get_client_nr`
 - Shared Memory Library, [531](#)
- `l4shmc_get_initialized_clients`
 - Shared Memory Library, [531](#)
- `l4shmc_get_signal`
 - Signals, [546](#)
- `l4shmc_is_chunk_clear`
 - Producer, [544](#)
- `l4shmc_is_chunk_ready`
 - Consumer, [538](#)
- `l4shmc_iterate_chunk`
 - Chunks, [536](#)
- `l4shmc_mark_client_initialized`
 - Shared Memory Library, [532](#)
- `l4shmc_rb_attach_receiver`
 - `ringbuf.h`, [2484](#)
- `l4shmc_rb_attach_sender`
 - `ringbuf.h`, [2485](#)
- `l4shmc_rb_deinit_buffer`
 - `ringbuf.h`, [2485](#)
- `l4shmc_rb_init_buffer`
 - `ringbuf.h`, [2485](#)
- `l4shmc_rb_init_receiver`
 - `ringbuf.h`, [2486](#)
- `l4shmc_rb_receiver_copy_out`
 - `ringbuf.h`, [2487](#)
- `l4shmc_rb_receiver_notify_done`
 - `ringbuf.h`, [2487](#)
- `l4shmc_rb_receiver_read_next_size`
 - `ringbuf.h`, [2487](#)
- `l4shmc_rb_receiver_wait_for_data`
 - `ringbuf.h`, [2487](#)
- `l4shmc_rb_sender_alloc_packet`
 - `ringbuf.h`, [2488](#)
- `l4shmc_rb_sender_commit_packet`
 - `ringbuf.h`, [2488](#)
- `l4shmc_rb_sender_next_copy_in`
 - `ringbuf.h`, [2489](#)
- `l4shmc_rb_sender_put_data`
 - `ringbuf.h`, [2489](#)
- `L4SHMC_RINGBUF_DATA`
 - `ringbuf.h`, [2483](#)
- `L4SHMC_RINGBUF_DATA_SIZE`
 - `ringbuf.h`, [2484](#)
- `L4SHMC_RINGBUF_HEAD`
 - `ringbuf.h`, [2484](#)
- `l4shmc_ringbuf_head_t`, [1667](#)
- `l4shmc_ringbuf_t`, [1668](#)
- `l4shmc_signal_cap`
 - Signals, [546](#)
- `l4shmc_trigger`
 - Producer, [552](#)
- `l4shmc_wait_any`
 - Consumer, [549](#)
- `l4shmc_wait_any_to`
 - Consumer, [549](#)
- `l4shmc_wait_any_try`
 - Consumer, [550](#)
- `l4shmc_wait_chunk`
 - Consumer, [539](#)
- `l4shmc_wait_chunk_to`
 - Consumer, [539](#)
- `l4shmc_wait_chunk_try`
 - Consumer, [540](#)
- `l4shmc_wait_signal`
 - Consumer, [550](#)
- `l4shmc_wait_signal_to`
 - Consumer, [550](#)
- `l4shmc_wait_signal_try`
 - Consumer, [551](#)
- `l4sigma0_debug_dump`
 - Sigma0 API, [554](#)
- `L4SIGMA0_IPCERROR`
 - Sigma0 API, [553](#)
- `l4sigma0_map_anypage`
 - Sigma0 API, [554](#)
- `l4sigma0_map_errstr`
 - Sigma0 API, [554](#)
- `l4sigma0_map_iomem`
 - Sigma0 API, [555](#)
- `l4sigma0_map_kip`
 - Sigma0 API, [555](#)
- `l4sigma0_map_mem`
 - Sigma0 API, [556](#)
- `L4SIGMA0_NOFPAGE`
 - Sigma0 API, [553](#)
- `L4SIGMA0_NOTALIGNED`
 - Sigma0 API, [553](#)
- `L4SIGMA0_OK`
 - Sigma0 API, [553](#)
- `l4sigma0_return_flags_t`
 - Sigma0 API, [553](#)
- `L4SIGMA0_SMALLERFPAGE`
 - Sigma0 API, [553](#)
- `l4util_add16`
 - Atomic Instructions, [571](#)
- `l4util_add16_res`
 - Atomic Instructions, [571](#)
- `l4util_add32`
 - Atomic Instructions, [571](#)
- `l4util_add32_res`
 - Atomic Instructions, [571](#)
- `l4util_add8`
 - Atomic Instructions, [572](#)
- `l4util_add8_res`
 - Atomic Instructions, [572](#)
- `l4util_and16`
 - Atomic Instructions, [572](#)

- l4util_and16_res
 - Atomic Instructions, [573](#)
- l4util_and32
 - Atomic Instructions, [573](#)
- l4util_and32_res
 - Atomic Instructions, [573](#)
- l4util_and8
 - Atomic Instructions, [574](#)
- l4util_and8_res
 - Atomic Instructions, [574](#)
- l4util_atomic_add
 - Atomic Instructions, [574](#)
- l4util_atomic_inc
 - Atomic Instructions, [575](#)
- l4util_backtrace
 - backtrace.h, [2791](#)
- l4util_bsf
 - Bit Manipulation, [587](#)
- l4util_bsr
 - Bit Manipulation, [588](#)
- l4util_btc
 - Bit Manipulation, [588](#)
- l4util_btr
 - Bit Manipulation, [588](#)
- l4util_bts
 - Bit Manipulation, [589](#)
- l4util_clear_bit
 - Bit Manipulation, [590](#)
- l4util_cmpxchg
 - Atomic Instructions, [575](#)
- l4util_cmpxchg16
 - Atomic Instructions, [576](#)
- l4util_cmpxchg32
 - Atomic Instructions, [576](#)
- l4util_cmpxchg64
 - Atomic Instructions, [577](#)
- l4util_cmpxchg8
 - Atomic Instructions, [577](#)
- l4util_complement_bit
 - Bit Manipulation, [590](#)
- l4util_cpu_capabilities
 - CPU related functions, [595](#)
- l4util_cpu_capabilities_nocheck
 - CPU related functions, [595](#)
- l4util_cpu_has_cpuid
 - CPU related functions, [596](#)
- l4util_dec16
 - Atomic Instructions, [578](#)
- l4util_dec16_res
 - Atomic Instructions, [578](#)
- l4util_dec32
 - Atomic Instructions, [578](#)
- l4util_dec32_res
 - Atomic Instructions, [578](#)
- l4util_dec8
 - Atomic Instructions, [579](#)
- l4util_dec8_res
 - Atomic Instructions, [579](#)
- l4util_find_first_set_bit
 - Bit Manipulation, [591](#)
- l4util_find_first_zero_bit
 - Bit Manipulation, [591](#)
- l4util_idt_desc_t, [1669](#)
- l4util_idt_header_t, [1670](#)
- l4util_in16
 - IA32 Port I/O API, [624](#)
- l4util_in32
 - IA32 Port I/O API, [625](#)
- l4util_in8
 - IA32 Port I/O API, [625](#)
- l4util_inc16
 - Atomic Instructions, [579](#)
- l4util_inc16_res
 - Atomic Instructions, [580](#)
- l4util_inc32
 - Atomic Instructions, [580](#)
- l4util_inc32_res
 - Atomic Instructions, [580](#)
- l4util_inc8
 - Atomic Instructions, [580](#)
- l4util_inc8_res
 - Atomic Instructions, [581](#)
- l4util_ins16
 - IA32 Port I/O API, [626](#)
- l4util_ins32
 - IA32 Port I/O API, [626](#)
- l4util_ins8
 - IA32 Port I/O API, [626](#)
- l4util_ioport_map
 - port_io.h, [1996](#)
- l4util_irq_acknowledge
 - irq.h, [2174](#), [2184](#)
- l4util_kip_for_each_feature
 - Kernel Interface Page API, [631](#)
- l4util_kip_kernel_abi_version
 - Kernel Interface Page API, [632](#)
- l4util_kip_kernel_has_feature
 - Kernel Interface Page API, [632](#)
- l4util_kip_kernel_is_ux
 - Kernel Interface Page API, [632](#)
- l4util_l4mod_info, [1671](#)
 - vbe_ctrl_info, [1672](#)
- l4util_l4mod_mod, [1672](#)
- L4util_l4mod_mod_flag_kernel
 - l4mod.h, [2826](#)
- L4util_l4mod_mod_flag_mask
 - l4mod.h, [2826](#)
- L4util_l4mod_mod_flag_roottask
 - l4mod.h, [2826](#)
- L4util_l4mod_mod_flag_sigma0
 - l4mod.h, [2826](#)
- L4util_l4mod_mod_flag_unspec
 - l4mod.h, [2826](#)
- l4util_l4mod_mod_info_flag
 - l4mod.h, [2826](#)
- l4util_mb_addr_range_t, [1673](#)

- [l4util_mb_apm_t](#), [1674](#)
- [l4util_mb_drive_t](#), [1675](#)
 - [drive_cylinders](#), [1676](#)
 - [drive_mode](#), [1676](#)
 - [drive_number](#), [1676](#)
- [l4util_mb_for_each_mmap_entry](#)
 - [mb_info.h](#), [2835](#)
- [l4util_mb_info_t](#), [1677](#)
- [L4UTIL_MB_MEMORY](#)
 - [mb_info.h](#), [2835](#)
- [l4util_mb_mod_t](#), [1678](#)
- [l4util_mb_vbe_ctrl_t](#), [1679](#)
- [l4util_mb_vbe_mode_t](#), [1680](#)
- [l4util_micros2l4to](#)
 - [util.h](#), [1969](#), [1973](#)
 - Utility Functions, [567](#)
- [l4util_next_power2](#)
 - Bit Manipulation, [592](#)
- [l4util_or16](#)
 - Atomic Instructions, [581](#)
- [l4util_or16_res](#)
 - Atomic Instructions, [581](#)
- [l4util_or32](#)
 - Atomic Instructions, [582](#)
- [l4util_or32_res](#)
 - Atomic Instructions, [582](#)
- [l4util_or8](#)
 - Atomic Instructions, [582](#)
- [l4util_or8_res](#)
 - Atomic Instructions, [582](#)
- [l4util_out16](#)
 - IA32 Port I/O API, [627](#)
- [l4util_out32](#)
 - IA32 Port I/O API, [627](#)
- [l4util_out8](#)
 - IA32 Port I/O API, [627](#)
- [l4util_outs16](#)
 - IA32 Port I/O API, [629](#)
- [l4util_outs32](#)
 - IA32 Port I/O API, [629](#)
- [l4util_outs8](#)
 - IA32 Port I/O API, [630](#)
- [l4util_rand](#)
 - Random number support, [634](#)
- [l4util_set_bit](#)
 - Bit Manipulation, [592](#)
- [l4util_splitlog2_hdl](#)
 - Utility Functions, [567](#)
- [l4util_splitlog2_size](#)
 - Utility Functions, [568](#)
- [l4util_srand](#)
 - Random number support, [634](#)
- [l4util_sub16](#)
 - Atomic Instructions, [583](#)
- [l4util_sub16_res](#)
 - Atomic Instructions, [583](#)
- [l4util_sub32](#)
 - Atomic Instructions, [583](#)
- [l4util_sub32_res](#)
 - Atomic Instructions, [584](#)
- [l4util_sub8](#)
 - Atomic Instructions, [584](#)
- [l4util_sub8_res](#)
 - Atomic Instructions, [584](#)
- [l4util_test_bit](#)
 - Bit Manipulation, [592](#)
- [l4util_xchg](#)
 - Atomic Instructions, [585](#)
- [l4util_xchg16](#)
 - Atomic Instructions, [585](#)
- [l4util_xchg32](#)
 - Atomic Instructions, [585](#)
- [l4util_xchg8](#)
 - Atomic Instructions, [586](#)
- [L4vbus](#), [707](#)
- [L4vbus GPIO functions](#), [437](#)
 - [l4vbus_gpio_config_get](#), [438](#)
 - [l4vbus_gpio_config_pad](#), [439](#)
 - [l4vbus_gpio_config_pull](#), [440](#)
 - [L4vbus_gpio_generic_func](#), [438](#)
 - [l4vbus_gpio_get](#), [440](#)
 - [l4vbus_gpio_multi_config_pad](#), [441](#)
 - [l4vbus_gpio_multi_get](#), [442](#)
 - [l4vbus_gpio_multi_set](#), [442](#)
 - [l4vbus_gpio_multi_setup](#), [443](#)
 - [L4VBUS_GPIO_PIN_PULL_DOWN](#), [438](#)
 - [L4VBUS_GPIO_PIN_PULL_NONE](#), [438](#)
 - [L4VBUS_GPIO_PIN_PULL_UP](#), [438](#)
 - [L4vbus_gpio_pull_modes](#), [438](#)
 - [l4vbus_gpio_set](#), [444](#)
 - [l4vbus_gpio_setup](#), [445](#)
 - [L4VBUS_GPIO_SETUP_INPUT](#), [438](#)
 - [L4VBUS_GPIO_SETUP_IRQ](#), [438](#)
 - [L4VBUS_GPIO_SETUP_OUTPUT](#), [438](#)
 - [l4vbus_gpio_to_irq](#), [445](#)
- [L4vbus PCI functions](#), [446](#)
 - [l4vbus_pci_cfg_read](#), [447](#)
 - [l4vbus_pci_cfg_write](#), [448](#)
 - [l4vbus_pci_irq_enable](#), [449](#)
 - [l4vbus_pcidev_cfg_read](#), [450](#)
 - [l4vbus_pcidev_cfg_write](#), [450](#)
 - [l4vbus_pcidev_irq_enable](#), [451](#)
- [L4vbus power management functions](#), [452](#)
 - [l4vbus_pm_resume](#), [452](#)
 - [l4vbus_pm_suspend](#), [453](#)
- [L4vbus::Device](#), [1684](#)
 - [bus_cap](#), [1687](#)
 - [dev_handle](#), [1687](#)
 - Device, [1686](#)
 - [device](#), [1688](#)
 - [device_by_hid](#), [1688](#)
 - [get_resource](#), [1689](#)
 - [is_compatible](#), [1690](#)
 - [next_device](#), [1691](#)
 - [operator!=](#), [1692](#)
 - [operator==](#), [1692](#)

- L4vbus::Gpio_module, 1693
 - config_pad, 1696
 - get, 1696
 - pin, 1697
 - set, 1698
 - setup, 1699
- L4vbus::Gpio_module::Pin_slice, 1700
- L4vbus::Gpio_pin, 1700
 - config_get, 1704
 - config_pad, 1704
 - config_pull, 1705
 - get, 1706
 - pin, 1706
 - set, 1706
 - setup, 1707
 - to_irq, 1708
- L4vbus::Icu, 1709
 - Src_dev_handle, 1712
 - Src_types, 1712
 - vicu, 1712
- L4vbus::Pci_dev, 1713
 - cfg_read, 1717
 - cfg_write, 1717
 - irq_enable, 1718
- L4vbus::Pci_host_bridge, 1719
 - cfg_read, 1723
 - cfg_write, 1723
 - irq_enable, 1724
- L4vbus::Pm< DEC >, 1725
 - pm_resume, 1727
 - pm_suspend, 1727
- L4vbus::Vbus, 1728
 - assign_dma_domain, 1735, 1736
 - release_ioport, 1737
 - request_ioport, 1737
 - root, 1738
- l4vbus_assign_dma_domain
 - L4 Vbus functions, 428
- L4VBUS_DEVICE_F_CHILDREN
 - vbus_types.h, 2873
- l4vbus_device_flags_t
 - vbus_types.h, 2873
- l4vbus_device_t, 1739
- L4vbus_dma_domain_assign_flags
 - L4 Vbus functions, 427
- L4VBUS_DMAD_BIND
 - L4 Vbus functions, 428
- L4VBUS_DMAD_KERNEL_DMA_SPACE
 - L4 Vbus functions, 428
- L4VBUS_DMAD_L4RE_DMA_SPACE
 - L4 Vbus functions, 428
- L4VBUS_DMAD_UNBIND
 - L4 Vbus functions, 428
- l4vbus_get_adr
 - L4 Vbus functions, 429
- l4vbus_get_device
 - L4 Vbus functions, 429
- l4vbus_get_device_by_hid
 - L4 Vbus functions, 430
- l4vbus_get_hid
 - L4 Vbus functions, 431
- l4vbus_get_next_device
 - L4 Vbus functions, 431
- l4vbus_get_resource
 - L4 Vbus functions, 433
- l4vbus_gpio_config_get
 - L4vbus GPIO functions, 438
- l4vbus_gpio_config_pad
 - L4vbus GPIO functions, 439
- l4vbus_gpio_config_pull
 - L4vbus GPIO functions, 440
- L4vbus_gpio_generic_func
 - L4vbus GPIO functions, 438
- l4vbus_gpio_get
 - L4vbus GPIO functions, 440
- l4vbus_gpio_multi_config_pad
 - L4vbus GPIO functions, 441
- l4vbus_gpio_multi_get
 - L4vbus GPIO functions, 442
- l4vbus_gpio_multi_set
 - L4vbus GPIO functions, 442
- l4vbus_gpio_multi_setup
 - L4vbus GPIO functions, 443
- L4VBUS_GPIO_PIN_PULL_DOWN
 - L4vbus GPIO functions, 438
- L4VBUS_GPIO_PIN_PULL_NONE
 - L4vbus GPIO functions, 438
- L4VBUS_GPIO_PIN_PULL_UP
 - L4vbus GPIO functions, 438
- L4vbus_gpio_pull_modes
 - L4vbus GPIO functions, 438
- l4vbus_gpio_set
 - L4vbus GPIO functions, 444
- l4vbus_gpio_setup
 - L4vbus GPIO functions, 445
- L4VBUS_GPIO_SETUP_INPUT
 - L4vbus GPIO functions, 438
- L4VBUS_GPIO_SETUP_IRQ
 - L4vbus GPIO functions, 438
- L4VBUS_GPIO_SETUP_OUTPUT
 - L4vbus GPIO functions, 438
- l4vbus_gpio_to_irq
 - L4vbus GPIO functions, 445
- L4VBUS_ICU_SRC_DEV_HANDLE
 - vbus.h, 2861
- l4vbus_icu_src_types
 - vbus.h, 2860
- L4VBUS_IFACE_SHIFT
 - vbus_interfaces.h, 2867
- l4vbus_iface_type_t
 - vbus_interfaces.h, 2867
- L4VBUS_INTERFACE_BUS
 - vbus_interfaces.h, 2868
- L4VBUS_INTERFACE_GENERIC
 - vbus_interfaces.h, 2868
- L4VBUS_INTERFACE_GPIO

- vbus_interfaces.h, [2868](#)
- L4VBUS_INTERFACE_ICU
 - vbus_interfaces.h, [2868](#)
- L4VBUS_INTERFACE_PCI
 - vbus_interfaces.h, [2868](#)
- L4VBUS_INTERFACE_PCIDEV
 - vbus_interfaces.h, [2868](#)
- L4VBUS_INTERFACE_PM
 - vbus_interfaces.h, [2868](#)
- l4vbus_is_compatible
 - L4 Vbus functions, [434](#)
- L4VBUS_NULL
 - vbus.h, [2860](#)
- l4vbus_pci_cfg_read
 - L4vbus PCI functions, [447](#)
- l4vbus_pci_cfg_write
 - L4vbus PCI functions, [448](#)
- l4vbus_pci_irq_enable
 - L4vbus PCI functions, [449](#)
- l4vbus_pciddev_cfg_read
 - L4vbus PCI functions, [450](#)
- l4vbus_pciddev_cfg_write
 - L4vbus PCI functions, [450](#)
- l4vbus_pciddev_irq_enable
 - L4vbus PCI functions, [451](#)
- l4vbus_pm_resume
 - L4vbus power management functions, [452](#)
- l4vbus_pm_suspend
 - L4vbus power management functions, [453](#)
- l4vbus_release_ioport
 - L4 Vbus functions, [434](#)
- l4vbus_request_ioport
 - L4 Vbus functions, [435](#)
- L4VBUS_RESOURCE_BUS
 - vbus_types.h, [2874](#)
- L4VBUS_RESOURCE_DMA_DOMAIN
 - vbus_types.h, [2874](#)
- L4VBUS_RESOURCE_F_MEM_MMIO_READ
 - vbus_types.h, [2874](#)
- L4VBUS_RESOURCE_F_MEM_MMIO_WRITE
 - vbus_types.h, [2874](#)
- L4VBUS_RESOURCE_F_MEM_R
 - vbus_types.h, [2874](#)
- L4VBUS_RESOURCE_F_MEM_W
 - vbus_types.h, [2874](#)
- l4vbus_resource_flags_t
 - vbus_types.h, [2873](#)
- L4VBUS_RESOURCE_GPIO
 - vbus_types.h, [2874](#)
- L4VBUS_RESOURCE_INVALID
 - vbus_types.h, [2874](#)
- L4VBUS_RESOURCE_IRQ
 - vbus_types.h, [2874](#)
- L4VBUS_RESOURCE_MAX
 - vbus_types.h, [2874](#)
- L4VBUS_RESOURCE_MEM
 - vbus_types.h, [2874](#)
- L4VBUS_RESOURCE_PORT
 - vbus_types.h, [2874](#)
- l4vbus_resource_t, [1740](#)
- l4vbus_resource_type_t
 - vbus_types.h, [2874](#)
- L4VBUS_ROOT_BUS
 - vbus.h, [2860](#)
- l4vbus_subinterface_supported
 - vbus_interfaces.h, [2868](#)
- l4vbus_vicu_get_cap
 - L4 Vbus functions, [436](#)
- L4vcpu::State, [1741](#)
 - add, [1742](#)
 - clear, [1742](#)
 - set, [1742](#)
 - State, [1741](#)
- L4vcpu::Vcpu, [1743](#)
 - cast, [1746](#)
 - entry_ip, [1746](#)
 - entry_sp, [1747](#)
 - ext_alloc, [1747](#)
 - i, [1748](#)
 - irq_disable_save, [1748](#)
 - irq_enable, [1748](#)
 - irq_restore, [1749](#)
 - is_irq_entry, [1750](#)
 - is_page_fault_entry, [1750](#)
 - r, [1750](#), [1751](#)
 - saved_state, [1751](#)
 - state, [1751](#), [1752](#)
 - task, [1752](#)
 - wait_for_event, [1753](#)
- l4vcpu_ext_alloc
 - Extended vCPU support, [649](#)
- l4vcpu_irq_disable
 - vCPU Support Library, [642](#)
- l4vcpu_irq_disable_save
 - vCPU Support Library, [643](#)
- l4vcpu_irq_enable
 - vCPU Support Library, [644](#)
- l4vcpu_irq_restore
 - vCPU Support Library, [645](#)
- l4vcpu_is_irq_entry
 - vCPU Support Library, [646](#)
- l4vcpu_is_page_fault_entry
 - vCPU Support Library, [646](#)
- l4vcpu_print_state
 - vCPU Support Library, [647](#)
- l4vcpu_wait_for_event
 - vCPU Support Library, [648](#)
- l4vio_net_p2p, a virtual network point-to-point link, [58](#)
- L4virtio, [708](#)
- L4virtio::Device, [1754](#)
 - config_queue, [1758](#)
 - device_config, [1759](#)
 - device_notification_irq, [1759](#)
 - register_ds, [1760](#)
 - set_status, [1761](#)
- L4virtio::Driver::Block_device, [1762](#)

- add_block, 1766
- process_request, 1767
- process_used_queue, 1768
- send_request, 1768
- setup_device, 1769
- start_request, 1770
- L4virtio::Driver::Block_device::Handle, 1771
- L4virtio::Driver::Device, 1772
 - bind_notification_irq, 1775
 - config_queue, 1776
 - driver_acknowledge, 1777
 - driver_connect, 1778
 - feature_negotiated, 1779
 - max_queue_size, 1780
 - register_ds, 1780
 - send, 1781
 - send_and_wait, 1782
 - wait, 1783
 - wait_for_next_used, 1784
- L4virtio::Driver::Virtio_net_device, 1785
 - finish_rx, 1789
 - rx_pkt, 1789
 - rx_queue_size, 1790
 - setup_device, 1790
 - tx, 1791
 - tx_queue_size, 1792
 - wait_rx, 1793
- L4virtio::Driver::Virtio_net_device::Packet, 1794
- L4virtio::Driver::Virtqueue, 1794
 - alloc_descriptor, 1799
 - desc, 1799
 - enqueue_descriptor, 1800
 - find_next_used, 1801
 - free_descriptor, 1801
 - init_queue, 1802, 1803
 - initialize_rings, 1804
- L4virtio::Ptr< T >, 1805
 - get, 1807
 - Invalid, 1807
 - Invalid_type, 1807
 - is_valid, 1808
- L4virtio::Svr::Bad_descriptor, 1809
 - Bad_address, 1810
 - Bad_descriptor, 1810
 - Bad_flags, 1810
 - Bad_next, 1810
 - Bad_rights, 1810
 - Bad_size, 1810
 - Error, 1810
 - message, 1810
- L4virtio::Svr::Block_dev_base< Ds_data >, 1811
 - Block_dev_base, 1815
 - finalize_request, 1816
 - get_writeback, 1817
 - process_request, 1817
 - set_blk_size, 1817
 - set_config_wce, 1818
 - set_discard, 1818
 - set_size_max, 1818
 - set_topology, 1819
 - set_write_zeroes, 1819
- L4virtio::Svr::Block_request< Ds_data >, 1820
 - data_size, 1820
 - next_block, 1821
- L4virtio::Svr::Data_buffer, 1822
 - copy_to, 1824
 - Data_buffer, 1823
 - done, 1824
 - set, 1824
 - skip, 1825
- L4virtio::Svr::Dev_config, 1826
 - change_queue_config, 1830
 - Dev_config, 1828, 1829
 - ds, 1830
 - get_cmd, 1830
 - guest_features, 1831
 - hdr, 1832
 - negotiated_features, 1832
 - qconfig, 1833
 - reset_cmd, 1834
 - reset_queue, 1834
 - set_device_needs_reset, 1835
 - set_status, 1836
 - status, 1837
- L4virtio::Svr::Dev_features, 1838
- L4virtio::Svr::Dev_status, 1839
 - running, 1840
- L4virtio::Svr::Device_t< DATA >, 1840
 - add_trusted_dataspaces, 1843
 - device_error, 1844
 - device_notify_irq, 1844
 - handle_mem_cmd_write, 1844
 - init_mem_info, 1845
 - register_driver_irq, 1845
 - reset_queue_config, 1846
 - setup_queue, 1846
- L4virtio::Svr::Driver_mem_list_t< DATA >, 1848
 - add, 1850
 - find, 1850
 - full, 1851
 - init, 1852
 - load_desc, 1852–1854
 - remove, 1855
- L4virtio::Svr::Driver_mem_region_t< DATA >, 1856
 - contains, 1859
 - Driver_mem_region_t, 1858
 - drv_base, 1860
 - ds, 1860
 - ds_offset, 1860
 - empty, 1861
 - flags, 1861
 - is_writable, 1861
 - local, 1861
 - local_base, 1862
 - size, 1862
- L4virtio::Svr::Request_processor, 1863

- current_flags, [1864](#)
- has_more, [1864](#)
- next, [1865](#)
- start, [1867](#), [1868](#)
- L4virtio::Svr::Virtqueue, [1870](#)
 - consumed, [1873](#)
 - desc, [1874](#)
 - desc_avail, [1874](#)
 - disable_notify, [1875](#)
 - enable_notify, [1875](#)
 - next_avail, [1876](#)
- L4virtio::Svr::Virtqueue::Head_desc, [1877](#)
 - desc, [1877](#)
 - operator Null_ptr_check const *, [1878](#)
 - valid, [1878](#)
- L4virtio::Virtqueue, [1879](#)
 - avail_align, [1882](#)
 - avail_size, [1882](#)
 - desc_align, [1883](#)
 - desc_size, [1883](#)
 - disable, [1884](#)
 - dump, [1885](#)
 - get_avail_idx, [1885](#)
 - get_tail_avail_idx, [1886](#)
 - no_notify_guest, [1886](#)
 - no_notify_host, [1886](#), [1887](#)
 - num, [1887](#)
 - ready, [1888](#)
 - setup, [1889](#)
 - setup_simple, [1890](#)
 - total_size, [1891](#), [1892](#)
 - used_align, [1893](#)
 - used_size, [1893](#)
- L4virtio::Virtqueue::Avail, [1894](#)
- L4virtio::Virtqueue::Avail::Flags, [1896](#)
 - no_irq_bfm_t, [1896](#)
- L4virtio::Virtqueue::Desc, [1897](#)
- L4virtio::Virtqueue::Desc::Flags, [1898](#)
 - indirect_bfm_t, [1899](#)
 - next_bfm_t, [1899](#)
 - write_bfm_t, [1899](#)
- L4virtio::Virtqueue::Used, [1900](#)
- L4virtio::Virtqueue::Used::Flags, [1901](#)
 - no_notify_bfm_t, [1901](#)
- L4virtio::Virtqueue::Used_elem, [1902](#)
 - Used_elem, [1902](#)
- l4virtio_block_config_t, [1903](#)
- l4virtio_block_discard_t, [1904](#)
- l4virtio_block_header_t, [1905](#)
- L4virtio_block_operations
 - L4 VIRTIO Block Device, [415](#)
- L4VIRTIO_BLOCK_S_IOERR
 - L4 VIRTIO Block Device, [416](#)
- L4VIRTIO_BLOCK_S_OK
 - L4 VIRTIO Block Device, [416](#)
- L4VIRTIO_BLOCK_S_UNSUPP
 - L4 VIRTIO Block Device, [416](#)
- L4virtio_block_status
 - L4 VIRTIO Block Device, [416](#)
- L4VIRTIO_BLOCK_T_DISCARD
 - L4 VIRTIO Block Device, [416](#)
- L4VIRTIO_BLOCK_T_FLUSH
 - L4 VIRTIO Block Device, [416](#)
- L4VIRTIO_BLOCK_T_GET_ID
 - L4 VIRTIO Block Device, [416](#)
- L4VIRTIO_BLOCK_T_IN
 - L4 VIRTIO Block Device, [416](#)
- L4VIRTIO_BLOCK_T_OUT
 - L4 VIRTIO Block Device, [416](#)
- L4VIRTIO_BLOCK_T_WRITE_ZEROES
 - L4 VIRTIO Block Device, [416](#)
- L4VIRTIO_CMD_CFG_QUEUE
 - L4 VIRTIO Transport Layer, [420](#)
- L4VIRTIO_CMD_MASK
 - L4 VIRTIO Transport Layer, [420](#)
- L4VIRTIO_CMD_NONE
 - L4 VIRTIO Transport Layer, [420](#)
- L4VIRTIO_CMD_SET_STATUS
 - L4 VIRTIO Transport Layer, [420](#)
- l4virtio_config_hdr_t, [1906](#)
 - status, [1907](#)
- l4virtio_config_queue
 - L4 VIRTIO Transport Layer, [422](#)
- l4virtio_config_queue_t, [1907](#)
 - L4 VIRTIO Transport Layer, [420](#)
- l4virtio_config_queues
 - L4 VIRTIO Transport Layer, [423](#)
- l4virtio_device_config
 - L4 VIRTIO Transport Layer, [423](#)
- l4virtio_device_config_ds
 - L4 VIRTIO Transport Layer, [423](#)
- L4virtio_device_ids
 - L4 VIRTIO Transport Layer, [421](#)
- l4virtio_device_notification_irq
 - L4 VIRTIO Transport Layer, [424](#)
- L4virtio_device_status
 - L4 VIRTIO Transport Layer, [422](#)
- L4virtio_feature_bits
 - L4 VIRTIO Transport Layer, [422](#)
- L4VIRTIO_FEATURE_CMD_CONFIG
 - L4 VIRTIO Transport Layer, [422](#)
- L4VIRTIO_FEATURE_VERSION_1
 - L4 VIRTIO Transport Layer, [422](#)
- L4VIRTIO_ID_9P
 - L4 VIRTIO Transport Layer, [421](#)
- L4VIRTIO_ID_BALLOON
 - L4 VIRTIO Transport Layer, [421](#)
- L4VIRTIO_ID_BLOCK
 - L4 VIRTIO Transport Layer, [421](#)
- L4VIRTIO_ID_CAIF
 - L4 VIRTIO Transport Layer, [421](#)
- L4VIRTIO_ID_CONSOLE
 - L4 VIRTIO Transport Layer, [421](#)
- L4VIRTIO_ID_CRYPT
 - L4 VIRTIO Transport Layer, [421](#)
- L4VIRTIO_ID_GPU

- L4 VIRTIO Transport Layer, [421](#)
- L4VIRTIO_ID_INPUT
 - L4 VIRTIO Transport Layer, [421](#)
- L4VIRTIO_ID_NET
 - L4 VIRTIO Transport Layer, [421](#)
- L4VIRTIO_ID_RNG
 - L4 VIRTIO Transport Layer, [421](#)
- L4VIRTIO_ID_RPMMSG
 - L4 VIRTIO Transport Layer, [421](#)
- L4VIRTIO_ID_RPROC_SERIAL
 - L4 VIRTIO Transport Layer, [421](#)
- L4VIRTIO_ID_SCSI
 - L4 VIRTIO Transport Layer, [421](#)
- L4VIRTIO_ID_SOCK
 - L4 VIRTIO Transport Layer, [421](#)
- L4VIRTIO_ID_VSOCK
 - L4 VIRTIO Transport Layer, [421](#)
- l4virtio_input_absinfo_t, [1909](#)
- l4virtio_input_config_t, [1909](#)
- l4virtio_input_devids_t, [1910](#)
- l4virtio_input_event_t, [1910](#)
- L4VIRTIO_IRQ_STATUS_CONFIG
 - L4 VIRTIO Transport Layer, [421](#)
- L4VIRTIO_IRQ_STATUS_VRING
 - L4 VIRTIO Transport Layer, [421](#)
- l4virtio_net_config_t, [1911](#)
- l4virtio_net_header_t, [1911](#)
- L4VIRTIO_OP_CONFIG_QUEUE
 - L4 VIRTIO Transport Layer, [421](#)
- L4VIRTIO_OP_DEVICE_CONFIG
 - L4 VIRTIO Transport Layer, [421](#)
- L4VIRTIO_OP_GET_DEVICE_IRQ
 - L4 VIRTIO Transport Layer, [421](#)
- L4VIRTIO_OP_REGISTER_DS
 - L4 VIRTIO Transport Layer, [421](#)
- L4VIRTIO_OP_SET_STATUS
 - L4 VIRTIO Transport Layer, [421](#)
- l4virtio_register_ds
 - L4 VIRTIO Transport Layer, [424](#)
- l4virtio_set_status
 - L4 VIRTIO Transport Layer, [425](#)
- L4VIRTIO_STATUS_ACKNOWLEDGE
 - L4 VIRTIO Transport Layer, [422](#)
- L4VIRTIO_STATUS_DEVICE_NEEDS_RESET
 - L4 VIRTIO Transport Layer, [422](#)
- L4VIRTIO_STATUS_DRIVER
 - L4 VIRTIO Transport Layer, [422](#)
- L4VIRTIO_STATUS_DRIVER_OK
 - L4 VIRTIO Transport Layer, [422](#)
- L4VIRTIO_STATUS_FAILED
 - L4 VIRTIO Transport Layer, [422](#)
- L4VIRTIO_STATUS_FEATURES_OK
 - L4 VIRTIO Transport Layer, [422](#)
- length
 - L4::lpc::Varg, [1080](#)
- Libedid_block_size
 - EDID parsing functionality, [391](#)
- libedid_check_header
 - EDID parsing functionality, [391](#)
- libedid_checksum
 - EDID parsing functionality, [392](#)
- Libedid_consts
 - EDID parsing functionality, [391](#)
- libedid_dump
 - EDID parsing functionality, [392](#)
- libedid_dump_standard_timings
 - EDID parsing functionality, [392](#)
- libedid_num_ext_blocks
 - EDID parsing functionality, [393](#)
- libedid_pnp_id
 - EDID parsing functionality, [393](#)
- libedid_prefered_resolution
 - EDID parsing functionality, [393](#)
- libedid_revision
 - EDID parsing functionality, [393](#)
- libedid_version
 - EDID parsing functionality, [394](#)
- Link
 - L4Re::Namespace, [1484](#)
- link
 - L4Re::Vfs::Directory, [1599](#)
- List_alloc
 - cxx::List_alloc, [849](#)
- list_alloc.h
 - l4la_alloc, [2828](#)
 - l4la_avail, [2828](#)
 - l4la_dump, [2828](#)
 - l4la_free, [2828](#)
 - l4la_init, [2829](#)
- load_desc
 - L4virtio::Svr::Driver_mem_list_t< DATA >, [1852](#)–[1854](#)
- local
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1861](#)
- local_base
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1862](#)
- local_id_received
 - L4::lpc::Gen_fpage< T >, [1020](#)
- log
 - L4Re::Env, [1444](#)
- Log interface, [474](#)
 - l4re_log_print, [475](#)
 - l4re_log_print_srv, [475](#)
 - l4re_log_printn, [476](#)
 - l4re_log_printn_srv, [477](#)
- Logging interface, [520](#)
- loop
 - L4::Server< LOOP_HOOKS >, [1232](#)
 - L4Re::Util::Registry_server< LOOP_HOOKS >, [1583](#)
- Low-Level Thread Functions, [633](#)
- Low_mask
 - cxx::Bitfield< T, LSB, MSB >, [762](#)
- lower_bound_node

- cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [800](#)
 - cxx::Bits::Bst< Node, Get_key, Compare >, [817](#)
- Lsb
 - cxx::Bitfield< T, LSB, MSB >, [762](#)
- lseek64
 - L4Re::Vfs::Regular_file, [1623](#)
- Lstr
 - L4::Factory::Lstr, [981](#)
- Mag, the GUI Multiplexer, [71](#)
- main_thread
 - L4Re::Env, [1444](#)
- make_cap
 - L4::lpc, [670](#)
- make_cap_full
 - L4::lpc, [670](#)
- make_cap_rw
 - L4::lpc, [671](#)
- make_cap_rws
 - L4::lpc, [672](#)
- make_ref_cap
 - L4Re Capability API, [518](#)
- make_ref_del_cap
 - L4Re Capability API, [519](#)
- make_shared_cap
 - L4Re, [693](#)
 - L4Re::Util, [705](#)
- make_shared_del_cap
 - L4Re, [694](#)
 - L4Re::Util, [705](#)
- make_unique_cap
 - L4Re, [695](#)
 - L4Re::Util, [705](#)
- make_unique_del_cap
 - L4Re, [695](#)
 - L4Re::Util, [705](#)
- map
 - L4::Task, [1258](#)
 - L4Re::Dataspace, [1421](#)
 - L4Re::Dma_space, [1433](#)
 - L4Re::Util::Dataspace_svr, [1545](#)
- map_hook
 - L4Re::Util::Dataspace_svr, [1546](#)
- map_region
 - L4Re::Dataspace, [1422](#)
- Mask
 - cxx::Bitfield< T, LSB, MSB >, [762](#)
- mask
 - L4::lcu, [992](#)
- Masks
- cxx::Bitfield< T, LSB, MSB >, [762](#)
- max
 - Small C++ Template Library, [560](#)
- max_free_slabs
 - cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, [750](#)
 - cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, [756](#)
- max_queue_size
 - L4virtio::Driver::Device, [1780](#)
- MB_ARD_MEMORY
 - mb_info.h, [2835](#)
- MB_ART_MEMORY
 - mb_info.h, [2835](#)
- mb_info.h
 - l4util_mb_for_each_mmap_entry, [2835](#)
 - L4UTIL_MB_MEMORY, [2835](#)
 - MB_ARD_MEMORY, [2835](#)
 - MB_ART_MEMORY, [2835](#)
- Mem
 - L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, [1294](#)
- mem_alloc
 - L4Re::Env, [1445](#)
- Mem_alloc_flags
 - L4Re::Mem_alloc, [1474](#)
- Mem_desc
 - L4::Kip::Mem_desc, [1155](#)
- Mem_type
 - L4::Kip::Mem_desc, [1155](#)
- Memory allocator, [477](#)
 - l4re_ma_alloc, [478](#)
 - l4re_ma_alloc_align, [479](#)
 - l4re_ma_alloc_align_svr, [480](#)
 - l4re_ma_flags, [478](#)
- Memory descriptors (C version), [185](#)
 - l4_kernel_info_get_mem_desc_end, [187](#)
 - l4_kernel_info_get_mem_desc_is_virtual, [188](#)
 - l4_kernel_info_get_mem_desc_start, [188](#)
 - l4_kernel_info_get_mem_desc_subtype, [188](#)
 - l4_kernel_info_get_mem_desc_type, [188](#)
 - l4_kernel_info_get_num_mem_descs, [189](#)
 - l4_kernel_info_mem_desc_t, [186](#)
 - l4_kernel_info_set_mem_desc, [189](#)
 - l4_mem_archspecific_acpi_nvs, [187](#)
 - l4_mem_archspecific_acpi_tables, [187](#)
 - l4_mem_archspecific_sub_type_common_t, [186](#)
 - l4_mem_info_acpi_rsd, [187](#)
 - l4_mem_info_sub_type_t, [187](#)
 - l4_mem_type_archspecific, [187](#)
 - l4_mem_type_bootloader, [187](#)
 - l4_mem_type_conventional, [187](#)
 - l4_mem_type_dedicated, [187](#)
 - l4_mem_type_info, [187](#)
 - l4_mem_type_reserved, [187](#)
 - l4_mem_type_shared, [187](#)
 - l4_mem_type_t, [187](#)
 - l4_mem_type_undefined, [187](#)
- Memory management - Data Spaces and the Region Map, [22](#)
- Memory operations., [319](#)
 - L4_mem_op_widths, [319](#)
 - l4_mem_read, [320](#)
 - L4_MEM_WIDTH_1BYTE, [319](#)
 - L4_MEM_WIDTH_2BYTE, [319](#)
 - L4_MEM_WIDTH_4BYTE, [319](#)

- l4_mem_write, [320](#)
- Memory related, [321](#)
 - l4_addr_consts_t, [324](#)
 - l4_bytes_to_mwords, [325](#)
 - L4_INVALID_ADDR, [325](#)
 - L4_LOG2_PAGESIZE, [322](#)
 - L4_LOG2_SUPERPAGESIZE, [322](#)
 - L4_PAGEMASK, [323](#)
 - L4_PAGESHIFT, [323](#)
 - l4_round_page, [325](#)
 - l4_round_size, [326](#)
 - L4_SUPERPAGEMASK, [323](#)
 - L4_SUPERPAGESHIFT, [324](#)
 - L4_SUPERPAGESIZE, [324](#)
 - l4_trunc_page, [327](#)
 - l4_trunc_size, [328](#)
- message
 - L4virtio::Svr::Bad_descriptor, [1810](#)
- Message Items, [352](#)
 - L4_ITEM_CONT, [354](#)
 - L4_ITEM_MAP, [353](#)
 - l4_map_control, [354](#)
 - L4_MAP_ITEM_GRANT, [354](#)
 - L4_MAP_ITEM_MAP, [354](#)
 - l4_map_obj_control, [355](#)
 - l4_msg_item_consts_t, [353](#)
 - L4_RCV_ITEM_LOCAL_ID, [354](#)
 - L4_RCV_ITEM_SINGLE_CAP, [354](#)
- Message Registers (MRs), [384](#)
- Message Tag, [356](#)
 - l4_msgtag, [360](#)
 - L4_MSGTAG_ERROR, [358](#)
 - L4_MSGTAG_FLAGS, [358](#)
 - L4_msgtag_flags, [358](#)
 - l4_msgtag_flags, [361](#)
 - l4_msgtag_has_error, [362](#)
 - l4_msgtag_is_exception, [362](#)
 - l4_msgtag_is_io_page_fault, [364](#)
 - l4_msgtag_is_page_fault, [365](#)
 - l4_msgtag_is_preemption, [365](#)
 - l4_msgtag_is_sigma0, [366](#)
 - l4_msgtag_is_sys_exception, [367](#)
 - l4_msgtag_items, [367](#)
 - l4_msgtag_label, [368](#)
 - L4_msgtag_protocol, [358](#)
 - L4_MSGTAG_SCHEDULE, [358](#)
 - l4_msgtag_t, [358](#)
 - L4_MSGTAG_TRANSFER_FPU, [358](#)
 - l4_msgtag_words, [369](#)
 - L4_platform_ctl_proto, [359](#)
 - L4_PROTO_ALLOW_SYSCALL, [359](#)
 - L4_PROTO_DEBUGGER, [359](#)
 - L4_PROTO_DMA_SPACE, [359](#)
 - L4_PROTO_EXCEPTION, [359](#)
 - L4_PROTO_FACTORY, [359](#)
 - L4_PROTO_IO_PAGE_FAULT, [359](#)
 - L4_PROTO_IOMMU, [359](#)
 - L4_PROTO_IRQ, [359](#)
 - L4_PROTO_IRQ_MUX, [359](#)
 - L4_PROTO_IRQ_SENDER, [359](#)
 - L4_PROTO_KOBJECT, [359](#)
 - L4_PROTO_LOG, [359](#)
 - L4_PROTO_META, [359](#)
 - L4_PROTO_NONE, [359](#)
 - L4_PROTO_PAGE_FAULT, [359](#)
 - L4_PROTO_PF_EXCEPTION, [359](#)
 - L4_PROTO_PLATFORM_CTL, [359](#)
 - L4_PROTO_PREEMPTION, [359](#)
 - L4_PROTO_SCHEDULER, [359](#)
 - L4_PROTO_SEMAPHORE, [359](#)
 - L4_PROTO_SIGMA0, [359](#)
 - L4_PROTO_SMCCC, [359](#)
 - L4_PROTO_SYS_EXCEPTION, [359](#)
 - L4_PROTO_TASK, [359](#)
 - L4_PROTO_THREAD, [359](#)
 - L4_PROTO_VM, [359](#)
- min
 - Small C++ Template Library, [560](#)
- mkdir
 - L4Re::Vfs::Directory, [1599](#)
- mmio_read
 - L4Re::Mmio_space, [1479](#)
- mmio_write
 - L4Re::Mmio_space, [1479](#)
- Mode
 - L4Re::Util::Event_t< PAYLOAD >, [1559](#)
- mode
 - L4::Uart, [1333](#)
- Mode_irq
 - L4Re::Util::Event_t< PAYLOAD >, [1559](#)
- Mode_polling
 - L4Re::Util::Event_t< PAYLOAD >, [1559](#)
- modify
 - L4drivers::Register_tmpl< BITS, BLOCK >, [1401](#)
- modify_senders
 - L4::Thread, [1269](#)
- Moe, the Root-Task, [45](#)
- mount
 - L4Re::Vfs::File_system, [1606](#)
 - L4Re::Vfs::Fs, [1610](#)
- move
 - L4::Cap< T >, [929](#)
 - L4::Cap_base, [937](#)
- Mr_bytes
 - L4::lpc::Msg, [675](#)
- Mr_words
 - L4::lpc::Msg, [675](#)
- Msb
 - cxx::Bitfield< T, LSB, MSB >, [762](#)
- msg_add
 - L4::lpc::Msg, [679](#)
- msg_get
 - L4::lpc::Msg, [679](#)
- Msg_ptr
 - L4::lpc::Msg_ptr< T >, [1062](#)
- msg_ptr

- L4::lpc, [672](#)
- msi_info
 - L4::lcu, [993](#)
- N
 - cxx::Bits::Direction, [826](#)
- Name-space API, [520](#)
- Name_max
 - L4Re::Inhibitor, [1464](#)
- Namespace interface, [481](#)
 - l4re_ns_query_srv, [482](#)
 - l4re_ns_query_to_srv, [483](#)
 - l4re_ns_register_flags, [482](#)
 - l4re_ns_register_obj_srv, [484](#)
- Ned, the Init Process, [49](#)
- negotiated_features
 - L4virtio::Svr::Dev_config, [1832](#)
- next
 - L4Re::Event_buffer_t< PAYLOAD >, [1458](#)
 - L4virtio::Svr::Request_processor, [1865](#)
- next_avail
 - L4virtio::Svr::Virtqueue, [1876](#)
- next_bfm_t
 - L4virtio::Virtqueue::Desc::Flags, [1899](#)
- next_block
 - L4virtio::Svr::Block_request< Ds_data >, [1821](#)
- next_device
 - L4vbus::Device, [1691](#)
- next_lock_info
 - L4Re::Inhibitor, [1464](#)
- next_timeout
 - L4::lpc_srv::Timeout_queue, [1121](#)
- no_demand
 - L4::Type_info::Demand, [1291](#)
- No_init
 - L4::Cap_base, [932](#)
- No_init_type
 - L4::Cap_base, [932](#)
- no_irq_bfm_t
 - L4virtio::Virtqueue::Avail::Flags, [1896](#)
- no_notify_bfm_t
 - L4virtio::Virtqueue::Used::Flags, [1901](#)
- no_notify_guest
 - L4virtio::Virtqueue, [1886](#)
- no_notify_host
 - L4virtio::Virtqueue, [1886](#), [1887](#)
- No_sync
 - L4Re::Dma_space, [1432](#)
- None
 - L4::Types::Flags< BITS_ENUM, UNDERLYING >, [1318](#)
 - L4Re::Dma_space, [1432](#)
- None_type
 - L4::Types::Flags< BITS_ENUM, UNDERLYING >, [1318](#)
- Normal
 - L4Re::Dataspace::F, [1425](#)
- NT_ASRS
 - ELF binary format, [615](#)
- NT_AUXV
 - ELF binary format, [615](#)
- NT_FPREGSET
 - ELF binary format, [615](#)
- NT_GWINDOWS
 - ELF binary format, [615](#)
- NT_LWPSINFO
 - ELF binary format, [615](#)
- NT_LWPSTATUS
 - ELF binary format, [615](#)
- NT_PLATFORM
 - ELF binary format, [615](#)
- NT_PRCRED
 - ELF binary format, [615](#)
- NT_PRFPXREG
 - ELF binary format, [615](#)
- NT_PRPSINFO
 - ELF binary format, [615](#)
- NT_PRSTATUS
 - ELF binary format, [615](#)
- NT_PRXREG
 - ELF binary format, [615](#)
- NT_PSINFO
 - ELF binary format, [615](#)
- NT_PSTATUS
 - ELF binary format, [615](#)
- NT_TASKSTRUCT
 - ELF binary format, [615](#)
- NT_UTSNAME
 - ELF binary format, [615](#)
- NT_VERSION
 - ELF binary format, [616](#)
- num
 - L4virtio::Virtqueue, [1887](#)
- num_interfaces
 - L4::Meta, [1184](#)
- NVMe server, [69](#)
- obj_cap
 - L4::Epiface, [961](#)
 - L4::Epiface_t0< RPC_IFACE, BASE >, [968](#)
 - L4::Irqp_t< Derived, BASE, bool >, [1152](#)
- Object Invocation, [328](#)
 - l4_ipc, [332](#)
 - l4_ipc_call, [333](#)
 - l4_ipc_receive, [335](#)
 - l4_ipc_reply_and_wait, [338](#)
 - l4_ipc_send, [339](#)
 - l4_ipc_send_and_wait, [341](#)
 - l4_ipc_sleep, [342](#)
 - l4_ipc_sleep_ms, [343](#)
 - l4_ipc_sleep_us, [344](#)
 - l4_ipc_wait, [345](#)
 - l4_sndfpage_add, [346](#)
 - l4_syscall_flags_t, [331](#)
 - L4_SYSF_CALL, [331](#)
 - L4_SYSF_NONE, [331](#)
 - L4_SYSF_OPEN_WAIT, [331](#)
 - L4_SYSF_RECV, [331](#)

- L4_SYSF_REPLY, [331](#)
- L4_SYSF_REPLY_AND_WAIT, [331](#)
- L4_SYSF_SEND, [331](#)
- L4_SYSF_SEND_AND_WAIT, [331](#)
- L4_SYSF_WAIT, [331](#)
- Object_registry
 - L4Re::Util::Object_registry, [1572](#)
- object_size
 - cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, [750](#)
 - cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, [756](#)
- objects_per_slab
 - cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, [750](#)
 - cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, [756](#)
- offset
 - l4_sched_cpu_set_t, [1365](#)
- operator l4_msgtag_t
 - L4::Factory::S, [984](#)
- operator new
 - Small C++ Template Library, [561](#)
- operator Null_ptr_check const *
 - L4virtio::Svr::Virtqueue::Head_desc, [1878](#)
- operator value_type
 - L4drivers::Ro_register_tmpl< BITS, BLOCK >, [1406](#)
- operator!
 - cxx::Bits::Direction, [826](#)
- operator!=
 - L4vbus::Device, [1692](#)
- operator<<
 - ipc_stream, [2107–2109](#)
- operator>>
 - ipc_stream, [2109, 2111–2114](#)
- operator()
 - cxx::Pair_first_compare< Cmp, Typ >, [864](#)
- operator->
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node, [806](#)
- operator=
 - L4drivers::Register_tmpl< BITS, BLOCK >, [1402](#)
 - L4Re::Rm::Unique_region< T >, [1519](#)
- operator==
 - L4Re::Video::Color_component, [1629](#)
 - L4Re::Video::Pixel_info, [1645](#)
 - L4vbus::Device, [1692](#)
- operator[]
 - cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >, [734](#)
 - cxx::Bitmap_base, [784, 785](#)
 - cxx::List< D, Alloc >, [847](#)
 - L4drivers::Register_block< MAX_BITS, BLOCK >, [1392](#)
 - L4drivers::Ro_register_block< MAX_BITS, BLOCK >, [1404](#)
- operator*
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node, [806](#)
- outchar
 - kdebug.h, [2681](#)
- outdec
 - kdebug.h, [2682](#)
- outhex12
 - kdebug.h, [2683](#)
- outhex16
 - kdebug.h, [2683](#)
- outhex20
 - kdebug.h, [2684](#)
- outhex32
 - kdebug.h, [2684](#)
- outhex64
 - kdebug.h, [2685](#)
- outhex8
 - kdebug.h, [2686](#)
- outnstring
 - kdebug.h, [2686](#)
- outstring
 - kdebug.h, [2687](#)
- outumword
 - kdebug.h, [2688](#)
- Overview, [1](#)
- Overwrite
 - L4Re::Namespace, [1484](#)
- p_flags
 - Elf32_Phdr, [899](#)
 - Elf64_Phdr, [909](#)
- p_type
 - Elf32_Phdr, [899](#)
 - Elf64_Phdr, [909](#)
- padding
 - L4Re::Video::Pixel_info, [1645](#)
- page_fault
 - L4::Pager, [1190](#)
- page_shift
 - L4Re::Util::Dataspace_svr, [1546](#)
- Pager
 - L4Re::Rm::F, [1514](#)
- pager
 - L4::Thread::Attr, [1280](#)
- Pair
 - cxx::Pair< First, Second >, [862](#)
- Pair_first_compare
 - cxx::Pair_first_compare< Cmp, Typ >, [863](#)
- parent
 - L4Re::Env, [1445, 1446](#)
- Parent API, [521](#)
- parse_cmdline
 - Comfortable Command Line Parsing, [597](#)
- Partly_resolved
 - L4Re::Namespace, [1483](#)
- PF_ARM_SB
 - ELF binary format, [606](#)
- PF_MASKOS
 - ELF binary format, [617](#)

- PF_MASKPROC
 - ELF binary format, [617](#)
- PF_R
 - ELF binary format, [617](#)
- PF_W
 - ELF binary format, [617](#)
- PF_X
 - ELF binary format, [617](#)
- Phys_space
 - L4Re::Dma_space, [1432](#)
- pin
 - L4vbus::Gpio_module, [1697](#)
 - L4vbus::Gpio_pin, [1706](#)
- Pinned
 - L4Re::Mem_alloc, [1475](#)
- Pixel_info
 - L4Re::Video::Pixel_info, [1641](#)
- pkg/drivers-frst/include/ARCH-amd64/asm_access.h, [1913](#)
- pkg/drivers-frst/include/ARCH-arm/asm_access.h, [1913](#)
- pkg/drivers-frst/include/ARCH-arm64/asm_access.h, [1914](#)
- pkg/drivers-frst/include/ARCH-mips/asm_access.h, [1915](#)
- pkg/drivers-frst/include/ARCH-ppc32/asm_access.h, [1915](#)
- pkg/drivers-frst/include/ARCH-sparc/asm_access.h, [1916](#)
- pkg/drivers-frst/include/ARCH-x86/asm_access.h, [1916](#)
- pkg/drivers-frst/include/asm_access_gen.h, [1917](#)
- pkg/drivers-frst/include/hw_mmio_register_block, [1917](#)
- pkg/drivers-frst/include/hw_register_block, [1918](#)
- pkg/drivers-frst/include/io_regblock.h, [1923](#)
- pkg/drivers-frst/include/io_regblock_port.h, [1926](#)
- pkg/drivers-frst/include/Makefile, [1940](#)
- pkg/drivers-frst/include/poll_timeout_counter.h, [1926](#)
- pkg/drivers-frst/uart/include/Makefile, [1940](#)
- pkg/drivers-frst/uart/include/uart_16550.h, [1927](#)
- pkg/drivers-frst/uart/include/uart_16550_dw.h, [1928](#)
- pkg/drivers-frst/uart/include/uart_base.h, [1929](#)
- pkg/drivers-frst/uart/include/uart_cadence.h, [1930](#)
- pkg/drivers-frst/uart/include/uart_dcc-v6.h, [1930](#)
- pkg/drivers-frst/uart/include/uart_dm.h, [1931](#)
- pkg/drivers-frst/uart/include/uart_dummy.h, [1931](#)
- pkg/drivers-frst/uart/include/uart_geni.h, [1932](#)
- pkg/drivers-frst/uart/include/uart_imx.h, [1932](#)
- pkg/drivers-frst/uart/include/uart_leon3.h, [1933](#)
- pkg/drivers-frst/uart/include/uart_linflex.h, [1934](#)
- pkg/drivers-frst/uart/include/uart_lpuart.h, [1934](#)
- pkg/drivers-frst/uart/include/uart_mvebu.h, [1935](#)
- pkg/drivers-frst/uart/include/uart_of.h, [1935](#)
- pkg/drivers-frst/uart/include/uart_omap35x.h, [1936](#)
- pkg/drivers-frst/uart/include/uart_pl011.h, [1936](#)
- pkg/drivers-frst/uart/include/uart_s3c2410.h, [1937](#)
- pkg/drivers-frst/uart/include/uart_sa1000.h, [1938](#)
- pkg/drivers-frst/uart/include/uart_sh.h, [1938](#)
- pkg/l4re-core/ned/lib/include/cmd_control, [1939](#)
- pkg/l4re-core/ned/lib/include/Makefile, [1940](#)
- Platform Control C API, [236](#)
 - l4_platform_ctl_cpu_allow_shutdown, [237](#)
 - l4_platform_ctl_cpu_disable, [238](#)
 - l4_platform_ctl_cpu_enable, [238](#)
 - l4_platform_ctl_system_shutdown, [239](#)
 - l4_platform_ctl_system_suspend, [240](#)
- pm_resume
 - L4vbus::Pm< DEC >, [1727](#)
- pm_suspend
 - L4vbus::Pm< DEC >, [1727](#)
- Poll_timeout_counter
 - L4::Poll_timeout_counter, [1197](#)
- Poll_timeout_kipclock
 - L4::Poll_timeout_kipclock, [1200](#)
- pop_front
 - cxx::Bits::Smart_ptr_list< ITEM >, [830](#)
 - cxx::H_list< T, POLICY >, [838](#)
 - cxx::S_list< T, POLICY >, [872](#)
- port_io.h
 - l4util_ioport_map, [1996](#)
- Ports
 - L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, [1294](#)
- print
 - L4Re::Log, [1470](#)
- println
 - L4Re::Log, [1470](#)
- process
 - L4Re::Util::Event_buffer_consumer_t< PAYLOAD >, [1551](#)
- process_request
 - L4virtio::Driver::Block_device, [1767](#)
 - L4virtio::Svr::Block_dev_base< Ds_data >, [1817](#)
- process_used_queue
 - L4virtio::Driver::Block_device, [1768](#)
- Producer, [540](#), [551](#)
 - l4shmc_chunk_ready, [541](#)
 - l4shmc_chunk_ready_sig, [541](#)
 - l4shmc_chunk_try_to_take, [541](#)
 - l4shmc_chunk_try_to_take_for_overwriting, [543](#)
 - l4shmc_chunk_try_to_take_for_writing, [543](#)
 - l4shmc_is_chunk_clear, [544](#)
 - l4shmc_trigger, [552](#)
- prog.mk - Application Role, [30](#)
- Program Input and Output, [23](#)
- Programming for L4Re, [9](#)
- PROTO_ANY
 - L4, [659](#)
- proto_dispatch
 - L4::Kobject_typeid< T >, [1177](#)
 - L4::Server_object_t< IFACE, BASE >, [1241](#)
- PROTO_EMPTY
 - L4, [659](#)
- protocols.h
 - L4RE_PROTO_DATASPACE, [2398](#)
 - L4RE_PROTO_DEBUG, [2398](#)
 - L4RE_PROTO_DMA_SPACE, [2398](#)
 - L4RE_PROTO_EVENT, [2398](#)

- L4RE_PROTO_GOOS, 2398
- L4RE_PROTO_INHIBITOR, 2398
- L4RE_PROTO_MMIO_SPACE, 2398
- L4RE_PROTO_NAMESPACE, 2398
- L4RE_PROTO_PARENT, 2398
- L4RE_PROTO_RM, 2398
- L4RE_PROTO_RSVD_1, 2398
- L4re_protocols, 2398
- pSLIM_BMAP_START_LSB
 - bitmap.h, 2298
- PT_DYNAMIC
 - ELF binary format, 617
- PT_GNU_EH_FRAME
 - ELF binary format, 617
- PT_GNU_RELRO
 - ELF binary format, 617
- PT_GNU_STACK
 - ELF binary format, 617
- PT_HIOS
 - ELF binary format, 617
- PT_HIPROC
 - ELF binary format, 617
- PT_INTERP
 - ELF binary format, 617
- PT_L4_AUX
 - ELF binary format, 617
- PT_L4_KIP
 - ELF binary format, 617
- PT_L4_STACK
 - ELF binary format, 617
- PT_LOAD
 - ELF binary format, 617
- PT_LOOS
 - ELF binary format, 617
- PT_LOPROC
 - ELF binary format, 617
- PT_NOTE
 - ELF binary format, 617
- PT_NULL
 - ELF binary format, 617
- PT_NUM
 - ELF binary format, 617
- PT_PHDR
 - ELF binary format, 617
- PT_SHLIB
 - ELF binary format, 617
- PT_TLS
 - ELF binary format, 617
- Pthread Support, 24
- ptr
 - cxx::Ref_ptr< T, CNT >, 869
- push_back
 - cxx::List_item, 853
- push_front
 - cxx::List_item, 854
- put
 - L4::Factory::S, 985, 986
 - L4::lpc::lostream, 1029
 - L4::lpc::Ostream, 1067, 1068
 - L4Re::Event_buffer_t< PAYLOAD >, 1458
- qconfig
 - L4virtio::Svr::Dev_config, 1833
- query
 - L4Re::Namespace, 1484
- query_log_name
 - L4::Debugger, 949
- query_log_typeid
 - L4::Debugger, 950
- Query_result_flags
 - L4Re::Namespace, 1483
- Query_timeout
 - L4Re::Namespace, 1483
- R
 - cxx::Bits::Direction, 826
 - L4Re::Dataspace::F, 1425
 - L4Re::Rm::F, 1513
- r
 - L4drivers::Register_block< MAX_BITS, BLOCK >, 1394
 - L4drivers::Ro_register_block< MAX_BITS, BLOCK >, 1404
 - L4Re::Video::Pixel_info, 1646
 - L4vcpu::Vcpu, 1750, 1751
- R_386_32
 - ELF binary format, 618
- R_386_COPY
 - ELF binary format, 618
- R_386_GLOB_DAT
 - ELF binary format, 618
- R_386_GOT32
 - ELF binary format, 618
- R_386_GOTOFF
 - ELF binary format, 618
- R_386_GOTPC
 - ELF binary format, 618
- R_386_JMP_SLOT
 - ELF binary format, 618
- R_386_NONE
 - ELF binary format, 618
- R_386_NUM
 - ELF binary format, 618
- R_386_PC32
 - ELF binary format, 618
- R_386_PLT32
 - ELF binary format, 618
- R_386_RELATIVE
 - ELF binary format, 618
- R_386_TLS_DTPMOD32
 - ELF binary format, 618
- R_386_TLS_DTPOFF32
 - ELF binary format, 618
- R_386_TLS_GD
 - ELF binary format, 618
- R_386_TLS_GD_32
 - ELF binary format, 618

- R_386_TLS_GD_CALL
 - ELF binary format, [618](#)
- R_386_TLS_GD_POP
 - ELF binary format, [618](#)
- R_386_TLS_GD_PUSH
 - ELF binary format, [618](#)
- R_386_TLS_GOTIE
 - ELF binary format, [618](#)
- R_386_TLS_IE
 - ELF binary format, [618](#)
- R_386_TLS_IE_32
 - ELF binary format, [618](#)
- R_386_TLS_LDM
 - ELF binary format, [618](#)
- R_386_TLS_LDM_32
 - ELF binary format, [618](#)
- R_386_TLS_LDM_CALL
 - ELF binary format, [618](#)
- R_386_TLS_LDM_POP
 - ELF binary format, [618](#)
- R_386_TLS_LDM_PUSH
 - ELF binary format, [618](#)
- R_386_TLS_LDO_32
 - ELF binary format, [618](#)
- R_386_TLS_LE
 - ELF binary format, [618](#)
- R_386_TLS_LE_32
 - ELF binary format, [618](#)
- R_386_TLS_TPOFF
 - ELF binary format, [618](#)
- R_386_TLS_TPOFF32
 - ELF binary format, [618](#)
- R_AARCH64_NONE
 - ELF binary format, [618](#)
- R_ARM_ABS12
 - ELF binary format, [619](#)
- R_ARM_ABS16
 - ELF binary format, [619](#)
- R_ARM_ABS32
 - ELF binary format, [619](#)
- R_ARM_ABS8
 - ELF binary format, [619](#)
- R_ARM_COPY
 - ELF binary format, [619](#)
- R_ARM_GLOB_DAT
 - ELF binary format, [619](#)
- R_ARM_GOT32
 - ELF binary format, [619](#)
- R_ARM_GOTOFF
 - ELF binary format, [619](#)
- R_ARM_GOTPC
 - ELF binary format, [619](#)
- R_ARM_JUMP_SLOT
 - ELF binary format, [619](#)
- R_ARM_NONE
 - ELF binary format, [619](#)
- R_ARM_NUM
 - ELF binary format, [619](#)
- R_ARM_PC24
 - ELF binary format, [619](#)
- R_ARM_PLT32
 - ELF binary format, [619](#)
- R_ARM_REL32
 - ELF binary format, [619](#)
- R_ARM_RELATIVE
 - ELF binary format, [619](#)
- R_ARM_THM_PC11
 - ELF binary format, [619](#)
- R_ARM_THM_PC9
 - ELF binary format, [619](#)
- R_X86_64_16
 - ELF binary format, [620](#)
- R_X86_64_32
 - ELF binary format, [619](#)
- R_X86_64_32S
 - ELF binary format, [620](#)
- R_X86_64_64
 - ELF binary format, [619](#)
- R_X86_64_8
 - ELF binary format, [620](#)
- R_X86_64_COPY
 - ELF binary format, [619](#)
- R_X86_64_DTPMOD64
 - ELF binary format, [620](#)
- R_X86_64_DTPOFF32
 - ELF binary format, [620](#)
- R_X86_64_DTPOFF64
 - ELF binary format, [620](#)
- R_X86_64_GLOB_DAT
 - ELF binary format, [619](#)
- R_X86_64_GOT32
 - ELF binary format, [619](#)
- R_X86_64_GOTPCREL
 - ELF binary format, [619](#)
- R_X86_64_GOTTPOFF
 - ELF binary format, [620](#)
- R_X86_64_JUMP_SLOT
 - ELF binary format, [619](#)
- R_X86_64_NONE
 - ELF binary format, [619](#)
- R_X86_64_PC16
 - ELF binary format, [620](#)
- R_X86_64_PC32
 - ELF binary format, [619](#)
- R_X86_64_PC8
 - ELF binary format, [620](#)
- R_X86_64_PLT32
 - ELF binary format, [619](#)
- R_X86_64_RELATIVE
 - ELF binary format, [619](#)
- R_X86_64_TLSGD
 - ELF binary format, [620](#)
- R_X86_64_TLSLD
 - ELF binary format, [620](#)
- R_X86_64_TPOFF32
 - ELF binary format, [620](#)

- R_X86_64_TPOFF64
 - ELF binary format, [620](#)
- RAM configuration, [68](#)
- Random number support, [633](#)
 - l4util_rand, [634](#)
 - l4util_srand, [634](#)
- rate
 - L4::Uart, [1333](#)
- rbegin
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [801](#)
 - cxx::Bits::Bst< Node, Get_key, Compare >, [818](#), [819](#)
- rcv_cap
 - L4::lpc_svr::Server_iface, [1113](#), [1114](#)
- rcv_endpoint.h
 - L4_RCV_EP_BIND_OP, [2720](#)
 - L4_rcv_ep_ops, [2720](#)
- read
 - L4::lpc, [673](#)
 - L4::Vcon, [1342](#)
 - L4drivers::Ro_register_tmpl< BITS, BLOCK >, [1406](#)
- read_with_flags
 - L4::Vcon, [1343](#)
- readv
 - L4Re::Vfs::Regular_file, [1623](#)
- ready
 - L4virtio::Virtqueue, [1888](#)
- realloc_rcv_cap
 - L4::lpc_svr::Server_iface, [1115](#)
 - L4Re::Util::Br_manager, [1527](#)
- Realtime API, [370](#)
- receive
 - L4::lpc::Istream, [1038](#)
 - L4::Irq, [1135](#)
- Receiver, [525](#)
- Ref_ptr
 - cxx::Ref_ptr< T, CNT >, [868](#)
- refresh
 - L4Re::Util::Video::Goos_svr, [1589](#)
 - L4Re::Video::View, [1649](#)
- Region
 - L4Re::Rm, [1503](#)
- Region map API, [522](#)
- Region map interface, [485](#)
 - l4re_rm_attach, [487](#)
 - l4re_rm_attach_srv, [488](#)
 - L4RE_RM_CACHING_SHIFT, [486](#)
 - l4re_rm_detach, [488](#)
 - l4re_rm_detach_ds, [490](#)
 - l4re_rm_detach_ds_unmap, [491](#)
 - l4re_rm_detach_srv, [492](#)
 - l4re_rm_detach_unmap, [492](#)
 - L4RE_RM_F_ATTACH_FLAGS, [486](#)
 - L4RE_RM_F_CACHE_BUFFERED, [486](#)
 - L4RE_RM_F_CACHE_NORMAL, [486](#)
 - L4RE_RM_F_CACHE_UNCACHED, [486](#)
 - L4RE_RM_F_CACHING, [486](#)
 - L4RE_RM_F_EAGER_MAP, [486](#)
 - L4RE_RM_F_IN_AREA, [486](#)
 - L4RE_RM_F_NO_ALIAS, [486](#)
 - L4RE_RM_F_PAGER, [486](#)
 - L4RE_RM_F_R, [486](#)
 - L4RE_RM_F_RESERVED, [486](#)
 - L4RE_RM_F_SEARCH_ADDR, [486](#)
 - l4re_rm_find, [493](#)
 - l4re_rm_find_srv, [494](#)
 - l4re_rm_flags_values, [486](#)
 - l4re_rm_free_area, [495](#)
 - l4re_rm_free_area_srv, [496](#)
 - L4RE_RM_REGION_FLAGS, [486](#)
 - l4re_rm_reserve_area, [496](#)
 - l4re_rm_reserve_area_srv, [497](#)
 - l4re_rm_show_lists, [498](#)
- Region_flag_shifts
 - L4Re::Rm, [1504](#)
- Region_flags
 - L4Re::Rm::F, [1513](#)
- Region_flags_mask
 - L4Re::Rm::F, [1514](#)
- register_del_irq
 - L4::Thread, [1270](#)
- register_driver_irq
 - L4virtio::Svr::Device_t< DATA >, [1845](#)
- register_ds
 - L4virtio::Device, [1760](#)
 - L4virtio::Driver::Device, [1780](#)
- Register_flags
 - L4Re::Namespace, [1483](#)
- register_irq_obj
 - L4::Registry_iface, [1209](#)
 - L4Re::Util::Object_registry, [1573](#)
- register_obj
 - L4::Registry_iface, [1209](#), [1210](#)
 - L4Re::Namespace, [1485](#)
 - L4Re::Util::Object_registry, [1573](#), [1574](#)
- Registry_server
 - L4Re::Util::Registry_server< LOOP_HOOKS >, [1582](#), [1583](#)
- release
 - cxx::Ref_ptr< T, CNT >, [869](#)
 - L4Re::Inhibitor, [1465](#)
 - L4Re::Rm::Unique_region< T >, [1520](#)
 - L4Re::Util::Counting_cap_alloc< COUNTERTYPE >, [1541](#)
 - L4Re::Util::Dataspace_svr, [1547](#)
- release_cap
 - L4::Task, [1259](#)
- release_ioport
 - L4vbus::Vbus, [1737](#)
- remove
 - cxx::Avl_tree< Node, Get_key, Compare >, [745](#)
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [802](#)
 - cxx::H_list< T, POLICY >, [839](#)

- cxx::List_item, [855](#)
 - L4::lpc_svr::Timeout_queue, [1122](#)
 - L4virtio::Svr::Driver_mem_list_t< DATA >, [1855](#)
- remove_all
 - cxx::Bits::Bst< Node, Get_key, Compare >, [820](#)
- remove_timeout
 - L4::lpc_svr::Server_iface, [1115](#)
 - L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >, [1129](#)
- remove_tree
 - cxx::Bits::Bst< Node, Get_key, Compare >, [821](#)
- rename
 - L4Re::Vfs::Directory, [1600](#)
- rend
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [803](#)
 - cxx::Bits::Bst< Node, Get_key, Compare >, [822](#)
- replace
 - cxx::H_list< T, POLICY >, [839](#)
- reply_and_wait
 - L4::lpc::lostream, [1029](#), [1030](#)
- Reply_compound
 - Server-Side IPC framework, [527](#)
- Reply_mode
 - Server-Side IPC framework, [526](#)
- Reply_separate
 - Server-Side IPC framework, [527](#)
- request_ioport
 - L4vbus::Vbus, [1737](#)
- reserve_area
 - L4Re::Rm, [1510](#), [1511](#)
- Reserved
 - L4::Kip::Mem_desc, [1155](#)
 - L4Re::Rm::F, [1514](#)
- reset
 - L4::lpc::lostream, [1031](#)
 - L4::lpc::lstream, [1039](#)
 - L4Re::Rm::Unique_region< T >, [1520](#)
- reset_cmd
 - L4virtio::Svr::Dev_config, [1834](#)
- reset_queue
 - L4virtio::Svr::Dev_config, [1834](#)
- reset_queue_config
 - L4virtio::Svr::Device_t< DATA >, [1846](#)
- Rights_mask
 - L4::lpc::Cap< T >, [1017](#)
 - L4Re::Dataspace::F, [1425](#)
 - L4Re::Rm::F, [1513](#)
- ringbuf.h
 - l4shmc_rb_attach_receiver, [2484](#)
 - l4shmc_rb_attach_sender, [2485](#)
 - l4shmc_rb_deinit_buffer, [2485](#)
 - l4shmc_rb_init_buffer, [2485](#)
 - l4shmc_rb_init_receiver, [2486](#)
 - l4shmc_rb_receiver_copy_out, [2487](#)
 - l4shmc_rb_receiver_notify_done, [2487](#)
 - l4shmc_rb_receiver_read_next_size, [2487](#)
 - l4shmc_rb_receiver_wait_for_data, [2487](#)
 - l4shmc_rb_sender_alloc_packet, [2488](#)
 - l4shmc_rb_sender_commit_packet, [2488](#)
 - l4shmc_rb_sender_next_copy_in, [2489](#)
 - l4shmc_rb_sender_put_data, [2489](#)
 - L4SHMC_RINGBUF_DATA, [2483](#)
 - L4SHMC_RINGBUF_DATA_SIZE, [2484](#)
 - L4SHMC_RINGBUF_HEAD, [2484](#)
- rm
 - L4Re::Env, [1446](#)
- rmdir
 - L4Re::Vfs::Directory, [1600](#)
- Ro
 - L4Re::Dataspace::F, [1425](#)
 - L4Re::Namespace, [1483](#)
- root
 - L4vbus::Vbus, [1738](#)
- round_order
 - L4, [663](#)
- Rs
 - L4Re::Namespace, [1484](#)
- run_thread
 - L4::Scheduler, [1221](#)
- running
 - L4virtio::Svr::Dev_status, [1840](#)
- Runtime_error
 - L4::Runtime_error, [1214](#)
- RW
 - L4Re::Dataspace::F, [1425](#)
 - L4Re::Rm::F, [1513](#)
- Rw
 - L4Re::Namespace, [1484](#)
- Rws
 - L4Re::Namespace, [1484](#)
- RWX
 - L4Re::Dataspace::F, [1425](#)
 - L4Re::Rm::F, [1514](#)
- RX
 - L4Re::Dataspace::F, [1425](#)
 - L4Re::Rm::F, [1513](#)
- rx_pkt
 - L4virtio::Driver::Virtio_net_device, [1789](#)
- rx_queue_size
 - L4virtio::Driver::Virtio_net_device, [1790](#)
- S
 - L4::Factory::S, [984](#)
- saved_state
 - L4vcpu::Vcpu, [1751](#)
- scan_zero
 - cxx::Bitmap< BITS >, [779](#)
 - cxx::Bitmap_base, [785](#)
- Scheduler, [241](#)
 - l4_sched_cpu_set, [243](#)
 - l4_sched_param, [244](#)
 - L4_SCHEDULER_CLASS_FIXED_PRIO, [243](#)
 - L4_SCHEDULER_CLASS_WFQ, [243](#)
 - L4_scheduler_classes, [242](#)
 - l4_scheduler_idle_time, [244](#)
 - L4_SCHEDULER_IDLE_TIME_OP, [243](#)

- l4_scheduler_info, [245](#)
- L4_SCHEDULER_INFO_OP, [243](#)
- l4_scheduler_info_with_classes, [246](#)
- l4_scheduler_is_online, [247](#)
- L4_scheduler_ops, [243](#)
- l4_scheduler_run_thread, [247](#)
- L4_SCHEDULER_RUN_THREAD_OP, [243](#)
- scheduler
 - L4Re::Env, [1447](#)
- screen_info
 - L4Re::Util::Video::Goos_svr, [1589](#)
- Search_addr
 - L4Re::Rm::F, [1513](#)
- segment.h
 - fiasco_amd64_segment_info, [1976](#)
 - fiasco_amd64_set_fs, [1977](#), [1981](#)
 - fiasco_amd64_set_segment_base, [1978](#), [1982](#)
 - L4_AMD64_SEGMENT_FS, [1975](#)
 - L4_AMD64_SEGMENT_GS, [1975](#)
 - L4_sys_segment, [1975](#)
 - L4_task_ldt_x86_consts, [1975](#), [1985](#)
 - L4_TASK_LDT_X86_ENTRY_SIZE, [1976](#), [1986](#)
 - L4_TASK_LDT_X86_MAX_ENTRIES, [1976](#), [1986](#)
- send
 - L4::lpc::Ostream, [1068](#)
 - L4::Vcon, [1343](#)
 - L4virtio::Driver::Device, [1781](#)
- send_and_wait
 - L4virtio::Driver::Device, [1782](#)
- send_request
 - L4virtio::Driver::Block_device, [1768](#)
- Sender, [525](#)
- Server
 - L4::Server< LOOP_HOOKS >, [1231](#)
- Server-Side IPC framework, [525](#)
 - Reply_compound, [527](#)
 - Reply_mode, [526](#)
 - Reply_separate, [527](#)
- server_iface
 - L4::Epiface, [961](#)
- set
 - cxx::Bitfield< T, LSB, MSB >, [764](#)
 - L4::Kip::Mem_desc, [1160](#)
 - L4::Poll_timeout_counter, [1198](#)
 - L4::Poll_timeout_kipclock, [1200](#)
 - l4_sched_cpu_set_t, [1365](#)
 - L4drivers::Register_tmpl< BITS, BLOCK >, [1402](#)
 - L4Re::Video::Color_component, [1629](#)
 - L4vbus::Gpio_module, [1698](#)
 - L4vbus::Gpio_pin, [1706](#)
 - L4vcpu::State, [1742](#)
 - L4virtio::Svr::Data_buffer, [1824](#)
- set_attr
 - L4::Vcon, [1344](#)
- set_bit
 - cxx::Bitmap_base, [786](#)
- set_blk_size
 - L4virtio::Svr::Block_dev_base< Ds_data >, [1817](#)
- set_config_wce
 - L4virtio::Svr::Block_dev_base< Ds_data >, [1818](#)
- set_device_needs_reset
 - L4virtio::Svr::Dev_config, [1835](#)
- set_dirty
 - cxx::Bitfield< T, LSB, MSB >, [764](#)
- set_discard
 - L4virtio::Svr::Block_dev_base< Ds_data >, [1818](#)
- set_fd
 - L4Re::Vfs::Fs, [1610](#)
- set_info
 - L4Re::Video::View, [1650](#)
- set_lock
 - L4Re::Vfs::Regular_file, [1624](#)
- set_mode
 - L4::lcu, [993](#)
- set_object_name
 - L4::Debugger, [951](#)
- set_raw
 - l4_vcon_attr_t, [1374](#)
- set_rcv_cap_flags
 - L4Re::Util::Br_manager, [1528](#)
- set_server
 - L4::Epiface, [962](#)
- set_size_max
 - L4virtio::Svr::Block_dev_base< Ds_data >, [1818](#)
- set_status
 - L4virtio::Device, [1761](#)
 - L4virtio::Svr::Dev_config, [1836](#)
- set_status_flags
 - L4Re::Vfs::Generic_file, [1613](#)
- set_topology
 - L4virtio::Svr::Block_dev_base< Ds_data >, [1819](#)
- set_unshifted
 - cxx::Bitfield< T, LSB, MSB >, [765](#)
- set_unshifted_dirty
 - cxx::Bitfield< T, LSB, MSB >, [766](#)
- set_viewport
 - L4Re::Video::View, [1650](#)
- set_write_zeroes
 - L4virtio::Svr::Block_dev_base< Ds_data >, [1819](#)
- setup
 - L4Re::Util::Counting_cap_alloc< COUNTERTYPE >, [1542](#)
 - L4vbus::Gpio_module, [1699](#)
 - L4vbus::Gpio_pin, [1707](#)
 - L4virtio::Virtqueue, [1889](#)
- setup_device
 - L4virtio::Driver::Block_device, [1769](#)
 - L4virtio::Driver::Virtio_net_device, [1790](#)
- setup_queue
 - L4virtio::Svr::Device_t< DATA >, [1846](#)
- setup_simple
 - L4virtio::Virtqueue, [1890](#)
- sh_flags
 - Elf32_Shdr, [902](#)
 - Elf64_Shdr, [912](#)
- sh_type

- Elf32_Shdr, [902](#)
- Elf64_Shdr, [912](#)
- Shared
 - L4::Kip::Mem_desc, [1155](#)
- Shared Memory Library, [527](#)
 - l4shmc_area_overhead, [528](#)
 - l4shmc_area_size, [528](#)
 - l4shmc_area_size_free, [529](#)
 - l4shmc_attach, [529](#)
 - l4shmc_chunk_overhead, [530](#)
 - l4shmc_connect_chunk_signal, [530](#)
 - l4shmc_create, [530](#)
 - l4shmc_get_client_nr, [531](#)
 - l4shmc_get_initialized_clients, [531](#)
 - l4shmc_mark_client_initialized, [532](#)
- Shared_cap
 - L4Re, [685](#)
 - L4Re::Util, [700](#)
- shared_cap
 - L4Re, [685](#)
 - L4Re::Util, [700](#)
- Shared_del_cap
 - L4Re, [685](#)
 - L4Re::Util, [701](#)
- shared_del_cap
 - L4Re, [686](#)
 - L4Re::Util, [701](#)
- SHF_ALLOC
 - ELF binary format, [620](#)
- SHF_ARM_COMDEF
 - ELF binary format, [620](#)
- SHF_ARM_ENTRYSECT
 - ELF binary format, [620](#)
- SHF_EXECINSTR
 - ELF binary format, [620](#)
- SHF_GROUP
 - ELF binary format, [621](#)
- SHF_INFO_LINK
 - ELF binary format, [620](#)
- SHF_LINK_ORDER
 - ELF binary format, [620](#)
- SHF_MASKOS
 - ELF binary format, [621](#)
- SHF_MASKPROC
 - ELF binary format, [621](#)
- SHF_MERGE
 - ELF binary format, [620](#)
- SHF_OS_NONCONFORMING
 - ELF binary format, [620](#)
- SHF_STRINGS
 - ELF binary format, [620](#)
- SHF_TLS
 - ELF binary format, [621](#)
- SHF_WRITE
 - ELF binary format, [620](#)
- shift
 - L4Re::Video::Color_component, [1629](#)
- Shift_type
 - cxx::Bitfield< T, LSB, MSB >, [761](#)
- SHN_ABS
 - ELF binary format, [621](#)
- SHN_COMMON
 - ELF binary format, [621](#)
- SHN_HIPROC
 - ELF binary format, [621](#)
- SHN_HIRESERVE
 - ELF binary format, [621](#)
- SHN_LOPROC
 - ELF binary format, [621](#)
- SHN_LORESERVE
 - ELF binary format, [621](#)
- SHN_UNDEF
 - ELF binary format, [621](#)
- SHT_DYNAMIC
 - ELF binary format, [621](#)
- SHT_DYNSYM
 - ELF binary format, [622](#)
- SHT_FINI_ARRAY
 - ELF binary format, [622](#)
- SHT_GROUP
 - ELF binary format, [622](#)
- SHT_HASH
 - ELF binary format, [621](#)
- SHT_HIOS
 - ELF binary format, [622](#)
- SHT_HIPROC
 - ELF binary format, [622](#)
- SHT_HIUSER
 - ELF binary format, [622](#)
- SHT_INIT_ARRAY
 - ELF binary format, [622](#)
- SHT_LOOS
 - ELF binary format, [622](#)
- SHT_LOPROC
 - ELF binary format, [622](#)
- SHT_LOUSER
 - ELF binary format, [622](#)
- SHT_NOBITS
 - ELF binary format, [621](#)
- SHT_NOTE
 - ELF binary format, [621](#)
- SHT_NULL
 - ELF binary format, [621](#)
- SHT_NUM
 - ELF binary format, [622](#)
- SHT_PREINIT_ARRAY
 - ELF binary format, [622](#)
- SHT_PROGBITS
 - ELF binary format, [621](#)
- SHT_REL
 - ELF binary format, [621](#)
- SHT_RELA
 - ELF binary format, [621](#)
- SHT_SHLIB
 - ELF binary format, [622](#)
- SHT_STRTAB

- ELF binary format, [621](#)
- SHT_SYMTAB
 - ELF binary format, [621](#)
- SHT_SYMTAB_SHNDX
 - ELF binary format, [622](#)
- shutdown
 - L4::Uart, [1333](#)
- si
 - l4_vcpu_regs_t, [1379](#)
- Sigma0 API, [552](#)
 - l4sigma0_debug_dump, [554](#)
 - L4SIGMA0_IPCERROR, [553](#)
 - l4sigma0_map_anypage, [554](#)
 - l4sigma0_map_errstr, [554](#)
 - l4sigma0_map_iomem, [555](#)
 - l4sigma0_map_kip, [555](#)
 - l4sigma0_map_mem, [556](#)
 - L4SIGMA0_NOFPAGE, [553](#)
 - L4SIGMA0_NOTALIGNED, [553](#)
 - L4SIGMA0_OK, [553](#)
 - l4sigma0_return_flags_t, [553](#)
 - L4SIGMA0_SMALLERFPAGE, [553](#)
- Sigma0, the Root-Pager, [45](#)
- signal
 - L4Re::Parent, [1492](#)
- Signals, [544](#)
 - l4shmc_add_signal, [545](#)
 - l4shmc_attach_signal, [545](#)
 - l4shmc_check_magic, [546](#)
 - l4shmc_get_signal, [546](#)
 - l4shmc_signal_cap, [546](#)
- size
 - L4::Kip::Mem_desc, [1161](#)
 - L4Re::Dataspace, [1423](#)
 - L4Re::Video::Color_component, [1630](#)
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1862](#)
- skip
 - L4::lpc::lstream, [1039](#)
 - L4virtio::Svr::Data_buffer, [1825](#)
- slab_size
 - cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, [750](#)
 - cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, [756](#)
- Small C++ Template Library, [558](#)
 - clamp, [559](#)
 - max, [560](#)
 - min, [560](#)
 - operator new, [561](#)
- Small_buf
 - L4::lpc::Small_buf, [1073](#), [1074](#)
- Smart_cap
 - L4::Smart_cap< T, SMART >, [1250](#)
- snd_base
 - L4::Cap_base, [938](#)
- Space_attrb
 - L4Re::Dma_space, [1432](#)
- Spaces and Mappings, [19](#)
- Split_ds
 - L4Re::Rm, [1504](#)
- Src_dev_handle
 - L4vbus::lcu, [1712](#)
- Src_types
 - L4vbus::lcu, [1712](#)
- ss
 - l4_exc_regs_t, [1354](#)
- stack
 - L4Re::Video::View, [1651](#)
- start
 - L4::Kip::Mem_desc, [1161](#)
 - L4virtio::Svr::Request_processor, [1867](#), [1868](#)
- start_request
 - L4virtio::Driver::Block_device, [1770](#)
- starts_with
 - cxx::String, [887](#)
- startup
 - L4::Uart, [1334](#)
- State
 - L4vcpu::State, [1741](#)
- state
 - L4vcpu::Vcpu, [1751](#), [1752](#)
- stats_time
 - L4::Thread, [1271](#)
- status
 - L4virtio::Svr::Dev_config, [1837](#)
 - l4virtio_config_hdr_t, [1907](#)
- STB_GLOBAL
 - ELF binary format, [622](#)
- STB_HIOS
 - ELF binary format, [622](#)
- STB_HIPROC
 - ELF binary format, [622](#)
- STB_LOCAL
 - ELF binary format, [622](#)
- STB_LOOS
 - ELF binary format, [622](#)
- STB_LOPROC
 - ELF binary format, [622](#)
- STB_WEAK
 - ELF binary format, [622](#)
- Str_cp_in
 - L4::lpc::Str_cp_in< T >, [1076](#)
- str_cp_in
 - L4::lpc, [673](#)
- String
 - cxx::String, [885](#)
- Strong
 - L4Re::Namespace, [1484](#)
- STT_FILE
 - ELF binary format, [623](#)
- STT_FUNC
 - ELF binary format, [623](#)
- STT_HIOS
 - ELF binary format, [623](#)
- STT_HIPROC

- ELF binary format, [623](#)
- STT_LOOS
 - ELF binary format, [623](#)
- STT_LOPROC
 - ELF binary format, [623](#)
- STT_NOTYPE
 - ELF binary format, [623](#)
- STT_OBJECT
 - ELF binary format, [623](#)
- STT_SECTION
 - ELF binary format, [623](#)
- sub_type
 - L4::Kip::Mem_desc, [1162](#)
- Super_pages
 - L4Re::Mem_alloc, [1475](#)
- supports
 - L4::Meta, [1184](#)
- switch_log
 - L4::Debugger, [951](#)
- switch_to
 - L4::Thread, [1271](#)
- symlink
 - L4Re::Vfs::Directory, [1601](#)
- system_shutdown
 - L4::Platform_control, [1195](#)
- system_suspend
 - L4::Platform_control, [1196](#)
- tag
 - L4::lpc::Istream, [1040](#)
 - L4::lpc::Ostream, [1069](#)
 - L4::lpc::Varg, [1080](#)
- take
 - L4Re::Util::Counting_cap_alloc< COUNTERTYPE
>, [1542](#)
 - L4Re::Util::Dataspace_svr, [1547](#)
- Task, [249](#)
 - L4_FP_ALL_SPACES, [250](#)
 - L4_FP_DELETE_OBJ, [250](#)
 - L4_FP_OTHER_SPACES, [250](#)
 - I4_task_add_ku_mem, [250](#)
 - I4_task_cap_equal, [251](#)
 - I4_task_cap_valid, [252](#)
 - I4_task_delete_obj, [253](#)
 - I4_task_map, [253](#)
 - I4_task_release_cap, [255](#)
 - I4_task_unmap, [256](#)
 - I4_task_unmap_batch, [257](#)
 - I4_task_vgicc_map, [258](#)
 - I4_unmap_flags_t, [250](#)
- task
 - L4Re::Env, [1447](#)
 - L4vcpu::Vcpu, [1752](#)
- test
 - L4::Poll_timeout_kipclock, [1201](#)
- test.mk - Test Application Role, [33](#)
- Thread, [259](#)
 - I4_thread_arm_set_tpidruro, [262](#)
 - L4_THREAD_CONTROL_ALIEN, [262](#)
 - L4_THREAD_CONTROL_BIND_TASK, [262](#)
 - L4_thread_control_flags, [261](#)
 - L4_THREAD_CONTROL_MR_IDX_BIND_TASK, [262](#)
 - L4_THREAD_CONTROL_MR_IDX_BIND_UTCB, [262](#)
 - L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER, [262](#)
 - L4_THREAD_CONTROL_MR_IDX_FLAG_VALS, [262](#)
 - L4_THREAD_CONTROL_MR_IDX_FLAGS, [262](#)
 - L4_THREAD_CONTROL_MR_IDX_PAGER, [262](#)
 - L4_thread_control_mr_indices, [262](#)
 - L4_THREAD_CONTROL_SET_EXC_HANDLER, [262](#)
 - L4_THREAD_CONTROL_SET_PAGER, [262](#)
 - L4_THREAD_CONTROL_UX_NATIVE, [262](#)
 - I4_thread_ex_regs, [263](#)
 - L4_THREAD_EX_REGS_CANCEL, [262](#)
 - L4_thread_ex_regs_flags, [262](#)
 - I4_thread_ex_regs_ret, [264](#)
 - I4_thread_ex_regs_ret_u, [265](#)
 - L4_THREAD_EX_REGS_TRIGGER_EXCEPTION, [262](#)
 - I4_thread_ex_regs_u, [266](#)
 - I4_thread_modify_sender_add, [267](#)
 - I4_thread_modify_sender_commit, [268](#)
 - I4_thread_modify_sender_start, [269](#)
 - I4_thread_register_del_irq, [270](#)
 - I4_thread_stats_time, [271](#)
 - I4_thread_switch, [271](#)
 - I4_thread_vcpu_control, [272](#)
 - I4_thread_vcpu_control_ext, [273](#)
 - I4_thread_vcpu_control_ext_u, [274](#)
 - I4_thread_vcpu_control_u, [275](#)
 - I4_thread_vcpu_resume_commit, [276](#)
 - I4_thread_vcpu_resume_start, [277](#)
 - I4_thread_yield, [278](#)
- Thread control, [279](#)
 - I4_thread_control_alien, [280](#)
 - I4_thread_control_bind, [280](#)
 - I4_thread_control_commit, [281](#)
 - I4_thread_control_exc_handler, [282](#)
 - I4_thread_control_pager, [283](#)
 - I4_thread_control_start, [283](#)
 - I4_thread_control_ux_host_syscall, [284](#)
- Thread Control Registers (TCRs), [388](#)
- thread.h
 - __L4UTIL_THREAD_FUNC, [2856](#)
- throw_error
 - L4Re, [696](#)
- throw_ipc_exception
 - L4, [664](#), [665](#)
- timed_out
 - L4::Poll_timeout_counter, [1198](#)
 - L4::Poll_timeout_kipclock, [1202](#)
- timeout
 - L4::lpc_svr::Timeout, [1118](#)

- timeout_expired
 - L4::lpc_svr::Timeout_queue, [1123](#)
- Timeouts, [370](#)
 - l4_ipc_timeout, [372](#)
 - L4_IPC_TIMEOUT_0, [372](#)
 - l4_rcv_timeout, [373](#)
 - l4_snd_timeout, [373](#)
 - l4_timeout, [373](#)
 - l4_timeout_abs, [374](#)
 - l4_timeout_get, [375](#)
 - l4_timeout_is_absolute, [376](#)
 - l4_timeout_rel, [376](#)
 - l4_timeout_rel_get, [377](#)
 - l4_timeout_s, [372](#)
 - l4_timeout_t, [372](#)
 - l4_utcb_mr64_idx, [377](#)
- Timestamp Counter, [634](#)
 - l4_busy_wait_ns, [635](#)
 - l4_busy_wait_us, [636](#)
 - l4_calibrate_tsc, [636](#)
 - l4_get_hz, [637](#)
 - l4_ns_to_tsc, [637](#)
 - l4_rdpmc, [638](#)
 - l4_rdpmc_32, [638](#)
 - l4_rdtsc, [638](#)
 - l4_rdtsc_32, [639](#)
 - l4_tsc_init, [639](#)
 - l4_tsc_to_ns, [640](#)
 - l4_tsc_to_s_and_ns, [640](#)
 - l4_tsc_to_us, [641](#)
- To_default
 - L4Re::Namespace, [1483](#)
- To_device
 - L4Re::Dma_space, [1432](#)
- to_irq
 - L4vbus::Gpio_pin, [1708](#)
- To_non_blocking
 - L4Re::Namespace, [1483](#)
- total_objects
 - cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, [753](#)
 - cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, [758](#)
- total_size
 - L4virtio::Virtqueue, [1891](#), [1892](#)
- trigger
 - L4::Triggerable, [1286](#)
- trunc_order
 - L4, [665](#)
- Trusted
 - L4Re::Namespace, [1484](#)
- Tutorial, [7](#)
- tx
 - L4virtio::Driver::Virtio_net_device, [1791](#)
- tx_queue_size
 - L4virtio::Driver::Virtio_net_device, [1792](#)
- type
 - L4::lpc::Varg, [1081](#)
 - L4::Kip::Mem_desc, [1162](#)
 - L4Re::Vfs::Be_file_system, [1596](#)
 - L4Re::Vfs::File_system, [1606](#)
- types
 - L4_TYPES_FLAGS_OPS_DEF, [2624](#)
- types.h
 - l4_capability_next, [2284](#)
- unbind
 - L4::lcu, [994](#)
 - L4::lommu, [1004](#)
- Uncacheable
 - L4Re::Dataspace::F, [1425](#)
- Undefined
 - L4::Kip::Mem_desc, [1155](#)
- Unique_cap
 - L4Re, [686](#)
 - L4Re::Util, [702](#)
- unique_cap
 - L4Re, [687](#)
 - L4Re::Util, [703](#)
- Unique_del_cap
 - L4Re, [687](#)
 - L4Re::Util, [703](#)
- unique_del_cap
 - L4Re, [687](#)
 - L4Re::Util, [704](#)
- Unique_region
 - L4Re::Rm::Unique_region< T >, [1517](#), [1518](#)
- unlink
 - L4Re::Namespace, [1486](#)
 - L4Re::Vfs::Directory, [1601](#)
- unlock_all_locks
 - L4Re::Vfs::Be_file, [1593](#)
 - L4Re::Vfs::Generic_file, [1614](#)
- unmap
 - L4::Task, [1260](#)
 - L4Re::Dma_space, [1435](#)
- unmap_batch
 - L4::Task, [1261](#)
- unmask
 - L4::lirq, [1136](#)
 - L4::lirq_eoi, [1139](#)
- unregister_obj
 - L4::Registry_iface, [1211](#)
 - L4Re::Util::Object_registry, [1575](#)
- up
 - L4::Semaphore, [1227](#)
- used_align
 - L4virtio::Virtqueue, [1893](#)
- Used_elem
 - L4virtio::Virtqueue::Used_elem, [1902](#)
- used_size
 - L4virtio::Virtqueue, [1893](#)
- utcb_area
 - L4Re::Env, [1447](#), [1448](#)
- util.h
 - l4_sleep, [1968](#), [1972](#)
 - l4util_micros2l4to, [1969](#), [1973](#)

- Utility Functions, [562](#)
 - [l4_sleep](#), [564](#)
 - [l4_touch_ro](#), [565](#)
 - [l4_touch_rw](#), [565](#)
 - [l4_usleep](#), [566](#)
 - [l4util_micros2l4to](#), [567](#)
 - [l4util_splitlog2_hdl](#), [567](#)
 - [l4util_splitlog2_size](#), [568](#)
- Uvmm, the virtual machine monitor, [60](#)
- ux_host_syscall
 - [L4::Thread::Attr](#), [1281](#)
- V_flags
 - [L4Re::Video::View](#), [1649](#)
- val
 - [cxx::Bitfield< T, LSB, MSB >](#), [767](#)
- val_dirty
 - [cxx::Bitfield< T, LSB, MSB >](#), [768](#)
- val_unshifted
 - [cxx::Bitfield< T, LSB, MSB >](#), [770](#)
- valid
 - [cxx::Bits::Base_avl_set< ITEM_TYPE, COM-PARE, ALLOC, GET_KEY >::Node](#), [806](#)
 - [L4virtio::Svr::Virtqueue::Head_desc](#), [1878](#)
- validate
 - [L4::Cap_base](#), [938](#), [939](#)
- value
 - [L4::lpc::Varg](#), [1081](#)
- Varg_list_ref
 - [L4::lpc::Varg_list_ref](#), [1086](#)
- vbe_ctrl_info
 - [l4util_l4mod_info](#), [1672](#)
- Vbus API, [523](#)
- vbus.h
 - [L4VBUS_ICU_SRC_DEV_HANDLE](#), [2861](#)
 - [l4vbus_icu_src_types](#), [2860](#)
 - [L4VBUS_NULL](#), [2860](#)
 - [L4VBUS_ROOT_BUS](#), [2860](#)
- vbus_interfaces.h
 - [L4VBUS_IFACE_SHIFT](#), [2867](#)
 - [l4vbus_iface_type_t](#), [2867](#)
 - [L4VBUS_INTERFACE_BUS](#), [2868](#)
 - [L4VBUS_INTERFACE_GENERIC](#), [2868](#)
 - [L4VBUS_INTERFACE_GPIO](#), [2868](#)
 - [L4VBUS_INTERFACE_ICU](#), [2868](#)
 - [L4VBUS_INTERFACE_PCI](#), [2868](#)
 - [L4VBUS_INTERFACE_PCIDEV](#), [2868](#)
 - [L4VBUS_INTERFACE_PM](#), [2868](#)
 - [l4vbus_subinterface_supported](#), [2868](#)
- vbus_types.h
 - [L4VBUS_DEVICE_F_CHILDREN](#), [2873](#)
 - [l4vbus_device_flags_t](#), [2873](#)
 - [L4VBUS_RESOURCE_BUS](#), [2874](#)
 - [L4VBUS_RESOURCE_DMA_DOMAIN](#), [2874](#)
 - [L4VBUS_RESOURCE_F_MEM_MMIO_READ](#), [2874](#)
 - [L4VBUS_RESOURCE_F_MEM_MMIO_WRITE](#), [2874](#)
 - [L4VBUS_RESOURCE_F_MEM_R](#), [2874](#)
 - [L4VBUS_RESOURCE_F_MEM_W](#), [2874](#)
 - [l4vbus_resource_flags_t](#), [2873](#)
 - [L4VBUS_RESOURCE_GPIO](#), [2874](#)
 - [L4VBUS_RESOURCE_INVALID](#), [2874](#)
 - [L4VBUS_RESOURCE_IRQ](#), [2874](#)
 - [L4VBUS_RESOURCE_MAX](#), [2874](#)
 - [L4VBUS_RESOURCE_MEM](#), [2874](#)
 - [L4VBUS_RESOURCE_PORT](#), [2874](#)
 - [l4vbus_resource_type_t](#), [2874](#)
- vcon.h
 - [L4_vcon_read_flags](#), [2767](#)
 - [L4_VCON_READ_SIZE_MASK](#), [2767](#)
 - [L4_VCON_READ_STAT_BREAK](#), [2767](#)
 - [L4_VCON_READ_STAT_DONE](#), [2767](#)
- vCPU API, [285](#)
 - [L4_VCPU_F_EXCEPTIONS](#), [288](#)
 - [L4_VCPU_F_FPU_ENABLED](#), [288](#)
 - [L4_VCPU_F_IRQ](#), [287](#)
 - [L4_VCPU_F_PAGE_FAULTS](#), [287](#)
 - [L4_VCPU_F_USER_MODE](#), [288](#)
 - [L4_VCPU_OFFSET_EXT_INFOS](#), [288](#)
 - [L4_VCPU_OFFSET_EXT_STATE](#), [288](#)
 - [L4_VCPU_SF_IRQ_PENDING](#), [288](#)
 - [L4_vcpu_state_flags](#), [287](#)
 - [L4_vcpu_state_offset](#), [288](#)
 - [L4_vcpu_sticky_flags](#), [288](#)
- vCPU Support Library, [641](#)
 - [l4vcpu_irq_disable](#), [642](#)
 - [l4vcpu_irq_disable_save](#), [643](#)
 - [l4vcpu_irq_enable](#), [644](#)
 - [l4vcpu_irq_restore](#), [645](#)
 - [l4vcpu_is_irq_entry](#), [646](#)
 - [l4vcpu_is_page_fault_entry](#), [646](#)
 - [l4vcpu_print_state](#), [647](#)
 - [l4vcpu_wait_for_event](#), [648](#)
- vcpu.h
 - [l4_vcpu_check_version](#), [2880](#)
- vcpu_control
 - [L4::Thread](#), [1272](#)
- vcpu_control_ext
 - [L4::Thread](#), [1273](#)
- vcpu_resume_commit
 - [L4::Thread](#), [1274](#)
- vcpu_resume_start
 - [L4::Thread](#), [1275](#)
- version
 - [l4_vcpu_state_t](#), [1382](#)
- vgicc_map
 - [L4::Vm](#), [1350](#)
- vicu
 - [L4vbus::lcu](#), [1712](#)
- Video API, [499](#)
 - [F_l4re_video_goos_auto_refresh](#), [501](#)
 - [F_l4re_video_goos_dynamic_buffers](#), [501](#)
 - [F_l4re_video_goos_dynamic_views](#), [501](#)
 - [F_l4re_video_goos_pointer](#), [501](#)
 - [F_l4re_video_view_above](#), [501](#)
 - [F_l4re_video_view_dyn_allocated](#), [501](#)

- F_l4re_video_view_flags_mask, 501
- F_l4re_video_view_none, 501
- F_l4re_video_view_set_background, 501
- F_l4re_video_view_set_buffer, 501
- F_l4re_video_view_set_buffer_offset, 501
- F_l4re_video_view_set_bytes_per_line, 501
- F_l4re_video_view_set_flags, 501
- F_l4re_video_view_set_pixel, 501
- F_l4re_video_view_set_position, 501
- l4re_video_goos_create_buffer, 501
- l4re_video_goos_create_view, 502
- l4re_video_goos_delete_buffer, 502
- l4re_video_goos_delete_view, 502
- l4re_video_goos_get_static_buffer, 502
- l4re_video_goos_get_view, 503
- l4re_video_goos_info, 503
- l4re_video_goos_info_flags_t, 500
- l4re_video_goos_refresh, 504
- l4re_video_view_get_info, 504
- l4re_video_view_info_flags_t, 501
- l4re_video_view_refresh, 504
- l4re_video_view_set_info, 505
- l4re_video_view_set_viewport, 505
- l4re_video_view_stack, 505
- l4re_video_view_t, 500
- view
 - L4Re::Video::Goos, 1637
- view_info
 - L4Re::Util::Video::Goos_svr, 1589
- Virtual Console, 289
 - l4_vcon_attr_t, 291
 - L4_VCON_ECHO, 291
 - l4_vcon_get_attr, 292
 - l4_vcon_get_attr_u, 293
 - L4_vcon_i_flags, 291
 - L4_VCON_ICANON, 291
 - L4_VCON_ICRNL, 291
 - L4_VCON_IGNCR, 291
 - L4_VCON_INLCR, 291
 - L4_vcon_l_flags, 291
 - L4_vcon_o_flags, 291
 - L4_VCON_OCRNL, 292
 - L4_VCON_ONLCR, 292
 - L4_VCON_ONLRET, 292
 - l4_vcon_read, 294
 - L4_VCON_READ_SIZE, 292
 - l4_vcon_read_u, 295
 - l4_vcon_read_with_flags, 296
 - l4_vcon_send, 297
 - l4_vcon_send_u, 298
 - l4_vcon_set_attr, 299
 - l4_vcon_set_attr_raw, 300
 - l4_vcon_set_attr_u, 300
 - L4_vcon_size_consts, 292
 - l4_vcon_write, 301
 - L4_VCON_WRITE_SIZE, 292
 - l4_vcon_write_u, 302
- Virtual Machines, 303
 - Virtual Registers (UTCBS), 378
 - l4_utcb_br, 380
 - l4_utcb_mr, 380
 - l4_utcb_t, 380
 - l4_utcb_tcr, 381
 - VM API for SVM, 304
 - VM API for TZ, 305
 - VM API for VMX, 305
 - L4_VM_VMX_BASIC_REG, 308
 - L4_vm_vmx_caps_regs, 307
 - l4_vm_vmx_clear, 308
 - L4_VM_VMX_CR0_FIXED0_REG, 308
 - L4_VM_VMX_CR0_FIXED1_REG, 308
 - L4_VM_VMX_CR4_FIXED0_REG, 308
 - L4_VM_VMX_CR4_FIXED1_REG, 308
 - L4_vm_vmx_dfl1_regs, 308
 - L4_VM_VMX_ENTRY_CTLTS_DFL1_REG, 308
 - L4_VM_VMX_EPT_VPID_CAP_REG, 308
 - L4_VM_VMX_EXIT_CTLTS_DFL1_REG, 308
 - l4_vm_vmx_field_len, 309
 - l4_vm_vmx_field_order, 309
 - l4_vm_vmx_get_caps, 310
 - l4_vm_vmx_get_caps_default1, 310
 - l4_vm_vmx_get_cr2_index, 311
 - L4_VM_VMX_MISC_REG, 308
 - L4_VM_VMX_NUM_CAPS_REGS, 308
 - L4_VM_VMX_NUM_DFL1_REGS, 308
 - L4_VM_VMX_PINBASED_CTLTS_DFL1_REG, 308
 - L4_VM_VMX_PROCBASED_CTLTS2_REG, 308
 - L4_VM_VMX_PROCBASED_CTLTS_DFL1_REG, 308
 - l4_vm_vmx_ptr_load, 311
 - l4_vm_vmx_read, 312
 - l4_vm_vmx_read_16, 313
 - l4_vm_vmx_read_32, 313
 - l4_vm_vmx_read_64, 314
 - l4_vm_vmx_read_nat, 315
 - L4_VM_VMX_TRUE_ENTRY_CTLTS_REG, 308
 - L4_VM_VMX_TRUE_EXIT_CTLTS_REG, 308
 - L4_VM_VMX_TRUE_PINBASED_CTLTS_REG, 308
 - L4_VM_VMX_TRUE_PROCBASED_CTLTS_REG, 308
 - L4_VM_VMX_VMCS_CR2, 307
 - L4_VM_VMX_VMCS_ENUM_REG, 308
 - L4_VM_VMX_VMCS_MSR_CSTAR, 307
 - L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE, 307
 - L4_VM_VMX_VMCS_MSR_LSTAR, 307
 - L4_VM_VMX_VMCS_MSR_STAR, 307
 - L4_VM_VMX_VMCS_MSR_SYSCALL_MASK, 307
 - L4_VM_VMX_VMCS_MSR_TSC_AUX, 307
 - L4_VM_VMX_VMCS_XCR0, 307
 - l4_vm_vmx_write, 315
 - l4_vm_vmx_write_16, 316
 - l4_vm_vmx_write_32, 317
 - l4_vm_vmx_write_64, 317

- l4_vm_vmx_write_nat, [318](#)
- W
 - L4Re::Dataspace::F, [1425](#)
 - L4Re::Rm::F, [1513](#)
- W_bits
 - cxx::Bitmap_base, [782](#)
- wait
 - L4::lpc::lstream, [1041](#), [1042](#)
 - L4::lirq, [1137](#)
 - L4virtio::Driver::Device, [1783](#)
- wait_for_event
 - L4vcpu::Vcpu, [1753](#)
- wait_for_next_used
 - L4virtio::Driver::Device, [1784](#)
- wait_rx
 - L4virtio::Driver::Virtio_net_device, [1793](#)
- Wd_16bit
 - L4Re::Mmio_space, [1479](#)
- Wd_32bit
 - L4Re::Mmio_space, [1479](#)
- Wd_64bit
 - L4Re::Mmio_space, [1479](#)
- Wd_8bit
 - L4Re::Mmio_space, [1479](#)
- Word_bytes
 - L4::lpc::Msg, [675](#)
- word_index
 - cxx::Bitmap_base, [786](#)
- write
 - L4::Uart, [1334](#)
 - L4::Vcon, [1345](#)
 - L4drivers::Register_tmpl< BITS, BLOCK >, [1402](#)
- write_bfm_t
 - L4virtio::Virtqueue::Desc::Flags, [1899](#)
- write_now
 - cxx, [654](#)
- writew
 - L4Re::Vfs::Regular_file, [1624](#)
- X
 - L4Re::Dataspace::F, [1425](#)
 - L4Re::Rm::F, [1513](#)
- x86 Virtual Registers (UTCB), [389](#)
 - L4_UTCB_BUF_REGS_OFFSET, [390](#)
 - L4_utcb_consts_x86, [390](#)
 - L4_UTCB_EXCEPTION_REGS_SIZE, [390](#)
 - L4_UTCB_GENERIC_BUFFERS_SIZE, [390](#)
 - L4_UTCB_GENERIC_DATA_SIZE, [390](#)
 - L4_UTCB_INHERIT_FPU, [390](#)
 - L4_UTCB_MSG_REGS_OFFSET, [390](#)
 - L4_UTCB_OFFSET, [390](#)
 - L4_UTCB_THREAD_REGS_OFFSET, [390](#)
- x86/l4/sys/__kip-arch.h, [1998](#)
- x86/l4/sys/__vcpu-arch.h, [2004](#), [2005](#)
- x86/l4/sys/cache.h, [2546](#)
- x86/l4/sys/consts.h, [2563](#)
- x86/l4/sys/ipc-invoke.h, [2885](#)
- x86/l4/sys/ktrace_events.h, [2012](#)
- x86/l4/sys/l4int.h, [2701](#), [2702](#)
- x86/l4/sys/linkage.h, [2017](#), [2018](#)
- x86/l4/sys/segment.h, [1983](#), [1986](#)
- x86/l4/sys/utcb.h, [2759](#), [2761](#)
- x86/l4/sys/vm.h, [2023](#)
- x86/l4/util/bitops_arch.h, [2027](#)
- x86/l4/util/cpu.h, [2033](#), [2034](#)
- x86/l4/util/idt.h, [1942](#), [1943](#)
- x86/l4/util/irq.h, [2183](#), [2185](#)
- x86/l4/util/l4_macros.h, [2037](#), [2038](#)
- x86/l4/util/mbi_argv.h, [2041](#), [2042](#)
- x86/l4/util/perform.h, [1950](#), [1951](#)
- x86/l4/util/port_io.h, [1990](#), [1992](#)
- x86/l4/util/rdtsc.h, [1960](#), [1962](#)
- x86/l4/util/spin.h, [1966](#), [1967](#)
- x86/l4/util/util.h, [1971](#), [1973](#)
- x86/l4f/l4/sys/ipc-l42-gcc3-nopic.h, [2885](#)
- x86/l4f/l4/sys/ipc.h, [2663](#), [2664](#)
- x86/l4f/l4/sys/segment.h, [1987](#), [1988](#)
- x86/l4f/l4/util/port_io.h, [1994](#), [1997](#)